FACULDADE DE ECONOMIA DA UNIVERSIDADE DE COIMBRA

Apoio à Decisão em Problemas de Programação Inteira e Inteira-mista Multiobjectivo: Contribuições Metodológicas

Maria João Teixeira Gomes Alves

Dissertação apresentada à Faculdade de Economia da Universidade de Coimbra para obtenção do grau de Doutor em Organização e Gestão de Empresas, especialidade de Investigação Operacional.

Coimbra – Portugal
2000

Índice

Agradecimentos

Ao Prof. João Clímaco quero expressar o meu maior agradecimento, não só pela orientação e acompanhamento deste trabalho, mas também pelo entusiasmo e ânimo que sempre me incutiu ao longo do trabalho.

À Faculdade de Economia da Universidade de Coimbra agradeço o apoio concedido durante o período de investigação, em particular a dispensa de serviço docente durante dois anos consecutivos. Endereço um agradecimento particular à Dra. Teresa Lello pelo esforço desenvolvido na correcção dos textos publicados em inglês, relacionados com este trabalho. Ao INESC de Coimbra agradeço o apoio logístico e financeiro de parte do trabalho, nomeadamente através do projecto PRAXIS XXI 2/2.1/MAT/465/94. Agradeço também à Fundação Calouste Gulbenkian a bolsa que me concedeu para apresentação de uma comunicação numa conferência internacional.

Aos meus colegas do INESC e da FEUC agradeço a amizade e o bom ambiente de trabalho que sempre me proporcionaram. Um agradecimento especial endereço aos Profs. João Paulo Costa e Carlos Henggeler Antunes por terem sugerido alterações em alguns capítulos desta dissertação.

Quero, por fim, expressar a minha gratidão a todos os meus amigos e familiares que de alguma forma contribuíram e me ajudaram neste trabalho, em particular o Zé Manel, os meus pais e as minhas irmãs.

Resumo

Neste trabalho propusemo-nos desenvolver novas metodologias para apoio à decisão em problemas de programação linear inteira e inteira-mista multiobjectivo (PLIMO e PLIMMO). A concepção de métodos que possibilitem um apoio à decisão eficaz neste tipo de problemas depara-se com várias dificuldades, uma vez que a complexidade computacional dos problemas com variáveis discretas se agrava na presença de múltiplos objectivos. Tendo em vista a concepção de abordagens interactivas, é, pois, importante que nos preocupemos simultaneamente com questões da programação matemática (relacionadas com o cálculo de soluções não dominadas do problema multiobjectivo), e com a condução do processo interactivo em que o agente de decisão tem um papel essencial.

No âmbito de um estudo teórico, começámos por abordar aspectos relacionados com o cálculo de soluções não dominadas de problemas de PLIMO/PLIMMO através da minimização da distância de Tchebycheff a pontos de referência. Desenvolvemos em seguida dois métodos interactivos de pontos de referência. O primeiro destina-se apenas a problemas de PLIMO (em que todas as variáveis são inteiras), ao passo que o segundo também se aplica a problemas de PLIMMO (em que há variáveis inteiras e variáveis contínuas). Os dois métodos diferem tecnicamente, partilhando as mesmas características de interacção com o agente de decisão. Procurámos estabelecer um protocolo simples de diálogo com o agente de decisão e reduzir o esforço computacional envolvido nas fases de cálculo de soluções não dominadas. Estes métodos são especialmente vocacionados para pesquisas direccionais, em que o agente de decisão tem apenas que indicar o objectivo que pretende melhorar relativamente a uma solução prévia. Para o cálculo das soluções não dominadas seguintes é usado um processo de análise de sensibilidade que, no primeiro método, é baseado em planos de corte e, no segundo, é baseado em *branch-and-bound*. A intenção é ajustar automaticamente o ponto de referência, projectando-o em seguida no conjunto das soluções não dominadas através da minimização da distância de Tchebycheff a esse ponto de referência (programa escalarizante de programação inteira ou inteira-mista). As sucessivas fases de cálculo não são independentes, tirando-se partido do cálculo anterior para a obtenção da solução seguinte. Em particular, o método baseado em *branch-and-bound* utiliza a anterior árvore de *branch-and-bound* para efectuar a análise de sensibilidade, fazendo em seguida operações de simplificação e ramificação da árvore conducentes à nova solução. Os resultados obtidos permitem-nos concluir que esta forma de proceder é eficaz na medida em que reduz o esforço computacional envolvido.

Desenvolvemos ainda uma abordagem baseada em meta-heurísticas para apoio à decisão em problemas multiobjectivo com variáveis binárias. Esta abordagem constitui uma extensão de algoritmos genéricos de *tabu search* e *simulated annealing* num contexto multicritério interactivo.

Os métodos referidos foram integrados num sistema computacional de apoio à decisão que implementámos no ambiente DELPHI para WINDOWS 95/98.

Abstract

In this work we have investigated new approaches for decision support in multiobjective integer linear programming (MOILP) and multiobjective mixed-integer linear programming (MOMILP) problems. The development of methods that provide an effective decision support for these problems faces two relevant types of difficulties, those caused by the existence of more than one objective function, and the computational complexity of the problems with discrete variables. It is, therefore, imperative to pay special attention to the issues concerning mathematical programming as well as the decision process in order to build effective interactive methods that help the decision maker (DM) to compromise and choose an acceptable solution.

We have investigated interactive reference point approaches for MOILP/MOMILP problems. The first approach is based on *cutting plane* techniques and is devoted to MOILP problems. We have further investigated another reference point approach based on *branch-and-bound*, which also addresses MOMILP problems. These approaches differ technically but share the same features in what concerns the interaction with the DM. Their aim is to provide a simple protocol to interact with the DM, without demanding too much information about his/her preferences, and to reduce the computational effort involved in the computation of nondominated solutions. Both methods are particularly interesting to perform directional searches, but the second one is more effective because it is more reliable. To start a directional search for nondominated solutions the DM chooses only the objective function he/she wishes to improve in relation to the previous nondominated solution. A sensitivity analysis procedure is then applied to adjust automatically the reference point and to project it onto the nondominated solutions set through the minimisation of the Tchebycheff distance to that reference point (an integer or mixed-integer scalarizing program). The successive computing phases are not independent since the procedure profits from the previous computation to produce the following solution. In particular, the branch-and-bound multiobjective approach uses the previous branch-and-bound tree to perform the sensitivity analysis and proceed to the computation of the new nondominated solution (using operations of simplification and branching of the tree). The results obtained evidence the effectiveness of this procedure in reducing the computational effort.

We have also developed an interactive metaheuristic approach to deal with multiobjective 0-1 linear problems, which is an extension of general *tabu search* and *simulated annealing* algorithms in an interactive multiobjective environment.

All the approaches have been integrated into a decision support software we have developed using the DELPHI developer for WINDOWS 95/98.

Capítulo I

Introdução

Nas últimas décadas, o desenvolvimento de métodos e sistemas de apoio à decisão dedicados a problemas com múltiplos critérios tem merecido, por parte dos investigadores, uma atenção crescente. De facto, tem sido frequentemente reconhecido que há vantagens em considerar mais do que um critério na modelação de problemas, no sentido de aproximar os modelos à realidade. Contudo, a relevância de uma abordagem multicritério vai para além do argumento "realista", possuindo um valor acrescentado intrínseco no processo de modelação e da análise do modelo, apoiando a reflexão e a criatividade face a um conjunto mais vasto de soluções potenciais (*cf.* BOUYSSOU, 1993).

Podemos distinguir dois grandes grupos de problemas multicritério a que se associam as correspondentes abordagens metodológicas: a análise multicritério sobre um conjunto finito de soluções explicitamente conhecido – que se designa habitualmente por análise multiatributo – e a programação matemática multicritério (ou multiobjectivo) em que o conjunto das soluções admissíveis é definido implicitamente por restrições analíticas. O nosso estudo reporta-se à programação matemática multiobjectivo.

De acordo com a natureza das variáveis (contínuas, discretas), dos coeficientes (determinísticos, estocásticos ou difusos), das funções objectivo (lineares, não lineares), das restrições (lineares, não lineares), da região admissível (convexa, não convexa), existe uma grande variedade de problemas de programação matemática multiobjectivo. O modelo mais estudado é o de programação linear multiobjectivo (PLMO), cujo tratamento apresenta menores dificuldades (quer do ponto de vista teórico, quer do ponto de vista computacional) se comparado com o de outros modelos de programação matemática multiobjectivo. Contudo, a modelação de um sistema real requer muitas vezes a incorporação de fenómenos discretos, como, por exemplo, a escolha de investimentos, níveis de produção, a existência de custos fixos, condições lógicas ou restrições disjuntivas. Torna-se, então, necessário recorrer a variáveis

discretas para modelar tais situações. Muitos desses problemas podem ser modelados como PLMO com restrições adicionais que impõem a condição de que todas ou algumas variáveis assumam apenas valores inteiros. Estes modelos classificam-se como modelos de *programação linear inteira multiobjectivo* (PLIMO) ou de *programação linear inteira-mista multiobjectivo* (PLIMMO). Distinguimos os dois tipos pela natureza das variáveis, considerando que, no primeiro, todas as variáveis são inteiras (que também designamos por caso inteiro puro) e, no segundo, há variáveis inteiras e variáveis contínuas. Há diversas áreas potenciais para aplicação da PLIMO e PLIMMO, tais como o planeamento e gestão de redes de serviços, a gestão da produção, a análise e avaliação de projectos (composição de carteiras de projectos de investimento) e o planeamento financeiro. Os modelos multiobjectivo inteiros-mistos são, contudo, os que se adaptam a um maior número de situações reais.

O nosso trabalho centra-se nas áreas da PLIMO e da PLIMMO.

Existem diversos métodos dedicados a problemas de programação matemática multiobjectivo. No entanto, muito trabalho está ainda por fazer, especialmente no que se refere aos problemas que não se enquadram no modelo geral de PLMO. Na verdade, os problemas de PLIMO/PLIMMO (ou outros problemas de optimização combinatória multiobjectivo) têm recebido pouca atenção, sobretudo se comparados com os de PLMO ou os modelos multiatributo. De igual forma constatamos que a optimização combinatória multiobjectivo tem sido bastante ignorada em comparação com a vasta literatura sobre problemas combinatórios monocritério.

Poder-se-ia esperar que a combinação de métodos de PLMO com técnicas de resolução de problemas de programação inteira monocritério resultasse automaticamente num método de PLIMO/PLIMMO. No entanto, tal não acontece, sobretudo porque as variáveis inteiras introduzem dificuldades específicas. Talvez mais curioso seja o facto de as dificuldades serem por vezes maiores se o problema admitir variáveis inteiras e variáveis contínuas (caso inteiro-misto), como veremos ao longo deste trabalho.

Na perspectiva das aplicações, são também poucos os estudos de casos publicados nestas áreas, não obstante o crescente reconhecimento de que muitos problemas de decisão são intrinsecamente multiobjectivo, e que as variáveis inteiras são indispensáveis na modelação de muitos problemas reais. No estudo sobre aplicações de programação matemática multiobjectivo de WHITE (1990), que contém 504 referências cobrindo o período de 1955 a 1986, poucos são os problemas que são modelados com variáveis discretas. Este facto deve-se, segundo ULUNGU E TEGHEM (1994), às dificuldades inerentes ao cálculo e à insuficiência de fundamentos teóricos.

Foi neste contexto que decidimos desenvolver o nosso trabalho nas áreas de PLIMO e PLIMMO. As principais motivações foram as suas potenciais aplicações e o facto de a investigação publicada nestas áreas ser ainda relativamente escassa. Propusemo-nos desenvolver

metodologias que fossem adequadas ao tratamento de problemas de PLIMO/PLIMMO tendo em vista apoiar eficazmente o agente de decisão (AD) envolvido no problema. Assumimos, neste trabalho, a existência de um único agente de decisão.

A resolução de um problema de optimização monocritério é meramente técnica e consiste em encontrar a solução óptima da respectiva função objectivo. O mesmo não se passa com os problemas multiobjectivo, uma vez que não existe, em geral, uma solução que optimiza simultaneamente todas as funções objectivo. Assim, o conceito de solução óptima dá lugar à noção de conjunto de soluções eficientes (não dominadas). Uma solução admissível diz-se eficiente se não existir uma outra qualquer solução admissível que melhore um dos objectivos sem piorar, pelo menos, um dos outros. O processo de decisão não se resume ao cálculo de soluções eficientes, havendo a necessidade de considerar as preferências do AD para a selecção de uma solução de compromisso final de entre o conjunto das soluções eficientes.

Os métodos multiobjectivo podem classificar-se de acordo com o tipo de intervenção que é requerida do AD, distinguindo-se três tipos: *(i)* métodos em que é feita uma agregação *a priori* das preferências do AD; *(ii)* métodos *geradores* de soluções eficientes, em que a intervenção do AD é feita *a posteriori*; *(iii)* métodos *interactivos*, em que há uma progressiva articulação das preferências do AD. Nos métodos do primeiro tipo, a agregação de preferências é feita antes do cálculo, transformando o problema multiobjectivo num problema monocritério. Assim, a decisão é feita *a priori*. Em contraponto, os métodos *geradores* determinam todo ou parte do conjunto das soluções eficientes que é depois apresentado ao AD. As preferências do AD são, portanto, manifestadas *a posteriori*. Os métodos *interactivos* alternam fases de cálculo e fases de diálogo com o AD. As fases de diálogo recolhem informação acerca das preferências do AD que é depois usada nas fases de cálculo seguintes. O fim do processo é determinado pela satisfação do AD com a solução que lhe é apresentada após uma fase de cálculo ou por alguma condição de paragem do algoritmo.

No que diz respeito aos problemas multiobjectivo com variáveis inteiras, a maior parte dos métodos geradores publicados na literatura é dedicada a problemas com variáveis binárias ou a problemas biobjectivo. De facto, existem vários obstáculos na construção de métodos geradores para problemas de PLIMO (com mais do que duas funções objectivo), o que se pode constatar pelas características dos métodos existentes. Estes requerem, em geral, a resolução de problemas auxiliares de complexidade elevada. Tendo em conta que um problema multiobjectivo pode ter centenas ou milhares de soluções eficientes, o esforço computacional pode então ser muito elevado, com a dificuldade adicional de que se tornará difícil para o AD analisar no final um volume tão grande de informação. Talvez por estas razões sejam poucas as abordagens geradoras de PLIMO/PLIMMO que encontrámos na literatura (mesmo para os casos particulares), sendo quase todas anteriores à década de 90. Atendendo às dificuldades, não só na construção, mas

também na utilização de um método gerador, os métodos *interactivos* têm tido um desenvolvimento e aceitação crescentes, espelhando a tendência actual da programação multicritério. Foi, aliás, fundamentalmente nas décadas de 80 e 90 que surgiu a maior parte dos métodos interactivos para PLIMO e PLIMMO, como daremos conta na revisão bibliográfica que apresentaremos adiante.

Também as abordagens que propomos nesta dissertação são interactivas. Tendo como objectivo principal apoiar eficazmente o AD no tratamento de problemas de PLIMO e PLIMMO, considerámos as seguintes linhas de orientação para o desenvolvimento dessas abordagens: (i) construir um protocolo simples de interacção com o AD, não exigindo demasiado esforço cognitivo do AD em cada fase de diálogo; (ii) permitir uma pesquisa livre de soluções eficientes, em que não há decisões irrevogáveis e o fim do processo de decisão é determinado pelo AD; (iii) poder alcançar qualquer solução eficiente do problema; (iv) exigir menor esforço computacional.

O ponto (iii) classifica um método como 'completo', segundo LEWANDOWSKI E WIERZBICKI (1988), e, de acordo com os mesmos autores, é uma das propriedades mais importantes que um método deve ter, à qual acrescentam a simplicidade computacional e o ser controlável. Nos problemas de PLIMO/PLIMMO podem existir soluções eficientes que não estão na fronteira do invólucro convexo (*'convex hull'*) da região admissível. Estas soluções são geralmente designadas por soluções eficientes *não suportadas* e não são alcançáveis através do tradicional 'método dos pesos' em que se optimizam somas pesadas das funções objectivo. Por conseguinte, as somas pesadas das funções objectivo são *funções escalarizantes* (função escalar substituta cuja solução óptima é uma solução eficiente do problema multiobjectivo) que, por si só, não permitem que um método de PLIMO ou PLIMMO seja 'completo'. Pelo contrário, podemos definir outras funções escalarizantes que projectam pontos de referência no conjunto não dominado e que permitem alcançar qualquer solução eficiente. Um ponto de referência do espaço dos objectivos pode ser constituído por níveis de aspiração que o AD gostaria de alcançar. Do ponto de vista técnico, as abordagens baseadas em pontos de referência são, pois, particularmente atractivas, pelo que optámos por este tipo de processo de cálculo.

Tomando por base o programa escalarizante que minimiza a distância de Tchebycheff (pesada ou não) a um dado ponto de referência do espaço dos objectivos, este programa pode ser parametrizado ao nível do ponto de referência e/ou ao nível dos pesos (no caso da métrica pesada). Como os problemas multiobjectivo em causa são de PLIMO ou PLIMMO, os programas escalarizantes resultantes são problemas monocritério de programação inteira ou inteira-mista. A optimização destes programas para diferentes parâmetros permite o cálculo de diferentes soluções eficientes. Julgamos, porém, que poderá não ser fácil para o AD expressar sempre as suas preferências sob a forma de pontos de referência ou pesos. Tratando-se de

problemas com soluções discretas, uma pequena alteração dos parâmetros (ponto de referência e/ou pesos) pode conduzir a soluções eficientes muito diferentes, ou pelo contrário, uma grande alteração dos parâmetros pode conduzir à mesma solução. Nesta última situação, podemos ainda falar em desperdício de esforço computacional, o que é relevante em problemas difíceis como os de programação inteira. Nestas circunstâncias, julgámos que seria útil para o AD um tipo de *pesquisa direccional* que permitisse calcular soluções eficientes distintas segundo uma dada direcção.

O tipo de pesquisa direccional, segundo direcções particulares, foi aquele que adoptámos com maior relevância nas nossas abordagens. Em cada interacção, o AD deve especificar apenas a função objectivo que pretende melhorar relativamente à solução eficiente anterior. Percebemos que, se o ponto de referência fosse ajustado convenientemente, mantendo iguais todas as componentes excepto a correspondente ao objectivo a melhorar, então obtinha-se uma nova solução eficiente. O ajuste do ponto de referência deveria ser o necessário para abandonar a solução anterior e alcançar a solução seguinte segundo aquela direcção. Mas, para que o processo fosse automático, seria necessário proceder a uma análise de sensibilidade. Alterado o ponto de referência, surgiria um novo programa escalarizante que apenas diferia do anterior no valor dum parâmetro, e cuja optimização conduziria a uma outra solução eficiente. Duas questões se nos colocaram, a nosso ver essenciais para o bom desempenho do método: "como proceder à análise de sensibilidade para ajustar automaticamente o ponto de referência?" e "como diminuir o esforço computacional na resolução do programa escalarizante seguinte, nomeadamente sem ter de recomeçar do início um novo processo de optimização?".

Começámos por desenvolver um método para problemas de PLIMO que recorre a técnicas de *planos de corte* na resolução dos programas escalarizantes (de programação inteira monocritério). Estas técnicas mostraram-se potencialmente interessantes para a incorporação de um processo de análise de sensibilidade e a resolução "encadeada" dos sucessivos programas escalarizantes durante uma pesquisa direccional. Assim, desenvolvemos um processo iterativo de análise de sensibilidade e cálculo que é usado em cada nova etapa de uma pesquisa direccional. Apesar de os princípios subjacentes ao método nos parecerem interessantes do ponto de vista teórico, este método revelou-se "frágil" na prática, o que se deve ao facto de usar exclusivamente planos de corte na resolução dos programas escalarizantes.

Para fazer face às dificuldades do primeiro método, construímos um outro que se baseia no uso de técnicas de *branch-and-bound*, e que se aplica tanto a problemas de PLIMO, como a problemas de PLIMMO. Tal como no método anterior, desenvolvemos um processo iterativo de análise de sensibilidade e cálculo que é usado para ajustar o ponto de referência e calcular novas soluções eficientes durante uma pesquisa direccional. A técnica usada nesta abordagem é substancialmente diferente da anterior. Todo o processo actua sobre a árvore de *branch-and-bound* que resolveu o último programa escalarizante e calculou a solução eficiente anterior, retirando-

lhe informação para a análise de sensibilidade e actualizando-a em seguida para obter uma nova solução eficiente. Essa actualização consiste em eliminar ramos antigos da árvore, que se revelem desnecessários, e criar novos ramos.

O método interactivo baseado em *branch-and-bound* é, em nosso entender, a contribuição principal deste trabalho. Para além da condução automática das pesquisas direccionais, o que já acontecia com a abordagem de planos de corte, os resultados computacionais são animadores. A resolução "encadeada" dos programas escalarizantes revelou-se eficaz (em termos de tempo computacional), se comparada com a optimização separada dos programas escalarizantes para os diferentes pontos de referência.

Do ponto de vista da interactividade com o AD, podemos apontar vantagens ao tipo de *pesquisa direccional* proposto, mas também lhe reconhecemos alguns pontos fracos.

Uma das principais vantagens é o facto de o ponto de referência ser ajustado automaticamente, o que liberta o AD da indicação explícita de novos pontos de referência. Assim, se o AD estiver interessado em conhecer soluções não dominadas próximas da actual, não precisará de dar "tiros no escuro" escolhendo pontos de referência que poderiam conduzir à mesma solução ou a soluções muito afastadas. Este tipo de pesquisa é particularmente útil para pesquisas locais, visto que o procedimento avança de uma solução não dominada para outra próxima dessa.

Julgamos, contudo, que pode ser difícil para o AD utilizar esta ferramenta para proceder a uma pesquisa estratégica, principalmente numa fase inicial do processo de decisão, em que é importante conhecer soluções que sirvam de "pontos de ancoragem" para as fases seguintes. Numa fase inicial da pesquisa poderá ser útil, por exemplo, optimizar várias somas pesadas das funções objectivo com pesos dispersos que determinem um conjunto inicial de soluções eficientes (suportadas). Uma outra desvantagem do tipo de pesquisa direccional proposto é a falta de controlo por parte do AD sobre as outras funções objectivo, uma vez que apenas escolhe uma delas para melhorar em cada instante. Apenas no caso biobjectivo o funcionamento é inequívoco, porque quando se melhora um dos objectivos piora-se o outro, pelo que é feito um "varrimento" sequencial de soluções eficientes num dos sentidos. Uma forma possível de conceder um maior controlo, em outros casos, é permitir que o AD introduza limitações adicionais nos valores das funções objectivo. A pesquisa prossegue depois em sub-regiões eficientes, o que corresponde na prática à consideração de outras direcções de pesquisa (diferentes das pré-definidas) na região eficiente original.

Os métodos de pontos de referência referidos atrás, um baseado em técnicas de planos de corte e outro em *branch-and-bound*, são os módulos principais de um sistema computacional de apoio à decisão. Trata-se de um programa integrado de computador que desenvolvemos em

DELPHI para ambiente WINDOWS 95/98. Este sistema inclui outras ferramentas que podem ser usadas em combinação com as anteriores, como a optimização de somas pesadas das funções objectivo e a introdução de limitações adicionais durante as pesquisas direccionais.

Incluímos ainda, neste sistema computacional, uma forma original de apresentação gráfica de resultados de problemas de PLIMO. Trata-se da representação de regiões de pontos de referência que conduzem à mesma solução não dominada, a que chamamos *regiões de indiferença*. Atendendo a que o universo das soluções de um problema de PLIMO é discreto, existem múltiplos pontos de referência que conduzem à mesma solução. São esses pontos que definem cada região de indiferença, que pode ser interpretada como um indicador da estabilidade da solução relativamente à variação do ponto de referência. A implementação desta ferramenta surgiu na sequência do estudo que fizemos sobre a definição de regiões de indiferença associadas aos pontos de referência de problemas PLIMO e que, por questões gráficas, está limitada a problemas com dois ou três objectivos.

O estudo das regiões de indiferença foi motivado pela preocupação, já manifestada anteriormente, de informar o AD de vias de pesquisa desnecessárias porque conduzem a soluções já conhecidas. A intenção é evitar a repetição de cálculos ao longo do processo de decisão, e o consequente desperdício de esforço computacional. Desenvolvemos um processo que actua de cada vez que é calculada uma solução eficiente. Com este processo não definimos toda a região de indiferença correspondente a uma solução, mas apenas sub-regiões dela. Esta informação não é, pois, completa; julgamos, porém, que é útil para o AD na medida em que delimita áreas de pontos de referência cuja escolha pode ser evitada para o conhecimento de novas soluções eficientes.

Numa perspectiva mais teórica, este estudo evidencia ainda algumas características do comportamento das abordagens de pontos de referência em problemas de PLIMO.

Para testar e avaliar melhor as novas metodologias, utilizámos o sistema computacional para analisar um problema multiobjectivo de localização-transporte. O problema não é original, tendo já sido estudado por COUTINHO-RODRIGUES ET AL. (1995). O modelo é baseado no trabalho de CURRENT E RATICK (1995) e considera diferentes critérios para seleccionar locais de abertura de estações de tratamento de materiais tóxicos e estabelecer as rotas de transporte dos materiais (e respectivas quantidades) desde os locais onde são gerados até às estações de tratamento.

Os modelos que integram simultaneamente questões de localização e estabelecimento de rotas constituem um progresso face àqueles que, convencionalmente, tratavam apenas um dos aspectos. Tratando-se de materiais tóxicos, o conhecido síndrome '*nimby*' (*not in my back yard* – "não no meu quintal") não se restringe às decisões de localização das estações de tratamento, como também às decisões dos percursos a utilizar e às quantidades transportadas. Assim, a incorporação de outros critérios, para além do custo, tais como o risco local e a distribuição de

riscos, são factores importantes. Porém, a complexidade computacional agrava-se pela introdução de vários critérios. O problema que abordamos neste trabalho segue um modelo de PLIMMO. Esta aplicação foi escolhida para ser analisada através das metodologias anteriormente descritas porque se enquadra na nossa área de estudo e envolve decisões de grande preocupação hoje em dia.

Nas experiências que fizemos com o problema de localização-transporte, usando o sistema computacional desenvolvido, notámos que o cálculo de cada solução eficiente era, na maior parte das vezes, praticamente instantâneo. Na verdade, este é um factor importante para o desenrolar do processo interactivo de decisão. Devemos, no entanto, referir que o problema em causa é de dimensão média e tem poucas variáveis inteiras. Apesar das vantagens resultantes do aproveitamento da árvore de *branch-and-bound* durante uma *pesquisa direccional*, a complexidade agrava-se em problemas de grandes dimensões ou com muitas variáveis inteiras. Pode mesmo tornar-se impraticável tratar estes problemas através de métodos que usem algoritmos de resolução exacta nas fases de cálculo. Uma alternativa para esses casos será a de recorrer a técnicas heurísticas para calcular soluções aproximadas das soluções eficientes em tempos computacionais razoáveis. Independentemente da heurística usada, a qualidade das aproximações é um factor importante a ter em conta, uma vez que estas técnicas não garantem a optimalidade da solução de um problema monocritério e, consequentemente, a obtenção de soluções eficientes em problemas multicritério.

Neste contexto, começámos por fazer experiências com *meta-heurísticas* em problemas monocritério. São muitas as dificuldades de construir heurísticas que sejam eficazes em problemas genéricos, pelo que nos limitamos aos problemas com variáveis binárias (0-1). Desenvolvemos e testamos duas versões de meta-heurísticas – uma de *simulated annealing* e outra de *tabu search* – para problemas (genéricos) de programação linear inteira 0-1. Estas versões foram depois integradas numa abordagem interactiva para problemas de programação linear inteira multiobjectivo com variáveis binárias (PLIMO 0-1).

Julgamos que esta abordagem é interessante para lidar com problemas de dimensões maiores ou para desenvolver uma pesquisa estratégica inicial, mesmo em problemas que possam ser tratados através de técnicas exactas. Neste último caso, funciona como um módulo de um sistema mais alargado, em que o AD tem a possibilidade de obter um conhecimento global do problema com um esforço computacional aceitável. Consequentemente, o AD pode delimitar zonas de pesquisa que considere mais interessantes, procedendo depois a uma pesquisa mais focada. Foi nesta perspectiva que integrámos, no mesmo sistema computacional, esta abordagem multiobjectivo com meta-heurísticas e os métodos de pontos de referência. O AD dispõe ainda de outras ferramentas de pesquisa "exacta", que podem ser úteis após uma pesquisa com meta-heurísticas. Uma das ferramentas consiste na verificação se uma dada solução (considerada

interessante pelo AD) obtida por uma meta-heurística é, de facto, eficiente. Do ponto de vista experimental, o sistema disponibiliza, desta forma, meios de aferição das aproximações obtidas pelas meta-heurísticas.

As contribuições que acabámos de referir constituem os assuntos centrais dos capítulos que se seguem. Esta dissertação encontra-se organizada da seguinte forma:

No **capítulo II** fazemos uma introdução aos conceitos fundamentais da programação multiobjectivo, dando particular destaque à programação linear inteira e inteira-mista multiobjectivo. Discutimos as diferenças entre as abordagens interactivas e não interactivas e apresentamos uma revisão bibliográfica de métodos interactivos para PLIMO e PLIMMO. Finalmente, apresentamos alguns comentários sobre a investigação existente na área, justificando e enquadrando algumas das opções que tomámos no nosso trabalho.

No **capítulo III** começamos por fazer uma breve introdução e revisão bibliográfica sobre planos de corte e análise de sensibilidade em programação inteira. Descrevemos e exemplificamos, em seguida, o método interactivo baseado em técnicas de planos de corte que desenvolvemos para problemas de PLIMO. Por fim, discutimos alguns resultados computacionais deste método.

No **capítulo IV** começamos por fazer uma breve introdução ao uso da técnica de *branch-and-bound* na resolução de problemas de programação inteira e inteira-mista. Apresentamos depois o método interactivo que desenvolvemos para problemas de PLIMO e PLIMMO baseado em *branch-and-bound*. Segue-se um exemplo ilustrativo e resultados computacionais. Propomos também um algoritmo que combina o método baseado em planos de corte e o de *branch-and-bound*, comentando o seu desempenho em relação ao dos métodos isolados. Ainda neste capítulo, abordamos algumas questões relacionadas com o protocolo de interacção com o AD.

No **capítulo V** apresentamos um estudo sobre a definição de regiões de indiferença associadas aos pontos de referência em problemas de PLIMO, e a sua integração com os métodos anteriores.

No **capítulo VI** analisamos um problema de localização-transporte através da metodologia de PLIMMO proposta no capítulo IV. Neste capítulo pretendemos também ilustrar o funcionamento do sistema computacional que implementámos, especialmente no que se refere ao interface com o utilizador.

No **capítulo VII** propomos uma metodologia interactiva, baseada em meta-heurísticas, para problemas de PLIMO com variáveis binárias (PLIMO 0-1). Começamos por fazer uma introdução às meta-heurísticas *simulated annealing* e *tabu search* e a algoritmos genéricos destas meta-heurísticas para problemas de optimização 0-1. Neste contexto, apresentamos duas novas propostas que designamos por *0-1SA* e *0-1TS*, respectivamente. Abordamos, em seguida, o uso de meta-heurísticas em problemas multiobjectivo, aludindo ao trabalho já existente nesta área.

Apresentamos, por fim, uma abordagem interactiva multiobjectivo que integra as versões *0-1SA* e *0-1TS*.

No **capítulo VIII** procuramos apresentar as principais conclusões deste trabalho e algumas vias de investigação futura.

Capítulo II

Programação linear inteira e inteira-mista multiobjectivo

– Fundamentos e métodos interactivos –

Neste capítulo procuramos fazer uma introdução à programação linear inteira e inteira-mista multiobjectivo. Omitimos as noções básicas da programação linear e da programação inteira (mista) monocritério, remetendo para algumas obras de referência nestas áreas: MURTY (1983) para a programação linear e NEMHAUSER E WOLSEY (1988) para a programação inteira. Apontamos ainda a obra de STEUER (1986) como importante referência da optimização multicritério, principalmente no que diz respeito à programação linear multiobjectivo.

Este capítulo está organizado da seguinte forma:

Na secção 1 apresentamos algumas noções básicas e fundamentos teóricos da programação matemática multiobjectivo, começando com os conceitos gerais e particularizando, em seguida, para os casos inteiro e inteiro-misto.

Na secção 2 caracterizamos e distinguimos as abordagens interactivas e as não interactivas.

Na secção 3 fazemos uma revisão bibliográfica de métodos interactivos dedicados a problemas de programação linear inteira e inteira-mista multiobjectivo. Terminamos o capítulo com alguns comentários críticos aos métodos existentes, enquadrando e justificando algumas das opções que tomámos no nosso trabalho.

II.1 FUNDAMENTOS

II.1.1 O problema com objectivos múltiplos

Consideremos o problema de programação matemática multiobjectivo com n variáveis de decisão $x=(x_1, x_2,..., x_n)$ e k funções objectivo (também designadas habitualmente por critérios). Assumimos, sem perda de generalidade, que as funções objectivo são a maximizar, sujeitas a restrições no espaço das variáveis de decisão expressas sob a forma de $x \in X$. Admitimos que o conjunto das soluções admissíveis, X, é fechado, limitado e não vazio. Matematicamente, o problema pode ser estabelecido da seguinte forma:

$$\begin{aligned} &\max \quad z_1 = f_1(x) \qquad\qquad (PMO)\\ &... \\ &\max \quad z_k = f_k(x) \\ &\text{s.a:} \quad x \in X \end{aligned}$$

Seja $Z \subset \Re^k$ o conjunto de todas as soluções admissíveis no espaço dos objectivos, i.e. $Z=\{(z_1,...,z_k) | z_i = f_i(x),\ i=1,...,k,\ x \in X\}$.

Num problema com múltiplos objectivos não existe, em geral, uma solução admissível que optimiza simultaneamente todas as funções objectivo, pelo que a noção de solução *óptima* dá lugar à noção de solução *eficiente* (também designada por não dominada, não inferior ou óptima de Pareto). Uma solução eficiente caracteriza-se por não existir uma outra solução admissível que melhore o valor de uma função objectivo sem piorar o valor de pelo menos uma outra.

Definição II.1 *Solução eficiente*

Uma solução $\bar{x} \in X$ diz-se eficiente sse não existe outra solução $x \in X$ tal que $f_i(x) \geq f_i(\bar{x})$ para todo o $i=1,...,k$ e $f_j(x) > f_j(\bar{x})$ para pelo menos um j.

Adoptando a terminologia de STEUER (1986), a designação de solução eficiente refere-se a soluções no espaço de decisão e a de *não dominada* a pontos do espaço dos objectivos.

Definição II.2 *Solução não dominada*

Uma solução $\bar{z} \in Z$ diz-se não dominada sse $\bar{z}_i = f_i(\bar{x})$, $i=1,...,k$ para $\bar{x} \in X$ eficiente.

De forma equivalente, uma solução admissível é não dominada quando não existe uma outra solução admissível que a *domine*, pressupondo a seguinte definição de *dominância*.

Definição II.3 *Dominância*

$\bar{z} \in \Re^k$ *domina* $z \in \Re^k$ sse $\bar{z}_i \geq z_i$ para todo o $i = 1,...,k$ e $\bar{z}_j > z_j$ para pelo menos um j.

Seja X_{ef} o *conjunto eficiente*, i.e. o subconjunto de X das soluções eficientes e Z_{nd} o *conjunto não dominado*, subconjunto de Z das soluções não dominadas.

Definição II.4 *Solução fracamente eficiente*

Uma solução $\bar{x} \in X$ diz-se fracamente eficiente se não existir uma outra solução $x \in X$ tal que $f_i(x) > f_i(\bar{x})$ para todo o $i = 1,...,k$.

Um conceito mais restrito de eficiência é o de solução *eficiente própria*.

Definição II.5 *Solução eficiente própria* (GEOFFRION, 1968)

Uma solução $\bar{x} \in X$ é eficiente própria se ela é eficiente e existe um número finito M>0 tal que, para cada função $f_i(x)$, $i = 1,...,k$, e cada $x \in X$ com $f_i(x) > f_i(\bar{x})$, se verifica $\dfrac{f_i(x) - f_i(\bar{x})}{f_j(\bar{x}) - f_j(x)} \leq M$ para algum j em que $f_j(x) < f_j(\bar{x})$.

As noções de solução *fracamente não dominada* e de solução *não dominada própria* decorrem directamente por analogia com a relação "solução eficiente – solução não dominada".

Suponhamos que $\bar{z} \in Z$ é não dominada própria. Pela definição II.5, para cada elemento $z \in Z$ e para cada índice i tal que $z_i > \bar{z}_i$, existe um índice j tal que $z_j < \bar{z}_j$ e $z_i - \bar{z}_i \leq M(\bar{z}_j - z_j)$ para algum M>0.

Uma solução eficiente/não dominada que não seja eficiente/não dominada própria chama-se *eficiente/não dominada imprópria*.

No caso de Z ser poliédrico (Z definido por um número finito de inequações lineares), o conjunto das soluções eficientes coincide com o conjunto das soluções eficientes próprias (ISERMANN, 1974). O mesmo se passa para Z discreto e finito, o que é uma consequência imediata da definição II.5. Contudo, em certos tipos de problemas multiobjectivo discretos com um número infinito de soluções e em problemas multiobjectivo não lineares podem ocorrer soluções eficientes impróprias (STEUER, 1986). O exemplo II.1 ilustra a situação de uma solução eficiente (não dominada) imprópria num problema não linear.

Exemplo II.1. Na figura II.1, Z_{nd} é a fronteira a traço mais grosso, sendo z^1 uma solução não dominada imprópria.

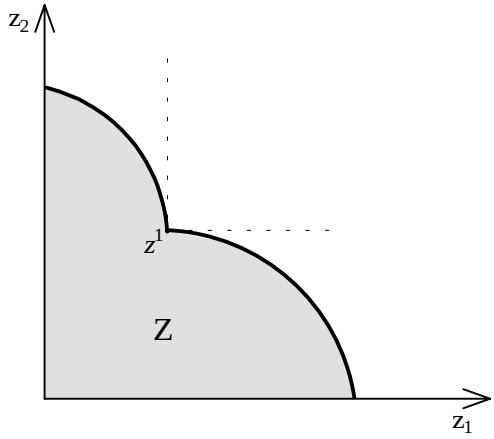


Fig. II.1 – Uma solução não dominada imprópria (In STEUER, 1986).

Uma outra noção importante é a distinção entre soluções eficientes *suportadas* e *não suportadas*. Em problemas de programação linear multiobjectivo todas as soluções eficientes são *suportadas*, mas em problemas com Z não convexo (em particular, problemas de programação linear inteira ou inteira-mista multiobjectivo) podem ocorrer soluções não dominadas que sejam *não suportadas.*

Podemos definir solução não dominada não suportada da seguinte forma:

Definição II.6 *Solução não dominada não suportada*

Uma solução $\bar{z} \in Z_{nd}$ é não suportada se ela for dominada por uma combinação convexa (não admissível) de soluções pertencentes a Z_{nd}.

Uma solução $\bar{z}$ não dominada não suportada corresponde a uma solução $\bar{x} \in X$ *eficiente não suportada*. As outras soluções não dominadas (eficientes) dizem-se *suportadas*. O exemplo II.2 ilustra a situação de existência de soluções não dominadas não suportadas num problema de programação inteira.

Exemplo II.2. Num problema em que as soluções não dominadas são as representadas na figura II.2, z^1, z^2 e z^4 são soluções não dominadas *suportadas* enquanto que z^3 é uma solução não dominada *não suportada*. Reparemos que z^3 é dominada por algumas combinações convexas de z^2 e z^4.

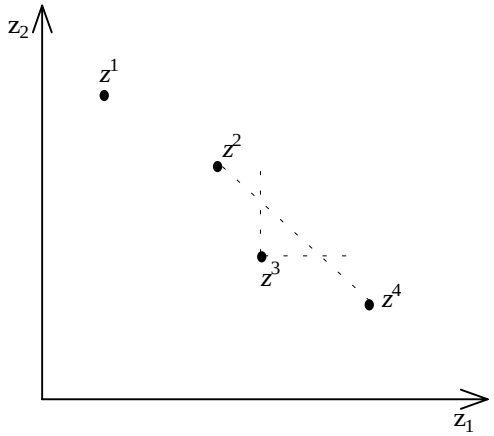


Fig. II.2 – Uma solução não dominada não suportada num problema com soluções discretas.

A existência de soluções não dominadas não suportadas pode também ser ilustrada com um problema de programação inteira-mista – exemplo II.3 – ou um problema não linear – exemplo II.4.

Exemplo II.3. A figura II.3 representa um problema de programação linear inteira-mista multiobjectivo onde, do conjunto das soluções não dominadas assinalado a traço mais grosso, as soluções z^1, z^2 e z^4 e as soluções do segmento de recta que une z^1 a z^2 são *suportadas*, enquanto que as soluções do segmento de recta que une z^3 a z^4 (excluindo os vértices) são *não suportadas*, visto que são dominadas por combinações convexas (não admissíveis) de z^2 e z^4; a solução z^3 é fracamente não dominada porque não existe nenhuma outra solução que seja estritamente melhor do que esta nas duas funções objectivo (condição de ser fracamente não dominada) sendo dominada (no sentido da definição II.3) por z^2.

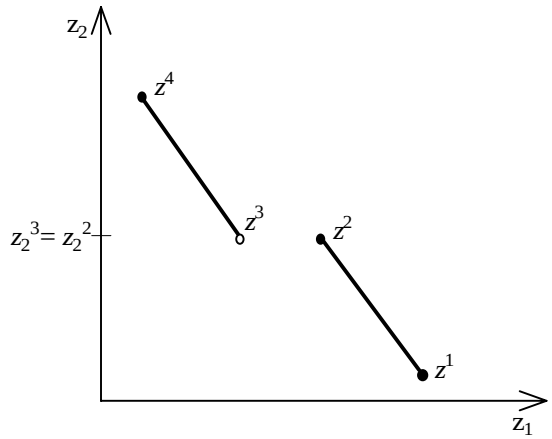


Fig. II.3 – Soluções não dominadas não suportadas num problema de programação inteira-mista.

Exemplo II.4. A figura II.4 representa um problema não linear; Z_{nd} é a fronteira desde z^2 a z^4; as soluções desde z^1 até z^2, exclusive, são fracamente não dominadas; as soluções desde z^2, exclusive, a z^3, exclusive, são não dominadas *não suportadas*; as soluções desde z^3 a z^4 são não dominadas *suportadas*.

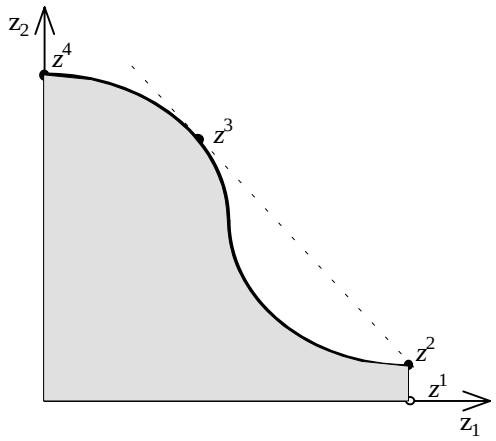


Fig. II.4 – Soluções não dominadas não suportadas num problema de programação não linear.

No que se segue, utilizaremos indiscriminadamente os termos *eficiente* e *não dominada* sempre que nos estejamos a referir a uma solução do problema multiobjectivo sem particularizar se o elemento é do espaço de decisão ou do espaço dos objectivos.

II.1.2 Caracterização do conjunto eficiente

Definidos os conceitos de solução eficiente, fracamente eficiente, eficiente própria e imprópria, e eficiente suportada e não suportada, interessa agora conhecer algumas das formas mais comuns de caracterizar o conjunto das soluções eficientes e que se traduzem em processos de cálculo destas soluções. Para tal, podemos definir programas *escalarizantes* em que as funções objectivo são agregadas temporariamente numa única função escalar. As funções escalarizantes dependem tipicamente não só dos valores das funções objectivo como também de parâmetros adicionais.

É bem conhecida a consideração de somas pesadas das funções objectivo e a sua utilização em programação linear multiobjectivo com variáveis contínuas (PLMO). Contudo, desse programa escalarizante não resulta um meio de caracterização completa do conjunto eficiente para todo o tipo de problemas. Consideremos o problema escalarizante P_w^1 de optimização de uma soma pesada das funções objectivo:

$$\max \quad \sum_{i=1}^{k} w_i f_i(x) \qquad (P_w^1)$$

$$\text{s.a:} \quad x \in X$$

$$\text{com} \quad w_i > 0,\ i=1,...,k$$

Da resolução de P_w^1 (com pesos $w_i > 0$) resulta sempre uma solução eficiente do problema multiobjectivo. Esta condição suficiente de eficiência é também condição necessária em PLMO, mas não o é em problemas com regiões admissíveis não convexas. Para X convexo, GEOFFRION (1968) provou a seguinte proposição:

Proposição II.1 (GEOFFRION, 1968): Seja X um conjunto convexo e $f_i(x)$, $i=1,...,k$ funções côncavas em X; $x^o \in X$ é eficiente própria sse x^o optimiza P_w^1 para algum vector w com componentes estritamente positivas.

Posteriormente, outros autores estabeleceram proposições mais especializadas do que esta. Um conjunto de referências destes trabalhos é fornecido em GAL E WOLF (1986).

Considera-se habitualmente em P_w^1 pesos normalizados, ou seja, $w \in W$ em que $W = \left\{ w \in \Re^k \;\middle|\; w_i > 0, \sum_{i=1}^{k} w_i = 1 \right\}$.

A abordagem das *somas pesadas* – assim a designaremos daqui em diante – não permite, porém, determinar soluções eficientes não suportadas. Relembramos que estas soluções não ocorrem em PLMO mas podem ocorrer em programas lineares com variáveis inteiras ou programas não lineares. Observemos, por exemplo, a figura II.2 onde não existe nenhum vector $w>0$ que conduza a z^3 através da optimização de P_w^1. O mesmo se passa com as soluções não dominadas não suportadas das figuras II.3 e II.4.

Independentemente do tipo de problema, se considerarmos em P_w^1 algum peso w_i nulo, deixamos de ter a garantia de obter uma solução eficiente. Esta situação pode ser observada na figura II.4 em que, se considerarmos $w_2=0$ e $w_1=1$, somos conduzidos ao segmento que une z^1 a z^2 onde apenas z^2 é não dominada. As outras soluções são fracamente não dominadas.

A abordagem das somas pesadas também não permite alcançar as soluções não dominadas impróprias, mesmo que suportadas, a não ser que se atribua um peso nulo a alguma função objectivo, com o eventual problema da falta de eficiência mencionado atrás – ver exemplo II.5.

> **Exemplo II.5.** A figura II.5 ilustra uma situação em que há uma solução não dominada imprópria (suportada), z^2, que só é possível alcançar pela abordagem das somas pesadas considerando peso nulo para $f_1(x)$.

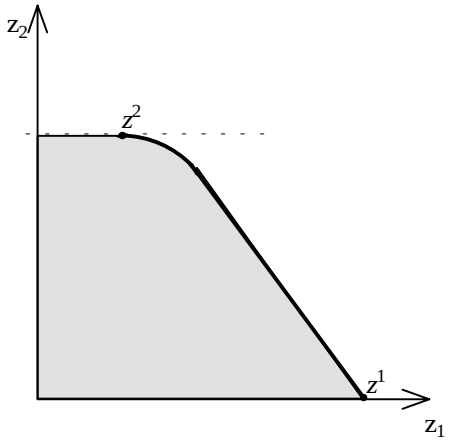


Fig. II.5 – A necessidade de um peso nulo para se alcançar uma solução não dominada imprópria.

Existem outros programas escalarizantes que permitem calcular soluções eficientes suportadas e não suportadas. Essas formas de caracterizar o conjunto das soluções eficientes baseiam-se na consideração de níveis para as funções objectivo, sejam eles níveis de *reserva* (valores que têm que ser necessariamente alcançados ou ultrapassados), níveis de *aspiração* (valores que se deseja alcançar mas que não são necessariamente alcançáveis) ou níveis de *referência*, que podem em geral ser interpretados como níveis de aspiração.

Pela consideração de níveis de *reserva* entende-se a imposição de limitações inferiores nos valores das funções objectivo. O resultado seguinte, apresentado por SOLAND (1979), enquadra de forma genérica a caracterização do conjunto eficiente com base em níveis de reserva:

Proposição II.2 (SOLAND, 1979): Seja g uma função real arbitrária definida em $\Re^k$ que é estritamente crescente em Z. Então $\bar{x}$ é uma solução eficiente sse for óptima do problema P_b^g para algum $b \in \Re^k$. P_b^g é definido da seguinte forma:

$$\begin{aligned} \max \quad & g\big(f_1(x),\ldots, f_k(x)\big) && (P_b^g) \\ \text{s.a:} \quad & f_i(x) \geq b_i && i=1,\ldots,k \\ & x \in X \end{aligned}$$

Uma particularização deste resultado é a abordagem das somas pesadas com restrições adicionais nos valores das funções objectivo:

$$\begin{aligned} \max \quad & \sum_{i=1}^{k} w_i f_i(x) && (P_{w,b}^1) \\ \text{s.a:} \quad & f_i(x) \geq b_i && i=1,\ldots,k \\ & x \in X \end{aligned}$$

em que $w \in W$.

A parametrização de $P^1_{w,b}$ em *b* (ou em *b* e *w*) permite determinar qualquer solução eficiente.

Muitas das caracterizações que usam níveis de *aspiração* ou de *referência* baseiam-se na métrica de Tchebycheff cujos desenvolvimentos iniciais se devem a BOWMAN (1976).

Comecemos por introduzir a métrica – simples ou pesada – de Tchebycheff (ou métrica L_∞). A distância entre dois pontos y^1 e y^2 segundo a métrica de Tchebycheff é $\left\|y^1 - y^2\right\|^\infty = \max_i \left|y_i^1 - y_i^2\right|$. No caso da métrica pesada, se considerarmos o vector de pesos $\lambda \geq 0$, temos que $\left\|y^1 - y^2\right\|^\infty_\lambda = \max_i \lambda_i \left|y_i^1 - y_i^2\right|$. O exemplo II.6 ilustra estes conceitos.

Exemplo II.6. A figura II.6 mostra os contornos dos pontos que se encontram à distância de 1 de $y \in \Re^2$ segundo a métrica pesada de Tchebycheff, considerando os seguintes pesos: $\lambda^1=(1,1)$ – equivalente à métrica simples; $\lambda^2=(1/3,2/3)$ e $\lambda^3=(3/4,1/4)$.

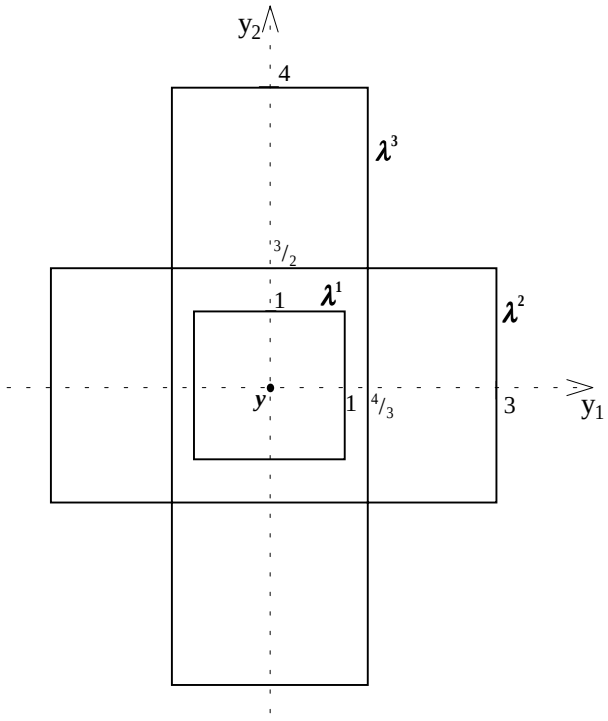


Fig. II.6 – Ilustração de contornos de isodistância da métrica pesada de Tchebycheff.

BOWMAN (1976) mostrou que a optimização paramétrica (em λ) da métrica pesada de Tchebycheff, aplicada às funções objectivo, gera todas as soluções eficientes.

Consideremos um ponto do espaço dos objectivos z^+, designado por *ponto de referência*, que satisfaz $z^+ \geq z$ para todo o $z \in Z$. Seja o seguinte problema escalarizante parametrizado em $\lambda \geq 0$:

$$\min \quad \max_{i=1,\ldots,k} \lambda_i \left|z_i^+ - f_i(x)\right| \qquad (P_\lambda^\infty)$$

$$\text{s.a:} \quad x \in X$$

Proposição II.3 (BOWMAN, 1976): Se $\bar{x} \in X$ é uma solução eficiente, então existe algum $\lambda = \bar{\lambda} \geq 0$ tal que $\bar{x}$ optimiza P_λ^∞.

A proposição II.3 indica que qualquer solução eficiente pode ser calculada através de uma parametrização adequada de P_λ^∞. Contudo, a optimalidade de P_λ^∞ não é condição suficiente de eficiência. A resolução de P_λ^∞ pode conduzir a soluções fracamente eficientes, como podemos ver no exemplo II.7.

Exemplo II.7. Na figura II.7 Z_{nd} é o segmento desde z^1 a z^2 incluindo os dois vértices. As soluções do segmento $]z^1, z^3]$ são fracamente não dominadas. Contudo, todas as soluções desde z^1 a z^4, em que apenas z^1 é não dominada, minimizam P_λ^∞ para o vector de pesos particular $\bar{\lambda}$ tal que $\alpha = \text{arctg}\left(\bar{\lambda}_1 / \bar{\lambda}_2\right)$. Ou – um outro exemplo – todas as soluções desde z^1 a z^3 minimizam P_λ^∞ para o vector de pesos $\lambda=(1,0)$.

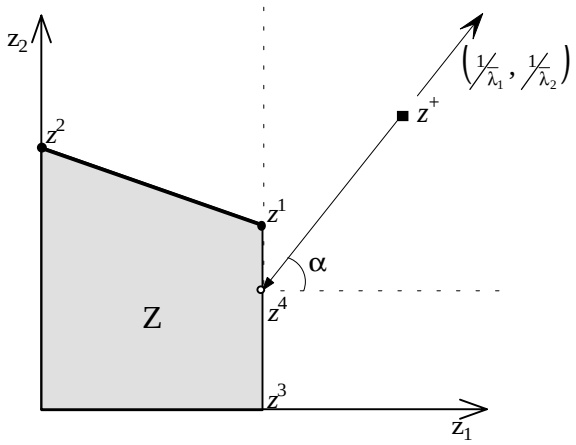


Fig. II.7 – P_λ^∞ pode conduzir a soluções fracamente não dominadas.

Com $z^+ \geq z$ para todo o $z \in Z$, os módulos de $\left|z_i^+ - f_i(x)\right|$ em P_λ^∞ podem ser retirados. O problema escalarizante P_λ^∞, que designaremos por *programa da métrica pesada de Tchebycheff*, pode ser escrito da seguinte forma equivalente:

$$\begin{aligned} \min \quad & \alpha && (P_\lambda^\infty) \\ \text{s.a:} \quad & \lambda_i\left(z_i^+ - f_i(x)\right) \leq \alpha \qquad i=1,\ldots,k \\ & x \in X \\ & \alpha \geq 0 \end{aligned}$$

O cálculo de soluções fracamente eficientes pode ser evitado substituindo-se a métrica pesada de Tchebycheff pela métrica pesada e *aumentada* de Tchebycheff (STEUER E CHOO, 1983). Esta é

definida por $\left\| z^+ - z \right\|_\lambda^\infty = \left\| z^+ - z \right\|_\lambda^\infty + \rho \sum_{i=1}^{k} (z_i^+ - z_i)$ em que ρ é uma quantidade positiva bastante pequena (mas computacionalmente significativa) – para ilustração, ver o exemplo II.8.

Exemplo II.8. A figura II.8 mostra os contornos da métrica pesada e aumentada de Tchebycheff para um dado $\lambda = (\lambda_1, \lambda_2) > (0,0)$, $\lambda_1 + \lambda_2 = 1$; $\theta_1 = \text{arctg}\left(\frac{\rho}{1-\lambda_1+\rho}\right)$ e $\theta_2 = \text{arctg}\left(\frac{\rho}{1-\lambda_2+\rho}\right)$.

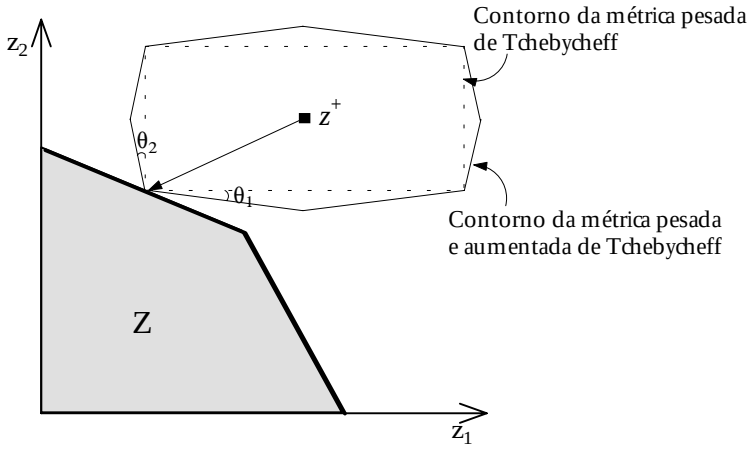


Fig. II.8 – Métrica pesada e aumentada de Tchebycheff.

Podemos então definir o problema escalarizante seguinte que designaremos por *programa da métrica pesada e aumentada de Tchebycheff*:

$$\begin{aligned} \min \quad & \alpha + \rho \sum_{i=1}^{k} \left(z_i^+ - f_i(x) \right) && (P_\lambda^{\infty,\rho}) \\ \text{s.a:} \quad & \lambda_i \left(z_i^+ - f_i(x) \right) \le \alpha \qquad i=1,...,k \\ & x \in X \\ & \alpha \ge 0 \\ & \text{com } \lambda_i \ge 0 \text{ para todo o } i = 1,...,k \text{ e } z^+ \ge z \text{ para todo o } z \in Z. \end{aligned}$$

Nota: A função objectivo de $P_\lambda^{\infty,\rho}$ pode ser substituída por $\alpha - \rho \sum_{i=1}^{k} f_i(x)$ visto que o termo restante é constante.

A optimização de $P_\lambda^{\infty,\rho}$ garante o cálculo de apenas soluções eficientes. Habitualmente usam-se pesos normalizados, i.e. $\lambda \in \Lambda^0 = \left\{ \lambda \in \Re^k \,\middle|\, \lambda_i \ge 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$.

Um ponto de referência frequentemente usado é a *solução ideal*, ponto do espaço dos objectivos, em geral não admissível, que optimiza simultaneamente todas as funções objectivo.

Definição II.7 *Solução ideal*

A solução ou ponto ideal $z^* \in \Re^k$ tem componentes z_i^*, $i=1,...,k$, definidas por $z_i^* = \max\{f_i(x) | x \in X\}$, $i=1,...,k$.

No âmbito da exposição da teoria de Tchebycheff, STEUER E CHOO (1983) e STEUER(1986) usam uma extensão da solução ideal que aqui designamos por *solução ideal estendida*.

Definição II.8 (STEUER E CHOO, 1983; STEUER, 1986) *Solução ideal estendida*

A solução ideal estendida $z^{**}=(z_1^{**},...,z_k^{**}) \in \Re^k$ é definida por $z_i^{**} = \max\{f_i(x) | x \in X\} + \varepsilon_i$, $i=1,...,k$, com constantes $\varepsilon_i \geq 0$; uma constante ε_i deve ser estritamente positiva pelo menos quando há mais do que uma solução não dominada que maximiza o objectivo i ou a única solução não dominada que maximiza o objectivo i também maximiza algum dos outros objectivos.

Considerando $\lambda \in \Lambda^0 = \left\{ \lambda \in \Re^k \;\middle|\; \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$ e a solução ideal estendida z^{**} como ponto de referência, STEUER E CHOO (1983) e STEUER (1986) provaram que existe sempre um ρ positivo que garante que qualquer solução não dominada de um problema com região admissível *discreta* ou *poliédrica* possa ser obtida a partir de $P_\lambda^{\infty,\rho}$. Assim, para ρ positivo suficientemente pequeno, além da optimização de $P_\lambda^{\infty,\rho}$ ($\lambda \in \Lambda^0$) ser condição suficiente de eficiência – o que é válido para o caso geral – é também condição necessária para os casos mencionados, ou seja, existe sempre um $\bar{\lambda} \in \Lambda^0$ tal que $\bar{x} \in X_{ef}$ optimiza $P_\lambda^{\infty,\rho}$ para $\lambda = \bar{\lambda}$ e $z^+ = z^{**}$.

Nota: Apesar de STEUER (1986) não indicar explicitamente a razão da necessidade de se usar a solução ideal estendida no lugar da solução ideal, podemos de facto constatar o seu interesse em casos particulares dos referidos na definição II.8. A título de curiosidade, apresentamos aqui um exemplo (da nossa autoria) de um problema de PLMO nessas circunstâncias – exemplo II.9.

Abordaremos na próxima secção, com maior detalhe, alguns resultados da teoria de Tchebycheff desenvolvida por STEUER E CHOO (1983) e STEUER (1986) para o caso discreto.

Exemplo II.9. Consideremos o seguinte problema de PLMO cujo espaço dos objectivos está representado na figura II.9 e a região não dominada (Z_{nd}) assinalada a cinzento.

$$\begin{aligned} \max \quad & z_1 = x_1 \\ \max \quad & z_2 = x_2 \\ \max \quad & z_3 = x_3 \\ \text{s.a:} \quad & x_1 + 3x_2 + 3x_3 \le 12 \\ & x_2 \le 3 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

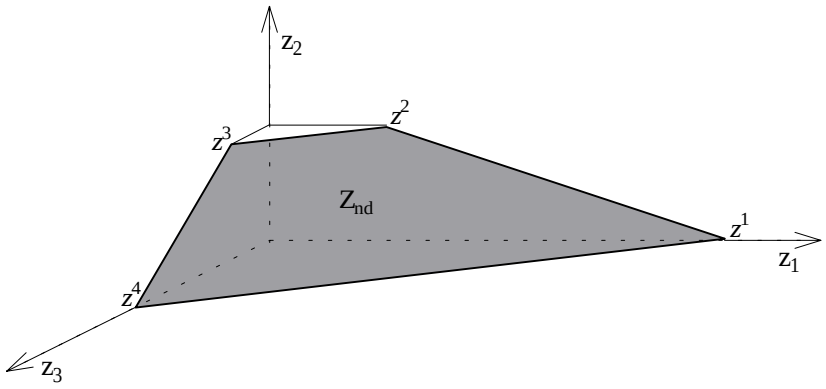


Fig. II.9 – Espaço dos objectivos do exemplo II.9.

$z^1 = (12,0,0)$; $z^2 = (3,3,0)$; $z^3 = (0,3,1)$; $z^4 = (0,0,4)$; $z^* = (12,3,4)$.

Como há mais do que uma solução não dominada que optimiza z_2, na definição de z^{**} o valor de ε_2 deve ser estritamente positivo enquanto que ε_1 e ε_3 podem ser nulos. Seja, por exemplo, $z^{**}=(12, 3.1, 4)$.
Vejamos então o inconveniente de usar z^*. Considerando z^* na métrica pesada, ou pesada e aumentada, de Tchebycheff, a determinação das soluções que optimizam z_2 é conseguida unicamente pela consideração de pesos nulos para z_1 e z_3: $\lambda_1=0$, $\lambda_2=1$, $\lambda_3=0$. No caso da métrica não aumentada, há soluções não dominadas e fracamente não dominadas que optimizam o respectivo problema escalarizante. Para a métrica aumentada, as soluções fracamente não dominadas são eliminadas e temos que:

$$\left\| z^* - z^2 \right\|_{\lambda=(0,1,0)}^{\infty} = 0+13\rho \qquad \text{e} \qquad \left\| z^* - z^3 \right\|_{\lambda=(0,1,0)}^{\infty} = 0 + 15\rho$$

Significa que, qualquer que seja $\rho>0$, é z^2 a solução não dominada que minimiza $P_{\lambda}^{\infty,\rho}$ para $\lambda=(0,1,0)$ e $z^+=z^*$. Consequentemente, não se consegue alcançar z^3 e as outras soluções não dominadas da aresta que une z^2 a z^3.
Contudo, se considerarmos $z^{**}=(12, 3.1, 4)$, existe um vector de pesos particular para cada solução não dominada tal que essa solução minimiza a distância a z^{**} segundo a métrica pesada de Tchebycheff. Em particular, o vector de pesos para z^3 é definido pela direcção diagonal de $\left\| z^{**} - z^3 \right\|_{\lambda}^{\infty}$, isto é, o vector normalizado de $\left(\frac{1}{12}, \frac{1}{0.1}, \frac{1}{3}\right)$ que é $\bar{\lambda}$ =(8x10^{-3}, 0.96, 0.032). Reparemos agora que

$$\left\| z^{**} - z^2 \right\|_{\lambda=\bar{\lambda}}^{\infty} = 0.128 + 13.1\rho \qquad \text{e} \qquad \left\| z^{**} - z^3 \right\|_{\lambda=\bar{\lambda}}^{\infty} = 0.096 + 15.1\rho$$

Logo, para ρ suficientemente pequeno, a solução não dominada z^3 é alcançável pela métrica pesada e aumentada de Tchebycheff.

A métrica pesada e aumentada e Tchebycheff não permite caracterizar por completo o conjunto das soluções eficientes em problemas com soluções eficientes impróprias. STEUER E CHOO (1983) e STEUER (1986) propõem uma abordagem *lexicográfica* com o programa da métrica pesada de Tchebycheff para tratar os casos em que a região admissível é não linear ou um conjunto discreto infinito. A abordagem lexicográfica pode também ser aplicada ao caso discreto finito ou poliédrico, mas tem a desvantagem de ser mais difícil de implementar pois são necessárias duas fases de optimização. Na 1ª fase, apenas se minimiza a variável α, ou seja, resolve-se o programa da métrica pesada (não aumentada) de Tchebycheff, P_{λ}^{∞}. De entre o conjunto das soluções óptimas de P_{λ}^{∞} existe sempre pelo menos uma que é eficiente, logo só é necessário passar à 2ª fase quando resultam óptimos alternativos da 1ª fase. A 2ª fase tem como intenção eliminar soluções fracamente eficientes obtidas na 1ª fase e consiste na minimização de $\sum_{i=1}^{k}\left(z_i^{+} - f_i(x)\right)$ ou, de forma equivalente, na maximização de $\sum_{i=1}^{k} f_i(x)$, dentro do conjunto das soluções que minimizam α.

Além da métrica pesada de Tchebycheff, aumentada ou na forma lexicográfica, com parametrização nos pesos, há outros programas escalarizantes do tipo *min-max* baseados em *pontos de referência* que permitem caracterizar o conjunto das soluções eficientes. Se os níveis de referência que compõem o ponto de referência forem usados como parâmetros de controlo, então a métrica de Tchebycheff altera a sua forma de dependência dos parâmetros e deve ser interpretada como uma *função escalarizante de realização* ('achievement scalarizing function' segundo LEWANDOWSKI E WIERZBICKI, 1988).

A minimização em X da *função escalarizante de realização* $\max_{i=1,\ldots,k} \lambda_i\left(q_i - f_i(x)\right)$ ou $\max_{i=1,\ldots,k} \lambda_i\left(q_i - f_i(x)\right) - \rho\sum_{i=1}^{k} f_i(x)$, $\lambda \geq 0$, produz soluções pelo menos fracamente eficientes, no primeiro caso, e soluções (estritamente) eficientes no segundo caso, qualquer que seja o ponto de referência $q \in \Re^k$. A minimização destas funções pode não representar a minimização de uma distância ao ponto de referência, já que q pode atravessar a fronteira não dominada e tornar-se atingível. Estas funções inserem-se numa classe a que LEWANDOWSKI E WIERZBICKI (1988) chamam de funções consistentes na ordem ('order-consistent achievement functions'). Uma função consistente na ordem mantém-se monótona mesmo quando o ponto de referência é admissível.

Os programas escalarizantes correspondentes às funções anteriores são os representados por $P_{q,\lambda}$ e $P_{q,\lambda}^{\rho}$, respectivamente:

$$\begin{aligned} \min \quad & \alpha && (P_{q,\lambda}) \\ \text{s.a:} \quad & \lambda_i\left(q_i - f_i(x)\right) \le \alpha \ , \ i=1,...,k \\ & x \in X \end{aligned}$$

$$\begin{aligned} \min \quad & \alpha - \rho\sum_{i=1}^{k} f_i(x) && (P^{\rho}_{q,\lambda}) \\ \text{s.a:} \quad & \lambda_i\left(q_i - f_i(x)\right) \le \alpha \ , \ i=1,...,k \\ & x \in X \end{aligned}$$

$P_{q,\lambda}$ ou $P^{\rho}_{q,\lambda}$ pode ser parametrizado em q com variação ou não de λ. No caso de λ ter componentes constantes, estas podem desempenhar o papel de factores de escala para as funções objectivo.

Notemos que a única diferença entre $P_{q,\lambda}$ e P^{∞}_{λ} (ou entre $P^{\rho}_{q,\lambda}$ e $P^{\infty,\rho}_{\lambda}$) está na variável α, que anteriormente era não negativa e agora é uma variável livre. Isso deve-se ao facto de o ponto de referência q poder ser um ponto qualquer de $\Re^k$, que não satisfaz necessariamente a condição $q \ge z$, $\forall z \in Z$. Citando WIERZBICKI (1998), o tipo de minimização de $P_{q,\lambda}$ e $P^{\rho}_{q,\lambda}$ não significa "chegar perto" no sentido tradicional mas "chegar perto ou ultrapassar".

Designaremos genericamente estes programas por *min-max*, que incluem naturalmente o programa da métrica pesada (simples ou aumentada) de Tchebycheff

II.1.3 O caso linear inteiro puro multiobjectivo (PLIMO)

Consideremos o modelo de programação linear inteira pura multiobjectivo (PLIMO):

$$\begin{aligned} \max \quad & z_1 = f_1(x) = c^1 x && \text{(PLIMO)} \\ & \ldots \\ \max \quad & z_k = f_k(x) = c^k x \\ \text{s.a:} \quad & x \in X \\ & X = \left\{ x \in \Re^n \mid Ax = b,\ x \ge 0,\ x \text{ inteiro} \right\} \end{aligned}$$

onde c^i, $i=1,...,k$, são os vectores $1\times n$ dos coeficientes das funções objectivo, A é a matriz $m\times n$ dos coeficientes das restrições e b é o vector $m\times 1$ dos termos independentes das restrições. Sem perda de generalidade, admitimos que A tem característica m e que as restrições foram previamente convertidas em igualdades. Todas as variáveis x_j, $j=1,...,n$ são inteiras; consideramos

ainda que os vectores c^i têm componentes inteiras o que implica que também $z_1,...,z_k$ assumem apenas valores inteiros. Seja X um conjunto limitado.

Designemos por C a matriz $k\times n$ cuja i-ésima linha é o vector c^i. Assim, cada ponto do espaço dos objectivos é dado por $z = Cx = (c^1x, c^2x,...,c^kx)$. Assumimos as definições anteriores de Z, X_{ef}, Z_{nd}, z^* e z^{**}.

Os problemas PLIMO não admitem soluções eficientes impróprias mas têm, em geral, soluções eficientes não suportadas devido à não convexidade de X. Consequentemente, a abordagem das *somas pesadas*, traduzida no programa paramétrico P_w^1, não permite caracterizar por completo o conjunto das soluções eficientes. No entanto, a introdução de restrições adicionais em P_w^1 já possibilita que sejam também alcançadas soluções eficientes não suportadas. Referimo-nos, portanto, ao problema $P_{w,b}^1$.

As abordagens baseadas em pontos de referência mencionadas atrás são adequadas para tratar este tipo de problemas, uma vez que permitem calcular qualquer solução eficiente. Considerando o *programa escalarizante da métrica pesada e aumentada de Tchebycheff*, $P_\lambda^{\infty,\rho}$ com $\lambda\in\Lambda^0$ e $z^+=z^{**}$ (solução ideal estendida), STEUER E CHOO (1983) e STEUER (1986) apresentam vários resultados teóricos para o problema PLIMO, que culminam na seguinte proposição.

Proposição II.4 (STEUER E CHOO, 1983): Seja ρ em $P_\lambda^{\infty,\rho}$ dado por

$$0 < \rho < \min_{z^p\in Z_{nd}}\left[\min_{z^h\in Z\setminus\{z^p\}}\left\{\frac{\alpha^{ph}-\alpha^{pp}}{\sum_{i=1}^{k}\left(z_i^h - z_i^p\right)}\middle|\sum_{i=1}^{k}\left(z_i^h - z_i^p\right)>0\right\}\right] \text{ em que}$$

$$\alpha^{pp} = \max_{i=1,...,k}\lambda_i^p\left(z_i^{**} - z_i^p\right),\ \alpha^{ph} = \max_{i=1,...,k}\lambda_i^p\left(z_i^{**} - z_i^h\right) \text{ e}$$

$$\lambda_i^p = \begin{cases} \dfrac{1}{\left(z_i^{**} - z_i^p\right)}\left[\displaystyle\sum_{i=1}^{k}\frac{1}{\left(z_i^{**} - z_i^p\right)}\right]^{-1} & \text{se } z_i^p \neq z_i^{**} \text{ para todo o } i \\ 1 & \text{se } z_i^p = z_i^{**} \\ 0 & \text{se } z_i^p \neq z_i^{**} \text{ mas } \exists j : z_j^p = z_j^{**} \end{cases}$$

$\bar{x}$ é uma solução eficiente do problema PLIMO sse existe $\lambda\in\Lambda^0$ tal que $\bar{x}$ é uma solução óptima de $P_\lambda^{\infty,\rho}$ com $z^+=z^{**}$. ■

Nota: Na prova de que existe $\lambda \in \Lambda^0$ tal que $\bar{x}$ optimiza $P_\lambda^{\infty,\rho}$, as componentes λ_i são definidas como λ_i^p no enunciado da proposição com z^p substituído por $\bar{z} = C\bar{x}$.

Concluímos, desta proposição, que qualquer solução calculada por $P_\lambda^{\infty,\rho}$ é eficiente e que é possível calcular-se qualquer solução eficiente através deste programa escalarizante. Contudo, a fórmula que dá o limite superior de ρ não é operacional.

Sempre que se pretenda determinar uma nova solução eficiente do problema PLIMO é necessário resolver o problema escalarizante para um $\lambda \in \Lambda^0$ diferente. Nada garante, no entanto, que um vector de pesos diferente não conduza à mesma solução. Tratando-se de problemas com soluções discretas e em número finito, há certamente um subconjunto de Λ^0 que dá acesso a cada solução eficiente. Na formulação de $P_\lambda^{\infty,\rho}$, os pesos λ_i surgem tanto nos coeficientes técnicos das restrições como nos termos independentes, o que torna difícil uma análise de sensibilidade a estes parâmetros, mesmo que fosse em programação linear contínua. Assim, pelo menos do ponto de vista técnico, é mais atractiva uma abordagem em que a parametrização se dá nos termos independentes das restrições, como acontece quando se varia o ponto de referência.

Consideremos, em primeiro lugar, o programa escalarizante que determina a solução admissível cujo ponto dos objectivos minimiza a distância a z^+ segundo a métrica *não pesada* de Tchebycheff; z^+ é um ponto de referência qualquer que satisfaz $z^+ \geq z$, $\forall z \in Z$, i.e., $z^+ \geq z^*$:

$$\begin{aligned} \min \quad & \alpha && (P_{z^+}^{\infty}) \\ \text{s.a:} \quad & z_i^+ - c^i x \leq \alpha \quad i=1,...,k \\ & x \in X \\ & \alpha \geq 0 \end{aligned}$$

Para a métrica aumentada de Tchebycheff, a função objectivo de $P_{z^+}^{\infty}$ é substituída por $\alpha + \rho \sum_{i=1}^{k} \left(z_i^+ - c^i x \right)$ ou, simplesmente, $\alpha - \rho \sum_{i=1}^{k} c^i x$:

$$\begin{aligned} \min \quad & \alpha - \rho \sum_{i=1}^{k} c^i x && (P_{z^+}^{\infty,\rho}) \\ \text{s.a:} \quad & z_i^+ - c^i x \leq \alpha \quad i=1,...,k \\ & x \in X \\ & \alpha \geq 0 \end{aligned}$$

As restrições $z_i^+ - c^i x \leq \alpha$ são equivalentes a $c^i x + \alpha \geq z_i^+$, pelo que a parametrização em z^+ se faz nos termos independentes das primeiras *k* restrições de $P_{z^+}^{\infty}$ ou $P_{z^+}^{\infty,\rho}$.

Apresentaremos em seguida resultados teóricos para $P_{z^+}^{\infty}$ e $P_{z^+}^{\infty,\rho}$ que seguem em forma e sequência a exposição feita por STEUER E CHOO (1983) para os problemas P_{λ}^{∞} e $P_{\lambda}^{\infty,\rho}$ (parametrizados em λ). Estes resultados têm como intenção mostrar que o programa paramétrico $P_{z^+}^{\infty,\rho}$ permite uma caracterização completa do conjunto das soluções eficientes do problema PLIMO. A simplicidade e potencialidades (nomeadamente ao nível de análise de sensibilidade) do problema $P_{z^+}^{\infty,\rho}$ motivou-nos esta análise e será este o tipo de problema escalarizante que usaremos nas abordagens interactivas que proporemos nos próximos capítulos. É esta a razão fundamental do maior destaque que damos aos resultados teóricos que se seguem.

Proposição II.5: Seja $Y=\{z \in Z \mid z=Cx$ e x é uma solução óptima de $P_{z^+}^{\infty}\}$. Então existe $\bar{z} \in Y$ tal que $\bar{z} \in Z_{nd}$.

> **Prova**: Seja $\bar{\alpha}$ o valor mínimo de α em $P_{z^+}^{\infty}$ e suponhamos que não existe $z \in Y$ tal que z é não dominado. Seja $\hat{z}$ um membro de Y que não é dominado por outro qualquer membro de Y. Se $\hat{z} \notin Z_{nd}$, então é porque existe um $\bar{z} \in Z_{nd}$ tal que $\bar{z} \geq \hat{z}$ e $\bar{z} \neq \hat{z}$ o que implica que $z_i^+ - \bar{z}_i \leq z_i^+ - \hat{z}_i$ para todo o *i*. Como $\hat{z} \in Y$, então $z_i^+ - \bar{z}_i \leq z_i^+ - \hat{z}_i \leq \bar{\alpha}$ para todo o *i*, o que implica que $\bar{z} \in Y$ ou então $\bar{\alpha}$ não é o valor mínimo de $P_{z^+}^{\infty}$, o que é uma contradição. Logo, existe algum $\bar{z} \in Y$ tal que $\bar{z} \in Z_{nd}$.

Nota: a prova desta proposição é semelhante à do teorema 3.1 de STEUER E CHOO (1983).

A *proposição II.5* estabelece que, de entre as soluções óptimas alternativas de $P_{z^+}^{\infty}$ para um dado z^+, existe pelo menos uma que é não dominada.

Quando minimizamos α, encontramos o menor (no sentido de subconjuntos) conjunto $\Phi(z^+,\alpha)=\{z \in \Re^k \mid z_i \in [z_i^+ - \alpha, +\infty)\}$ que intersecta Z, de entre a família de conjuntos $\{\Phi(z^+,\alpha)\}_{\alpha \geq 0}$.

Consideremos o seguinte:

(i) $z^a \in Z_{nd}$, $z^b \in Z$, $z^a \neq z^b$

(ii) $z^{a+} = z^a + \theta e^k$, em que $\theta = \max_{i=1,\ldots,k} (z_i^* - z_i^a)$ e e^k é um vector de 1's de dimensão k. Logo, $z^{a+} \geq z^* \geq z$, $\forall z \in Z$.

(iii) $\alpha^{aa} = \max_{i=1,\ldots,k} (z_i^{a+} - z_i^a)$

(iv) $\alpha^{ab} = \max_{i=1,\ldots,k} (z_i^{a+} - z_i^b)$

Lema II.6: Sejam (i), (ii) e (iii). Então não existe um $z^b \in Z$, $z^b \neq z^a$ que pertence a $\Phi(z^{a+}, \alpha^{aa}) = \{z \in \Re^k : z_i \in [z_i^{a+} - \alpha^{aa}, +\infty)\}$.

Prova: O ponto de referência é z^{a+}, logo as primeiras k restrições de $P_{z^{a+}}^{\infty}$ para z^a ficam $z_i^{a+} - z_i^a \leq \alpha$, $i=1,\ldots,k$ e, por (ii), $z_i^a + \theta - z_i^a \leq \alpha$, $i=1,\ldots,k$. Assim, $\theta \leq \alpha$; $\alpha^{aa} = \theta$ e $\Phi(z^{a+}, \alpha^{aa}) = \{z \in \Re^k : z_i \in [z_i^a, +\infty)\}$. Como $z^a \in Z_{nd}$, então não existe $z^b \in Z$, $z^b \neq z^a$, tal que $z^b \in \Phi(z^{a+}, \alpha^{aa})$.

Lema II.7: Sejam (i), (ii), (iii) e (iv). Então $\alpha^{aa} < \alpha^{ab}$ qualquer que seja $z^b \in Z$.

Prova: Como $z^a \in Z_{nd}$, então pelo lema II.6 não existe outro $z \in Z$ pertencente a $\Phi(z^{a+}, \alpha^{aa})$. Todos os $z^b \in Z$, $z^b \neq z^a$, pertencem a superconjuntos de $\Phi(z^{a+}, \alpha^{aa})$ e, consequentemente, $\alpha^{aa} < \alpha^{ab}$ $\forall z^b \in Z$.

Dos *lemas II.6* e *II.7* concluímos que z^a é a única solução não dominada que minimiza $P_{z^+}^{\infty}$ para $z^+ = z^{a+}$. A *proposição II.8* estabelecerá agora um resultado análogo para $P_{z^+}^{\infty,\rho}$, definindo para tal um limite superior para a constante ρ.

Proposição II.8: Seja $z^a \in Z_{nd}$. Então z^a é o único ponto que minimiza $P_{z^+}^{\infty,\rho}$ para $z^+ = z^{a+}$ definido como em (ii) e $0 < \rho < \rho^a = \min_{z^b \in Z \setminus \{z^a\}} \left\{ \frac{\alpha^{ab} - \alpha^{aa}}{\sum_{i=1}^{k} (z_i^b - z_i^a)} \middle| \sum_{i=1}^{k} (z_i^b - z_i^a) > 0 \right\}$ com α^{aa} e α^{ab} definidos como em (iii) e (iv), respectivamente.

Prova: Pelo lema II.7 ($\alpha^{aa} < \alpha^{ab}$) e pela construção de ρ^a no enunciado, $\rho^a > 0$. Suponhamos uma solução qualquer $z^b \in Z$, $z^b \neq z^a$. O valor da função objectivo de $P_{z^+}^{\infty,\rho}$ para z^b é $\alpha^{ab} - \rho \sum_{i=1}^{k} z_i^b$. Sabemos que z^a será a única solução óptima de

$P_{z^{a+}}^{\infty,\rho}$ se $\alpha^{aa} - \rho \sum_{i=1}^{k} z_i^a < \alpha^{ab} - \rho \sum_{i=1}^{k} z_i^b$, $\forall z^b \in Z \setminus \{z^a\}$. Como $\alpha^{aa} < \alpha^{ab}$, a optimalidade de z^a só poderá ser posta em causa se $\sum_{i=1}^{k} \left(z_i^b - z_i^a\right) > 0$. Para assegurar a optimalidade (única) de z^a devemos ter $\rho \sum_{i=1}^{k} \left(z_i^b - z_i^a\right) < \alpha^{ab} - \alpha^{aa}$, $\forall z^b \in Z \setminus \{z^a\}$, ou seja $\rho < \dfrac{\alpha^{ab} - \alpha^{aa}}{\sum_{i=1}^{k} \left(z_i^b - z_i^a\right)}$ sempre que $\sum_{i=1}^{k} \left(z_i^b - z_i^a\right) > 0$. Sendo assim, $\rho < \rho^a$ com ρ^a definido como no enunciado da proposição garante a optimalidade de z^a e a sua unicidade.

Proposição II.9: Seja $0 < \rho < \min_{z^p \in Z_{nd}} \left[\min_{z^h \in Z \setminus \{z^p\}} \left\{ \dfrac{\alpha^{ph} - \alpha^{pp}}{\sum_{i=1}^{k} \left(z_i^h - z_i^p\right)} \middle| \sum_{i=1}^{k} \left(z_i^h - z_i^p\right) > 0 \right\} \right]$ com α^{pp} e α^{ph} definidos de acordo com (iii) e (iv), respectivamente. Então, x^a é eficiente sse existe z^+ tal que x^a é uma solução óptima de $P_{z^+}^{\infty,\rho}$.

Prova: Seja $z^a = Cx^a$.

$\Rightarrow$ directamente da proposição II.8 com $z^+ = z^{a+}$ definido como em (ii) e porque $\rho < \rho^a$.

$\Leftarrow$ Suponhamos que x^a não é eficiente e que minimiza $P_{z^+}^{\infty,\rho}$ para um dado z^+. Com z^a dominado existe algum $z^b \in Z$ tal que $z^b \geq z^a$ e $z^b \neq z^a$. Isto implica que $z^+ - z^b \leq z^+ - z^a$ e $\rho \sum_{i=1}^{k} z_i^b > \rho \sum_{i=1}^{k} z_i^a$. O valor da função objectivo de $P_{z^+}^{\infty,\rho}$ é então menor em z^b do que em z^a, o que contradiz a optimalidade de x^a. Logo x^a é eficiente.

A *proposição II.9* permite concluir que é possível gerar qualquer solução eficiente do problema PLIMO através da minimização de distâncias não pesadas de Tchebycheff a pontos de referência, e considerando apenas pontos de referência superiores ou iguais à solução ideal. Esta não é uma limitação porque poder-se-iam usar outros pontos de referência. Apenas não é obrigatório faze-lo para caracterizar por completo o conjunto das soluções eficientes.

Notemos ainda que z^{a+} tal como foi definido em (ii) é sempre um ponto de referência inteiro, o que permite definir o programa escalarizante como inteiro puro. Sendo Z finito e composto apenas por pontos inteiros, se z^+ for restrito em $P_{z^+}^{\infty,\rho}$ a pontos inteiros, então uma outra expressão pode ser deduzida para delimitar ρ superiormente. Reparemos, em primeiro lugar, que a expressão que limita os valores admissíveis para ρ na proposição II.9 não depende de soluções dominadas, e por isso o operador $\min_{z^h \in Z \setminus \{z^p\}}$ pode ser substituído por $\min_{z^h \in Z_{nd} \setminus \{z^p\}}$. Vejamos porquê:

> Se $z^h \notin Z_{\text{nd}}$, então existe $z^r \in Z$ que o domina, ou seja $z_i^r \geq z_i^h$ para todo o $i = 1,...,k$ e para pelo menos um i a desigualdade é estrita. Então $\sum_{i=1}^{k}\left(z_i^r - z_i^p\right) > \sum_{i=1}^{k}\left(z_i^h - z_i^p\right)$. O valor de α^{pp} não se altera e $\alpha^{pr} = \max_{i=1,...,k}\left(z_i^{p+} - z_i^r\right) \leq \max_{i=1,...,k}\left(z_i^{p+} - z_i^h\right) = \alpha^{ph}$. Logo, $\frac{\alpha^{pr} - \alpha^{pp}}{\sum_{i=1}^{k}\left(z_i^r - z_i^p\right)} < \frac{\alpha^{ph} - \alpha^{pp}}{\sum_{i=1}^{k}\left(z_i^h - z_i^p\right)}$.

Este resultado foi observado por KALISZEWSKI (1994) relativamente à expressão que limita ρ no teorema de STEUER E CHOO (1983), que nós designámos por proposição II.4 (notemos que essa expressão apenas difere da nossa na definição de α^{pp} e α^{ph}). KALISZEWSKI (1994) acrescentou ainda que o denominador $\sum_{i=1}^{k}\left(z_i^h - z_i^p\right)$ tem como limite superior $k\delta$ com $\delta = \max_i \max_{z,z' \in Z} \left\| z_i - z_i' \right\|^{\text{E}}$ em que $\|.\|^{\text{E}}$ denota a norma Euclideana. Em particular para Z finito e composto por vectores inteiros, este limite superior pode ser diminuído para $\delta(k-1)-1$. Na verdade, como z^h e z^p são não dominados, existe pelo menos um índice s tal que z_s^h é menor do que z_s^p de pelo menos 1 o que justifica a expressão anterior. Voltando às condições da proposição II.9, temos ainda que o numerador $\alpha^{ph} - \alpha^{pp}$ é sempre maior ou igual a 1. Consequentemente, uma outra fórmula para valores admissíveis de ρ em $P_{z^+}^{\infty,\rho}$, que garante a geração de todas as soluções não dominadas de um problema PLIMO, é $0 < \rho < \frac{1}{\delta(k-1)-1}$.

Até agora considerámos apenas pontos de referência inatingíveis que satisfazem a condição $z^+ \geq z^*$. Mas o ponto de referência pode ser qualquer desde que a variável α seja definida em $P_{z^+}^{\infty}$ ou $P_{z^+}^{\infty,\rho}$ sem restrição de sinal. Referimo-nos aos problemas escalarizantes *min-max* $P_{q,\lambda}$ e $P_{q,\lambda}^{\rho}$,

definidos na secção anterior, com $\lambda_i=1$, $\forall i$. Nestes casos, apenas não devemos falar em minimização da métrica (aumentada) de Tchebycheff.

Facilmente se prova que, dado um ponto de referência qualquer $q \in \Re^k$, a solução eficiente resultante do programa escalarizante com "α livre" é igual à que se obtém para o ponto de referência $q+\theta e^k$ com θ positivo suficientemente grande para que $q+\theta e^k \geq z^*$. Apenas varia o valor de α. É, pois, sempre possível usar o programa escalarizante de Tchebycheff aumentado $P_{z^+}^{\infty,\rho}$ desde que o ponto de referência seja previa e convenientemente ajustado, o que é tecnicamente vantajoso porque se trabalha com a variável α não negativa em vez de a considerar livre.

II.1.4 Extensão ao caso linear inteiro-misto multiobjectivo (PLIMMO)

Consideremos o modelo de programação linear inteira-mista multiobjectivo (PLIMMO):

$$\begin{aligned} &\max \quad z_1 = f_1(x) = c^1 x \qquad \text{(PLIMMO)}\\ &\ldots\\ &\max \quad z_k = f_k(x) = c^k x\\ &\text{s.a:} \quad x \in X\\ &\qquad X = \left\{ x \in \Re^n \mid Ax = b,\, x \geq 0,\, x_j \text{ inteiro},\, j \in I \right\}, \end{aligned}$$

em que I é o conjunto dos índices das variáveis inteiras, $I \subset \{1,...,n\}$, $I \neq \varnothing$; c^i, $i=1,...,k$, (de dimensão $1\times n$), A (de dimensões $m\times n$ e característica m) e b ($m\times 1$) são matrizes de componentes reais que representam, respectivamente, os coeficientes das funções objectivo, os coeficientes das restrições e os termos independentes das restrições. Tal como anteriormente, consideramos C a matriz $k\times n$ cuja i-ésima linha é o vector c^i e as definições anteriores de Z, X_{ef}, Z_{nd}, z^* e z^{**}. Assumimos também que X é limitado.

Tal como o problema PLIMO, também PLIMMO não admite soluções eficientes impróprias, mas pode ter soluções eficientes não suportadas, em virtude da não convexidade de X. Se, por um lado, as formas de caracterização do conjunto eficiente que são válidas para PLIMO são, em geral, válidas para PLIMMO, existe uma questão, de relevância essencialmente teórica, que cria uma dificuldade adicional neste caso relativamente ao anterior. Veremos adiante que as dificuldades da passagem da PLIMO para a PLIMMO também se podem estender ao desenvolvimento de métodos, nomeadamente métodos interactivos. As particularidades do caso inteiro-misto multiobjectivo têm sido bastante ignoradas pelos investigadores, mesmo nos estudos mais teóricos em que os autores particularizam os casos poliédrico, discreto, não linear,

discreto infinito, mas não se referem ao caso inteiro-misto. TEGHEM E KUNSCH (1986A) referem a não existência de técnicas específicas para caracterizar o conjunto das soluções eficientes em PLIMMO, o que os leva a considerar, no seu artigo de revisão, apenas o caso inteiro puro.

Qualquer solução eficiente do problema PLIMMO pode ser alcançada através de uma parametrização adequada do programa das somas pesadas com restrições adicionais nos valores das funções objectivo ($P^1_{\lambda,b}$).

Também P^{∞}_{λ}, $P^{\infty}_{z^+}$ ou $P_{q,\lambda}$, tal como foram definidos anteriormente, podem gerar qualquer solução eficiente do problema PLIMMO mas, mais uma vez, uma solução calculada por estes programas pode ser fracamente eficiente. Como em outros casos, esta questão pode ser ultrapassada pela consideração de $P^{\infty,\rho}_{\lambda}$, $P^{\infty,\rho}_{z^+}$ e $P^{\rho}_{q,\lambda}$ em substituição de P^{∞}_{λ}, $P^{\infty}_{z^+}$ e $P_{q,\lambda}$, respectivamente. Contudo, é aqui que reside a principal diferença face ao caso de PLIMO. A dificuldade consiste em estabelecer um limite superior para o valor de ρ tal que o programa escalarizante consiga alcançar toda e qualquer solução eficiente e, consequentemente, validar as proposições II.4 e II.9 para o caso PLIMMO.

Num problema PLIMMO podem existir "pequenos" subconjuntos de soluções não dominadas que o programa da métrica aumentada (pesada ou não) de Tchebycheff não consegue calcular, mesmo considerando ρ muito pequeno. Essas soluções não dominadas situam-se próximo de alguma solução fracamente não dominada. Devemos, porém, salientar que esta questão tem interesse essencialmente teórico. Na prática, podemos atribuir a ρ um valor de tal maneira pequeno que o agente de decisão não discrimine essas soluções não dominadas da solução fracamente não dominada "vizinha". Esta questão é ilustrada no exemplo II.10.

> **Exemplo II.10.** A figura II.10 mostra o comportamento da métrica (pesada ou não) aumentada de Tchebycheff num problema PLIMMO. O conjunto das soluções não dominadas é constituído pelo segmento desde z^1 a z^2, incluindo os dois vértices, e o segmento de z^3 a z^4, incluindo z^4 mas excluindo z^3 que é fracamente não dominada. Para um dado ρ, as soluções não dominadas entre z^3 e $z^{3'}$ não são possíveis de alcançar porque é z^2 que optimiza o programa escalarizante. Diminuir o valor de ρ corresponde a aproximar $z^{3'}$ da solução fracamente não dominada z^3, mas há sempre soluções entre $z^{3'}$ e z^3 que não são possíveis de alcançar. Estas soluções são claramente desfavorecidas na razão de perda/ganho em relação a z^2, já que $\left(z_1^2 - z_1^{3'}\right) \big/ \left(z_2^{3'} - z_2^2\right)$ é muito elevado.

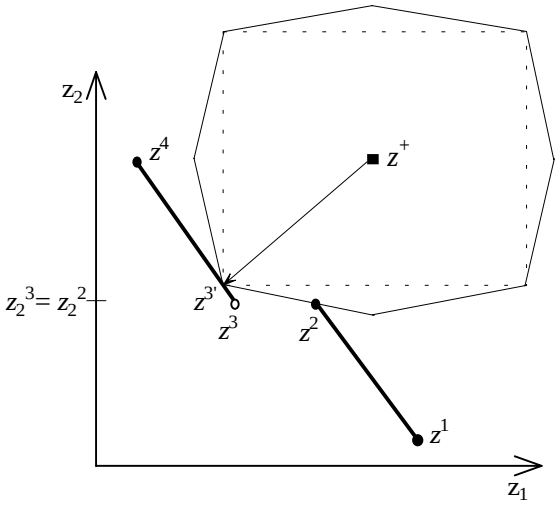


Fig. II.10 – Uso da métrica aumentada de Tchebycheff em PLIMMO.

Este tipo de situação pode também ocorrer em problemas não lineares para as soluções não dominadas próprias próximas de soluções não dominadas impróprias. Em ambos os casos (PLIMMO ou não linear) um tipo de abordagem *lexicográfica*, como a proposta por STEUER E CHOO (1983), ultrapassa esta questão com os inconvenientes práticos de poder requerer dois estádios de optimização.

Mas, apesar de tudo, os programas $P_{\lambda}^{\infty,\rho}$, $P_{z^+}^{\infty,\rho}$ ou $P_{q,\lambda}^{\rho}$ podem ser usados para estabelecer uma condição necessária e suficiente de determinação de qualquer solução eficiente que não ultrapasse uma determinada razão (pré-estabelecida) de perda/ganho relativamente a outras soluções. Referimo-nos a soluções que satisfazem a condição de eficiência própria (definição II.5) para M definido *a priori*. Estas soluções enquadram-se na definição de solução *eficiente* ε*–própria* de LEWANDOWSKI E WIERZBICKI (1988) em que ε representa um número positivo pequeno tal que $M=1+1/\varepsilon$. Este conceito de eficiência, o mais restrito de todos os apresentados, é, segundo WIERZBICKI (1998), o mais prático de todos apesar de ser o mais difícil de expressar teoricamente.

Em suma, os programas escalarizantes baseados na métrica não aumentada de Tchebycheff podem calcular qualquer solução eficiente mas também calculam soluções fracamente eficientes – incluímos aqui os programas min-max que, de uma forma mais geral consideram funções escalarizantes de realização (P_{λ}^{∞}, $P_{z^+}^{\infty}$, $P_{q,\lambda}$); as abordagens lexicográficas permitem calcular qualquer solução eficiente evitando as fracamente eficientes; as formas aumentadas ($P_{\lambda}^{\infty,\rho}$, $P_{z^+}^{\infty,\rho}$, $P_{q,\lambda}^{\rho}$) são uma boa substituição prática das anteriores. Neste último caso, devemos escolher para ρ um valor positivo bastante pequeno mas computacionalmente significativo. Notemos que, embora seja possível estabelecer fórmulas que limitam superiormente o valor de ρ nos casos de

PLMO e PLIMO, elas não são operacionais, o que tem como consequência que o mesmo tipo de prática deva ser adoptada na selecção do valor de ρ.

II.2 ABORDAGENS INTERACTIVAS VERSUS NÃO INTERACTIVAS

Apesar de as formas de caracterização do conjunto das soluções eficientes constituírem importantes resultados teóricos, elas não oferecem um meio explícito de apoio à decisão em problemas com objectivos múltiplos, em particular problemas de PLIMO e de PLIMMO.

Nas últimas décadas tem sido desenvolvido muito trabalho no sentido de criar métodos dedicados a problemas com múltiplos critérios. Mas, não podemos deixar de referir a pouca atenção que a programação inteira e inteira-mista multiobjectivo, e os problemas combinatórios multicritério em geral, tem merecido por parte dos investigadores, principalmente se comparada com a programação linear multiobjectivo com variáveis contínuas (PLMO) ou os modelos multiatributo (onde o conjunto das alternativas é definido explicitamente). Este facto é bem reconhecido no artigo de revisão de ULUNGU E TEGHEM (1994), em que os autores referem que o "espírito multiobjectivo" não prevalece ainda na optimização combinatória e há ainda muito a fazer nesta direcção. As dificuldades inerentes a estes problemas não são fáceis de lidar, o que constitui uma segunda razão para que a optimização combinatória multiobjectivo tenha sido substancialmente ignorada em comparação com a vasta literatura sobre problemas combinatórios monocritério.

De facto, as dificuldades resultantes da complexidade computacional destes problemas agravam-se com a passagem do mono ao multicritério. É que, enquanto o caso monocritério se resume a um problema de optimização, o conhecimento de soluções eficientes de um problema multiobjectivo envolve, em geral, a optimização de vários problemas escalarizantes de igual ou maior complexidade.

No contexto da programação matemática multiobjectivo, a atenção dos investigadores tem recaído essencialmente na PLMO. A introdução de fenómenos discretos em modelos lineares multiobjectivo conduz a problemas de PLIMO ou PLIMMO que não podem, em geral, ser tratados pelos métodos de PLMO. A construção de metodologias adequadas a problemas de PLIMO e PLIMMO envolve mais do que uma simples combinação de métodos de PLMO com técnicas de programação inteira. Esta questão acentua-se no caso de PLIMMO porque, além do conjunto das soluções admissíveis não ser convexo (tal como em PLIMO) não se pode usufruir das vantagens de uma situação em que todas as soluções são discretas. Existem, pois, abordagens multiobjectivo para problemas de PLIMO que não são aplicáveis a problemas de PLIMMO (exemplos dessas abordagens são o método de KLEIN E HANNAN (1982) que gera todas a

soluções eficientes do problema e o método interactivo de MARCOTTE E SOLAND (1980, 1986), que abordaremos mais à frente).

Os métodos multicritério podem classificar-se de acordo com o tipo de intervenção que é requerida do agente de decisão (AD). Podem distinguir-se três grandes classes de métodos conforme a articulação das preferências do AD é feita *a priori*, *a posteriori*, ou progressivamente.

No primeiro tipo, toda a informação de preferências é indicada pelo AD antes do tratamento computacional do problema. Apesar de existir modelação multiobjectivo, o problema é depois tratado através de um sistema de relações consolidadas, por exemplo, através da construção de uma função utilidade.

No segundo tipo, todo o conjunto das soluções eficientes (ou um subconjunto deste) é calculado e apresentado ao AD para avaliação. Estes métodos são habitualmente nomeados de métodos *geradores*.

No terceiro tipo, através da interacção com o AD há uma captura progressiva das suas preferências. Como moderador no diálogo pode existir um analista que auxilia o AD na interpretação de resultados e exposição das suas preferências, no caso de este não estar familiarizado com a metodologia ou o sistema computacional usado. Estes métodos – vulgarmente designados por métodos *interactivos* – alternam fases de cálculo com fases de diálogo com o AD. Nas fases de cálculo são construídas propostas (uma ou mais soluções) que são depois submetidas ao AD. O AD deve então reagir e indicar informação acerca das suas preferências que será incorporada na fase de cálculo seguinte.

A dificuldade em utilizar métodos do primeiro tipo reside na obtenção *a priori* de informação de preferências do AD para se poder construir um sistema de relações que agregue, numa única dimensão, todos os critérios explicitamente considerados no modelo.

Também os métodos *geradores* têm sido alvo de numerosas críticas devido ao elevado esforço computacional envolvido no cálculo exaustivo das soluções eficientes. É ainda relevante o facto de o AD ser "inundado" de alternativas no final do processo (eventualmente algumas delas com valores próximos entre si), o que dificulta a análise e uma escolha final.

No âmbito da PLIMO/PLIMMO, foi nas décadas de 70 e 80 que foi desenvolvida a maioria dos métodos *geradores*. Sem preocupação de exaustividade, referimos os seguintes métodos que julgamos serem os mais conhecidos:

- métodos de PASTERNAK E PASSY (1973), BITRAN (1977, 1979), DECKRO E WINKOFSKY (1983), KIZILTAN E YUCAOGLU (1983) e WHITE (1984) para problemas de PLIMO com variáveis 0-1;
- métodos de KLEIN E HANNAN (1982), VILLAREAL E KARWAN (1981) e CHALMET ET AL. (1986) para problemas de PLIMO.

Observamos que grande parte dos métodos *geradores* de PLIMO se restringem a problemas com variáveis binárias, o que facilmente se compreende pela maior facilidade em usar técnicas de enumeração face a outros problemas inteiros ou inteiros-mistos. Este facto é visível no artigo de revisão de RASMUSSEN (1986) dedicado à programação 0-1 multiobjectivo. Há métodos que tiram partido do caso bicritério, como por exemplo os métodos de PASTERNAK E PASSY (1973) e CHALMET ET AL. (1986).

Alguns dos métodos referidos usam um processo construtivo, acrescentando iterativamente novas soluções ao conjunto eficiente. É o caso dos métodos de PASTERNAK E PASSY (1973), BITRAN (1977, 1979), KLEIN E HANNAN (1982), WHITE (1984) e CHALMET ET AL. (1986). Por exemplo, o método de KLEIN E HANNAN (1982) calcula todas as soluções eficientes de um problema usando um processo iterativo que restringe sucessivamente a região admissível através de restrições que não são satisfeitas pelas soluções eficientes anteriores. Estas restrições resultam de condições dos tipos 'e' e 'ou'. Como a formalização de condições 'ou' é feita à custa de variáveis auxiliares binárias, a complexidade do problema monocritério que se resolve em cada iteração vai aumentando.

Outros métodos trabalham com soluções 'potencialmente' eficientes, e só quando termina o processo de cálculo é que se conhece quais são as soluções realmente eficientes – VILLAREAL E KARWAN (1981), DECKRO E WINKOFSKY (1983) e KIZILTAN E YUCAOGLU (1983).

Grande parte dos métodos *geradores* que mencionámos estão resumidos no artigo de TEGHEM E KUNSCH (1986[A]) sobre metodologias de caracterização do conjunto eficiente em problemas de PLIMO. As principais características destes métodos encontram-se também descritas no recente estudo de FERREIRA (1997) sobre métodos de PLIMO/PLIMMO. TEGHEM E KUNSCH (1986[A]) concluem que não será errado dizer que nenhum destes métodos tem apetência para lidar convenientemente com problemas de grandes dimensões. Tal afirmação não constitui surpresa se atendermos à complexidade dos problemas de PLIMO. Importa ainda referir que nenhum destes métodos se aplica ao caso inteiro-misto. Apenas recentemente MAVROTAS E DIAKOULAKI (1998) propuseram um método gerador para o caso 0-1 misto. A técnica consiste em gerar e guardar soluções 'potencialmente' não dominadas e eliminar sucessivamente as dominadas através de comparações par a par, até que no final restem apenas as não dominadas. Os resultados computacionais apresentados neste artigo ilustram bem o esforço computacional envolvido na geração completa do conjunto eficiente: problemas aleatórios com 50 restrições, 25 variáveis contínuas, 25 variáveis binárias e 2 funções objectivo requereram um tempo computacional médio de 1 h 5 min num computador Pentium a 150 MHz. Reconhecendo a impraticabilidade desta abordagem para problemas maiores, os autores sugerem a intervenção do AD para restringir o âmbito da pesquisa, nomeadamente impondo limitações nos valores das funções objectivo e estipulando uma grelha de filtros no processo gerador.

Atendendo às dificuldades, já referidas, dos métodos geradores e o elevado número de soluções não dominadas que um problema pode ter, os métodos *interactivos* têm tido um desenvolvimento crescente. Eles ilustram a tendência actual da programação multicritério – uma evolução no sentido do apoio à decisão, ou seja, no sentido de ajudar o AD a encontrar uma solução de compromisso satisfatória. A classificação de um método como interactivo ou não interactivo nem sempre é inequívoca. Existem métodos que têm como intenção gerar um subconjunto representativo de soluções não dominadas – métodos geradores de acordo com a definição atrás – mas que poderiam ser facilmente incorporados numa estrutura interactiva. O método biobjectivo de SOLANKI (1991), que introduziremos adiante, é um bom exemplo de uma abordagem desse tipo.

Foi essencialmente no início da década de 80 que os investigadores começaram a dedicar-se ao desenvolvimento de métodos *interactivos* para PLIMO e PLIMMO. Cedo se aperceberam das vantagens dos métodos interactivos neste tipo de problemas, já reveladas anteriormente em outros problemas (como por exemplo, em problemas de PLMO) e que, por maioria de razão, se afiguram promissores nos problemas com variáveis inteiras. Este tipo de métodos permite reduzir o esforço computacional, especialmente relevante em problemas de PLIMO e de PLIMMO de grandes dimensões, e apoiam o AD na escolha da sua solução preferida. TEGHEM E KUNSCH (1986[B]), não reclamando exaustividade, fazem uma revisão de métodos interactivos para PLIMO e PLIMMO publicados até ao final de 1985. Dos métodos apresentados, seis ao todo, o primeiro data de 1980 (VILLAREAL ET AL., 1980). Abrangendo o mesmo período de tempo, uma outra revisão deve-se a RASMUSSEN (1986), que se debruçou sobre a programação 0-1 multiobjectivo. Este trabalho classifica 27 métodos, não se confinando aos métodos interactivos. Na verdade, de entre os métodos apresentados e de acordo com a classificação de Rasmussen, menos de metade são interactivos e, enquanto que os não interactivos se aplicam, na sua maioria, exclusivamente ao caso 0-1, os métodos interactivos destinam-se a PLIMO e aplicam-se ao caso 0-1 por ser um caso particular desse. Uma revisão mais recente da área encontra-se na obra de FERREIRA (1997), onde se adopta a taxonomia dos métodos proposta por CLÍMACO ET AL. (1997). Um outro trabalho de revisão específico à programação multiobjectivo inteira-mista deve-se a ALVES E CLÍMACO (1998[A]). Este trabalho inclui resumos daqueles que julgamos serem os principais métodos interactivos de PLIMMO, classificando-os segundo o processo de cálculo das soluções eficientes e o tipo de interacção com o AD.

Nos métodos *interactivos* o conjunto das soluções não dominadas é progressivamente explorado através da alternância entre fases de cálculo e fases de diálogo, até que uma proposta seja considerada satisfatória pelo AD ou pelo algoritmo. Apesar de terem uma estrutura formal comum, podem distinguir-se algumas concepções que justificam o uso da interactividade.

Importa, pois, introduzir os maiores paradigmas seguidos pelos autores dos métodos interactivos.

Alguns autores admitem que as preferências do AD podem ser representadas por uma *função utilidade implícita*. Por conseguinte, o processo interactivo tem como intenção descobrir o 'óptimo' (ou uma aproximação deste) de uma função utilidade implícita do AD. A convergência para esse 'óptimo' requer que o AD não se contradiga nas suas respostas ao longo do processo interactivo.

Em contraste com este tipo de abordagens, existem aquelas que têm como intenção permitir uma *aprendizagem* progressiva do conjunto das soluções não dominadas e das preferências do AD. Este tipo foi chamado de *comunicação aberta* em ALVES E CLÍMACO (1998[A]), designação esta inspirado no conceito de 'open exchange' definido por FEYERABEND (1975). Estas abordagens não procuram a convergência para o 'óptimo' de uma função utilidade implícita, mas apenas ajudar o AD a evitar a pesquisa de soluções não dominadas que não são interessantes para ele, concentrando-se naquelas que vão ao encontro das suas preferências. Não há, assim, decisões irrevogáveis durante o processo de decisão, podendo o AD voltar atrás sempre que o desejar. O AD é chamado em cada interacção para dar algumas indicações para a pesquisa de novas soluções não dominadas, podendo eventualmente introduzir restrições adicionais (por exemplo, restrições temporárias nos valores das funções objectivo). O processo termina quando o AD entender que já conhece o suficiente acerca do conjunto das soluções não dominadas e encontrou uma *solução de compromisso satisfatória*. Usando a terminologia de ROY (1987), a "convergência" dá lugar à "criação". O processo interactivo é um processo construtivo e não a busca orientada por algo pré-existente.

VANDERPOOTEN (1992) defende um modelo que se inicia com uma *fase de aprendizagem*, na qual o AD aprende acerca do seu problema e das suas preferências através de uma *exploração livre*. Quando a estrutura de preferências estiver bem estabilizada, o AD pode prosseguir para uma *fase de procura* onde a exploração é dirigida e tem como intenção detectar a melhor solução de compromisso. Este processo pode ser retroactivo pelo que deve ser visto como uma sucessão de fases de aprendizagem e fases de procura. Segundo Vanderpooten, a maior parte dos procedimentos propostos na literatura favorecem, ou a fase de procura, ou a fase de aprendizagem. Esta distinção nas duas concepções do uso da interactividade não difere muito da que apresentámos no parágrafo anterior. Os procedimentos orientados para a *fase de procura* são aqueles em que é claramente assumido que a estrutura de preferências do AD é pré-existente e representada por uma *função utilidade implícita* ou, não se assumindo especificamente o tipo de estrutura preferencial, supõe-se que ela permanece estável. Ou seja, a concepção orientada para a procura ajuda o AD a determinar uma prescrição. Os procedimentos orientados para a *fase de aprendizagem* rejeitam a hipótese de que a estrutura de preferências do AD é pré-existente e

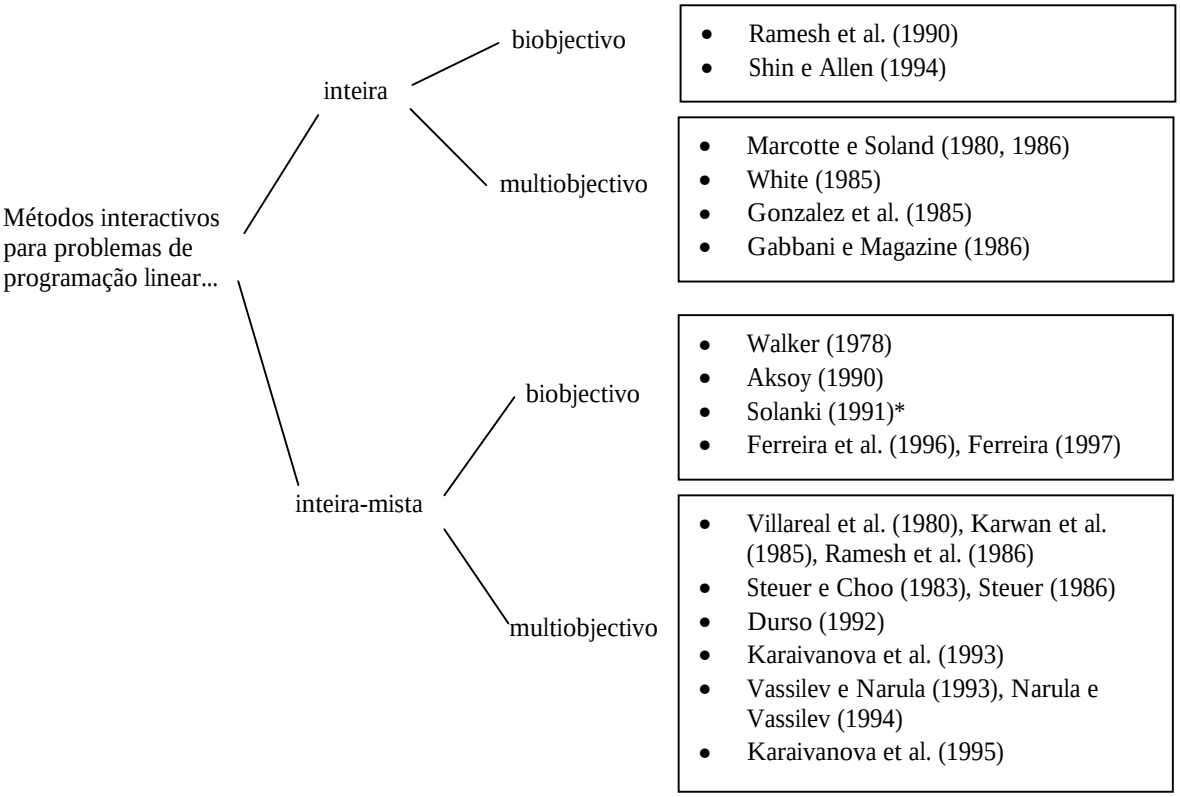
estável. Projectados, acima de tudo, para apoiar a aprendizagem das preferências, estes procedimentos auxiliam o AD a explorar o conjunto das alternativas sob a forma de tentativa-erro. VANDERPOOTEN (1992) observa que é interessante a mistura dos dois tipos podendo ser implementada por um mecanismo de "orientar por omissão", livremente interrompido e re-orientado pelo AD.

Na secção seguinte introduziremos os métodos interactivos de PLIMO e PLIMMO encontrados na bibliografia consultada, caracterizando-os segundo o tipo de protocolo com o AD e o processo usado para o cálculo das soluções eficientes.

II.3 MÉTODOS INTERACTIVOS PARA PLIMO E PLIMMO

II.3.1 Revisão e categorização

Foi essencialmente nas décadas de 80 e 90 que surgiram as propostas de métodos interactivos para PLIMO e PLIMMO e, como podemos observar na lista apresentada na figura II.11, a investigação nesta área não é vasta. Sem pretensão de exaustividade, julgamos serem estes os principais métodos, ou pelo menos os mais divulgados pela comunidade científica da área.



(* Apesar de não interactivo, o método de SOLANKI (1991) poderia facilmente ser enquadrado num protocolo interactivo e por isso é aqui incluído).

Fig. II.11 – Métodos interactivos para PLIMO e PLIMMO mais divulgados.

Dos métodos interactivos que se aplicam a problemas de PLIMO e/ou PLIMMO, podemos distinguir aqueles que se dedicam a problemas em que todas as variáveis são inteiras (PLIMO) e aqueles que se aplicam também ao caso inteiro-misto (PLIMMO) ou até a problemas mais genéricos (como por exemplo, o método de STEUER E CHOO, 1983). É esta a distinção que fazemos na taxonomia apresentada na figura II.11, pelo que todos os métodos incluídos na classe '*inteira-mista*' também tratam problemas da classe '*inteira*'. Não distinguimos o caso particular de variáveis binárias porque não encontrámos nenhum método interactivo dedicado exclusivamente a este tipo de problemas. Distinguimos ainda o caso '*biobjectivo*' e o caso '*multiobjectivo*', já que o primeiro é naturalmente de aplicabilidade mais limitada. Resta ainda salientar o facto de que todos estes métodos se aplicam a problemas lineares, podendo alguns deles também tratar problemas não lineares. Estes casos serão explicitamente assinalados no resumo que se segue de cada um dos métodos.

De acordo com o exposto na secção II.1, o uso de somas pesadas das funções objectivo não permite alcançar soluções eficientes não suportadas, mas a consideração de restrições adicionais, designadamente nos valores das funções objectivo, já permite ultrapassar esta dificuldade. Alguns autores utilizam processos de cálculo que se enquadram neste tipo (*grupo I*). Muitos nem sequer consideram parametrização ao nível dos pesos mas apenas nas restrições adicionais, como veremos a seguir. Um outro tipo de processo de cálculo de soluções eficientes baseia-se na métrica de Tchebycheff (pesada e/ou aumentada) ou, mais genericamente, usa funções escalarizantes de realização que projectam pontos de referência no conjunto das soluções não dominadas (*grupo II*). As abordagens que usam este segundo tipo de processo de cálculo são em geral mais abrangentes, adequando-se a problemas inteiros, inteiros-mistos, lineares ou não lineares. Nestes dois grupos incluem-se quase todos os métodos interactivos. Exclui-se o método de SHIN E ALLEN (1994) que usa uma técnica particular de cálculo para problemas biobjectivo.

Alguns dos métodos pressupõem uma função utilidade implícita, ou pelo menos uma estrutura de preferências pré-existente estável. Outros são orientados para a aprendizagem assumindo um protocolo de comunicação aberta com o AD.

Na breve descrição de cada um dos métodos tentaremos caracterizá-los segundo dois aspectos: processo de cálculo e tipo de protocolo interactivo. Dentro de cada classe definida na taxonomia da figura II.11, os métodos surgirão por ordem cronológica.

II.3.1.1 Programação linear inteira biobjectivo

RAMESH ET AL. (1990)

RAMESH ET AL. (1990) propõem um método interactivo dedicado a problemas de programação linear inteira biobjectivo (PLIB). Assume-se a existência de uma *função utilidade* implícita do AD, pseudo-côncava e não decrescente.

A metodologia emprega uma versão modificada do método de Zionts-Wallenius para problemas de PLMO (ZIONTS E WALLENIUS, 1983) num contexto de *branch-and-bound*. O método de Zionts-Wallenius (de PLMO) usa somas pesadas das funções objectivo para o cálculo de soluções eficientes, reduzindo progressivamente o conjunto dos pesos através de restrições impostas a partir de informação de preferências do AD. Esta informação resulta da comparação de pares de soluções não dominadas e da avaliação de *vectores de compensação* (tendências de variação unitária das funções objectivo ao longo de arestas que têm origem numa solução não dominada e conduzem a outras soluções não dominadas).

O método de PLIB de Ramesh et al. começa por relaxar as condições de integralidade e aplicar o método de Zionts-Wallenius à relaxação linear do problema de PLIB. Se a solução considerada 'óptima' (para a função utilidade implícita) for inteira, então será também 'óptima' para o problema original. Caso contrário, é conduzida uma pesquisa de *branch-and-bound*. Partindo de uma solução incumbente inteira inicial $z^1 = \left(z_1^1, z_2^1\right)$, obtida heuristicamente, a região admissível relaxada é partida em 2 subconjuntos, mutuamente exclusivos. Estes subconjuntos são obtidos, respectivamente, pela adição das restrições $f_1(x) \geq z_1^1$ e $(f_1(x) \leq z_1^1 - \varepsilon \ \wedge \ f_2(x) \geq z_2^1)$, sendo ε um valor positivo pequeno. A pesquisa é então conduzida separadamente em cada subconjunto. Os problemas candidatos a investigação na pesquisa *branch-and-bound* são gerados através da imposição de limitações superiores ou inferiores em variáveis com valor não inteiro. Cada problema associado a um nodo da árvore de *branch-and-bound* é um problema biobjectivo relaxado linearmente que será resolvido usando a estratégia do método de Zionts-Wallenius. Além das restrições habituais do método de Zionts-Wallenius no conjunto dos pesos, podem também ser impostas outras restrições num problema candidato (biobjectivo relaxado). São restrições no espaço dos objectivos resultantes da comparação de um par de soluções não dominadas ou outras restrições de índole global que permitem excluir a avaliação de soluções adjacentes e *vectores de compensação*. Um nodo não será ramificado se a solução 'óptima' do respectivo problema candidato for preterida relativamente à solução incumbente ou se, pelo contrário, for inteira e preferida à incumbente, procedendo-se neste caso à actualização da solução incumbente.

Esta metodologia insere-se naquele que designámos de *grupo I* no que diz respeito à geração de soluções eficientes.

SHIN E ALLEN (1994)

Neste artigo é proposto um método interactivo para problemas de programação matemática inteira biobjectivo. O problema pode ser linear ou não linear com funções objectivo côncavas. Assume-se que é convexa a região admissível sem as restrições de integralidade.

Este método pretende isolar a melhor solução de compromisso para a *função utilidade* implícita através da avaliação sucessiva de pares de soluções não dominadas. Com base nas respostas do AD da comparação de pares de soluções, eliminam-se regiões de pesquisa através da imposição de restrições nos valores das funções objectivo. É utilizado um processo particular de cálculo que determina a solução não dominada suportada que está mais próxima, para a direita ou para a esquerda, da solução anterior (relembramos que são problemas biobjectivo). Esta solução designa-se por ASN ('Associated Supported Nondominated'). Dado um ponto não dominado z^s, determina-se um ponto ASN para a direita de z^s da seguinte forma: graficamente, traça-se uma semi-recta horizontal com início em z^s e roda-se para baixo (no sentido dos ponteiros do relógio) até intersectar um ponto admissível – é então esse o ponto ASN à direita de z^s (ver figura II.12). Matematicamente, um ponto ASN obtém-se pela optimização de um problema auxiliar não linear (mesmo quando o problema biobjectivo é linear). Analogamente se determina um ponto ASN para a esquerda de z^s.

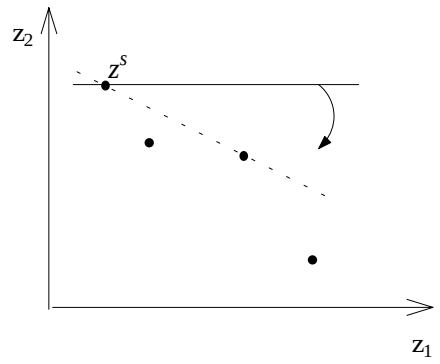


Fig. II.12 - Ilustração do processo de cálculo de soluções não dominadas por SHIN E ALLEN (1994).

O algoritmo interactivo começa por determinar a solução não dominada que maximiza $f_2(x)$ e considera-a a solução preferida actual. Segue-se um processo iterativo com os seguintes passos: (1) Determina-se um ponto ASN da solução preferida. Se não existir nenhum (para a direita ou para a esquerda), termina. Sejam z^s e z^t , uma a solução preferida e a outra a respectiva ASN, considerando que z^s está à esquerda de z^t (ou seja, $z_2^s > z_2^t$). (2) Se o AD preferir z^s a z^t, impõe-se a restrição $f_2(x) > z_2^t$, se preferir z^t impõe-se a restrição $f_1(x) > z_1^s$ e, se elas forem indiferentes para o AD, impõem-se as restrições $f_1(x) \geq z_1^s$ e $f_2(x) > z_2^t$. Actualiza-se a solução preferida para a próxima iteração e regressa a (1).

(Nota: a redução da região admissível permite que sejam também alcançadas as soluções não suportadas através deste processo de cálculo).

II.3.1.2 Programação linear inteira multiobjectivo

MARCOTTE E SOLAND (1980, 1986)

O método de Marcotte e Soland é aplicável a problemas multiobjectivo em que o conjunto das soluções admissíveis é convexo ou discreto. Como os próprios autores referem, o método não se aplica ao caso inteiro-misto. A informação requerida do AD consiste na comparação de pares de soluções, num método que assume uma estrutura de preferências estável, mas que não pressupõe necessariamente a existência de uma função utilidade implícita. É um método que usa uma estrutura de *branch-and-bound*, em que cada nodo da árvore representa um subproblema multiobjectivo cuja região admissível é um subconjunto da região admissível original, obtido através de restrições nos objectivos. As soluções eficientes são geradas através da optimização de somas pesadas das funções objectivo dentro de cada subconjunto da região admissível (*grupo I*).

Seja N^j o nodo da árvore de *branch-and-bound* em análise associado a Z^j, subconjunto de Z (região admissível no espaço dos objectivos) e β^j o respectivo ponto ideal. O primeiro passo consiste em encontrar uma solução não dominada $z^j \in Z^j$. Para tal, optimiza-se em Z^j uma soma pesada das funções objectivos com pesos estritamente positivos. Se $z^j \neq \beta^j$, o nodo é ramificado criando-se tantos filhos quantas as componentes de z^j que são estritamente menores do que as de β^j (em geral k). O filho i de N^j herda a região admissível do pai, Z^j, com a restrição adicional $f_i(x) > z_i^j$ (esta restrição é operacionalizada de modo distinto no caso convexo e no caso discreto). Significa, portanto, que as regiões admissíveis dos filhos de N^j não são necessariamente disjuntas. As soluções ideais dos nodos funcionam como limites superiores para os respectivos ramos e o algoritmo pressupõe que elas sejam inseridas numa lista por ordem decrescente de preferência do AD. Esta lista define a ordem de selecção dos nodos para análise. Definindo a solução incumbente como a solução não dominada preferida pelo AD, de entre as obtidas até ao momento, um nodo não será analisado se a respectiva solução ideal não for preferida relativamente à incumbente. Desta forma, o algoritmo termina se $z^j = \beta^j$ ou a solução incumbente for preferida em relação à solução ideal à cabeça da lista de preferências, ou ainda, se a lista estiver vazia.

Os autores apresentam versões especializadas para os casos convexo e discreto (que inclui os problemas de PLIMO) demonstrando, para cada caso, que o algoritmo é finito. Apesar de os autores não explicarem a razão do método não ser aplicável ao caso inteiro-misto, podemos de

facto perceber que nenhuma das duas versões é adequada ao caso inteiro-misto. A título de curiosidade, tentaremos fazer uma breve ilustração deste facto, através de exemplos para os vários casos. O exemplo II.11 mostra a aplicação do método a um problema de PLMO (versão para o caso convexo); o exemplo II.12 mostra a aplicação do método a um problema de PLIMO (versão para o caso discreto); o exemplo II.13 ilustra que nenhuma das versões é adequada a um problema de PLIMMO. É nossa intenção alertar também para as eventuais dificuldades que podem decorrer de abordagens como esta, que usam somas pesadas das funções objectivo, quando aplicadas ao caso inteiro-misto[1].

Exemplo II.11. Seja o problema de PLMO ilustrado na figura II.13, em que z^1 e z^2 são as soluções não dominadas que optimizam individualmente cada uma das funções objectivo e definem $\beta^j = (z_1^1, z_2^2)$. Para calcular uma solução não dominada $z^j \in Z^j$, Marcotte e Soland propõem que se optimize uma soma pesada de z_1 e z_2 em Z^j, considerando as restrições adicionais $z_1 \geq \alpha_1$ e $z_2 \geq \alpha_2$, em que $\alpha_1 = (z_1^1 + z_1^2)/2$ e $\alpha_2 = (z_2^1 + z_2^2)/2$. Esta forma de proceder permite que seja encontrada uma solução "central" (ver figura II.13). Como as somas ponderadas das funções objectivo conduzem a soluções básicas da região admissível em causa, estas restrições são importantes para evitar que se determine novamente z^1 ou z^2. Na ramificação seguinte, Z^j será partido em 2 subconjuntos, um considerando $z_1 \geq z_1^j$ e outro com $z_2 \geq z_2^j$, mas apenas se $\beta_i^j - z_i^j \geq \delta_i$ (com δ_i>0 definido pelo AD) para i=1 e 2, respectivamente. Este processo de cálculo garante a existência de z^j, porque Z^j é convexo. Assegura ainda que a solução não dominada "central" que se irá obter para a partição *i* de Z^j melhora z_i de pelo menos δ_i / k, pelo que se conclui que a árvore de *branch-and-bound* é finita.

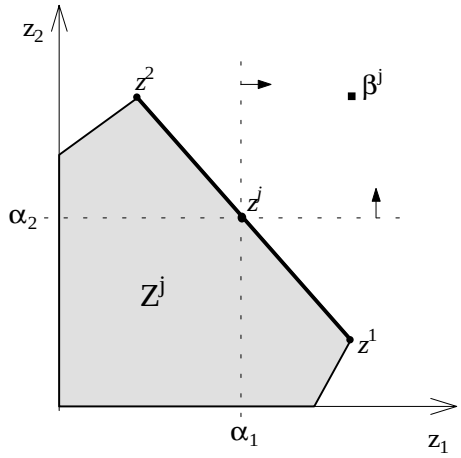


Fig. II.13 – Ilustração do método de Marcotte e Soland no caso convexo.

Exemplo II.12. Seja agora o exemplo de PLIMO ilustrado na figura II.14 em que $\beta^j = (z_1^1, z_2^2)$. Marcotte e Soland propõem o cálculo de uma solução não dominada $z^j \in Z^j$ através da optimização em Z^j de uma soma pesada das funções objectivo. Na ramificação

[1] Estes exemplos são da nossa autoria e, por isso, da nossa inteira responsabilidade.

seguinte, Z^j será partido em 2 subconjuntos, adicionando-se a restrição $z_1 \geq z_1^j + \delta_1$ ao primeiro e $z_2 \geq z_2^j + \delta_2$ ao segundo, com δ_1, δ_2 escalares positivos pequenos. Sendo {z^1, z^2, z^3, z^4} o conjunto das soluções não dominadas de Z^j, então z^j seria uma das soluções z^1, z^2 ou z^3 (já que z^4 é não suportada). Suponhamos, por hipótese, que $z^j = z^3$. Z^j é então partido em dois subconjuntos, um com $z_1 \geq z_1^3 + \delta_1$ (subconjunto 1) e que contém z^1 e z^4, e outro em que $z_2 \geq z_2^3 + \delta_2$ (subconjunto 2) e que contém z^2. Desta forma, como a partição *i* só existe se $z_i^j < \beta_i^j$, cada partição (a que corresponde um nodo da árvore de *branch-and-bound*) contém sempre pelo menos uma solução não dominada.

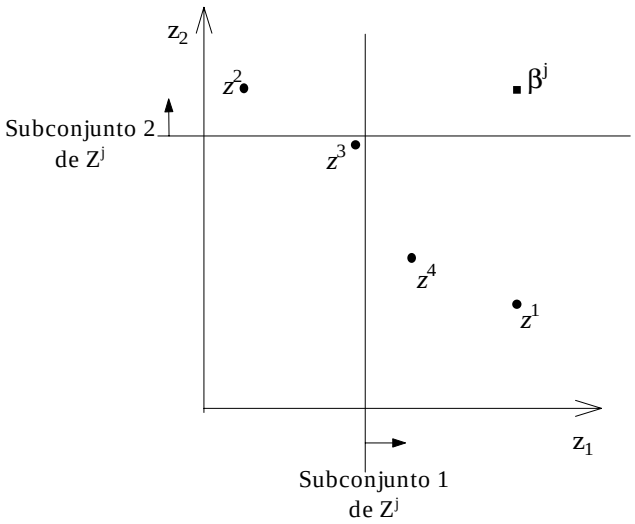


Fig. II.14 – Ilustração do método de Marcotte e Soland no caso discreto.

Exemplo II.13. Vejamos agora o resultado da aplicação separada das versões do método de Marcotte e Soland para os casos convexo e discreto ao exemplo de PLIMMO ilustrado na figura II.15. As soluções não dominadas de Z^j são $[z^1,z^3] \cup]z^4,z^2]$ e a solução ideal é definida por $\beta^j = (z_1^1, z_2^2)$. Relativamente ao cálculo de uma solução não dominada $z^j \in Z^j$:

- a versão para o caso convexo, aplicável a PLMO, implica a consideração de $z_1 \geq \alpha_1$ e $z_2 \geq \alpha_2$, o que define neste caso uma região admissível vazia (ver figura II.15);
- na versão para o caso discreto, aplicável a PLIMO, z^j seria igual a z^1, z^2 ou z^3. Suponhamos, por hipótese, que $z^j = z^3$. Então Z^j será partido em dois subconjuntos, um considerando $z_1 \geq z_1^3 + \delta_1$ (subconjunto 1) e outro considerando $z_2 \geq z_2^3 + \delta_2$ (subconjunto 2). Mas, cada um destes subconjuntos é convexo, onde se verifica que esta versão para o caso discreto já não é adequada. Vejamos porquê: tomemos, por exemplo, o subconjunto 1 definido por $[z^1,z^{3'}]$, em que $z^{3'}$ está muito próxima de z^3 porque $z_1^{3'} = z_1^3 + \delta_1$ com δ_1 positivo pequeno. Ao optimizar uma qualquer soma pesada das funções objectivo neste subconjunto o resultado seria z^1 ou $z^{3'}$. Qualquer uma destas soluções tem uma componente igual à da solução ideal deste subconjunto. Assim, na ramificação seguinte, o subconjunto 1 criaria apenas 1 filho com o subconjunto $[z^1,z^{3''}]$ ($z^{3''}$ muito próxima de $z^{3'}$) ou $[z^{1'},z^{3'}]$ ($z^{1'}$ muito próxima de z^1). O processo continuaria de modo semelhante, tendo como consequência uma sucessiva e extensa descendência de nodos que apresentariam ao AD soluções muito próximas umas das outras.

Em suma, nenhuma destas versões é adequada a problemas de PLIMMO.

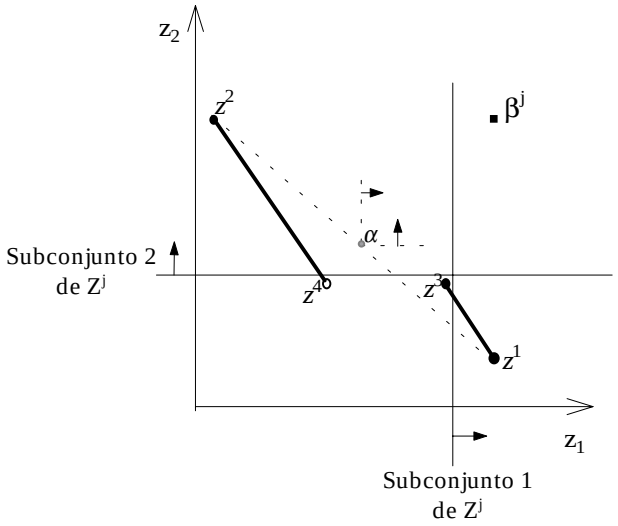


Fig. II.15 – Ilustração das dificuldades do método de Marcotte e Soland em PLIMMO.

WHITE (1985)

O método de White baseia-se no *branch-and-bound* interactivo de Marcotte e Soland (versão publicada em 1980 e revista em 1986) e numa extensão da relaxação Lagrangeana de problemas escalares para problemas vectoriais. Aplica-se, tal como o antecessor, a problemas de programação linear inteira multiobjectivo. O objectivo principal do uso das técnicas Lagrangeanas é encontrar limites para as funções objectivo que ajudem a eliminar soluções 'não-óptimas' para a *função utilidade* implícita (monótona crescente).

No método de *branch-and-bound* de Marcotte e Soland a solução não dominada incumbente é comparada com o ponto ideal de cada nodo da árvore ao qual está associado um subconjunto da região admissível. O nodo não será analisado mais detalhadamente, nem ramificado, se a solução incumbente for preferida à solução ideal desse nodo. A solução ideal fornece, assim, limites superiores para os valores dos objectivos naquele ramo da árvore. O método Lagrangeano tem como intenção estreitar estes limites de modo a que se eliminem mais facilmente partes da árvore que possam não ser interessantes para o AD por não conduzirem a soluções 'óptimas'.

Herdando todo o modo de operação do método de Marcotte e Soland, o método de White enquadra-se no *grupo I* no que diz respeito ao processo de cálculo de soluções eficientes.

GONZALEZ ET AL. (1985)

Este trabalho apresenta um procedimento interactivo para problemas de PLIMO. A informação de preferências do AD é extraída, em cada interacção, a partir da indicação da solução menos preferida de um conjunto reduzido de soluções eficientes candidatas. Assume-se que existe uma *função utilidade* implícita do AD.

O procedimento divide-se em duas fases. A primeira fase calcula apenas soluções eficientes suportadas optimizando, para o efeito, somas pesadas das funções objectivo. A segunda fase calcula soluções eficientes não suportadas através da abordagem das somas pesadas das funções objectivo com restrições auxiliares que eliminam soluções já conhecidas (*grupo I*).

A pesquisa inicia-se com o cálculo das k soluções não dominadas que maximizam individualmente cada função objectivo, constituindo o conjunto N^*. Forma-se então o hiperplano que passa pelos k pontos de N^* e, a partir deste, definem-se pesos para a soma pesada das funções objectivo cujo gradiente é perpendicular ao hiperplano (se os pesos não forem todos positivos faz-se uma perturbação). Determina-se a solução não dominada z^j que optimiza essa soma pesada. Se $z^j \notin N^*$ e for preferida pelo AD relativamente a algum elemento de N^*, então z^j irá substituir em N^* a solução menos preferida, e o processo de cálculo repete-se; caso contrário, termina a 1ª fase.

Para o cálculo de soluções eficientes não suportadas da 2ª fase, optimiza-se uma função pesada das funções objectivo, $F(x)$, cujo gradiente é perpendicular ao hiperplano que passa pelos k pontos de N^*, considerando a restrição adicional $F(x) \leq P$ (P é obtido pela subtracção de uma quantidade fraccionária à constante do hiperplano de suporte). O processo de cálculo de soluções eficientes não suportadas pode continuar reduzindo-se, para o efeito, o valor de P de modo a tornar inadmissíveis os pontos já gerados. O processo termina quando o AD o desejar ou a solução encontrada for dominada por alguma outra solução já calculada.

Nota: Se a 1ª fase terminar com $z^j \notin N^*$ não preferida pelo AD relativamente a qualquer elemento de N^* , então a 2ª fase concentra-se na pesquisa de soluções não suportadas próximas da preferida de N^*, ou seja $F(x)$ será a função que gerou essa solução de N^*.

GABBANI E MAGAZINE (1986)

Gabbani e Magazine propõem, neste artigo, uma abordagem interactiva para problemas de PLIMO em que as soluções eficientes são calculadas através de uma heurística. O algoritmo é uma adaptação do método de contracção do cone dos critérios (ou redução do conjunto dos pesos) desenvolvido em 1977 por Steuer para problemas de PLMO (este método encontra-se também descrito em STEUER, 1986). Assume-se que o AD tem uma *função utilidade linear* implícita.

O algoritmo de Steuer consiste basicamente no seguinte: (i) selecciona automaticamente $2k+1$ vectores de pesos w uniformemente distribuídos no conjunto dos pesos daquela iteração, inicialmente igual a $W = \left\{ w \in \Re^k \;\middle|\; w_i > 0, \sum_{i=1}^{k} w_i = 1 \right\}$; (ii) resolve os $2k+1$ problemas das somas pesadas das funções objectivo; (iii) pede ao AD para seleccionar a solução preferida de entre as

calculadas e o conjunto dos pesos é reduzido em torno do vector de pesos que conduziu à solução escolhida. Este é um processo iterativo onde se repetem os passos de (i) a (iii) até se verificar alguma das condições de paragem do algoritmo. Na proposta de Gabbani e Magazine para PLIMO, os problemas de programação inteira de (ii) são resolvidos de forma heurística, numa tentativa de diminuir o esforço computacional envolvido. Não há, contudo, qualquer garantia de que as soluções obtidas sejam óptimas para a respectiva soma pesada dos objectivos, e portanto eficientes do problema multiobjectivo. Além disso, trata-se de uma abordagem de somas pesadas simples pelo que não calcula soluções eficientes não suportadas. A experiência relatada com este método limitou-se a problemas 0-1 de mochila (*knapsack*) multidimensional para os quais se usou uma heurística específica.

II.3.1.3 Programação linear inteira-mista biobjectivo

WALKER (1978)

O método de Walker é dedicado a problemas de programação matemática linear e não linear biobjectivo. Assume uma *função utilidade* implícita do AD para a qual pretende encontrar a respectiva solução 'óptima'. Do AD são requeridas apenas respostas do tipo "sim/não" no que diz respeito ao interesse em melhorar o valor de uma função objectivo. O processo de cálculo consiste na optimização de somas pesadas das funções objectivo, não alcançando portanto soluções eficientes não suportadas. Considerando o parâmetro w para formar a função escalar $f_1(x) + wf_2(x)$, o método começa por calcular as soluções que maximizam esta função para $w^0=0$ e $w^1=M$ (número positivo muito grande). Segue-se um processo iterativo de ajuste de w^0 e w^1, até determinar $w^*=w^0=w^1$ a que o autor chama de valor óptimo do parâmetro w. O ajuste de w^0 e w^1 é feito com base nas respostas do AD face à sua satisfação com os valores de $f_1(x)$ e $f_2(x)$ nas soluções que lhe vão sendo apresentadas.

AKSOY (1990)

O método interactivo de Aksoy destina-se a problemas inteiros-mistos biobjectivo (incluindo o caso não linear) e emprega um esquema de *branch-and-bound* para dividir o (sub)conjunto das soluções não dominadas associado a cada nodo da árvore em dois subconjuntos disjuntos. O processo de ramificação procura fazer a bissecção do intervalo de valores não dominados de z_2 no nodo sob exploração, testando em primeiro lugar se existe um ponto não dominado cujo valor de z_2 esteja no centro do intervalo. Ou seja, se o intervalo para z_2 no nodo N^j for $[l_2^j, u_2^j]$, verifica-se se existe alguma solução não dominada tal que $z_2 = (l_2^j + u_2^j)/2$. Se esta solução existir, ela é usada para dividir em dois o subconjunto não dominado de N^j. Caso contrário, o

subconjunto será dividido usando dois pontos não dominados cujos valores de z_2 estejam mais próximos (um para cima e outro para baixo) do valor médio. Para o cálculo destas soluções é usado um processo lexicográfico, em que se optimiza uma das funções objectivo de cada vez, limitando a outra. Assim, o processo de cálculo pode ser visto como a optimização de somas pesadas das funções objectivo, usando apenas os vectores de pesos (1,0) e (0,1) e impondo restrições adicionais nos valores das funções objectivo (*grupo I*). A forma lexicográfica destina-se a garantir que se obtêm soluções não dominadas, já que se usam pesos nulos.

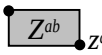
O processo interactivo requer que o AD faça comparações par a par de soluções não dominadas de modo a determinar o nodo que se irá ramificar a seguir, e que o AD escolha/actualize a solução incumbente – solução não dominada preferida de entre as já conhecidas. Assume-se que as preferências do AD são coerentes, transitivas e invariantes durante todo o processo, tendo como intenção optimizar uma *função utilidade* implícita do AD.

SOLANKI (1991)

O método de Solanki procura gerar um subconjunto representativo, ou seja, bem distribuído, de soluções não dominadas de problemas de programação linear inteira-mista biobjectivo. O algoritmo proposto combina a função da métrica pesada de Tchebycheff com os princípios básicos do método NISE (COHON ET AL., 1979) para problemas lineares biobjectivo. O método NISE constrói uma aproximação da fronteira não dominada através do cálculo sucessivo de soluções que optimizam somas pesadas das funções objectivo. Nesse método, um par de soluções candidatas a adjacentes é considerado uma boa aproximação da região não dominada intermédia se a distância das combinações convexas desse par à solução ideal dessa região for inferior a uma quantidade pré-estabelecida. É esta distância que dá uma medida do *erro* da aproximação. Se, por outro lado, a distância não for aceitável, então calcula-se uma nova solução não dominada intermédia.

A não convexidade da região admissível em problemas inteiros e inteiros-mistos faz com que a medida do *erro* usada no NISE deixe de ser válida. Por outro lado, o processo de cálculo usado no NISE não permite capturar soluções não suportadas. Estes factos levaram Solanki a adoptar o programa escalarizante da métrica pesada e aumentada de Tchebycheff e a modificar a medida do *erro*. Este método insere-se, pois, no *grupo II* no que diz respeito ao processo de cálculo de soluções não dominadas.

Conhecido um par de soluções não dominadas (z^a, z^b), $z_1^a > z_1^b$, candidatas a adjacentes (porque não se conhece nenhuma solução intermédia), o *erro* associado a esse par pode ser medido por $\max\left\{\left(z_1^a - z_1^b\right)/R_1, \left(z_2^b - z_2^a\right)/R_2\right\}$ onde R_i, i=1,2, é a diferença entre o máximo e o mínimo de z_i no conjunto não dominado e tem como função a normalização dos valores.

Solanki propõe uma medida do *erro* mais aproximada nos casos em que se consegue inferir alguma sub-região inadmissível dentro da região Z^{ab} – rectângulo definido por z^b no canto superior esquerdo e z^a no canto inferior direito: 

. Se o *erro* é superior ao *erro máximo* permitido, então calcula-se uma nova solução não dominada, seja z^c, através do programa escalarizante da métrica pesada e aumentada de Tchebycheff, com um vector de pesos e um ponto de referência particulares que asseguram que z^c pertence à região Z^{ab}. Se $z^c = z^a$ ou $z^c = z^b$, significa que não existe outra solução não dominada entre estas duas e o *erro* associado a este par é nulo. Caso contrário, o par anterior é substituído (z^a,z^c) e (z^c,z^b). A aproximação do conjunto não dominado é progressivamente melhorada, diminuindo-se os *erros* associados aos pares. Em cada iteração, o método escolhe para análise o par de soluções com maior *erro*.

O método proposto por Solanki não é interactivo porque o AD apenas tem de especificar inicialmente o *erro máximo* permitido, o que define a condição de paragem do algoritmo. Mas, na nossa opinião, ele poderia facilmente ser enquadrado num protocolo interactivo. Para tal, poderia ser o AD a escolher o par de soluções a analisar em cada iteração, e a decidir interactivamente o fim ou continuação do algoritmo, não necessitando de escolher *a priori* o *erro máximo*.

FERREIRA ET AL. (1996), FERREIRA (1997)

Trata-se de um método interactivo para problemas de programação linear inteira-mista biobjectivo que é orientado para a aprendizagem progressiva e selectiva do conjunto das soluções não dominadas. Assume um protocolo de *comunicação aberta* com o AD.

Esta abordagem começa por determinar as soluções não dominadas que optimizam individualmente cada uma das funções objectivo. Em cada interacção o AD é chamado a indicar a região que pretende pesquisar através da selecção de um par de soluções não dominadas candidatas a adjacentes, ou especificando directamente limites inferiores para os valores das funções objectivo. Em qualquer um dos casos, optimiza-se uma soma pesada das funções objectivo na região indicada pelo AD, que é reduzida relativamente à região admissível original por restrições nos valores das funções objectivo (*grupo I*). O conhecimento de novas soluções não dominadas permite eliminar progressivamente regiões do espaço dos objectivos, quer por dominância quer por inadmissibilidade. FERREIRA (1997) propõe ainda o uso da métrica pesada de Tchebycheff, em alternativa às somas pesadas, para calcular uma solução não dominada dentro de uma dada região (à semelhança do que acontece no método de SOLANKI, 1991). Esta técnica tem como principal vantagem permitir que regiões mais alargadas sejam eliminadas por inadmissibilidade.

II.3.1.4 Programação linear inteira-mista multiobjectivo

VILLAREAL ET AL. (1980), KARWAN ET AL. (1985), RAMESH ET AL. (1986)

VILLAREAL ET AL. (1980) desenvolveram um método interactivo para problemas de PLIMMO que utiliza uma estrutura de *branch-and-bound*. Este método foi posteriormente melhorado por KARWAN ET AL (1981) e RAMESH ET AL. (1986). Tal como o método bicritério de RAMESH ET AL. (1990), apresentado atrás, estas versões são extensões do método de Zionts-Wallenius de PLMO. Começando por aplicar o método de Zionts-Wallenius à relaxação linear do problema de PLIMMO, o método procede depois a uma fase de *branch-and-bound* que termina quando for encontrada uma solução inteira que satisfaça as preferências do AD. Assume-se a existência de uma *função utilidade* implícita e as preferências do AD são capturadas através de comparações de pares de soluções e avaliação de *vectores de compensação* (ver descrição anterior do método de RAMESH ET AL., 1990). À luz da função utilidade subjacente, são tomadas sucessivas decisões no sentido de *(i)* aplicar novamente o método de Zionts-Wallenius à relaxação linear de um subproblema multiobjectivo associado a um nodo da árvore de *branch-and-bound* ou *(ii)* continuar a ramificar através da imposição de limitações numa variável que viole a condição de integralidade. A geração de soluções dentro de cada nodo é feita no contexto do método de Zionts-Wallenius, pelo que são usadas somas pesadas das funções objectivo (*grupo I*).

STEUER E CHOO (1983), STEUER (1986)

STEUER E CHOO (1983) propuseram neste trabalho – também descrito em STEUER(1986) – um método interactivo genérico para problemas de programação matemática multiobjectivo, o que inclui os casos de PLIMO e de PLIMMO. O método assume uma *função utilidade* implícita do AD sem restrições particulares na forma.

A estratégia do procedimento interactivo consiste na amostragem de subconjuntos progressivamente mais concentrados de soluções não dominadas. Em cada interacção, o AD selecciona a solução que prefere de entre uma amostra, de tamanho pré-definido, de soluções não dominadas. Estas soluções são obtidas pela optimização do programa escalarizante da métrica pesada e aumentada de Tchebycheff $P_{\lambda}^{\infty,\rho}$ (ou pela abordagem lexicográfica se o problema for não linear). O ponto de referência é a solução ideal estendida, z^{**}, e são usados vectores de pesos, λ, dispersos no conjunto dos pesos de cada iteração. O conjunto dos pesos inicial é Λ^0. A solução preferida pelo AD em cada interacção é usada para definir um vector de pesos em torno do qual se concentra e reduz o conjunto dos pesos para a iteração seguinte. A redução do conjunto dos pesos depende de um factor de convergência definido *a priori*. O procedimento termina após um número pré-definido de iterações.

DURSO (1992)

O método apresentado por Durso é uma modificação do método de *branch-and-bound* de MARCOTTE E SOLAND (1986). Esta nova versão aplica-se tanto a problemas de PLIMO como a problemas de PLIMMO, o que é conseguido pela alteração da técnica de cálculo das soluções não dominadas, a principal diferença entre os dois algoritmos. Enquanto que o método de Marcotte e Soland usa somas pesadas das funções objectivo (em sub-regiões delimitadas por restrições adicionais nos valores das funções objectivo), o método de Durso usa a métrica pesada e aumentada de Tchebycheff (nas mesmas sub-regiões) com pesos todos iguais para que as soluções sejam centrais (enquadrando-se no *grupo II*).

Para cada nodo N^j da árvore de *branch-and-bound* são primeiro calculadas as k soluções não dominadas que definem o respectivo ponto ideal, β^j. O AD indica, em cada interacção, o nodo a analisar através da selecção do ponto ideal preferido (não sendo, portanto, requerida uma ordenação prévia dos nodos segundo preferência dos pontos ideais como no método de Marcotte e Soland). O processo de análise de um nodo começa por resolver o programa da métrica aumentada de Tchebycheff para determinar uma solução não dominada 'central' da região admissível associada a esse nodo. O AD deve então escolher a solução preferida, $\hat{z}$, de entre as $k+1$ (as k primeiras mais a 'central'); esta escolha é semelhante à que se efectua no método de Marcotte e Soland. São criados tantos filhos (no máximo k) quantas as componentes i de $\hat{z}$ que são menores do que as de β^j de pelo menos uma quantidade δ_i (dada pelo AD). O filho que restringe a i-ésima função objectivo acrescenta a restrição $f_i(x) \geq \hat{z}_i + \delta$ (com δ um escalar positivo pequeno) à região admissível herdada do pai.

Esta abordagem interactiva pode ser vista como um procedimento de *comunicação aberta* que termina quando o AD tiver encontrado uma solução de compromisso satisfatória.

KARAIVANOVA ET AL. (1993)

A abordagem proposta neste artigo é dedicada a problemas de PLIMO e de PLIMMO. Trata-se de uma adaptação do algoritmo de STEUER E CHOO (1983), mas em que os programas escalarizantes de Tchebycheff são resolvidos heuristicamente, não havendo portanto a garantia de que as soluções obtidas sejam não dominadas. São apresentados resultados computacionais com problemas gerados aleatoriamente tendo em vista a comparação do método proposto com o de Steuer e Choo, ambos implementados num computador pessoal equivalente a um IBM/XT com coprocessador. Foi definida uma função utilidade para cada problema e registaram-se os tempos médios requeridos para resolver os problemas. Observou-se que em problemas pequenos (3 ou 6 variáveis inteiras, 15 ou 30 restrições, 3 ou 6 objectivos) o procedimento exacto de Steuer e Choo era mais rápido, o que já não acontecia em problemas maiores. Nos três

problemas com maior número de variáveis – 30 variáveis inteiras, 3 restrições e 3 objectivos – o tempo médio do procedimento heurístico foi de 4 h e 47 min, ao passo que 8h não foram suficientes para que o método de Steuer e Choo conseguisse resolver algum desses problemas.

II.3.1.5 Métodos dedicados a PLIMO que também se aplicam a PLIMMO

Alguns investigadores desenvolveram recentemente outros métodos interactivos para problemas de PLIMO que são também aplicáveis a PLIMMO. Neste tipo de abordagens incluem-se os métodos de VASSILEV E NARULA (1993), NARULA E VASSILEV(1994) e KARAIVANOVA ET AL. (1995). Na nossa opinião, são procedimentos de *comunicação aberta* com o AD que partilham algumas características centrais, como a projecção de pontos de referência no conjunto das soluções não dominadas (apesar de ser operacionalizada de diferentes formas) e o tipo de informação de preferências requerida do AD. Esta informação reside, fundamentalmente, na especificação de níveis de aspiração (pontos de referência) e níveis de reserva para os valores das funções objectivo, inserindo-se no *grupo II* quanto ao processo de geração de soluções não dominadas. Estas abordagens reflectem a tendência actual de uma preocupação crescente em não exigir demasiado do AD, procurando também diminuir o esforço computacional. Tendo como preocupação reduzir o número de programas inteiros (mistos) monocritério a resolver, os autores desenvolveram métodos que optimizam apenas um programa escalarizante em cada interacção. Além disso, algumas destas abordagens são '*contínuas-inteiras*' (NARULA E VASSILEV, 1994 e KARAIVANOVA ET AL., 1995); trabalham a maior parte do tempo com soluções contínuas da relaxação linear do problema e, sempre que o AD considere interessante uma dada solução contínua, o algoritmo determina em seguida a solução inteira não dominada que lhe está mais próxima (de acordo com a métrica de Tchebycheff).

São estes métodos que passamos a descrever com algum detalhe.

VASSILEV E NARULA (1993), NARULA E VASSILEV (1994)

O método de VASSILEV E NARULA (1993) opera do seguinte modo:

(i) Calcula uma solução não dominada inicial $\hat{z}$. Por uma questão de simplicidade, a solução inicial é obtida pela projecção do ponto de referência de componentes nulas no conjunto não dominado, i.e. $\max\left\{\alpha+\rho\sum_{i=1}^{k} f_i(x) \,\middle|\, f_i(x)\geq\alpha, i=1,...,k,\ x\in X\right\}$ (equivalente ao programa *min-max* $P^{\rho}_{q,\lambda}$ com q um vector de 0's e λ um vector de 1's).

(ii) Se o AD considerar $\hat{z}$ satisfatória, termina; caso contrário, o AD deve especificar um novo ponto de referência q de componentes superiores a $\hat{z}$ nas funções objectivo que pretende melhorar (índices i que constituirão o conjunto H), componentes inferiores a $\hat{z}$ nos objectivos

que está disposto a piorar ($i \in$ L) e componentes iguais a $\hat{z}$ nos objectivos que gostaria que se mantivessem iguais ($i \in$ E).

(iii) Baseado nos valores de q e da última solução $\hat{z}$, é definido o programa escalarizante seguinte que determina uma nova solução, pelo menos fracamente não dominada; atribui essa solução a $\hat{z}$ e regressa a (ii).

$$\begin{aligned}
\max \quad & \alpha \\
\text{s.a:} \quad & f_i(x) - (q_i - \hat{z}_i)\alpha \geq \hat{z}_i && i \in \text{H} \quad (\text{em que } q_i > \hat{z}_i) \\
& f_i(x) \geq q_i - \theta(\hat{z}_i - q_i) && i \in \text{L} \quad (\text{em que } q_i < \hat{z}_i) \\
& f_i(x) = \hat{z}_i && i \in \text{E} \quad (\text{em que } q_i = \hat{z}_i) \\
& x \in X \\
& \alpha \geq 0
\end{aligned}$$

com $\theta \geq 0$ (constante).

É proposta também uma alteração deste programa que garante que a solução obtida é não dominada. Os autores observam que, se o AD especificar q tal que $q_i > \hat{z}_i$ para todo o i, o resultado do programa anterior é novamente $\hat{z}$.

NARULA E VASSILEV (1994) propõem uma alteração deste algoritmo no sentido de diminuir o esforço computacional. Assim, em vez de determinar soluções inteiras nas fases de cálculo (i) e (iii), são calculadas uma ou mais soluções contínuas – soluções não dominadas da relaxação linear do problema multiobjectivo – que são apresentadas ao AD. No passo (iii), o problema escalarizante pode ser resolvido para $\theta = 0$ e para outros valores paramétricos de θ usando um algoritmo 'standard' de programação linear paramétrica. O AD pode continuar a pesquisa de soluções contínuas ou requerer o cálculo da solução não dominada inteira mais próxima – no sentido *min-max* (ou de Tchebycheff) – de uma solução contínua que considere interessante.

KARAIVANOVA ET AL. (1995)

Neste trabalho são propostos dois métodos baseados na projecção de pontos de referência no conjunto não dominado, em que, o primeiro opera apenas com soluções inteiras, e o segundo com soluções contínuas e inteiras.

A filosofia subjacente ao primeiro método é próxima da de VASSILEV E NARULA (1993) mas é implementada de forma diferente. O programa escalarizante usado por VASSILEV E NARULA (1993) maximiza a menor diferença estandardizada à última solução, nos valores dos objectivos que o AD pretende melhorar. Por seu lado, KARAIVANOVA ET AL. (1995) minimiza a maior diferença estandardizada a um ponto de referência, nos valores dos mesmos objectivos. Os níveis de referência são usados explicitamente como limites inferiores para os outros objectivos.

Se $\hat{z}$ designar a última solução calculada e q for um ponto de referência dado pelo AD, em que apenas as componentes de índices $i \in \mathrm{H}$ são superiores a $\hat{z}$, então o programa escalarizante que se resolve é o seguinte:

$$\begin{aligned} \min \quad & \alpha \\ \text{s.a:} \quad & f_i(x) + (q_i - \hat{z}_i)\alpha \geq q_i \qquad i \in \mathrm{H} \\ & f_i(x) \geq q_i \qquad i \notin \mathrm{H} \\ & x \in X \\ & \alpha \geq 0 \end{aligned}$$

Este programa calcula uma solução pelo menos fracamente não dominada que é apresentada ao AD. Se o AD considerar a solução não satisfatória, então deve indicar um novo ponto de referência e o processo repete-se. Como este processo opera apenas com soluções inteiras, os autores designaram-no por método 'inteiro puro'.

O segundo método tem como intenção fornecer mais informação ao AD, com maior eficácia computacional, uma preocupação já manifestada por NARULA E VASSILEV (1994). Os autores designaram-no por método 'contínuo-inteiro'. Propõe-se viajar na fronteira não dominada contínua (soluções não dominadas da relaxação linear do problema multiobjectivo) usando o conhecido método 'Pareto Race' (KORHONEN E WALLENIUS, 1988) para problemas de PLMO. O 'Pareto Race' determina sucessivas soluções contínuas através da projecção de pontos de uma direcção de referência $q + t\Delta d$, em que $t \geq 0$ é um parâmetro e $\Delta d \in \Re^k$ é um vector direcção. Quando o AD encontrar uma solução satisfatória para o problema contínuo, então o método calcula a solução não dominada inteira mais próxima (no sentido de Tchebycheff) dessa solução contínua. O cálculo faz-se através da resolução do programa *min-max* $P_{q,\lambda}$ (ou $P^{\rho}_{q,\lambda}$) com q o ponto dos objectivos da solução contínua.

Como os próprios autores referem, o método 'contínuo-inteiro' é computacionalmente mais eficaz que o método 'inteiro puro', mas tem o inconveniente de o AD operar a maior parte do tempo no espaço contínuo, o que pode não ser satisfatório. São apresentados resultados de testes do método 'inteiro puro' num computador pessoal AT486. Foram gerados 10 problemas aleatórios para cada combinação de 2, 4 ou 6 objectivos, 15 ou 30 restrições e 15 ou 30 variáveis inteiras. Foi para os problemas com 4 objectivos, 15 restrições e 15 variáveis que se registou o menor dos tempos médios (52 seg para 3 iterações). No que diz respeito ao maior dos tempos médios, este foi alcançado nos problemas com 2 objectivos, 30 restrições e 30 variáveis (3 h e 34 min para 3 iterações).

II.3.1.6 Breve referência a outros métodos

Além dos métodos interactivos aqui apresentados, existem ainda outras abordagens interactivas baseadas em pontos de referência que podem ser aplicadas a problemas de PLIMO ou de PLIMMO, desde que sejam incorporados algoritmos adequados à resolução dos respectivos programas escalarizantes inteiros (ou inteiros mistos) monocritério.

Podemos começar por referir o método STEM, desenvolvido por BENAYOUN ET AL. (1971) para problemas de PLMO. Este método usa uma estratégia de redução da região admissível através da imposição de limitações adicionais nos valores das funções objectivo. Estas limitações são incluídas no programa escalarizante da métrica pesada de Tchebycheff que determina, em cada iteração, a solução da região reduzida que está mais próxima do ponto ideal. Tendo em conta os posteriores resultados teóricos sobre o uso da métrica de Tchebycheff (e extensões) em problemas com variáveis inteiras, a extensão do STEM a problemas de PLIMO ou de PLIMMO é possível sem grandes dificuldades. No artigo de revisão de TEGHEM E KUNSCH (1986[B]) os autores reconhecem as potencialidades de abordagens do tipo STEM, alegando a possibilidade de se manter, a níveis razoáveis, tanto o esforço numérico como aquele que é requerido por parte do AD. Podemos, de resto, considerar uma abordagem do tipo STEM o primeiro método proposto por KARAIVANOVA ET AL. (1995).

No trabalho de L'HOIR E TEGHEM (1995), apesar de o problema abordado ser de PLMO, os autores referem que o método interactivo usado – chamado MOMIX – foi desenvolvido especialmente para problemas de PLIMMO. Após o cálculo de uma primeira solução não dominada através do método STEM, as fases interactivas do MOMIX são integradas numa árvore de *branch-and-bound* que inclui duas etapas. A 1ª etapa consiste numa pesquisa em profundidade e tem como intenção determinar uma boa solução de compromisso. A 2ª etapa é um procedimento de retrocesso para confirmar o grau de satisfação do AD ou encontrar uma solução de compromisso melhor. Assim, a característica principal do MOMIX é o uso de um *branch-and-bound* interactivo cuja filosofia foi previamente introduzida por MARCOTTE E SOLAND (1986). A região admissível é reduzida de um nodo da árvore para um seu filho através de limitações nos valores das funções objectivo. Dentro da região admissível associada a cada nodo é calculada uma solução não dominada como no método STEM.

Também a abordagem de STEUER ET AL. (1993) é extensível a problemas de PLIMO e de PLIMMO. Propõem uma combinação do método de STEUER E CHOO (1983) com um outro método a que os autores chamam método de 'Vectores de Aspiração'. Sendo o primeiro método "conduzido pelo algoritmo" – cuja filosofia provavelmente tem maior utilidade nas primeiras iterações porque permite uma amostragem de soluções não dominadas dispersas – o segundo é "orientado por aspirações" – o que poderá ser mais útil nas últimas iterações em que o AD

pretende encontrar uma solução de compromisso final. O programa escalarizante usado no método de 'Vectores de Aspiração' consiste na minimização da distância pesada e aumentada de Tchebycheff à solução ideal (como em STEUER E CHOO, 1983), mas o vector de pesos é determinado a partir de um vector de aspirações para os critérios escolhido pelo AD.

As métodos referidos são apenas alguns exemplos de abordagens interactivas cujas metodologias são suficientemente genéricas para que possam ser aplicadas a problemas de PLIMO ou de PLIMMO, apesar de terem sido desenvolvidos, implementados ou testados, para problemas de PLMO.

Existem ainda sistemas computacionais com metodologias genéricas, mas que se aplicam apenas a problemas particulares de PLIMO/PLIMMO porque utilizam algoritmos específicos para resolver os programas escalarizantes monocritério. Estes algoritmos tiram partido das estruturas especiais dos problemas, como é o caso, por exemplo, do sistema DINAS ('Dynamic Interactive Network Analysis System') que trata problemas multiobjectivo de 'transbordo' com localização de serviços (OGRYCZAK ET AL., 1989, 1992).

DINAS insere-se na família de sistemas DIDAS ('Dynamic Interactive Decision Analysis & Support') desenvolvidos na década de 80 por um grupo de investigadores para tratar diferentes tipos de problemas. LEWANDOWSKI ET AL. (1989) apresentam os princípios metodológicos, teoria matemática e variantes de implementação das várias aplicações dos sistemas da família DIDAS. Os sistemas DIDAS baseiam-se em procedimentos interactivos de aspirações para os objectivos (pontos de referência). Em particular, o DINAS é um procedimento interactivo que o AD controla através de dois vectores de parâmetros: níveis de aspiração q^a e níveis de reserva q^r. Em cada iteração resolve-se o programa escalarizante seguinte:

$$\min \quad \max_{j=1,\ldots,k} u_j(z, q^a, q^r) + \frac{\rho}{k} \sum_{j=1}^{k} u_j(z, q^a, q^r)$$

$$\text{s.a:} \quad z = Cx, \; x \in X$$

em que ρ é um número positivo arbitrariamente pequeno e u_j é uma função que mede o desvio do resultado face às expectativas do AD para o objectivo j. A função u_j usada em DINAS é a de WIERZBICKI (1986). Nesta função, os níveis de reserva q^r funcionam como limites 'soft', uma vez que não é obrigatório que a solução obtida seja superior a q^r. No entanto, há uma penalização muito elevada para resultados piores que os níveis de reserva. Os programas escalarizantes são problemas de programação linear inteira-mista com uma estrutura especial herdada do problema multiobjectivo de transbordo–localização. Consequentemente, o algoritmo usado para os resolver, designado por TRANSLOC, é um *branch-and-bound* que tira partido dessa estrutura especial.

II.3.2 Comentários aos métodos anteriores e enquadramento das novas contribuições

Do estudo de revisão de métodos interactivos para problemas de PLIMO e de PLIMMO que acabámos de apresentar podemos tirar algumas ilações, permitindo-nos fazer algum juízo crítico acerca da investigação existente nesta área.

Os método interactivos pretendem ultrapassar as principais dificuldades dos métodos geradores, nomeadamente ao nível do esforço computacional envolvido e da sobrecarga cognitiva do AD. Todavia, muitas das abordagens propostas requerem ainda um considerável esforço computacional que, pelo tempo despendido, pode comprometer a interactividade com o AD. Outras exigem demasiada intervenção do AD, ou ainda vêm a sua aplicabilidade limitada a problemas biobjectivo ou inteiros puros. Alguns métodos não são 'completos' no sentido de poder gerar qualquer solução eficiente, o que, segundo LEWANDOWSKI E WIERZBICKI (1988) é uma das três propriedades mais importantes que um método deve ter. As outras propriedades são a 'controlabilidade' e a 'simplicidade computacional'.

Nos parágrafos seguintes tentaremos particularizar estas questões mencionando os métodos em que se colocam algumas destas dificuldades ou limitações.

O esforço computacional é muito elevado nos métodos que requerem a resolução independente de vários programas inteiros (ou inteiros mistos) em cada iteração ou interacção. É o que acontece, por exemplo, nos métodos de STEUER E CHOO (1983), MARCOTTE E SOLAND (1986) e DURSO (1992). No primeiro, resolvem-se $2k$ programas escalarizantes em cada iteração e, nos outros dois, resolvem-se $k+1$ programas em cada nodo que se analisa da árvore de *branch-and-bound* (destinando-se os k primeiros programas à determinação da solução ideal de cada nodo). Também nas extensões do método de Zionts-Wallenius a problemas com variáveis inteiras (RAMESH ET AL., 1990; VILLAREAL ET AL., 1980; KARWAN ET AL., 1985; RAMESH ET AL., 1986) o esforço computacional é grande. Colocam-se ainda demasiadas questões ao AD, sendo algumas delas difíceis de responder, como a avaliação de 'vectores de compensação' que não se referem ao problema original mas a subproblemas relaxados. A este propósito, TEGHEM E KUNSCH (1986[B]) comentaram que, se o método de Zionts-Wallenius já faz muitas perguntas ao AD em problemas de PLMO, então em problemas de PLIMO pode tomar proporções dramáticas porque o número de questões aumenta muito depressa com o número de nodos da árvore de *branch-and-bound*.

A limitação da aplicabilidade é outra dificuldade que se coloca em algumas abordagens. Na nossa opinião, existem abordagens biobjectivo que se afiguram promissoras, como por exemplo as de AKSOY (1990), SOLANKI (1991) e FERREIRA ET AL. (1996), tanto do ponto de vista

cognitivo como computacional, mas o facto de se destinarem exclusivamente a problemas com duas funções objectivo torna-as naturalmente limitadas na prática. De facto, as dificuldades são mais facilmente ultrapassáveis no caso biobjectivo do que em multiobjectivo. Por exemplo, o método de AKSOY (1990) tem uma estrutura de *branch-and-bound* semelhante à do método multiobjectivo de MARCOTTE E SOLAND (1986) e as dificuldades são significativamente maiores no método de Marcotte e Soland. Como consequência, o método de Aksoy é aplicável a problemas inteiros mistos e o de Marcotte e Soland não é.

Há alguns métodos que não permitem calcular soluções eficientes não suportadas. Nesta classe incluem-se os métodos de WALKER (1978) e GABBANI E MAGAZINE (1986) baseados em somas pesadas das funções objectivo.

Tendo em vista diminuir o esforço computacional, foram propostas algumas abordagens em que os programas escalarizantes são resolvidos por técnicas heurísticas: GABBANI E MAGAZINE (1986) e KARAIVANOVA ET AL. (1993). Esta constitui uma via de investigação naturalmente interessante, mas é importante que seja aferida a qualidade das soluções uma vez que estas não são necessariamente eficientes. Nenhum dos dois trabalhos aborda esta questão e apenas o artigo de KARAIVANOVA ET AL. (1993) apresenta tempos computacionais da abordagem proposta.

Os recentes métodos de VASSILEV E NARULA (1993), NARULA E VASSILEV(1994) e KARAIVANOVA ET AL. (1995) revelam a preocupação actual em não exigir demasiada informação de preferências do AD em cada interacção, tentando ao mesmo tempo diminuir o esforço computacional. Começam por resolver apenas um programa escalarizante em cada interacção mas concluem que, mesmo um só problema inteiro (misto) monocritério de cada vez, pode ser moroso. NARULA E VASSILEV (1994) e KARAIVANOVA ET AL. (1995) (2º método) preconizam então abordagens contínuas-inteiras em que a maior parte do tempo é gasto no cálculo de soluções não dominadas contínuas do problema multiobjectivo relaxado linearmente. No entanto, não se sabe *a priori* se as soluções contínuas estão perto ou longe das inteiras "mais próximas", pelo que pode existir um elevado desperdício de tempo em busca de informação pouco relevante para o problema em causa. É esta, em nosso entender, a maior desvantagem deste tipo de abordagens contínuas-inteiras.

A experiência computacional relatada é muito reduzida. Dos trabalhos publicados na década de 90, aqueles em que os tempos computacionais podem de alguma forma ser avaliados face aos computadores de hoje, apenas KARAIVANOVA ET AL. (1993) e KARAIVANOVA ET AL. (1995) apresentam resultados computacionais. Os primeiros referem-se a um computador equivalente a um IBM/XT com coprocessador e, neste computador, foram necessárias mais de 4 h para

resolver (no sentido de optimizar uma suposta função utilidade implícita) heuristicamente problemas com 3 objectivos, 3 restrições e 30 variáveis inteiras. No método de KARAIVANOVA ET AL. (1995) o tempo médio de cálculo de uma solução não dominada de problemas com 4 objectivos, 15 restrições e 15 variáveis inteiras foi superior a 1 h, num PC486.

Muitas das abordagens, em particular as mais recentes, baseiam-se em *pontos de referência/níveis de aspiração* para os objectivos, recorrendo a funções de Tchebycheff ou, mais genericamente, a funções escalarizantes de realização do tipo *min-max*. De facto, sobressaem as vantagens deste tipo de abordagens pela sua generalidade. Recordemos o caso do método de MARCOTTE E SOLAND (1986) que usa somas pesadas das funções objectivo com restrições adicionais nas funções objectivo. Este método não se adequa ao caso inteiro-misto, mas a substituição das somas pesadas por distâncias de Tchebycheff a pontos de referência, proposta por DURSO (1992), permitiu resolver a questão. Observemos ainda a segunda abordagem de FERREIRA (1997) que, pelo facto de usar a métrica de Tchebycheff, consegue detectar zonas maiores de inadmissibilidade do que a primeira abordagem que usa somas pesadas das funções objectivo.

Segundo WIERZBICKI (1998), as abordagens baseadas em *pontos de referência* podem ser vistas como uma generalização da programação por metas ('goal programming'). As abordagens de pontos de referência procuram preservar as maiores vantagens da programação por metas e ultrapassar as desvantagens de base, ou seja, o cálculo de soluções não eficientes quando as metas são atingíveis e dominadas. Poder-se-ia impor a condição de que as metas estivessem suficientemente distantes do conjunto das soluções admissíveis, mas obrigar a que as metas sejam irrealistas significa perder a vantagem básica da programação por metas – ser intuitivamente apelativa. Nas abordagens de pontos de referência, o significado de "chegar perto" do ponto de referência é mais lato, podendo ser interpretado como "chegar perto ou ultrapassar" (WIERZBICKI, 1998). Isto relaciona-se com o conceito de decisões *satisfatórias* de SIMON (1957) – usado para descrever a forma como as pessoas tomam decisões – e com o conceito de decisões *quase-satisfatórias* (LEWANDOWSKI E WIERZBICKI, 1988). De acordo com Simon, os agentes de decisão desenvolvem progressivamente, através da aprendizagem, *níveis de aspiração* para os vários resultados importantes nas suas decisões. Segundo WIERZBICKI (1998), há estudos que revelam que os agentes de decisão usam repetidamente, na prática, vários níveis de referência, não só *níveis de aspiração* mas também *níveis de reserva*. Também NAKAYAMA (1997) preconiza a filosofia da satisfação ou quase-satisfação em contraste com a assunção de uma *função utilidade* implícita onde se presume que as preferências do AD satisfazem certos axiomas matemáticos. É que estes axiomas geralmente não se verificam durante o processo de decisão, porque o juízo de valor do AD pode ser incoerente, o que decorre naturalmente do facto da informação disponível ir aumentando durante o processo de decisão.

Tendo em conta as vantagens técnicas das abordagens de pontos de referência, pela sua generalidade, e as potencialidades oferecidas ao nível da interacção com o AD, propusémo-nos desenvolver abordagens interactivas para problemas de PLIMO e de PLIMMO, tendo por base as seguintes preocupações:

- Criar um protocolo simples de interacção com o AD que assenta na especificação de níveis de aspiração (*ponto de referência*) para os objectivos ou simples indicação de um objectivo que o AD pretende melhorar relativamente a uma solução não dominada. Neste último caso, fica a cargo do algoritmo o ajuste do ponto de referência. O AD poderá ainda impor limitações adicionais nos valores das funções objectivo (*níveis de reserva*) de modo a delimitar zonas de pesquisa que considere mais interessantes.
- Não assumindo a existência de qualquer função utilidade implícita, pretendemos uma *comunicação aberta* com o AD orientada para a aprendizagem. O processo de decisão termina apenas quando o AD considerar que encontrou uma solução de compromisso satisfatória.
- Diminuir o esforço computacional pela resolução de programas escalarizantes paramétricos de forma dependente, ou seja, aproveitando cálculos anteriores para obter as soluções seguintes através de técnicas de análise de sensibilidade e pós-optimização. A análise de sensibilidade tem também como missão ajustar o ponto de referência quando o AD indica apenas o objectivo que pretende melhorar. Uma vez que os problemas tratados admitem soluções discretas e vários pontos de referência conduzem à mesma solução, é importante que o ponto de referência seja ajustado convenientemente de modo a que a solução não dominada obtida seja diferente da anterior. O ajuste automático do ponto de referência deverá, pois, facilitar a intervenção do AD e evitar repetições de cálculos, permitindo que se poupe esforço computacional.

Nos dois capítulos que se seguem proporemos abordagens interactivas com estas características. A primeira, apresentada no capítulo III, usa técnicas de *planos de corte* para resolver os programas escalarizantes inteiros monocritério, e destina-se exclusivamente a problemas de PLIMO. A segunda, apresentada no capítulo IV, usa técnicas de *branch-and-bound* e é mais genérica e robusta que a primeira, aplicando-se tanto a problemas de PLIMO como a problemas de PLIMMO.

Capítulo III

Método interactivo para PLIMO baseado em planos de corte

Este capítulo é parcialmente baseado em ALVES E CLÍMACO (1997) e ALVES E CLÍMACO (1999A), e apresenta um método interactivo que desenvolvemos para problemas de programação linear inteira pura multiobjectivo (PLIMO). Este método combina a utilização de programas escalarizantes de Tchebycheff, planos de corte e técnicas de análise de sensibilidade.

As abordagens multiobjectivo baseadas em pontos de referência podem ser particularmente interessantes para o tratamento de problemas com variáveis inteiras uma vez que permitem alcançar qualquer solução não dominada, suportada ou não suportada. Este tipo de abordagens usa habitualmente programas escalarizantes baseados na métrica de Tchebycheff ou, mais genericamente, programas escalarizantes de realização do tipo *min-max* que projectam pontos de referência no conjunto das soluções não dominadas. As abordagens interactivas, a nosso ver as mais adequadas para tratar os problemas de PLIMO, incorporam informação sobre as preferências do AD nos referidos programas escalarizantes para produzir novas soluções não dominadas. Essa informação consiste na especificação de um novo ponto de referência – como, por exemplo, no método de VASSILEV E NARULA (1993) – ou na indicação implícita ou explícita de 'pesos' que alteram a direcção de projecção de um ponto de referência fixo – é o caso, por exemplo, do método de STEUER E CHOO (1983) que utiliza a métrica pesada de Tchebycheff com variação dos pesos. A alteração do ponto de referência ou do vector de pesos não garante, porém, que a solução não dominada seguinte seja diferente da anterior. Relembramos que os problemas tratados são de programação inteira, em que o conjunto das soluções é discreto, existindo portanto um subconjunto de valores dos parâmetros (sejam eles pontos de referência ou pesos) correspondente a cada solução. Assim, para além da necessidade de se resolver um

novo problema monocritério de programação inteira de cada vez que se alteram os parâmetros, não sabemos *a priori* se não existirá uma repetição de cálculos. Esta questão agrava as dificuldades computacionais do tratamento dos problemas de PLIMO.

A abordagem que propomos tenta mitigar essas dificuldades, incorporando mecanismos de *análise de sensibilidade* para a alteração dos parâmetros – no nosso caso, o ponto de referência – e resolvendo os sucessivos programas escalarizantes de forma "encadeada" para diminuir o esforço computacional envolvido. A análise de sensibilidade tem como intenção facilitar a intervenção do AD, porque altera automaticamente o ponto de referência quando o AD está interessado em conhecer soluções não dominadas próximas da actual. Isto evita que o AD escolha pontos de referência que possivelmente conduziriam à mesma solução.

O programa escalarizante usado para o cálculo de soluções não dominadas consiste na minimização da distância aumentada (não pesada) de Tchebycheff a um ponto de referência, programa $P_{z^+}^{\infty,\rho}$ (de acordo com a definição na secção II.1.3). A parametrização do programa escalarizante através do ponto de referência permite alcançar qualquer solução não dominada e apresenta, a nosso ver, duas importantes vantagens face a uma variação de pesos na métrica pesada de Tchebycheff:

- O cálculo de soluções não dominadas que melhoram sucessivamente um dado objectivo pode ser obtido incrementando-se a correspondente componente do ponto de referência e mantendo-se as outras componentes iguais. Este resultado (apresentado adiante na proposição III.4) permite que se estabeleça um diálogo simples, em que o AD apenas tem que indicar o objectivo que gostaria de melhorar relativamente à solução não dominada anterior.
- Apesar de a programação inteira oferecer grandes dificuldades para qualquer tipo *análise de sensibilidade*, é mais fácil lidar com variações nas componentes do ponto de referência do que nos pesos porque, enquanto que as primeiras surgem nos termos independentes das restrições, os segundos surgem em coeficientes técnicos das restrições. E, como referimos atrás, é importante que se disponha de algum mecanismo de análise de sensibilidade que detecte, pelo menos parcialmente, pontos de referência que conduzem à mesma solução.

A técnica adoptada para resolver os programas escalarizantes é a de *planos de corte* . Apesar de os planos de corte terem demonstrado grandes limitações práticas como técnica de resolução isolada de problemas (genéricos) de programação inteira, eles facilitam a incorporação da *análise de sensibilidade* e a resolução "encadeada" dos sucessivos programas escalarizantes. Essa foi a principal razão que nos motivou a usar planos de corte neste trabalho. Além disso, há estudos recentes que têm demonstrado o valor dos planos de corte na resolução de certas classes de

problemas inteiros puros ou 0-1 mistos, ou na sua integração em esquemas de *branch-and-bound*. A versão actual do método usa apenas *planos de corte fraccionários de Gomory* (GOMORY, 1963) e *desigualdades de cobertura mínima estendidas* (CROWDER ET AL., 1983), em que estas últimas apenas se aplicam a problemas com variáveis 0-1. Alguns dos novos desenvolvimentos da área dos planos de corte não se enquadram de forma directa na nossa abordagem multiobjectivo e por isso não foram utilizados. Mas a abordagem que propomos não está limitada aos planos de corte usados actualmente, podendo ser incorporados outros tipos de desigualdades válidas. Neste contexto, é de referir ainda a importância do *pré-processamento* de problemas. O pré-processamento é um processo inicial de simplificação que não deve ser negligenciado na resolução de problemas de programação inteira, independentemente do método usado a seguir. Alguns dos trabalhos publicados na literatura da área, que obtiveram êxito na prática, consideram o pré-processamento de problemas antes da geração de cortes. No nosso trabalho incluímos algumas dessas técnicas de pré-processamento.

Na secção 1 deste capítulo fazemos uma breve revisão das técnicas planos de corte, mencionando de forma sucinta alguns dos trabalhos mais recentes nesta área.

A *análise de sensibilidade* incluída na nossa abordagem tem como intenção ajustar automaticamente o ponto de referência no programa escalarizante, de modo a garantir que se encontrará, na fase de cálculo seguinte, uma solução não dominada diferente da anterior. Propomos um processo iterativo que alterna a actualização do ponto de referência e o cálculo com planos de corte, culminando numa nova solução não dominada. Esta foi uma forma que encontrámos para dar resposta ao caso particular com que nos deparámos de análise de sensibilidade ao termo independente de uma restrição do programa escalarizante $P_{z^+}^{\infty,\rho}$. No caso geral, trata-se de uma questão de difícil resolução, como veremos adiante na secção 2. Nessa secção fazemos uma breve revisão da investigação anterior na área da análise de sensibilidade/paramétrica em programação inteira que se relaciona mais de perto com o nosso problema. Pretendemos alertar para as principais dificuldades existentes neste domínio, apresentando alguns resultados teóricos conhecidos e abordagens paramétricas com algoritmos de planos de corte.

Em suma, a combinação do programa escalarizante de Tchebycheff, parametrizado no ponto de referência, com técnicas de planos de corte e análise de sensibilidade é especialmente útil para pesquisar soluções não dominadas "consecutivas", propiciando *pesquisas direccionais* e/ou *locais*. É este o tipo de pesquisa a que se destina fundamentalmente o novo método interactivo que desenvolvemos para PLIMO, e que apresentamos na secção 3 deste capítulo. Na secção 3.1 começamos por estabelecer e demonstrar alguns resultados teóricos nos quais assentam as

principais características do método. Estes resultados referem-se ao uso da métrica de Tchebycheff em PLIMO, relacionando-se também com o uso de planos de corte. Na secção 3.2 descrevemos o método interactivo, dando particular destaque ao processo de análise de sensibilidade. Apresentamos em seguida um exemplo ilustrativo na secção 3.3. As características principais da implementação computacional e os testes efectuados são descritos na secção 3.4.

Na secção 4 apresentamos algumas conclusões deste trabalho.

III.1 PLANOS DE CORTE – REVISÃO[1]

III.1.1 Desigualdades válidas em programação inteira

Consideremos o problema de programação inteira definido da seguinte forma:

$$\max \{cx \mid x \in X\} \qquad \text{(P III.1)}$$
$$\text{com } X = \{x \in \Re^n \mid Ax \le b,\ x \ge 0,\ x \text{ inteiro}\}$$

em que A e b têm coeficientes racionais.

Utilizaremos a notação $\lfloor \bullet \rfloor$ para designar o maior número inteiro não superior a $\bullet$.

Um conceito fundamental nos planos de corte é o de *desigualdade válida*.

Definição III.1 *Desigualdade válida*

Uma desigualdade $\pi x \le \pi_0$ é uma desigualdade válida para X se $\pi x \le \pi_0$ se verifica para todo o $x \in X$.

Como desigualdades válidas para X de (P III.1) temos, em particular, aquelas que definem o invólucro convexo ('convex hull') de X que é sempre um poliedro.

Definição III.2 *Invólucro convexo*

Dado um conjunto $X \subseteq \Re^n$, o invólucro convexo de X, designado por $conv(X)$, é definido por

$$\left\{ x \mid x = \sum_{i=1}^{t} \lambda_i x^i,\ \sum_{i=1}^{t} \lambda_i = 1,\ \lambda_i \ge 0 \text{ para } i = 1,...,t \text{ considerando todos os subconjuntos finitos } \{x^1,...,x^t\} \text{ de } X \right\}.$$

[1] As secções III.1.1 e III.1.2 baseiam-se nas obras de NEMHAUSER E WOLSEY (1988) e WOLSEY (1998).

O problema (P III.1) poderia ser resolvido através de $\max\{cx \mid x \in conv(X)\}$ porque qualquer solução básica que optimize este problema é óptima de (P III.1). Contudo, não existe em geral nenhuma forma simples de caracterizar as desigualdades necessárias para descrever *conv*(X).

Uma desigualdade válida para (P III.1) é também a *desigualdade C-G* (Chvátal-Gomory) obtida pelo procedimento de Chvátal-Gomory. Para descrever este procedimento, consideremos $X = S \cap \{x \in \Re^n \mid x \text{ inteiro}\}$ em que $S = \{x \in \Re^n \mid Ax \le b, x \ge 0\}$ e A é uma matriz $m \times n$ de colunas $(A_1,...,A_n)$. Se $u \in \Re^m$, $u \ge 0$, então:

(i) $\sum_{j=1}^{n} uA_j x_j \le ub$ é válida para S porque $u \ge 0$ e $\sum_{j=1}^{n} A_j x_j \le b$

(ii) $\sum_{j=1}^{n} \lfloor uA_j \rfloor x_j \le ub$ é válida para S porque $x \ge 0 \Rightarrow \sum_{j=1}^{n} (uA_j - \lfloor uA_j \rfloor) x_j \ge 0$

(iii) $\sum_{j=1}^{n} \lfloor uA_j \rfloor x_j \le \lfloor ub \rfloor$ é válida para X porque x é inteiro $\Rightarrow \sum_{j=1}^{n} \lfloor uA_j \rfloor x_j$ é inteiro

A desigualdade (iii) é uma *desigualdade C-G*. Ela pode ser adicionada a $Ax \le b$ e o processo pode repetir-se para outras combinações de desigualdades originais e/ou geradas.

Qualquer desigualdade válida para X pode ser obtida pela aplicação do procedimento de Chvátal-Gomory num número finito de vezes. Esta proposição encontra-se demonstrada em NEMHAUSER E WOLSEY (1988) e WOLSEY (1998).

Designemos por *plano de corte* uma desigualdade válida para X que corta soluções não inteiras de S. Genericamente, um algoritmo de *planos de corte* tem os seguintes passos:

Início: Seja t=0 e $S^0 = S$.

Iteração t: Determina a solução óptima x^t do problema relaxado: $\max\{cx \mid x \in S^t\}$. Se x^t é inteira, então é essa a solução óptima de (P III.1) e termina. Caso contrário, se for encontrada uma desigualdade válida para X, $\pi x \le \pi_0^t$ que corte x^t, ou seja, tal que $\pi x^t > \pi_0^t$, faz-se $S^{t+1} = S^t \cap \{x: \pi x \le \pi_0^t\}$ e aumenta-se t. Se não, termina.

Um algoritmo de planos de corte pode ter como intenção terminar só quando for encontrada uma solução inteira (como o algoritmo de Gomory que apresentaremos em seguida) ou servir apenas para melhorar a formulação do problema num processo de reformulação automática. Neste último caso, poderá servir como entrada a um algoritmo de *branch-and-bound*.

III.1.2 Algoritmo de planos de corte fraccionários de Gomory

Consideremos o problema de programação inteira em que todas as restrições foram convertidas em igualdades,

$$\max\{cx \mid Ax = b,\ x \geq 0,\ x \text{ inteiro}\}$$

onde A e b têm coeficientes inteiros.

No final da década de 50, Gomory desenvolveu um algoritmo baseado em planos de corte para a resolução de problemas inteiros (GOMORY, 1963). A ideia consiste em resolver a relaxação linear do problema, i.e. $\max\{cx \mid Ax = b,\ x \geq 0\}$, encontrando uma base óptima para esse problema. Se a solução óptima da relaxação linear, x^*, não for inteira, então escolhe-se uma variável básica com valor não inteiro e gera-se uma *desigualdade C-G* que corte x^* a partir da equação associada a essa variável básica.

Dada uma base óptima para a relaxação linear, o problema pode ser rescrito da seguinte forma:

$$\begin{aligned} \max \quad & \bar{a}_{00} + \sum_{j \in NB} \bar{a}_{0j} x_j && \text{(P III.2)} \\ \text{s.a:} \quad & x_{Bi} + \sum_{j \in NB} \bar{a}_{ij} x_j = \bar{b}_i \qquad i=1,...,m \\ & x \geq 0 \text{ e inteiro} \end{aligned}$$

em que $\bar{a}_{0j}$ são os coeficientes da 'linha dos custos reduzidos' ('$c_j - z_j$') e que verificam $\bar{a}_{0j} \leq 0$, $\forall j \in NB$, (porque a base é óptima para o problema relaxado); NB designa o conjunto dos índices das variáveis não básicas; $\bar{b}_i \geq 0$ para $i=1,...,m$ (porque a solução é primal admissível). Se x^* não é inteira, então é porque existe alguma linha i com $\bar{b}_i$ não inteiro. A *desigualdade C-G* para a linha i é:

$$x_{Bi} + \sum_{j \in NB} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor$$

Sabendo que $x_{Bi} = \bar{b}_i - \sum_{j \in NB} \bar{a}_{ij} x_j$, esta desigualdade pode ser rescrita de modo a eliminar x_{Bi}:

$$\bar{b}_i - \sum_{j \in NB} \bar{a}_{ij} x_j + \sum_{j \in NB} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor$$

$$\Leftrightarrow \qquad \sum_{j\in NB}\left(\bar{a}_{ij}-\lfloor\bar{a}_{ij}\rfloor\right)x_j \geq \left(\bar{b}_i-\lfloor\bar{b}_i\rfloor\right)$$

$$\Leftrightarrow \qquad \sum_{j\in NB} f_{ij}x_j \geq f_{i0}$$

onde $f_{ij}=\bar{a}_{ij}-\lfloor\bar{a}_{ij}\rfloor, j\in NB,\ f_{i0}=\bar{b}_i-\lfloor\bar{b}_i\rfloor$, verificando-se que $0\leq f_{ij}<1$ e $0<f_{i0}<1$.

A $\sum_{j\in NB} f_{ij}x_j \geq f_{i0}$ chama-se *plano de corte de Gomory*, ou simplesmente *corte de Gomory*. Como as variáveis não básicas $j\in NB$ são nulas em x^*, então $\sum_{j\in NB} f_{ij}x_j^* = 0$, pelo que x^* não verifica o plano de corte de Gomory.

Se a equação $x_{n+1}+\sum_{j\in NB}(-f_{ij})x_j = -f_{i0}$ (corte de Gomory convertido em igualdade) for acrescentada ao sistema do problema (P III.2), considerando x_{n+1} a respectiva variável básica, então a solução primal deixa de ser admissível sendo necessário tornar não básica x_{n+1} para repor a admissibilidade.

Este processo de geração de cortes pode ser enquadrado no algoritmo geral de planos de corte apresentado atrás. GOMORY (1963) provou a convergência finita do algoritmo (*c.f.* NEMHAUSER E WOLSEY, 1988) se as linhas a partir das quais se geram os cortes forem escolhidas apropriadamente.

Quando se adiciona um corte, seja o t-ésimo corte, a variável desvio x_{n+t} é inicialmente básica, mas passará a não básica no sentido de repor a admissibilidade primal. Se x_{n+t} se tornar novamente básica com valor positivo numa iteração subsequente, então significa que o corte já não está activo. Este pode ser eliminado do problema assim como a variável x_{n+t}. Devemos ainda referir que, na prática, é geralmente vantajoso adicionar vários cortes em cada iteração do que um só de cada vez.

Na sequência deste trabalho, Gomory estabeleceu ainda planos de cortes para programação inteira-mista.

III.1.3 Outros desenvolvimentos

Os planos de corte pareceram inicialmente muito promissores, mas rapidamente se revelaram ineficazes computacionalmente. Como vários autores afirmaram, são matematicamente elegantes mas apresentam propriedades pobres de convergência e a maior dificuldade advém, não do número de iterações, mas sim dos erros da aritmética em computador (BALAS ET AL., 1996[B]). A

construção de planos de corte de Gomory tem um baixo custo computacional mas estes são, em geral, pouco profundos. As dificuldades agravam-se quando se começam a tornar muito próximos uns dos outros, de iteração para iteração.

Em 1971, Balas propõe uma nova classe de planos de corte para programação inteira (BALAS, 1971). Estes cortes, designados por *cortes de intersecção*, são definidos da seguinte forma: dada uma solução $\bar{x}$ não inteira e óptima da relaxação linear do problema, define-se o hipercubo unitário que contém $\bar{x}$ e vértices inteiros e define-se a hiperesfera que circunscreve o hipercubo; esta hiperesfera intersecta em n pontos independentes as n semi-rectas com origem em $\bar{x}$ que contêm as n arestas do conjunto admissível relaxado (considerando-se $\bar{x}$ não degenerado); o hiperplano que passa por esses n pontos de intersecção define um corte válido para o problema inteiro. Neste trabalho mostra-se que os cortes fraccionários de Gomory são casos particulares dos cortes de intersecção em virtude da hiperesfera poder ser substituída por outra hipersuperfície convexa com determinadas propriedades. Estes planos de corte, bem como as *desigualdades disjuntivas* que se baseiam na investigação levada a cabo por Balas sobre a disjunção de poliedros, também na década de 70, serviram de base a desenvolvimentos mais recentes que revelaram sucesso na prática. As desigualdades disjuntivas são desigualdades válidas para um conjunto que é a união de dois conjuntos disjuntos.

A área dos planos de corte continuou a despertar o interesse de muitos investigadores e trabalhos posteriores demonstraram a sua capacidade na resolução de problemas mais específicos, ou quando inseridos num algoritmo de enumeração. Destacamos o trabalho de Crowder-Johnson-Padberg (CROWDER ET AL., 1983) pelos notáveis resultados computacionais numa colecção de problemas inteiros 0-1 com uma única característica comum, o facto de serem esparsos. O algoritmo de resolução usa sequencialmente três técnicas: pré-processamento de problemas, geração de planos de corte e *branch-and-bound*. Os principais planos de corte usados são designados por *desigualdades de cobertura mínima estendidas*. Estas consistem em aproximações do invólucro convexo para o problema de mochila (*knapsack*) 0-1. As *desigualdades de cobertura* para politopos 0-1 *knapsack* foram desenvolvidas em simultâneo por vários autores e publicadas em 1975 (*c.f.* WOLSEY, 1998). Mas, o maior passo surgiu posteriormente com o uso destas desigualdades por CROWDER ET AL. (1983) em problemas de programação 0-1. A identificação de cortes neste algoritmo consiste no seguinte: dada a solução óptima do problema da relaxação linear, $\bar{x}$, tenta-se encontrar *desigualdades de cobertura mínima* (para definição, ver o anexo III.A) que cortem $\bar{x}$; caso não existam, tenta-se encontrar um outro tipo de desigualdades, designadas por *desigualdades de configuração (1,k)*. Este processo procura identificar uma desigualdade por cada linha da matriz original das restrições, que será depois sujeita a um processo de *extensão* (também chamado de *elevação*) e adicionada ao problema de relaxação linear. O problema aumentado é

então re-optimizado. O procedimento repete-se até que seja obtida uma solução 0-1 ou não se consiga encontrar mais nenhum corte, ou ainda, se o ganho na função objectivo for demasiado pequeno. Nos dois últimos casos, inicia-se um processo de *branch-and-bound*. A vantagem das desigualdades usadas é que preservam a esparsidade das matrizes, ao contrário do que acontece com os tradicionais planos de corte que são tipicamente densos. Além disso, têm sempre coeficientes inteiros pequenos.

Também JOHNSON ET AL. (1985) abordam a resolução de problemas de programação inteira 0-1 através de pré-processamento, geração de desigualdades de cobertura e *branch-and-bound*. Neste trabalho, os problemas tratados são particulares seguindo modelos de planeamento hierárquico de grande escala.

Desenvolvimentos mais recentes incluem a generalização de desigualdades de cobertura para *knapsacks* 0-1 com restrições de limite superior generalizadas (GU ET AL, 1998) e para *knapsacks* inteiros (CERIA ET AL., 1998). Neste último trabalho investiga-se o uso de planos de corte em problemas com variáveis inteiras genéricas. Mostra-se como se podem gerar planos de corte a partir de restrições do tipo *knapsack*, estendendo depois essas desigualdades (à semelhança das desigualdades de cobertura mínima estendidas para variáveis 0-1). Cada um destes planos de corte resulta da solução de um problema de *knapsack* inteiro (que é resolvido heuristicamente) e alguns cálculos auxiliares para os quais se usa programação dinâmica. Os autores exploram ainda o uso de planos de corte inteiros-mistos de Gomory e a consideração destes e dos anteriores para "fortalecer" a formulação de um problema antes de uma pesquisa de *branch-and-bound*. Dos resultados computacionais apresentados, onde se inclui a comparação com um *branch-and-bound* simples, verifica-se que o uso dos dois tipos de cortes com *branch-and-bound* é a melhor estratégia para a maior parte dos casos.

Um outro tipo de desigualdade válida usado em programação 0-1 mista chama-se *desigualdade de cobertura de fluxo*. VAN ROY E WOLSEY (1987) desenvolveram um sistema experimental que implementa este tipo de desigualdade. Neste trabalho, os autores descrevem uma abordagem que consiste no uso de planos de corte para reformular e resolver problemas 0-1 mistos. Após a reformulação, o problema é resolvido por *branch-and-bound*. Esta abordagem foi motivada em parte pelo sucesso do trabalho de outros autores, como por exemplo, o de CROWDER ET AL. (1983). Os autores abordam não só a geração de cortes para restrições 0-1 puras (*desigualdades de cobertura mínima estendidas*), como também a geração de cortes para restrições 0-1 mistas. Assim, é alargada a noção de cobertura (cobertura generalizada) para linhas 0-1 mistas. Tanto nas desigualdades de cobertura mínima (estendidas) como nas de *cobertura de fluxo (estendidas)*, o esforço computacional envolvido na geração de cortes não é muito elevado porque o problema de separação de *knapsack* é resolvido heuristicamente (problema auxiliar para gerar uma desigualdade que corte a solução da relaxação linear).

GOEMANS (1989) estuda um modelo inteiro-misto que consiste numa restrição 0-1 mista, à qual se juntam restrições de limite superior variável para as variáveis contínuas. Neste trabalho, é derivada uma classe de inequações válidas a partir das desigualdades de cobertura de fluxo usadas por VAN ROY E WOLSEY (1987).

BOYD (1994) apresenta uma técnica para gerar planos de corte (designados por cortes de Fenchel) para programas inteiros que se baseia na habilidade em optimizar uma função linear num poliedro. O algoritmo proposto é um algoritmo puro de planos de corte. Os planos de corte usados são desigualdades fortes mas, em contrapartida, requerem algum esforço computacional na sua geração. Para determinar um corte é necessário resolver um problema auxiliar para o qual se usa programação generalizada que aproxima sucessivamente uma função côncava por um conjunto de segmentos lineares por troços. Segundo o autor, a programação generalizada pode, contudo, revelar-se inadequada para dimensões elevadas do problema auxiliar, o que está relacionado essencialmente com a densidade da matriz original.

Recentemente têm sido propostos métodos para obter formulações mais "robustas" de problemas 0-1 mistos, que se relacionam com o trabalho de Balas na programação disjuntiva e nos cortes de intersecção (BALAS, 1971). Um exemplo destes métodos deve-se a BALAS ET AL. (1993). Os autores propõem um algoritmo de planos de corte, em que cada corte é obtido a partir da solução de um programa linear com cerca de duas vezes o tamanho da relaxação linear do problema original. O objectivo desse programa é escolher o corte mais 'fundo' de entre uma dada família de desigualdades violadas pela solução não inteira actual. Trata-se de um processo que *eleva* o problema a um espaço de maior dimensão, onde uma formulação mais conveniente pode proporcionar uma relaxação mais "estreita". Esta é depois *projectada* no espaço original. Na prática, sob o nome de *lift-and-project* (elevar e projectar), foi desenvolvido um algoritmo de planos de corte disjuntivos.

Como já referimos atrás, muitas das técnicas de planos de corte têm o seu papel mais importante numa fase de reformulação do problema. Se, durante essa fase, não se atingir a solução óptima, então avança-se para um algoritmo de enumeração (do tipo *branch-and-bound*) com uma formulação "fortalecida". É, pois, uma espécie de pré-processamento iterativo que pode ou não terminar na solução óptima. Este processo difere do tradicional pré-processamento (BREARLY ET AL., 1975) que consiste, basicamente, em percorrer as linhas e as colunas do problema para identificar linhas redundantes, detectar variáveis que se podem tornar fixas, incrementar (decrementar) limites inferiores (superiores) de variáveis e modificar coeficientes de matrizes. O tradicional pré-processamento não deve, aliás, ser negligenciado na resolução de um problema de programação inteira (seja qual for o método usado a seguir) porque pode

simplificar o problema. Notemos que alguns dos trabalhos citados que obtiveram êxito na prática incluem, antes da geração de cortes, o pré-processamento de problemas (por exemplo, CROWDER ET AL., 1983 e JOHNSON ET AL., 1985).

Para além dos algoritmos que geram planos de corte antes de se iniciar a resolução através de *branch-and-bound*, existem também aqueles em que os planos de corte são gerados ao longo da árvore de *branch-and-bound*. Estes algoritmos são habitualmente designados por *branch-and-cut*. Na prática, deve ser estabelecido um compromisso entre a geração de cortes, que implica um maior esforço computacional em cada nodo, e a ramificação.

BALAS ET AL. (1996[A]) investigam o desempenho computacional da incorporação de planos de corte do tipo *lift-and-project* (BALAS ET AL., 1993) num enquadramento *branch-and-cut* para problemas 0-1 mistos. Os autores defendem que os cortes gerados num nodo devem ser globalmente válidos para que o algoritmo *branch-and-cut* seja eficaz. Caso contrário, seria necessário tratar e guardar separadamente os cortes relativos a cada nodo da árvore de pesquisa, o que requereria um grande esforço computacional (tanto em tempo como em memória). Nessa perspectiva, foi desenvolvido um processo que começa por gerar um corte válido para um nodo da árvore, 'elevando-o' em seguida de modo a tornar-se válido para todo o problema. Os testes computacionais, em problemas 0-1 puros e mistos, revelaram que o código é eficaz. O seu desempenho em várias instâncias foi tão bom ou superior ao de alguns dos melhores códigos inteiros-mistos disponíveis até ao momento. Em alguns casos foi mesmo mais eficaz que algoritmos actuais específicos para determinados problemas.

Os cortes inteiros-mistos de Gomory foram também reavivados como um meio computacional num contexto de *branch-and-cut* para problemas 0-1 mistos (BALAS ET AL, 1996[B]). Neste trabalho, os autores mostram que é possível estender estes cortes por forma a que, sendo gerados num nodo, sejam globalmente válidos. Este resultado é válido apenas para programas 0-1 mistos e, segundo os autores, não é extensível ao caso geral com variáveis inteiras (puro ou misto). Os autores salientam ainda que não é conhecido nenhum resultado semelhante para o caso geral. Para os testes computacionais foram usados os problemas testados por BALAS ET AL. (1996[A]). O *branch-and-cut* de BALAS ET AL. (1996[B]) conseguiu resolver mais instâncias do que vários algoritmos de *branch-and-bound* puro, mas menos (4 em 29) do que o algoritmo de BALAS ET AL.(1996[A]). Porém, o algoritmo de BALAS ET AL. (1996[B]) conseguiu superar em tempo computacional o algoritmo de BALAS ET AL. (1996[A]) em quase metade das instâncias. Os autores questionam-se então: "porque é que os cortes de Gomory funcionam bem nas nossas experiências se isso ainda não tinha acontecido no passado?" As razões que apontam como as mais prováveis para explicar este facto são as seguintes: (i) são gerados vários cortes de cada vez e não um em cada re-optimização; (ii) os cortes são gerados em diferentes nodos da árvore o que ajuda a afastarem-se de um efeito de paralelização; (iii) os cortes são globalmente válidos (numa

experiência computacional com cortes locais, o código falhou em cerca de metade dos problemas); (iv) os cortes são usados selectivamente, ou seja aqueles que são localmente redundantes não são incluídos, o que permite evitar algumas dificuldades numéricas; (v) os códigos de resolução de programas lineares são mais robustos agora do que há 30 anos atrás.

Estes trabalhos mais recentes abrem novas perspectivas na utilização dos planos de corte, mesmo para os mais clássicos, deixando bem visível que a área dos planos de corte continua a ser alvo do interesse de muitos investigadores.

Esta breve introdução e revisão que fizemos a trabalhos desenvolvidos na área dos planos de corte não é de forma alguma exaustiva. Pretendemos apenas apontar os principais tipos de planos de corte usados, os problemas a que se adequam e o seu desempenho. É de salientar que existem vários outros trabalhos nesta área que são específicos a problemas particulares, como por exemplo, o problema da árvore de Steiner, caixeiro viajante, etc.

O método interactivo para problemas de PLIMO que apresentaremos neste capítulo usa planos de corte na resolução dos programas inteiros escalarizantes. Por questões de simplicidade na geração dos cortes, os planos de corte adoptados são os cortes fraccionários de Gomory e as desigualdades de cobertura mínima estendidas de Crowder-Johnson-Padberg (CROWDER ET AL., 1983). Estas últimas apenas se adequam aos programas escalarizantes de problemas multiobjectivo com variáveis 0-1. Como veremos, o método não tira partido da especificidade dos planos de corte usados e, por isso, não há qualquer limitação que o obrigue a confinar-se a estes tipos de planos de corte. Pela mesma razão, remetemos para o anexo III.A a descrição das desigualdades de cobertura mínima estendidas.

III.2 REVISÃO DE ANÁLISE DE SENSIBILIDADE / PARAMÉTRICA EM PROGRAMAÇÃO INTEIRA

Nesta secção fazemos uma referência à investigação anterior na área da análise de sensibilidade/paramétrica em programação inteira (PI). Cingimo-nos àquela que de alguma forma se relacione com o nosso trabalho, nomeadamente resultados teóricos e trabalho desenvolvido no contexto dos planos de corte.

WANG E WORNG (1996) distinguem a análise de sensibilidade da análise paramétrica, considerando que a primeira tem como intenção encontrar o maior intervalo de tolerância de um parâmetro para o qual a solução ou base óptima actual se mantém, ao passo que a análise paramétrica pretende identificar a sequência de soluções óptimas para os correspondentes intervalos de tolerância de um parâmetro. A análise de sensibilidade ou paramétrica é geralmente feita em relação aos coeficientes da função objectivo ou aos termos independentes das restrições e, qualquer uma delas, quando efectuada em PI, apresenta dificuldades acrescidas relativamente

ao mesmo tipo de análise em programação linear (PL). Como um modelo de PI não tem preços sombra ou variáveis duais com interpretação comparável ao que acontece em PL, quando se pretende determinar a influência na solução óptima da alteração, por exemplo, do nível de um recurso, então é geralmente necessário resolver o problema com o recurso alterado.

Atendendo a que a função objectivo do problema escalarizante de PI que iremos tratar é a minimizar, comecemos por definir os seguintes problemas paramétricos genéricos como sendo de minimização, um com parametrização em "*b*" e outro em "*c*":

$$z(\theta)= \min\{cx \mid Ax \geq b+\theta r,\ x\geq 0,\ x_j \text{ inteiro},\ j\in I\} \qquad (P_\theta)$$

$$z'(\theta)= \min\{(c+\theta f)x \mid Ax \geq b,\ x\geq 0,\ x_j \text{ inteiro},\ j\in I\} \qquad (P^\theta)$$

em que, sem perda de generalidade, $0\leq\theta\leq 1$. P_θ e P^θ são programas paramétricos inteiros, se I contiver os índices de todas as variáveis, ou paramétricos inteiros-mistos se I representar apenas um subconjunto dos índices das variáveis.

Um dos artigos mais reconhecidos nesta área é o de GEOFFRION E NAUSS (1977) que apresenta resultados teóricos do comportamento de P_θ e P^θ. Relativamente a P_θ, foram estabelecidas as seguintes proposições:

Proposição III.1 (GEOFFRION E NAUSS, 1977): Quando $r \geq 0$, o valor óptimo de P_θ, i.e. $z(\theta)$, é monótono não decrescente para $\theta \in [0,1]$, e qualquer solução admissível para $\overline{\theta}$ mantém-se admissível para todos os $\theta \leq \overline{\theta}$.

Proposição III.2 (GEOFFRION E NAUSS, 1977): Quando $r \geq 0$, qualquer solução óptima de P_θ para $\overline{\theta}$ continua óptima em todos os $\theta \geq \overline{\theta}$ para os quais a solução continua admissível.

A proposição III.2 surge como corolário de uma outra mais genérica que diz que, se uma solução óptima x^* de um problema (P) se mantiver admissível num outro problema (Q), que tem a mesma função objectivo de (P) e região admissível contida na de (P), então x^* é uma solução óptima de (Q).

Quanto a P^θ, Geoffrion e Nauss apresentam, de entre vários resultados, a seguinte proposição:

Proposição III.3 (GEOFFRION E NAUSS, 1977): O valor óptimo de P^θ para $0\leq \theta \leq 1$, i.e. $z'(\theta)$, representa uma função linear por troços, contínua e côncava no seu domínio finito.

Tendo em conta este último resultado, e atendendo a que alterações de θ em P^{θ} não alteram a região admissível, podemos facilmente concluir que as dificuldades em lidar com P^{θ} são menores do que com P_{θ}. Este facto é visível no trabalho de JENKINS (1982) que desenvolveu um método para análise paramétrica de P^{θ} e P_{θ} em programação inteira-mista. O método consiste na resolução do problema inteiro-misto para determinados valores do parâmetro θ, ligando depois os resultados através de análise paramétrica clássica de PL no sentido de completar a análise de P^{θ} ou P_{θ} para a variação contínua do parâmetro. É usada uma regra heurística para reduzir o número de programas individuais a resolver e reduzir, desta forma, o esforço computacional total. A regra heurística assume o seguinte: se $x^*(\theta')=x^*(\theta'')$, então esta solução é óptima para todo o $\theta' \leq \theta \leq \theta''$ (em que $x^*(\theta)$ designa os valores das variáveis inteiras na solução óptima de P^{θ} ou P_{θ} para um dado θ). Com esta regra, a análise paramétrica *completa* requer que sejam resolvidos $2p$-1 problemas inteiros-mistos (em qualquer um dos casos), em que p é o número de diferentes soluções inteiras encontradas. No caso de P^{θ}, as propriedades de $z'(\theta)$ (linear por troços e côncava) poderiam ser usadas para verificar se $z'(\theta)$ foi encontrado para todo o $0 \leq \theta \leq 1$, apesar de o procedimento completo poder requerer a resolução de muitos problemas inteiros-mistos. Mas, não existe infelizmente nenhum resultado comparável com a concavidade de $z'(\theta)$ que possa ser facilmente aplicado para provar que P_{θ} foi resolvido para todo o $0 \leq \theta \leq 1$. Segundo Jenkins, não resta, pois, alternativa se não usar a regra heurística. Contudo, o pressuposto da regra heurística não se verifica sempre, como mostra WANG E HORNG (1996) através de um contra-exemplo (em que r tem componentes positivas e negativas). Esta questão despertou o interesse de outros investigadores.

Baseando-se no trabalho de JENKINS (1982), WANG E HORNG (1996) propõem um algoritmo para a parametrização completa de P_{θ} em programação inteira pura. Utilizam um passo para θ que se baseia no seguinte facto: quando há uma perturbação em algum b_i, a solução óptima mantém-se a não ser que a parte inteira de $b_i+\theta r_i$ se altere. Assim, cada valor de $\theta \in [0,1]$ que produz $b_i+\theta r_i$ inteiro para algum i é um candidato principal de θ, provando-se que entre dois candidatos principais de θ adjacentes existe no máximo uma solução óptima de P_{θ}. De acordo com os autores, esta é uma questão merecedora de estudos futuros porque o passo de θ proposto é um valor "prudente". Acrescentam que, embora não o consigam determinar, o passo deveria ser a variação exacta de b que conduzisse a outras soluções inteiras. Isso permitiria reduzir consideravelmente o esforço computacional.

Ainda na sequência do trabalho de JENKINS (1982), CREMA (1998, 1999) apresenta resultados que permitem verificar se a análise do problema paramétrico P_{θ} é completa em programação

inteira-mista. O algoritmo de JENKINS (1982) determina uma função $g(\theta)$ tal que $g(\theta) \geq z(\theta)$ para todo o $\theta \in [0,1]$ e $g(\theta)=z(\theta)$ em alguns valores de θ. CREMA (1998) define um problema auxiliar de programação linear inteira-mista que permite verificar se $g(\theta)=z(\theta)$ para um intervalo $\theta \in [\alpha_0,\alpha_l] \subseteq [0,1]$ em que g é contínua. Quando $g(\theta)\neq z(\theta)$ para algum θ, o problema auxiliar determina θ^* tal que existe uma solução associada a θ^* não encontrada anteriormente. Nesse caso, essa solução é usada para melhorar $g(\theta)$. O problema auxiliar considera θ uma variável de decisão, incluindo ainda outras variáveis auxiliares contínuas e binárias. No artigo de CREMA (1999) é apresentada uma versão especializada deste algoritmo para problemas de programação inteira 0-1.

CREMA (1997) desenvolveu também um método de análise multiparamétrica dos termos independentes das restrições para problemas de programação inteira. Se considerarmos o problema P_θ definido atrás, θ é agora um vector tal que a parametrização das restrições toma a forma $Ax \geq b+\theta$, com $\theta \in \Omega=\{\theta \in \Re^m: L \leq \theta \leq U\}$. O algoritmo baseia-se no seguinte princípio: seja (S^1) o problema P_θ com θ interpretado como um vector de variáveis de decisão $(Ax-\theta \geq b)$ e (x^1,θ^1) uma solução óptima de (S^1); então x^1 é uma solução óptima de P_θ para todo o θ tal que $\theta \in \Omega \cap W^1$ com $W^1=\{\theta : Ax^1-b \geq \theta\}$. W^1 define um subconjunto de vectores θ para o qual x^1 é óptima. Definindo depois (S^2) que restringe (S^1) de modo a que $\theta \in \Omega - W^1$ (restrições do tipo 'ou' que são formuladas à custa de variáveis binárias auxiliares), determina-se x^2 e W^2, novo subconjunto de vectores θ para o qual x^2 é óptima. Assim, considerando sucessivamente $\theta \in (\Omega \cap W^p)-(W^1 \cup W^2 ... \cup W^{p-1})$ consegue-se uma parametrização completa. Isto não significa que os intervalos $W^1,...,W^p$ conduzam a p soluções inteiras distintas, porque uma solução pode surgir mais do que uma vez.

Todos os procedimentos mencionados (JENKINS, 1982, WANG E HORNG, 1996, CREMA, 1997, 1998, 1999) visam uma parametrização completa de P_θ com $0 \leq \theta \leq 1$, resolvendo separadamente cada problema inteiro ou inteiro-misto da sequência obtida para certos valores de θ. A análise é, pois, independente do algoritmo usado para resolver os problemas de programação inteira (mista). Existem, no entanto, métodos que tiram partido do algoritmo de programação inteira (mista) usado, seja ele de planos de corte ou um algoritmo de enumeração como o *branch-and-bound*. Referiremos no próximo capítulo os que usam algoritmos de enumeração (mesmo que combinados com planos de corte), referindo aqui aqueles que usam apenas planos de corte.

KLEIN E HOLM (1979) mostram como os planos de corte de Gomory, introduzidos durante a resolução de um problema de programação linear inteira (ou inteira-mista), podem ser

modificados de modo a continuarem válidos quando se alteram os termos independentes de restrições do problema (independentemente da variação ser no sentido de reduzir ou alargar a região admissível). Assim, quando se alteram as componentes de b, alteram-se também recursivamente os termos independentes dos cortes que tinham sido anteriormente introduzidos (através de operações com as matrizes inversas das bases do processo iterativo), e recalculam-se os valores das variáveis da base actual. Se estes valores forem não negativos e inteiros, então a solução é óptima; caso contrário, prossegue-se introduzindo novos cortes. Subjacente ao método está o seguinte resultado (dos mesmos autores): é possível definir cortes de Gomory a partir de linhas 'não admissíveis' do quadro simplex (ou seja, a partir de equações violadas pela solução actual). NAUSS (1979) tira partido deste resultado para resolver o problema paramétrico discreto (P_{r^k} : $\min\{cx \mid Ax \geq b+r^k,\ x \geq 0,\ x_j$ inteiro, $j \in I\}$), em que há K variações discretas em b, ($b+r^k$, $k=1,...,K$). O algoritmo proposto adiciona K colunas à direita do quadro simplex, uma para cada problema, e trabalha simultaneamente com todas essas colunas. Assim, de cada vez que se adiciona um corte, ele é do tipo $\sum_{j \in NB} f_{ij} x_j \geq \left(f_{i0}^1, f_{i0}^2, ..., f_{i0}^K\right)$.

BAILEY E GILLETT (1980) propõem um algoritmo para análise paramétrica completa de P_θ, usando planos de corte de Gomory, em que há uma progressiva contracção da região admissível. Defendem que o maior benefício do uso de planos de corte é o facto de se trabalhar com um só quadro de tamanho reduzido. Considerando apenas o caso em que $r \geq 0$ (em P_θ), aproveitam o resultado de GEOFFRION E NAUSS (1977) (exposto na proposição III.2) de que a solução óptima para θ permanecerá óptima para $\theta' > \theta$ enquanto for admissível. Quando isso deixa de acontecer, a região admissível é contraída (i.e. reduzida) através da alteração do termo independente de uma ou mais restrições por forma a eliminar a solução actual.

Recorrendo também aos planos de corte de Gomory e a facetas de *knapsack* (desigualdades de cobertura de Crowder-Johnson-Padberg para problemas 0-1), JENKINS (1987) apresenta um método de análise paramétrica da variação dos coeficientes da função objectivo de um problema inteiro (puro). O problema paramétrico em causa é P^θ com todas as variáveis inteiras. Uma das grandes vantagens de P^θ face a P_θ é que a variação de θ não altera o conjunto das soluções admissíveis e, consequentemente, os cortes introduzidos previamente permanecem válidos. Por outro lado, a concavidade de $z'(\theta)$ pode ser explorada para calcular $z'(\theta)$ para todo o $\theta \in [0,1]$. Jenkins já havia explorado este resultado na análise paramétrica de P^θ em programação inteira-mista (JENKINS, 1982), formalizando mais tarde (em 1986) um procedimento especializado para PI pura. O procedimento requeria a identificação da solução óptima do problema inteiro num determinado número de valores do parâmetro e, nas experiências anteriores, os problemas

tinham sido resolvidos separadamente usando um código comercial de *branch-and-bound*. Segundo o autor, as tentativas que fez para usar a árvore de enumeração de um problema de PI para guiar a pesquisa em *branch-and-bound* de outro problema não foram particularmente animadoras. Por isso, e motivado pelo trabalho prévio de KLEIN E HOLM (1979), JENKINS (1987) desenvolveu um método que aproveita os planos de corte para uma análise de sensibilidade clássica de PL aplicada a um dado $\overline{\theta}$. A intenção é determinar o domínio de optimalidade em PL de $x^*(\overline{\theta})$, a solução óptima para $\overline{\theta}$. O método foi testado em 1000 problemas 0-1 e 1000 problemas inteiros com 10 a 20 variáveis e 6 a 30 restrições. Compararam-se os resultados da resolução dos problemas inteiros de forma separada (com planos de corte), sob a forma cumulativa (em que se continua a adicionar cortes ao último problema resolvido), e ainda acrescentando-se análise de sensibilidade para evitar resolver todos os problemas. Dos resultados conclui-se que o maior benefício deriva da abordagem cumulativa havendo também algum benefício, mas menor, quando se inclui análise de sensibilidade.

Em resumo, os trabalhos referidos visam resolver o programa de programação linear inteira (ou inteira-mista) parametrizado nos coeficientes da função objectivo ou nos termos independentes das restrições. Consideram em geral uma sequência de programas inteiros (mistos) para diferentes valores do(s) parâmetro(s). Cada um desses programas é resolvido separadamente (JENKINS, 1982, WANG E HORNG, 1996, CREMA, 1997, 1998, 1999) ou de forma dependente/simultânea usando, por exemplo, planos de corte (KLEIN E HOLM, 1979, NAUSS, 1979, BAILEY E GILLETT, 1980, JENKINS, 1987). No segundo caso o tratamento é mais fácil para a parametrização na função objectivo, visto que a região admissível não se altera (JENKINS, 1987). A garantia de uma parametrização completa (com um parâmetro) nos termos independentes das restrições envolve, em geral, um esforço computacional muito elevado porque, ou se considera um passo muito pequeno para o parâmetro (como fez WANG E HORNG, 1996), ou é necessário resolver um problema auxiliar inteiro-misto de dimensão superior ao original para verificar se há alguma outra solução entre dois valores de parâmetros mais espaçados (como propõe CREMA, 1998). Nenhuma destas abordagens é, pois, computacionalmente simples para dar resposta à seguinte questão: "qual a maior variação do termo independente de uma restrição que mantém óptima a solução de um problema com variáveis inteiras?".

III.3 MÉTODO DE PONTOS DE REFERÊNCIA PARA PLIMO QUE USA PLANOS DE CORTE

III.3.1 Resultados teóricos

Consideremos o modelo de programação linear inteira pura multiobjectivo (PLIMO) definido na secção II.1.3, em que todas as variáveis x_j, $j=1,...,n$, e as funções objectivo z_i, $i=1,...,k$, assumem apenas valores inteiros:

$$\begin{aligned} &\max \quad z_1 = c^1 x \qquad\qquad (\text{PLIMO})\\ &\ldots\\ &\max \quad z_k = c^k x\\ &\text{s.a:} \quad x \in X\\ &\qquad X = \left\{ x \in \Re^n \mid Ax = b,\ x \geq 0,\ x \text{ inteiro} \right\} \end{aligned}$$

Seja C a matriz $k \times n$ cuja i-ésima linha é o vector c^i de componentes inteiras. Assumimos as definições anteriores (apresentadas no capítulo II) de Z, X_{ef}, Z_{nd}, z^* e z^{**}.

Consideremos o programa escalarizante que determina a solução $x \in X$ do problema PLIMO cujo ponto do espaço dos objectivos, $z = Cx$, é o mais próximo do ponto de referência z^+ segundo a métrica *não pesada* de Tchebycheff. Sem perda de generalidade[2], assumimos que z^+ satisfaz $z^+ \geq z$, $\forall z \in Z$, ou seja, $z^+ \geq z^*$:

$$\begin{aligned} &\min \quad \alpha \qquad\qquad (P^{\infty}_{z^+})\\ &\text{s.a:} \quad c^i x + \alpha \geq z^+_i \qquad i=1,...,k\\ &\qquad x \in X\\ &\qquad \alpha \geq 0 \end{aligned}$$

Apesar de $P^{\infty}_{z^+}$ poder gerar soluções fracamente eficientes, de entre as soluções óptimas alternativas para um dado z^+, existe sempre pelo menos uma solução que é eficiente (proposição II.5). Além disso, qualquer solução eficiente pode ser obtida através de uma parametrização adequada de $P^{\infty}_{z^+}$ (lema II.7). O cálculo de soluções fracamente eficientes pode ser evitado pelo uso da métrica aumentada de Tchebycheff, $P^{\infty,\rho}_{z^+}$, em que ρ é uma constante positiva suficientemente pequena:

---

[2] Como vimos na secção II.1.3, pode ser usado um qualquer ponto de referência se considerarmos o programa *min-max* que apenas difere de $P^{\infty}_{z^+}$ por α não ter restrição de sinal. Referimo-nos ao programa $P_{q,\lambda}$ com $\lambda = (1,1,...,1)$. Mas, qualquer ponto de referência pode ser substituído por outro equivalente que satisfaz $z^+ \geq z^*$ e que produz o mesmo resultado através de $P^{\infty}_{z^+}$.

$$\begin{aligned} \min \quad & \alpha - \rho \sum_{i=1}^{k} c^i x && (P_{z^+}^{\infty,\rho}) \\ \text{s.a:} \quad & c^i x + \alpha \geq z_i^+ \qquad i=1,...,k \\ & x \in X \\ & \alpha \geq 0 \end{aligned}$$

Qualquer solução eficiente do problema PLIMO pode ser obtida a partir de $P_{z^+}^{\infty,\rho}$ e qualquer solução óptima de $P_{z^+}^{\infty,\rho}$ é eficiente (proposição II.9).

Por definição, os programas $P_{z^+}^{\infty}$ e $P_{z^+}^{\infty,\rho}$ são de programação linear inteira-mista dado que α é uma variável contínua. Esta variável assumirá apenas valores inteiros se forem sempre usados pontos de referência inteiros (relembramos que os elementos de c^i, $i=1,...,k$, são inteiros).

Por razões de simplicidade e clareza de prova, as proposições seguintes serão estabelecidas para $P_{z^+}^{\infty}$ em vez de $P_{z^+}^{\infty,\rho}$.

Proposição III.4: Seja x^a uma solução eficiente do problema PLIMO que optimiza $P_{z^{a+}}^{\infty}$ e seja $z^{b+} = \left(z_1^{a+},...,z_p^{a+}+\theta_p,...,z_k^{a+}\right)$ com $\theta_p > 0$. Então, verifica-se (i) ou (ii) estabelecidos por:

(i) x^a optimiza $P_{z^{b+}}^{\infty}$;

(ii) existe $x^b \in X_{ef}$ que optimiza $P_{z^{b+}}^{\infty}$ tal que $c^p x^b > c^p x^a$.

Prova: Seja α^a o valor óptimo de α em $P_{z^{a+}}^{\infty}$ e $z^a = C x^a$.

CASO 1: Se $z_p^{a+} - z_p^a = \alpha^a$ então,

$(x^a, \alpha^a + \theta_p)$ é uma solução admissível em $P_{z^{b+}}^{\infty}$. Se ela não for óptima (o que significa que (i) não se verifica), então existe uma outra solução eficiente (x^b, α^b) que optimiza $P_{z^{b+}}^{\infty}$. Seja $z^b = C x^b$. Como $\alpha^b < \alpha^a + \theta_p$ e $z_p^{b+} - z_p^b \leq \alpha^b \Leftrightarrow z_p^{a+} + \theta_p - z_p^b \leq \alpha^b$, então $z_p^{a+} - z_p^b < \alpha^a$. Pela hipótese do *caso 1*, $z_p^{a+} - z_p^a = \alpha^a$, logo $z_p^b > z_p^a \Leftrightarrow c^p x^b > c^p x^a$.

CASO 2: $z_p^{a+} - z_p^a < \alpha^a$; consideremos separadamente duas situações:

2.1) se $\theta_p \leq \alpha^a - z_p^{a+} + z_p^a$, então

$$z_p^{b+} - z_p^a \leq z_p^{a+} + (\alpha^a - z_p^{a+} + z_p^a) - z_p^a \Leftrightarrow z_p^{b+} - z_p^a \leq \alpha^a \qquad (1)$$

Como, $z_i^{b+} = z_i^{a+}$, $\forall i \neq p$ verifica-se que $z_i^{b+} - z_i^a \leq \alpha^a$, $\forall i \neq p$ (2)

Por *(1)* e *(2)*, (x^a, α^a) é admissível em $P^{\infty}_{z^{b+}}$. Por outro lado, qualquer solução admissível $(\tilde{x}, \tilde{\alpha})$ em $P^{\infty}_{z^{b+}}$ é também admissível para $P^{\infty}_{z^{a+}}$. Como (x^a, α^a) optimiza $P^{\infty}_{z^{a+}}$, então $\alpha^a \leq \tilde{\alpha}$ e consequentemente (x^a, α^a) é também óptima para $P^{\infty}_{z^{b+}}$ – situação (i).

2.2) Se $\theta_p > \alpha^a - z_p^{a+} + z_p^a$, então

seja $\theta_p = (\alpha^a - z_p^{a+} + z_p^a) + \delta_p$ em que $\delta_p > 0$.

Seja $z^{c+} = \left(z_1^{a+}, ..., z_p^{a+} + (\alpha^a - z_p^{a+} + z_p^a), ..., z_k^{a+}\right)$. De acordo com a prova do *caso 2.1*, (x^a, α^a) optimiza $P^{\infty}_{z^{c+}}$ e verifica-se $z_p^{c+} - z_p^a = \alpha^a$.

Logo, o resultado produzido por $z^{b+} = \left(z_1^{c+}, ..., z_p^{c+} + \delta_p, ..., z_k^{c+}\right)$ relativamente ao de z^{c+} recai no *caso 1*.

Nota: Se substituirmos a condição $c^p x^b > c^p x^a$ por $c^p x^b \geq c^p x^a$ em (ii) da proposição III.4, então esta proposição é também válida para $P^{\infty,\rho}_{z^+}$, assumindo que ρ é suficientemente pequeno.

Proposição III.5: Seja $z^{a+} = \left(z_1^{a+}, ..., z_p^{a+}, ..., z_k^{a+}\right)$ um ponto de referência de componentes inteiras, $z^{b+} = \left(z_1^{a+}, ..., z_p^{a+} + 1, ..., z_k^{a+}\right)$ e $z^{c+} = \left(z_1^{a+}, ..., z_p^{a+} + \theta_p, ..., z_k^{a+}\right)$ com $0 < \theta_p < 1$. Então, não existe nenhuma solução eficiente que optimize $P^{\infty}_{z^{c+}}$ sem optimizar , pelo menos, $P^{\infty}_{z^{a+}}$ ou $P^{\infty}_{z^{b+}}$.

Prova: Suponhamos que existe $x^c \in X_{ef}$ que optimiza $P^{\infty}_{z^{c+}}$ e não optimiza $P^{\infty}_{z^{a+}}$ nem $P^{\infty}_{z^{b+}}$. Seja $z^c = C x^c$ e sejam α^a, α^b e α^c os valores óptimos de α em $P^{\infty}_{z^{a+}}$, $P^{\infty}_{z^{b+}}$ e $P^{\infty}_{z^{c+}}$, respectivamente.

CASO 1: se $z_p^{c+} - z_p^c = \alpha^c$, então

α^c não é inteiro e $z_i^{a+} - z_i^c \leq \lfloor \alpha^c \rfloor$, $\forall\, i \neq p$, *(3)*

em que $\lfloor \alpha^c \rfloor$ designa a parte inteira de α^c. Seja $\langle \alpha^c \rangle = \alpha^c - \lfloor \alpha^c \rfloor$, i.e. a parte fraccionária de α^c.

Neste caso, $z_p^c = z_p^{c+} - \alpha^c = \underbrace{z_p^{a+} - \lfloor \alpha^c \rfloor}_{\text{inteiro} \Rightarrow} + \underbrace{\theta_p - \langle \alpha^c \rangle}_{\text{inteiro}}$, $0 < \theta_p < 1$ e $0 < \langle \alpha^c \rangle < 1$,

o que implica que $\theta_p - \langle \alpha^c \rangle = 0$ e $z_p^c = z_p^{a+} - \lfloor \alpha^c \rfloor$ *(4)*

Por *(3)* e *(4)*, $(x^c, \lfloor \alpha^c \rfloor)$ é admissível para $P_{z^{a+}}^{\infty}$. Seja (x^a, α^a) uma solução óptima de $P_{z^{a+}}^{\infty}$. Então $(x^a, \alpha^a + \theta_p)$ é admissível para $P_{z^{c+}}^{\infty}$. Mas, $\alpha^a < \lfloor \alpha^c \rfloor$ porque x^c não minimiza $P_{z^{a+}}^{\infty}$. Logo, $\alpha^a + \theta_p < \lfloor \alpha^c \rfloor + \theta_p = \lfloor \alpha^c \rfloor + \langle \alpha^c \rangle = \alpha^c$

o que contradiz a hipótese de que (x^c, α^c) minimiza $P_{z^{c+}}^{\infty}$.

CASO 2: se $z_p^{c+} - z_p^c < \alpha^c$, então

existe $J \subseteq \{1,\ldots,k\} \setminus \{p\}$ tal que $z_j^{c+} - z_j^c = \alpha^c$, $j \in J$ e α^c é inteiro;

$z_j^{c+} - z_j^c = \alpha^c \Leftrightarrow z_j^{b+} - z_j^c = \alpha^c$, $j \in J$. *(5)*

Para todo o $i \notin J \cup \{p\}$, $z_i^{b+} - z_i^c < \alpha^c$. *(6)*

Para p, $z_p^{c+} - z_p^c < \alpha^c \Leftrightarrow z_p^{b+} - z_p^c < \alpha^c - \theta_p + 1$. O lado esquerdo da desigualdade é inteiro o que implica que $z_p^{b+} - z_p^c \leq \alpha^c$. *(7)*

Por *(5)*, *(6)* e *(7)*, (x^c, α^c) é admissível em $P_{z^{b+}}^{\infty}$. Seja (x^b, α^b) uma solução óptima de $P_{z^{b+}}^{\infty}$. Esta solução é admissível para $P_{z^{c+}}^{\infty}$ e $\alpha^b < \alpha^c$ porque (x^c, α^c) não minimiza $P_{z^{b+}}^{\infty}$ o que contraria a hipótese de que (x^c, α^c) minimiza $P_{z^{c+}}^{\infty}$.

Nota: A proposição III.5 é válida também para $P_{z^+}^{\infty,\rho}$.

Proposição III.6: Se os pontos de referência z^{a+} e z^{b+} satisfazem $z^{b+} \geq z^{a+}$ então todos os planos de corte introduzidos durante a resolução de $P_{z^{a+}}^{\infty}$ são desigualdades válidas para $P_{z^{b+}}^{\infty}$.

Prova: Qualquer solução $(\tilde{x}, \tilde{\alpha})$ admissível para $P_{z^{b+}}^{\infty}$ satisfaz $\tilde{x} \in X$ e $c^i \tilde{x} + \tilde{\alpha} \geq z_i^{b+}$, $i=1,\ldots,k$. Como $z_i^{b+} \geq z_i^{a+}$ para todo o i, $(\tilde{x}, \tilde{\alpha})$ é também admissível para $P_{z^{a+}}^{\infty}$. Logo, o conjunto admissível de $P_{z^{b+}}^{\infty}$ é um subconjunto do de $P_{z^{a+}}^{\infty}$, pelo que a proposição é verdadeira.

Nota: A proposição III.6 é válida também para $P_{z^+}^{\infty,\rho}$.

Em resumo, podemos concluir o seguinte:

(i) Para obter soluções não dominadas que melhorem uma dada função objectivo, relativamente à solução não dominada anterior, pode usar-se um processo que se baseia no incremento da correspondente componente do ponto de referência mantendo as outras componentes iguais (proposição III.4). Um resultado semelhante foi estabelecido para PLMO por METEV E YORDANOVA (1993).

(ii) Podemos usar apenas pontos de referência inteiros (e incrementos inteiros) sem que haja perda de soluções não dominadas intermédias (proposição III.5). Isto conduz a que todas as variáveis dos programas escalarizantes $P_{z^+}^{\infty}$ e $P_{z^+}^{\infty,\rho}$ sejam inteiras e que estes sejam tratados como inteiros puros (apenas o valor da função objectivo de $P_{z^+}^{\infty,\rho}$ é não inteiro o que não traz dificuldades acrescidas).

(iii) Considerando (i) e (ii), os planos de corte de Gomory que forem introduzidos durante a resolução de um programa escalarizante são desigualdades válidas para o programa escalarizante seguinte (proposição III.6). Não obstante, estas desigualdades são em geral mais fracas para o novo problema.

Além dos planos de corte de Gomory, usamos também desigualdades de cobertura mínima estendidas para problemas de PLIMO com variáveis 0-1. Como os programas escalarizantes de Tchebycheff não são 0-1 puros, estas desigualdades são sempre introduzidas a partir das restrições do problema de PLIMO, ou seja, apenas as restrições $Ax = b$ são usadas para gerar desigualdades de cobertura mínima estendidas.

III.3.2 Descrição do método

As principais características do método interactivo que propomos para problemas de PLIMO são as seguintes:

- **Utiliza um protocolo simples para interagir com o AD** que requer, em cada interacção, a indicação de um novo ponto de referência ou apenas um objectivo que o AD deseje melhorar face à solução não dominada anterior. Neste último caso, prossegue-se com uma *pesquisa direccional* (tirando partido do resultado da proposição III.4) em que as soluções não dominadas são sucessivamente melhores para o objectivo escolhido. A solução obtida em cada interacção de uma pesquisa direccional é a que está mais próxima da anterior segundo uma trajectória particular. Por esta razão dizemos que estas soluções são 'próximas' e são 'consecutivas' na pesquisa direccional.
- **Identifica intervalos para as componentes do ponto de referência que conduzem à mesma solução não dominada**, o que permite o ajuste automático do ponto de

referência durante uma pesquisa direccional. Para tal, é feita uma *análise de sensibilidade* a uma componente do ponto de referência no programa escalarizante.

- **Aproveita cálculos anteriores para prosseguir no cálculo de outras soluções não dominadas**. Os sucessivos programas escalarizantes de uma pesquisa direccional são resolvidos de forma "encadeada" utilizando-se um processo cumulativo de introdução de *planos de corte* (tendo em conta o resultado da proposição III.6).
- **Trabalha com pontos de referência inteiros**, o que não é uma limitação porque não implica perda de soluções não dominadas (proposição III.5) e conduz a programas escalarizantes inteiros puros (a menos da função objectivo), uma característica importante para efeitos de análise de sensibilidade e uso de planos de corte.

III.3.2.1 Algoritmo

O método começa por sugerir ao AD um ponto de referência z^+ para projectar no conjunto das soluções não dominadas (*passo* 0). Por uma questão de simplicidade de cálculo, o ponto de referência sugerido é o ponto ideal da relaxação linear truncado para valores inteiros. O AD pode alterar esse ponto de referência (*passo 1*). Nesta altura, ou em qualquer outra fase do processo de decisão, o AD pode escolher um ponto de referência qualquer, atingível ou não. No entanto, como o programa escalarizante usado para projectar o ponto de referência é o programa de Tchebycheff $P_{z^+}^{\infty,\rho}$, o ponto de referência poderá ter que ser ajustado (*passo 2*) de modo a verificar $z^+ \geq z^*$.

> Nota: Trata-se de um ajuste cujo interesse é meramente técnico. Tem a vantagem de permitir que se restrinja a variável α do programa escalarizante a valores não negativos. Se este ajuste não fosse feito, poder-se-ia usar o z^+ original desde que se definisse α como variável livre. Nesse caso, estaríamos a usar um programa escalarizante de realização *min-max*.

Após o cálculo de uma solução não dominada através da resolução de $P_{z^+}^{\infty,\rho}$ (*passo 3*), o AD pode optar por indicar explicitamente outros pontos de referência (regressando ao *passo 1*) ou por fazer uma *pesquisa direccional,* indicando, para o efeito, uma função objectivo z_p que deseja melhorar relativamente à solução não dominada que lhe foi apresentada (*passo 4*). Neste último caso, o ponto de referência é actualizado automaticamente, incrementando-se z_p^+ através de um processo iterativo de análise de sensibilidade e re-optimização de $P_{z^+}^{\infty,\rho}$ (*passo* 5). O processo termina com uma nova solução não dominada que melhora z_p face à anterior. O AD pode continuar a pesquisar soluções segundo a mesma direcção (continuar no *passo* 5), mudar de direcção (*passo 4*) ou diversificar a pesquisa escolhendo directamente outros pontos de referência

(*passo 1*). Termina quando o AD considerar que encontrou uma solução de compromisso satisfatória.

O diagrama da figura III.1 esquematiza o algoritmo proposto.

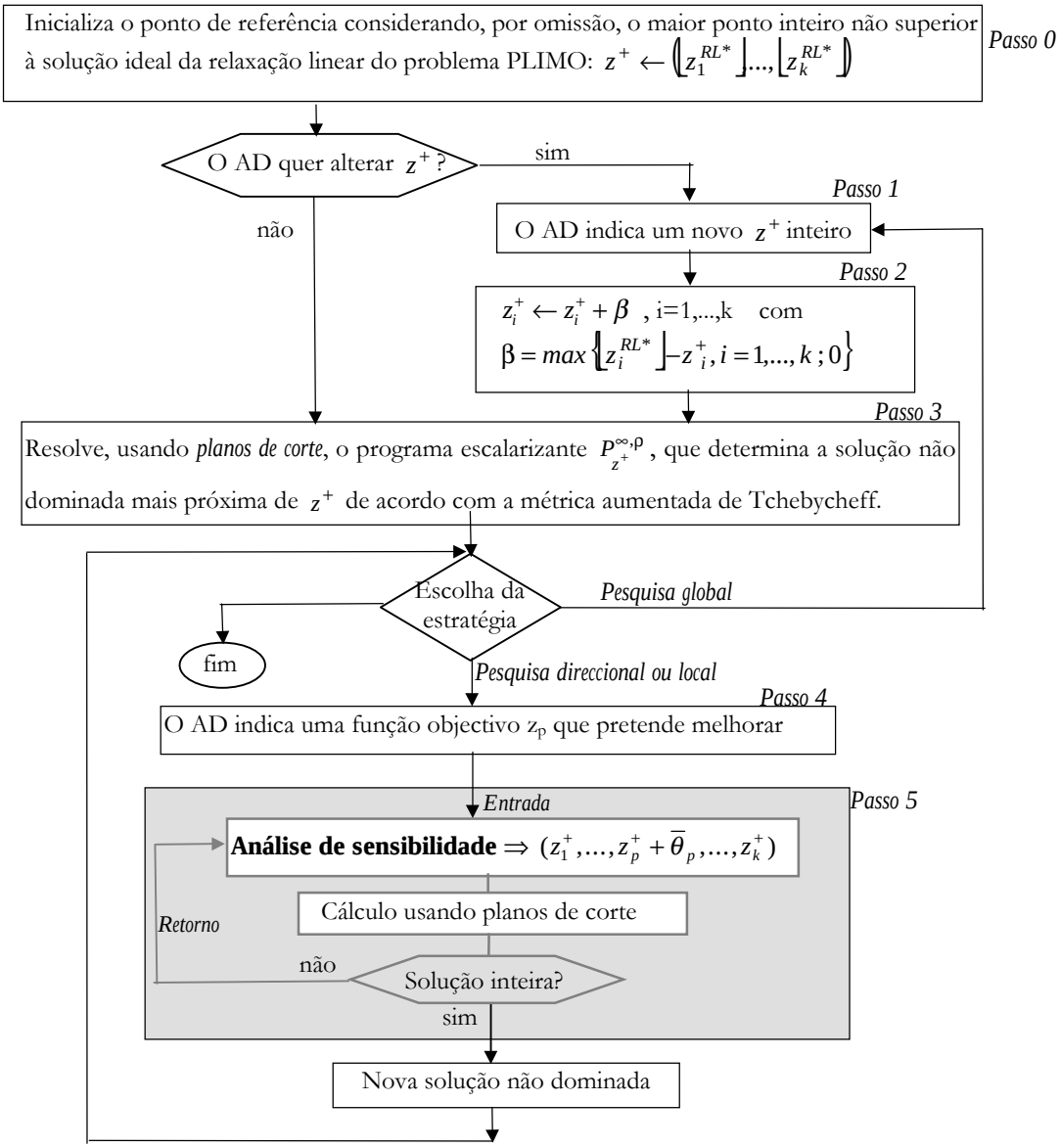


Fig. III.1 – Algoritmo do método interactivo para PLIMO baseado em planos de corte.

O algoritmo anterior representa apenas uma proposta no que diz respeito ao protocolo de interacção com o AD. A parte principal é o modo como se efectuam as pesquisas direccionais, designadamente o processo iterativo do *passo 5* que envolve fases de análise de sensibilidade e re-optimização. Este processo será descrito a seguir.

III.3.2.2 Processo iterativo de análise de sensibilidade e re-optimização

De cada vez que o AD especifica uma função objectivo z_p que pretende melhorar, seria desejável conhecer o maior valor inteiro que, somado à componente p do actual ponto de referência, conduziria ainda à solução não dominada actual. Denotando esse valor por θ_p^{max}, significaria então que, para $\bar{\theta}_p = \theta_p^{max} + 1$, teríamos a garantia de encontrar uma solução não dominada 'próxima' mas diferente da anterior, e com valor superior para z_p (de acordo com a proposição III.4). Mas, infelizmente, não é conhecido nenhum processo de análise de sensibilidade que dê resposta a esta questão de uma forma computacionalmente simples. Por isso, propomos um procedimento que aproxima iterativamente o valor de θ_p^{max}, e consequentemente $\bar{\theta}_p$, até se encontrar uma nova solução não dominada.

Por uma questão de clareza de exposição, e sem perda de generalidade, consideremos o problema $P_{z^+}^{\infty}$ em vez de $P_{z^+}^{\infty,\rho}$. Relembramos que a função objectivo do primeiro é simplesmente a minimização de α, variável que representa a distância de Tchebycheff (L_∞) do ponto do espaço dos critérios ao ponto de referência.

Seja $s_i \geq 0$ a variável desvio da restrição $c^i x + \alpha \geq z_i^+$, $i \in \{1,\ldots,k\}$ em $P_{z^+}^{\infty}$. Como z^+ e c^i, i=1,...,k, são vectores inteiros, então as variáveis s_i assumem apenas valores inteiros.

Suponhamos que o ponto de referência inteiro actual é z^{a+} e x^a a solução eficiente actual (com z^a=Cx^a o ponto não dominado); (x^a,α^a) é uma solução óptima de $P_{z^{a+}}^{\infty}$ para a qual $s_i = s_i^a$, i=1,…,k.

Designemos por $z^{a+} + \theta_p$ o vector $\left(z_1^{a+},\ldots, z_p^{a+} + \theta_p,\ldots, z_k^{a+}\right)$ e consideremos ainda que $(P_{z^{a+}}^{\infty})^{\text{RL+PC}}$ representa o programa obtido pela introdução de planos de corte na relaxação linear de $P_{z^{a+}}^{\infty}$ de modo a que (x^a,α^a) também minimiza $(P_{z^{a+}}^{\infty})^{\text{RL+PC}}$.

Utilizaremos a notação $\lceil \bullet \rceil$ para designar o menor número inteiro não inferior a $\bullet$.

No processo iterativo de análise de sensibilidade há duas situações distintas a considerar: a *entrada*, isto é, a primeira iteração do processo em que a solução actual (e de partida) é inteira; o *retorno*, que representa uma outra qualquer iteração, em que a solução não é inteira.

ENTRADA:

Se $\boldsymbol{s_p^a > 0}$ ($s_p^a = \alpha^a - z_p^{a+} + z_p^a$) então,

(x^a, α^a) minimiza $P^{\infty}_{z^{a+}+\theta_p}$ para $0 \le \theta_p \le s_p^a$ (de acordo com a prova da proposição III.4 – *CASO 2.1*). Neste caso, consideramos $\theta_p^{max} = s_p^a$, $\overline{\theta}_p = \theta_p^{max} + 1$ e $z^{b+} = z^{a+} + \overline{\theta}_p$.

Sabemos ainda que a distância L_∞ da solução não dominada anterior z^a ao novo ponto de referência z^{b+} é $\alpha^a + 1$, porque esta distância é dada por $\max \{ \max_{i \neq p} (z_i^{a+} - z_i^a),\ z_p^{a+} + s_p^a + 1 - z_p^a \} = \max \{ \alpha^a,\ \alpha^a + 1 \} = \alpha^a + 1$. Por outro lado, o valor óptimo de α em $P^{\infty}_{z^{b+}}$ será sempre maior ou igual a α^a, visto que z^{b+} é um ponto mais afastado do que z^{a+}. Consequentemente, o valor óptimo de α em $P^{\infty}_{z^{b+}}$, seja α^b, satisfaz

$$\alpha^a \le \alpha^b \le \alpha^a + 1$$

o que é equivalente a $\alpha^b = \alpha^a \ \vee\ \alpha^b = \alpha^a + 1$, porque α assume apenas valores inteiros.

Se $\boldsymbol{s_p^a = 0}$ então,

uma variação $\theta_p > 0$ em $(P^{\infty}_{z^{a+}})^{\text{RL+PC}}$ implica que o valor óptimo de α neste programa linear – denotado por $(P^{\infty}_{z^{a+}+\theta_p})^{\text{RL+PC}}$ – seja igual ou superior a $\alpha^a + \pi_p^a\, \theta_p$, em que a igualdade se verifica para valores de θ_p que mantêm admissível a base actual; π_p^a denota o valor óptimo da variável dual π_p associada à restrição p de $(P^{\infty}_{z^{a+}})^{\text{RL+PC}}$. Como $(P^{\infty}_{z^{a+}+\theta_p})^{\text{RL+PC}}$ é uma relaxação de $P^{\infty}_{z^{a+}+\theta_p}$ então, qualquer que seja $\theta_p > 0$, $\alpha^a + \pi_p^a\, \theta_p$ constitui um limite inferior para o valor óptimo de α em $P^{\infty}_{z^{a+}+\theta_p}$ (relembramos que os problemas são a minimizar). Como apenas se usam pontos de referência inteiros, o que implica que α seja uma variável inteira, então este limite inferior pode ser ajustado para $\alpha^a + \lceil \pi_p^a \theta_p \rceil$. Por outro lado, ($x^a, \alpha^a + \theta_p$) é admissível para $P^{\infty}_{z^{a+}+\theta_p}$. Logo, $\alpha^a + \theta_p$ é um limite superior para o valor óptimo de α em $P^{\infty}_{z^{a+}+\theta_p}$. Podemos, portanto, concluir que x^a minimiza $P^{\infty}_{z^{a+}+\theta_p}$ para pelo menos os valores de θ_p (inteiros) que satisfazem $\lceil \pi_p^a \theta_p \rceil = \theta_p$, o que leva a considerar $\theta_p^{max} = \max \{ \theta_p \text{ inteiro} \mid \lceil \pi_p^a \theta_p \rceil = \theta_p \}$, $\overline{\theta}_p = \theta_p^{max} + 1$ e

$z^{b+} = z^{a+} + \overline{\theta}_p$. Como o intervalo de valores admissíveis para π_p é [0,1], a lista seguinte dá os valores de θ_p^{max} em função de π_p^a:

$$\begin{array}{l|ll}
\theta_p^{max} = 0 & \Leftarrow & \pi_p^a = 0 \\
\theta_p^{max} = 1 & \Leftarrow & 0 < \pi_p^a \leq 1/2 \\
\theta_p^{max} = 2 & \Leftarrow & 1/2 < \pi_p^a \leq 2/3 \\
\ldots & & \\
\theta_p^{max} = t & \Leftarrow & t-1/t < \pi_p^a \leq t/t+1 \\
\ldots & & \\
\theta_p^{max} \to +\infty & \Leftarrow & \pi_p^a = 1
\end{array}$$

Concluímos ainda que o valor óptimo de α em $P^{\infty}_{z^{b+}}$, seja α^b, satisfaz

$$\alpha^a + \overline{\theta}_p - 1 \leq \alpha^b \leq \alpha^a + \overline{\theta}_p$$

o que é equivalente a $\alpha^b = \alpha^a + \overline{\theta}_p - 1 \vee \alpha^b = \alpha^a + \overline{\theta}_p$, porque α assume apenas valores inteiros.

Um caso particular é quando $\pi_p^a = 1$, em que $\theta_p^{max} = \infty$. Significa, neste caso, que x^a é a solução eficiente que maximiza a função objectivo z_p do problema PLIMO.

Conclusão: Em ambos os casos ($s_p^a > 0$ ou $s_p^a = 0$), determinamos um valor $\overline{\theta}_p$ tal que todos os pontos de referência $z^{a+} + \theta_p$, com θ_p inteiro e menor do que $\overline{\theta}_p$, conduzem à solução não dominada anterior. Como o resultado para $z^{b+} = z^{a+} + \overline{\theta}_p$ é ainda desconhecido, este será o ponto de referência seguinte. Desta análise retiramos um limite superior *U* para o valor óptimo de α em $P^{\infty}_{z^{b+}}$, que é $\alpha^a + 1$ quando $s_p^a > 0$, e $\alpha^a + \overline{\theta}_p$ quando $s_p^a = 0$. O limite superior representa a distância L_∞ da solução não dominada anterior, z^a, ao novo ponto de referência z^{b+}.

Após determinado o ponto de referência, z^{b+}, altera-se o quadro simplex actualizando a base e a solução correspondente. Suponhamos que a solução resultante é $(\overline{x}, \overline{\alpha})$ que satisfaz as condições de admissibilidade e de optimalidade para o problema linear representado no quadro. Se $\overline{\alpha}$ não for inteiro, regressa-se à fase de análise de sensibilidade (*retorno*). Se $\overline{\alpha}$ for inteiro mas $\overline{x}$ violar alguma condição de integralidade, então são introduzidos novos planos de corte. Este processo de introdução de novos cortes, e eliminação dos antigos que deixam de estar activos, repete-se até se encontrar uma nova solução inteira (x^b), ou $\overline{\alpha}$ tomar um valor não inteiro. Neste último caso, regressa-se à análise de sensibilidade (*retorno*).

RETORNO:

Nesta altura é conhecido U, limite superior (inteiro) para α^b em $P^{\infty}_{z^{b+}}$, tal que $\alpha^b = U-1 \vee \alpha^b = U$. Esta informação não foi incluída no quadro simplex, ela apenas auxilia a análise de sensibilidade.

Como $\overline{\alpha}$ não é inteiro, então $\lceil \overline{\alpha} \rceil = U$ e podemos concluir que x^a ainda optimiza $P^{\infty}_{z^{b+}}$. Perante esta situação, θ_p^{max} pode ser incrementado de 1 ou de um valor superior, determinado de acordo com o seguinte:

Designemos por δ_p um aumento (inteiro) face ao θ_p^{max} anterior. Seja $\overline{\pi}_p$ o valor da variável dual π_p associada à restrição p do programa linear do quadro simplex actual. Podemos, então, estabelecer um limite inferior e um limite superior para o valor óptimo α em $P^{\infty}_{z^{b+}+\delta_p}$. O limite inferior é $\lceil \overline{\alpha} + \overline{\pi}_p \delta_p \rceil$ e o limite superior é $U+\delta_p$. Assim, para valores (inteiros) de δ_p que satisfaçam $\lceil \overline{\alpha} + \overline{\pi}_p \delta_p \rceil = U+\delta_p$, há a garantia de que x^a ainda optimiza $P^{\infty}_{z^{b+}+\delta_p}$.

Por esta razão, θ_p^{max}, $\overline{\theta}_p$ e, consequentemente, a componente p de z^{b+} são incrementados de $\delta_p^{max} = 1 + \max \{ \delta_p \text{ inteiro} \mid \lceil \overline{\alpha} + \overline{\pi}_p \delta_p \rceil = U + \delta_p \}$. O limite superior U é actualizado para $U+\delta_p^{max}$.

Nota: Se usarmos $P^{\infty,\rho}_{z^+}$ em vez de $P^{\infty}_{z^+}$, devemos substituir o valor da variável dual π_p por um outro ligeiramente diferente, que designaremos por π'_p, e que é dado por $(-\overline{a}_{\alpha,s_p})$, i.e. o simétrico do elemento que está no cruzamento da linha de α, com a coluna de s_p no quadro simplex actualizado. A análise de sensibilidade avalia o efeito produzido em α e não no valor da função objectivo do programa escalarizante.

III.3.3 Exemplo ilustrativo

Nesta secção ilustraremos, através de um pequeno exemplo, o funcionamento do método interactivo dando particular destaque ao processo de análise de sensibilidade e re-optimização que acabámos de descrever. O exemplo escolhido é um problema biobjectivo com duas variáveis inteiras:

$$\max z_1 = x_1 - x_2$$

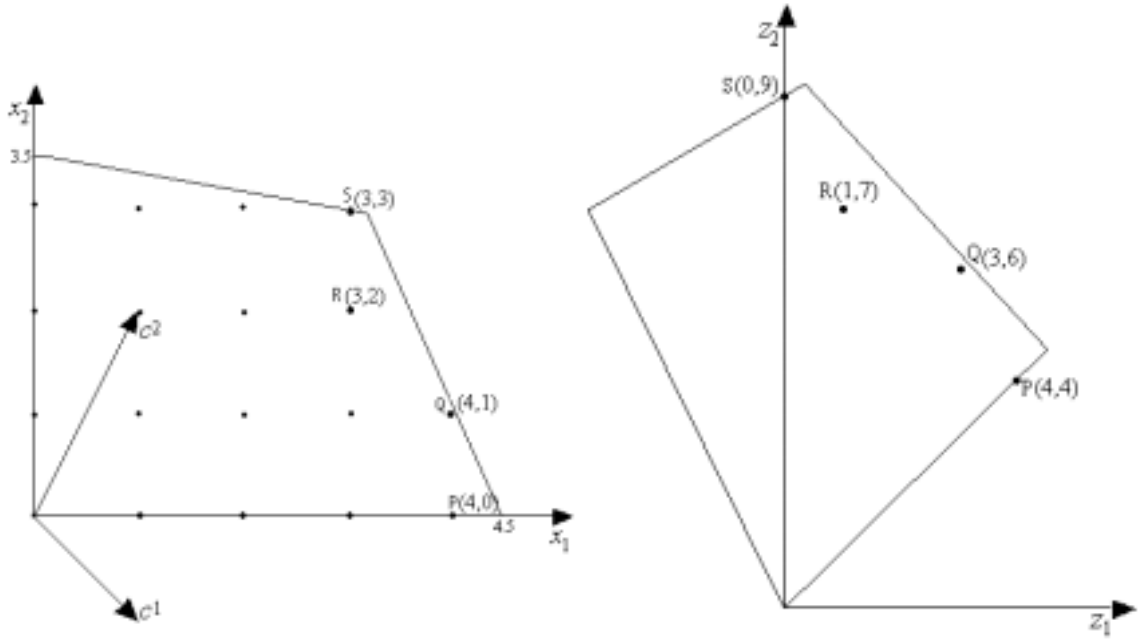
$$\max z_2 = x_1 + 2x_2$$

$$\text{s.a:} \quad x_1 + 6x_2 \leq 21$$

$$14x_1 + 6x_2 \leq 63$$

$$x_1,\ x_2 \geq 0 \text{ e inteiras}$$

A figura III.2 mostra as soluções eficientes (P, Q, R e S) no espaço das variáveis de decisão *(a)* e os correspondentes pontos não dominados no espaço dos objectivos *(b)*.



(a) *(b)*

Fig. III.2 – Espaço das variáveis de decisão e espaço dos objectivos do exemplo.

A aplicação de um fase de pré-processamento (para mais detalhes, ver a secção III.3.4) a este problema tem como efeito acrescentar limites superiores nas duas variáveis de decisão: $x_1 \leq 4$ e $x_2 \leq 3$.

O programa escalarizante usado é $P_{z^+}^{\infty,\rho}$ (considerando ρ=0.001) cuja formulação é a seguinte (em que todas as restrições já foram convertidas em igualdades):

$$\min \{\alpha - 0.002x_1 - 0.001x_2\}$$

s.a: $x_1 - x_2 + \alpha - s_1 = z_1^+$

$x_1 + 2x_2 + \alpha - s_2 = z_2^+$

$x_1 + 6x_2 + x_3 = 21$

$14x_1 + 6x_2 + x_4 = 63$

$0 \le x_1 \le 4$

$0 \le x_2 \le 3$

$\alpha, s_1, s_2, x_3, x_4 \ge 0$

todas as variáveis são inteiras

Para um dado ponto de referência $z^+=(z_1^+, z_2^+)$ inteiro, o programa escalarizante $P_{z^+}^{\infty,\rho}$ será resolvido usando planos de corte de Gomory e o método dual-simplex para variáveis limitadas. Como é sabido, no método simplex ou dual-simplex para variáveis limitadas, as variáveis não básicas podem tomar o valor do seu limite inferior ou do limite superior. Consequentemente, a expressão de um plano de corte de Gomory construído a partir de um quadro onde existem variáveis não básicas com valor não nulo é diferente da apresentada na secção III.1.2. Omitiremos aqui a sua dedução, apresentando apenas a expressão final:

Sejam NB^L e NB^U os conjuntos dos índices das variáveis não básicas, respectivamente nos limites inferiores (L_j) e nos limites superiores (U_j).

Seja $x_{B_i} + \sum_{j \in NB^L \cup NB^U} \bar{a}_{ij} x_j = \bar{b}_i$ a linha i do quadro simplex correspondente à variável básica x_{B_i} cujo valor é $\bar{x}_{B_i} = \bar{b}_i - \sum_{j \in NB^L} \bar{a}_{ij} L_j - \sum_{j \in NB^U} \bar{a}_{ij} U_j$. Se $\bar{x}_{B_i}$ não for inteiro, então podemos construir um corte de Gomory a partir da linha i cuja expressão é

$$\sum_{j \in NB^L} \langle \bar{a}_{ij} \rangle x_j - \sum_{j \in NB^U} \langle -\bar{a}_{ij} \rangle x_j \quad \ge \quad \langle \bar{x}_{B_i} \rangle + \sum_{j \in NB^L} \langle \bar{a}_{ij} \rangle L_j - \sum_{j \in NB^U} \langle -\bar{a}_{ij} \rangle U_j ,$$

em que $\langle \bar{a}_{ij} \rangle = \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor$.

Suponhamos que o ponto de referência escolhido para iniciar a pesquisa é $\boldsymbol{z^{a+}=(6,10)}$. O quadro simplex óptimo da relaxação linear de $P_{z^+}^{\infty,\rho}$ para (6,10) é:

$\bar{x}_{NB}$	0	0	0	0	
	x4	s2	s1	$\bar{b}$	$\bar{x}_B$
α	-0.07143	-0.47619	-0.52381	3.40476	3.40476
x2	0	-0.33333	0.33333	1.33333	1.33333
x3	-0.07143	1.85714	-1.85714	9.071428	9.071428
x1	0.07143	0.14286	-0.14286	3.92857	3.92857
c-z	0.071571	0.476143	0.523857		

Como a solução não é inteira, introduzem-se planos de corte até ser alcançada uma solução inteira. A estratégia adoptada consiste em introduzir vários cortes de cada vez, um corte por cada variável cuja parte fraccionária seja superior a um determinado limiar, por exemplo, 0.1. Assim, no quadro anterior são introduzidos 3 cortes definidos a partir das linhas de α, x_2 e x_1:

$\bar{x}_{NB}$	0	0	0	0	
	x4	s2	s1	$\bar{b}$	$\bar{x}_B$
α	-0.07143	-0.47619	-0.52381	3.40476	3.40476
x2	0	-0.33333	0.33333	1.33333	1.33333
x3	-0.07143	1.85714	-1.85714	9.071428	9.071428
x1	0.07143	0.14286	-0.14286	3.92857	3.92857
y1	-0.92857	-0.52381	-0.47619	-0.40476	-0.40476
y2	0	-0.66667	-0.33333	-0.33333	-0.33333
y3	-0.07143	-0.14286	-0.85714	-0.928571	-0.928571
c-z	0.071571	0.476143	0.523857		

Os 3 cortes escritos em função das varáveis principais têm as seguintes formulações:

$$12x_1 + 5x_2 - \alpha \leq 50 \qquad \text{(variável desvio } y_1\text{)}$$

$$x_1 + x_2 + \alpha \geq 9 \qquad \text{(variável desvio } y_2\text{)}$$

$$-x_2 + \alpha \geq 3 \qquad \text{(variável desvio } y_3\text{)}$$

Após duas iterações do dual-simplex, o quadro tem a seguinte configuração:

$\bar{x}_{NB}$	**4**	0	0	0	
	x1	s2	y3	$\bar{b}$	$\bar{x}_B$
α	0.33333	-0.33333	-0.66667	5.33333	**4**
x2	0.33333	-0.33333	0.33333	2.33333	**1**
x3	-1	2	-2	7	**11**
x4	12	2	-2	49	**1**
y1	10.66667	1.33333	-2.33333	43.66667	1
y2	-0.33333	-0.66667	-0.33333	-1.33333	0
s1	-1	0	-1	-3	1
c-z	-0.33500	0.33300	0.66700		

Os planos de corte correspondentes a y_1 e y_2 não são necessários e podem ser eliminados.

Este é o **ponto Q** na figura III.2: x^a=(4, 1,11,1); s^a=(1,0); α^a=4; z^a=(3,6).

Suponhamos que o AD pretende conhecer soluções não dominadas que sejam melhores do que esta em z_1. Para o efeito, o procedimento vai aumentar a 1ª componente do ponto de referência. Neste momento o ponto de referência é (6,10) e passará a ser $(6+\bar{\theta}_1,10)$ com $\bar{\theta}_1 = \theta_1^{max} + 1$, em que θ_1^{max} é um valor inteiro de que temos a garantia que ainda conduz à solução eficiente anterior.

Análise de sensibilidade – entrada: estamos perante uma situação em que s_1 é básica pelo que uma alteração do termo independente da 1ª restrição apenas altera o valor de s_1 na solução básica actual. Logo, para o ponto de referência (7,10), s_1=0 e todas as outras variáveis mantêm os seus valores, verificando-se que z^a é uma solução não dominada que minimiza a distância L_∞ a este ponto de referência. Para o ponto de referência (8,10), s_1= –1 e esta base deixa de ser admissível. Em suma, como $s_1^a = 1$, então $\theta_1^{max} = s_1^a = 1$, $\bar{\theta}_1 = 2$ e o novo ponto de referência será $\boldsymbol{z^{b+}=(8,10)}$. Sabemos, desde já, que os possíveis valores óptimos para α serão: $\boldsymbol{\alpha^b=4 \vee \alpha^b=5}$. Feita a actualização do último quadro simplex para o novo ponto de referência, o quadro resultante é o seguinte:

$\bar{x}_{NB}$	4	0	0	0	
	x1	s2	y3	$\bar{b}$	$\bar{x}_B$
α	0.33333	-0.33333	-0.66667	5.33333	4
x2	0.33333	-0.33333	0.33333	2.33333	1
x3	-1	2	-2	7	11
x4	12	2	-2	49	1
s1	-1	0	(-1)	-5	-1 ↵
c-z	-0.33500	0.33300	0.66700		

$\bar{x}_{NB}$	4	0	0	0	
	x1	s2	s1	$\bar{b}$	$\bar{x}_B$
α	1	-0.33333	-0.66667	8.66667	4.66667
x2	0	-0.33333	0.33333	0.66667	0.66667
x3	1	2	-2	17	13
x4	14	2	-2	59	3
c-z	-1.00200	0.33300	0.66700		

Ainda não foi alcançada uma solução inteira mas já se consegue perceber que a solução eficiente anterior, x^a, optimiza $P^{\infty}_{z^{b+}}$ para z^{b+}=(8,10), porque $\alpha^b \geq \lceil 4.66667 \rceil$=5, sendo 5 também o seu limite superior.

Análise de sensibilidade – retorno: sabemos que $\pi'_1 = -\bar{a}_{\alpha,s_1}$ =0.66667 (simétrico do elemento que está na intersecção da linha de α com a coluna de s_1, e que é ligeiramente diferente de π_1=0.667); a solução eficiente anterior mantém-se para os pontos de referência (8+δ_1,10), com $\delta_1 \geq 0$ e inteiro, tais que $\lceil 4.66667+0.66667\delta_1 \rceil = 5+\delta_1$; o maior valor de δ_1 nestas circunstâncias é 1, pelo que se conclui que a solução não dominada z^a minimiza a distância L_∞ aos pontos de referência (8,10) e (9,10), correspondentes a δ_1=0 e δ_1=1, respectivamente. O ponto de referência z^{b+} é então ajustado para (**10,10**) e os valores possíveis para o óptimo de α^b são agora **6** ou **7**.

O quadro simplex anterior actualizado para o ponto de referência z^{b+}=(10,10) é o seguinte:

$\bar{x}_{NB}$	**4**	0	0	0	
	x1	s2	s1	$\bar{b}$	$\bar{x}_B$
α	1	-0.33333	-0.66667	10	**6**
x2	0	-0.33333	0.33333	0	**0**
x3	1	2	-2	21	**17**
x4	14	2	-2	63	**7**
c-z	-1.00200	0.33300	0.66700		

Foi atingida uma nova solução não dominada que, como se esperava, melhora z_1 relativamente à solução anterior. Este é o **ponto P** na figura III.2: x^b=(4, 0, 17, 7); s^b=(0,0); α^b=6; z^b=(4, 4).

Suponhamos que o AD pretende continuar a conhecer soluções não dominadas que melhorem z_1. Há então que determinar $\bar{\theta}_1 = \theta_1^{max} + 1$, inteiro positivo, para formar o novo ponto de referência z^{c+}=(10+ $\bar{\theta}_1$,10).

Análise de sensibilidade – entrada: $\lceil 0.66667\ \theta_1 \rceil = \theta_1$ verifica-se para θ_1=1 e θ_1=2, logo os pontos de referência (11,10) e (12,10) conduzem a x^b. Então, θ_1^{max} =2 e $\bar{\theta}_1$ =3, e o procedimento prossegue com o ponto de referência $\boldsymbol{z^{c+}=(13,10)}$. Começa por fazer a devida actualização no quadro simplex. Sabemos agora que $\boldsymbol{\alpha^c=8 \vee \alpha^c=9}$.

$\bar{x}_{NB}$	4	0	0	0	
	x1	s2	s1	$\bar{b}$	$\bar{x}_B$
α	1	-0.33333	-0.66667	12	8
x2	0	-0.33333	0.33333	-1	-1 ↵
x3	1	2	-2	27	23
x4	14	2	-2	69	13
c-z	-1.00200	0.33300	0.66700		

$\bar{x}_{NB}$	4	0	0	0	
	x1	x2	s1	$\bar{b}$	$\bar{x}_B$
α	1	-1	**-1**	13	9
s2	0	-3	-1	3	3
x3	1	6	0	21	17
x4	14	6	0	63	7
c-z	-1.00200	0.99900	1		

Este é ainda o ponto P. Estamos perante uma situação em que π_1' (igual a π_1) é **1**, o que significa que, qualquer que seja o ponto de referência ($13+\theta_1$, 10), o respectivo valor óptimo de α satisfará $9+ \mathbf{1}.\theta_1 \leq \alpha \leq 9+\theta_1$ (o limite superior é determinado pela solução anterior, x^b). Por isso concluímos que, qualquer que seja o aumento da 1ª componente do ponto de referência, a solução eficiente obtida será sempre x^b (ponto P). Esta é a solução eficiente que optimiza z_1 no problema PLIMO.

III.3.4 Implementação computacional e testes

O método interactivo proposto para problemas de PLIMO foi implementado no ambiente de desenvolvimento DELPHI para Windows 95/98 num PENTIUM II (350MHz). Esta aplicação computacional inclui um editor de problemas em formato de 'folha de cálculo', procedimentos gráficos para interagir com o utilizador e procedimentos de cálculo. Neste último grupo insere-se o pré-processamento de problemas, geração de planos de corte, análise de sensibilidade e a resolução de problemas de PL através do algoritmo dual-simplex para variáveis limitadas.

O pré-processamento de problemas consiste em efectuar recursivamente um conjunto de testes e regras simples que permitem, em alguns casos, reduzir o problema, designadamente no que se refere a: (i) remover restrições redundantes; (ii) substituir restrições por limites de variáveis; (iii) fixar variáveis; (iv) remover ou estreitar limites de variáveis; (v) reduzir coeficientes. A implementação dos primeiros 4 itens baseia-se em BREARLY ET AL. (1975) com algumas

adaptações para problemas multiobjectivo. O item (v) segue de perto JOHNSON ET AL. (1985) e é apenas aplicável a problemas de PLIMO 0-1.

A estratégia adoptada para a inserção de planos de corte na resolução dos programas escalarizantes é a seguinte:

• Se o problema de PLIMO for 0-1, tenta-se encontrar desigualdades de cobertura mínima estendidas que sejam violadas pela solução actual (segundo o processo descrito no anexo III.A). Se não existir nenhuma, introduzem-se planos de corte de Gomory. No caso dos problemas inteiros genéricos, introduzem-se apenas planos de corte de Gomory.

• São introduzidos vários planos de corte de Gomory em cada quadro simplex (um por cada variável com valor não inteiro). Esta forma de proceder segue a sugestão de BALAS ET AL. (1996[b]) e, experiências preliminares que efectuámos, revelaram resultados superiores aos da consideração de um só corte de cada vez (escolhido de acordo com várias heurísticas). Na nossa implementação excluímos, contudo, planos de corte que sejam violados pela solução actual por uma quantidade abaixo de um determinado limiar (desde que não sejam todos).

• As desigualdades de cobertura mínima estendidas e os planos de corte de Gomory são guardados temporariamente num 'depósito' de planos de corte activos. Antes de os guardar, os planos de corte são traduzidos para função das variáveis principais, o que possibilita refazer do início as estruturas de dados usadas no dual-simplex. Todas as restrições do 'depósito' têm coeficientes inteiros.

Foram incluídas duas implementações do (dual) simplex (com opção de variáveis limitadas) obtidas a partir de *software* disponível na 'Internet', e que podem ser usadas em alternativa. A primeira consiste numa tradução para a linguagem Pascal do código em C do *LP_SOLVE* (obtido a partir de ftp://ftp.ics.ele.yue.nl/pub/lp_solve/) e a outra foi retirada do código da aplicação *LP_OPTIMIZER* desenvolvida em ambiente DELPHI por Markus Weidenauer. O código fonte completo e informações sobre o desempenho deste *software* podem ser obtidos a partir de http://www.netcologne.de/~nc-weidenma/readme.htm. Este código é mais robusto que o primeiro, sendo por isso usado na maior parte dos testes computacionais que efectuámos.

Para uma melhor ilustração do funcionamento do sistema computacional, consideremos um problema hipotético de localização de estações de tratamento de lixo em que se pretende determinar os locais para instalar as estações, de entre 40 sítios potenciais. O problema tem 3 funções objectivo f_1, f_2 e f_3, a minimizar, que medem o custo, o risco e o número total de estações que cobrem cada região, respectivamente. As restrições seguem o modelo de cobertura em que todas as variáveis são 0-1.

O sistema começa por apresentar a solução ideal do problema relaxado linearmente, truncada a valores inteiros, $z^{RL*}=(1585\times10^3, 1565\times10^3, 42)$, como sugestão para um ponto de referência inicial. Suponhamos que, em vez deste, é escolhido o ponto de referência $z^{+}=(1585\times10^3, 1200\times10^3, 42)$ que privilegia f_2 (o risco) relativamente ao ponto de referência sugerido pelo sistema. A partir deste ponto de referência é calculada a primeira solução não dominada e apresentada na janela principal do sistema (figura III.3).

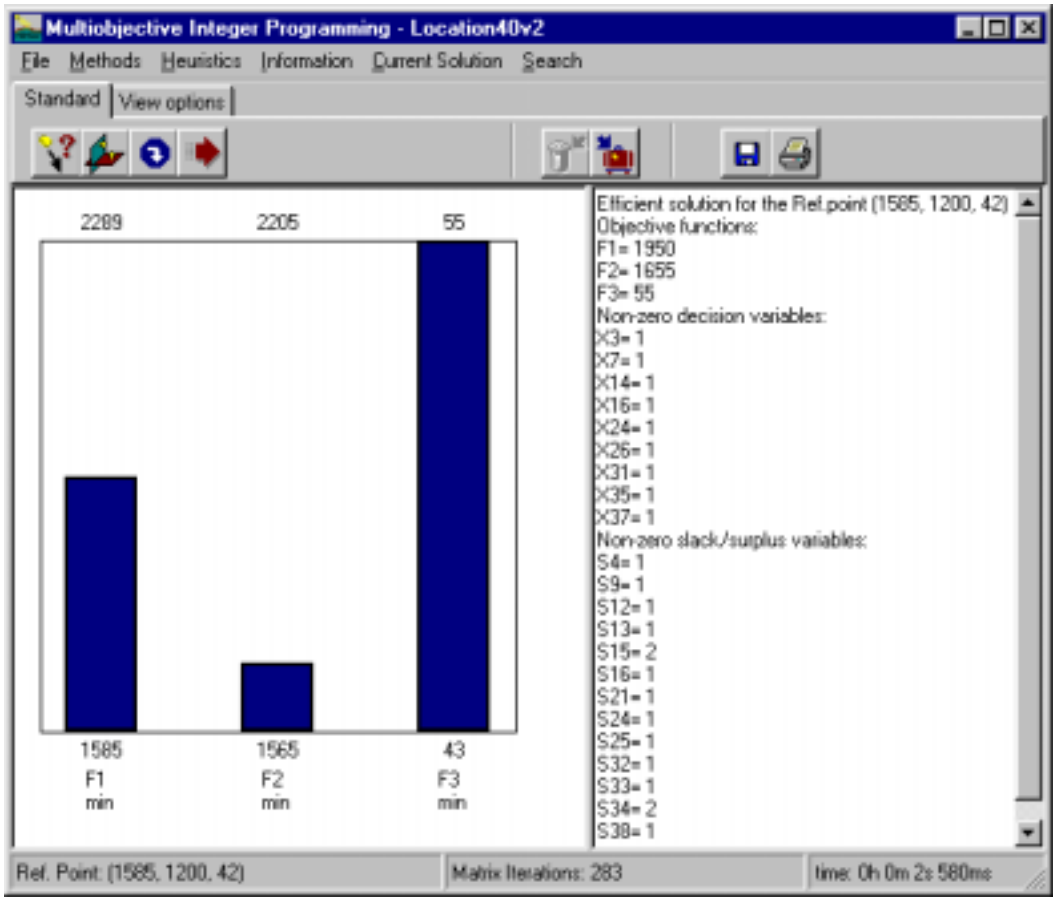


Fig. III.3 – Janela principal do sistema com a primeira solução do problema.

Na janela da figura III.3 há dois botões principais para conduzir uma pesquisa direccional, um para alterar a direcção,  (através da indicação da função objectivo que se pretende melhorar em cada instante), e outro para indicar que se deseja continuar a pesquisa na mesma direcção,  .

Suponhamos que se escolhe, em primeiro lugar, a diminuição de f_1 como direcção de pesquisa e, posteriormente, a diminuição de f_3. A tabela III.1 mostra os resultados obtidos desta pesquisa.

Interacção	*Estratégia*	*Ponto de referência* $(z_1^+ \times 10^3, z_2^+ \times 10^3, z_3^+)$	*Solução:* $(f_1 \times 10^3, f_2 \times 10^3, f_3)$	Estações abertas
(1)	• início	(1585, 1200, 42)	sol. 1: (1950,1655,55)	3, 7, 14, 16, 24, 26, 31, 35, 37
(2)	• diminuir f_1	⇒(1440, 1200, 42)	sol. 2: (1728, 1709, 49)	2, 7, 11, 24, 26, 31, 35, 37
(3)	• diminuir f_1	⇒ (1174, 1200, 42)	sol. 3: (1695, 1753, 48)	7, 12, 14, 24, 26, 31, 35, 37
(4)	• diminuir f_1	⇒ (1044, 1200, 42)	sol. 4: (1610, 1850, 52)	7, 11,12, 24, 26, 31, 35, 37
(5)	• diminuir f_3	⇒ (1044, 1200, -600)	Regressa à sol. 3	
(6)	• diminuir f_3	⇒ (1044, 1200, -609)	sol. 5: (1700, 1788, 46)	7, 12, 20, 24, 26, 34, 37

Tabela III.1 – Alguns resultados do problema de localização.

Relembramos que, excepto o primeiro ponto de referência, todos os outros são alterados automaticamente pelo sistema. Tal como referimos atrás na descrição do método, a solução obtida em cada interacção de uma *pesquisa direccional* é aquela que mais se aproxima da anterior segundo a trajectória seguida, que é uma trajectória de melhoramento de uma dada função objectivo previamente escolhida. Para este conceito de 'proximidade' das soluções, tem influência não só a diferença de valores no objectivo escolhido (em que esta diferença é sempre no sentido do melhoramento), mas também a diferença nos outros objectivos. Consequentemente, à excepção do caso biobjectivo, a solução seguinte de uma pesquisa direccional poderá não ser a solução eficiente que apresenta a menor diferença da solução anterior no objectivo escolhido para melhoramento. Observamos este facto nos resultados da tabela III.1, em que a solução 5 tem um valor de f_1 mais próximo da solução 2 do que a solução 3 e, no entanto, foi a solução 3 que se obteve na interacção (3).

A figura III.4 mostra uma outra janela do sistema com um gráfico de barras para as 5 soluções calculadas.

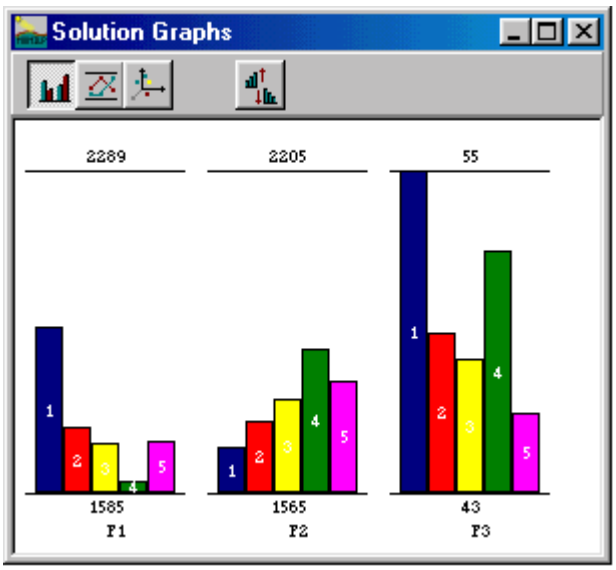


Fig. III.4 – Gráfico com as 5 soluções calculadas para o problema de localização.

Podemos fazer também uma análise técnica desta pesquisa. A tabela III.2 apresenta, para cada interacção da pesquisa direccional, a variação provocada automaticamente no ponto de

referência, o número de iterações de análise de sensibilidade que foram necessárias para essa actualização e o tempo computacional, que inclui o tempo gasto nas fases de análise de sensibilidade e nas de re-optimização. A última coluna da tabela apresenta o tempo computacional necessário para a resolução independente (também com planos de corte) do programa escalarizante final de cada interacção, ou seja, aquele a que corresponde o ponto de referência final dessa interacção (este processo é designado por *re-início*).

Como podemos observar, por vezes uma pequena variação do ponto de referência é o suficiente para que seja obtida uma nova solução – é caso da interacção (6) – e noutras vezes é necessária uma grande variação do ponto de referência para conseguir escapar da solução anterior – é o caso da interacção (5). Este facto evidencia a vantagem de o processo ser automático. Imaginemos como poderia ser difícil para o AD "descobrir" que teria de diminuir a 3ª componente do ponto de referência de 42 para –600, se mantivesse as outras componentes iguais, e pretendesse melhorar f_3 relativamente à solução 4. E se, após verificar a necessidade desta variação, o AD fizesse, por analogia, uma variação semelhante na interacção seguinte – seria levado directamente ao óptimo de f_3, um "salto" talvez maior do que aquele que ele desejaria.

Relativamente ao tempos de cálculo, o tempo de uma interacção da pesquisa direccional é, em média, superior ao de *re-início*. Contudo, não nos podemos alhear do facto dos primeiros incluírem a análise de sensibilidade e re-optimização. Além disso, estes tempos são sempre melhores do que o somatório dos tempos necessários para o *re-início* com todos os pontos de referência intermédios (ou mesmo uma parte deles) que a análise de sensibilidade fez evitar.

Pesquisa direccional: Interacção	*variação do ponto de referência (em valor absoluto)*	*número de iterações de análise de sensibilidade*	*tempo total (seg)*	*tempo de re-início com o ponto de referência final da interacção*
(2)	145 na 1ª componente	55	9.0	9.6
(3)	266 na 1ª componente	45	5.8	2.0
(4)	130 na 1ª componente	98	19.9	7.0
(5)	642 na 3ª componente	2	0.2	0.6
(6)	9 na 3ª componente	6	0.6	0.2

Tabela III.2 – Análise técnica da pesquisa feita para o problema de localização.

O principal ponto fraco desta abordagem é a instabilidade numérica que advém do uso das técnicas de planos de corte. Isto limita, na prática, a abordagem a problemas de pequenas dimensões.

Foram feitas outras experiências computacionais com problemas de PLIMO gerados aleatoriamente. Considerámos um total de 42 problemas dos seguintes tipos: problemas de mochila, i.e. *knapsack* 0-1 ('KnapS_'), *knapsack* com variáveis inteiras ('knapI_'), *knapsack* 0-1 multidimensional ('KnapM_' e 'Pet_'), problemas de cobertura ('Cover_'), e problemas de embalagem ('Pack_'). Estes problemas têm entre 10 e 100 variáveis, 1 a 40 restrições e 2 ou 3

funções objectivo, e não sofreram qualquer redução na fase de pré-processamento. As regras usadas na geração aleatória dos coeficientes dos problemas estão descritas no anexo III.B. Fizemos vários testes, quer de resolução independente dos programas escalarizantes para vários pontos de referência dados explicitamente, quer para pesquisas direccionais.

Fizemos 10 experiências de resolução independente do programa escalarizante monocritério para cada um dos problemas de PLIMO, considerando pontos de referência diferentes. A tabela III.3 indica, para cada problema, as respectivas dimensões sob o formato $n*m*k$ ('nº de variáveis*nº de restrições*nº de funções objectivo'), e sumaria os resultados, indicando o número de experiências (em 10) com sucesso. As experiências com sucesso são aquelas em que não ocorreram erros numéricos e terminaram com sucesso numa solução não dominada. Como podemos observar, a optimização terminou sem sucesso em 22 dos 420 testes efectuados.

Problema	***n*m*k***	**Sucessos**	**Problema**	***n*m*k***	**Sucessos**
KnapS10	10*1*2	10	Cover40_20	40*20*2	8
KnapS10_3FO	10*1*3	10	Cover50_20	50*20*2	8
KnapS15	15*1*2	9	Cover60_10	60*10*2	10
KnapS20	20*1*2	8	Cover60_20a	60*20*2	10
KnapI10	10*1*2	10	Cover60_20b	60*20*2	10
KnapI10_3FO	10*1*3	10	Cover60_30a	60*30*2	10
KnapI15	15*1*2	10	Cover60_30b	60*30*2	10
KnapI20	20*1*2	10	Cover70_20a	70*20*2	10
KnapI30	30*1*2	10	Cover70_20b	70*20*2	9
KnapI100	100*1*2	10	Cover70_30a	70*30*2	10
Pet1	10*6*2	10	Cover70_30c	70*30*2	10
Pet4	20*10*2	9	Pack20_20	20*20*2	10
Pet5	28*10*2	10	Pack40_10a	40*10*2	8
KnapM10	10*5*2	10	Pack40_10b	40*10*2	10
KnapM20	20*10*2	5	Pack40_20	40*20*2	10
Cover9_7	9*7*2	10	Pack50_10	50*10*2	10
Cover20_20	20*20*2	10	Pack60_10	60*10*2	7
Cover20_20_3	20*20*3	10	Pack70_10	70*10*2	10
Cover30_10	30*10*2	10	Pack80_10	80*10*2	10
Cover30_20	30*20*2	10	Pack100_10	100*10*2	10
Cover40_10	40*10*2	10	Pack100_20	100*20*2	7

Tabela III.3 – Testes individuais dos programas escalarizantes com 10 pontos de referência diferentes.

No anexo III.C incluímos resultados de experiências da abordagem multiobjectivo, que consistiram numa breve pesquisa direccional para cada um dos problemas anteriores (semelhante à ilustrada na tabela III.1 para o problema de localização).

De todos os testes efectuados, notamos que há, em geral, um melhor comportamento em problemas com matrizes esparsas e quando os coeficientes das restrições são da mesma ordem de grandeza dos coeficientes das funções objectivo. A abordagem falhou em aproximadamente 6% de todos os testes efectuados dentro do nosso conjunto de problemas. A resolução "encadeada" dos sucessivos programas escalarizantes de uma pesquisa direccional, em que há um processo cumulativo de introdução de planos de corte, não parece aumentar as dificuldades numéricas.

III.4 CONCLUSÕES

Neste capítulo apresentámos uma nova abordagem interactiva para tratar problemas de programação linear inteira pura multiobjectivo. Tirando partido das técnicas de planos de corte e das características do programa escalarizante de Tchebycheff, desenvolvemos um processo iterativo de análise de sensibilidade que tem como intenção identificar intervalos para os pontos de referência que conduzem à mesma solução não dominada. Assim, se o AD estiver interessado em conhecer soluções não dominadas próximas da actual, não precisará de indicar explicitamente pontos de referência que poderiam conduzir à mesma solução ou a soluções muito afastadas. Esta ferramenta é especialmente útil para efectuar pesquisas *locais* perto das soluções não dominadas preferidas pelo AD, ou para pesquisas *direccionais* com carácter mais global. O AD apenas indica o objectivo que pretende melhorar em cada momento, e é o método que actualiza automaticamente o ponto de referência através de análise de sensibilidade. Os sucessivos programas escalarizantes são resolvidos de forma "encadeada", usando-se para o efeito uma introdução cumulativa de planos de corte.

A utilização de planos de corte como única técnica de resolução dos problemas escalarizantes de programação inteira torna esta abordagem pouco robusta (porque é muito sensível a erros numéricos), o que limita naturalmente a sua aplicabilidade. A resolução "encadeada" dos sucessivos programas escalarizantes de uma pesquisa direccional não parece, porém, aumentar as dificuldades numéricas. Por conseguinte, quanto mais fiáveis forem as técnicas de planos de corte usadas, melhor será o desempenho da abordagem multiobjectivo. Os planos de corte usados até ao momento são apenas os cortes de Gomory e desigualdades de cobertura mínima estendidas, mas outros tipos de planos de corte poderiam ser experimentados.

ANEXO III.A

Desigualdades de cobertura mínima estendidas de Crowder-Johnson-Padberg[3]

Consideremos o programa inteiro com variáveis binárias (0-1) definido da seguinte forma:

$$\max \{cx \mid x \in X\} \qquad \text{(PIB)}$$
$$\text{com } X = \{x \mid Ax \le b,\ x \in \{0,1\}^n\},$$

em que a matriz A ($m \times n$) e os vectores b e c têm coeficientes racionais. Seja $\bar{x}$ a solução óptima da relaxação linear de (PIB), que não é uma solução inteira. Seja N={1,...,*n*}.

Para cada restrição de $Ax \le b$ em que os coeficientes não são todos 0, 1 ou –1, tenta-se gerar uma desigualdade válida para X que seja violada por $\bar{x}$. Trata-se de uma *desigualdade de cobertura mínima*, violada por $\bar{x}$, que será depois sujeita a um processo de extensão formando uma *desigualdade de cobertura mínima estendida*.

A designação de *desigualdade de cobertura mínima* decorre directamente da definição de *cobertura mínima*.

Definição III.3 *Cobertura* e *Cobertura mínima*

Consideremos genericamente uma restrição $\sum_{j \in N} a_j x_j \le a_0$ onde a_j são números positivos racionais e x_j variáveis que assumem apenas valores 0 ou 1. Um conjunto S⊆N é uma *cobertura* se $\sum_{j \in S} a_j > a_0$. Uma cobertura é *mínima* se S\{i} não é uma cobertura, qualquer que seja $i \in$ S, ou seja se se verifica $\sum_{j \in S} a_j - a_i \le a_0$ para todo o $i \in$ S.

Definição III.4 *Desigualdade de cobertura mínima*

Se S é uma cobertura mínima em relação a $\sum_{j \in N} a_j x_j \le a_0$, então cada solução 0-1 que satisfaz esta restrição também satisfaz $\sum_{j \in S} x_j \le |S| - 1$ com $|S|$ o cardinal de S. À desigualdade $\sum_{j \in S} x_j \le |S| - 1$ chama-se *desigualdade de cobertura mínima*.

---

[3] Esta secção segue de perto CROWDER ET AL. (1983).

Passemos então a descrever o procedimento de construção de *desigualdades de cobertura mínima* usado por CROWDER ET AL. (1983). O procedimento analisa uma restrição do problema (PIB) de cada vez, a partir da qual procura construir uma desigualdade válida para X que corte $\bar{x}$.

Seja $\sum_{j\in N} a_{ij}x_j \le b_i$ a restrição i de $Ax \le b$ a ser analisada. Assume-se que todos os coeficientes a_{ij} são não negativos. Se isso não acontecer, começa-se por fazer uma mudança de variável do tipo $x_j \leftarrow 1-x_j$ para as variáveis que têm coeficientes negativos. Esta substituição altera a solução $\bar{x}$ para $\bar{y}$.

Seja $N_1=\{j\in N : \bar{y}_j > 0, a_{ij} \neq 0\}$ e $N_0=\{j\in N : \bar{y}_j = 0, a_{ij} \neq 0\}$. Como se pretende identificar uma restrição válida para (PIB) que seja violada por $\bar{y}$, se $N_1=\varnothing$ ou se $a_{ij}=1$ para todo o $j \in N_1$, então nada há a fazer com a restrição i e avança-se para a seguinte. Caso contrário, tenta-se encontrar um cobertura mínima com índices de N_1, com base no seguinte facto: "existe uma desigualdade de cobertura mínima que corta $\bar{y}$ sse o valor óptimo da função objectivo do problema *(Knap1)* for menor do que 1".

$$\min\left\{\sum_{j\in N_1}(1-\bar{y}_j)s_j \,\middle|\, \sum_{j\in N_1} a_{ij}s_j > b_i,\ s_j\in\{0,1\}, \forall j\in N_1\right\} \qquad \textit{(Knap1)}$$

Começa-se por resolver o problema *(Knap1)* como um de programação linear (substituindo $s_j\in\{0,1\}$ por $0\le s_j\le 1$). Trata-se de um problema semelhante a um de *knapasack* 0-1 com uma desigualdade estrita, mas que pode facilmente ser transformada numa do tipo "≥". A relaxação linear de *(Knap1)* resolve-se por inspecção, atribuindo o valor 1 às variáveis por ordem crescente de $(1-\bar{y}_j)/a_{ij}$ e ficando a última das positivas com o valor residual entre 0 e 1. Arredondando para 1 a variável com valor fraccionário, obtém-se uma cobertura para $\sum_{j\in N} a_{ij}x_j \le b_i$ (cobertura formada pelos índices das variáveis com valor 1), que se transforma numa *cobertura mínima* pela alteração de algumas variáveis de 1 para 0.[4]

Obtida a *cobertura mínima*, S, tenta-se estendê-la a outros elementos, primeiro de N_1, se $N_1\backslash S\neq\varnothing$, e depois a N_0, no caso da desigualdade de cobertura cortar a solução $\bar{y}$. A extensão da cobertura tem como intenção tornar a desigualdade mais forte. Pretende-se, pois, determinar

[4] No nosso trabalho consideramos a alteração das variáveis de 1 para 0 por ordem decrescente de $(1-\bar{y}_j)/a_{ij}$ (que mantêm a condição de cobertura) até se conseguir uma cobertura mínima.

coeficientes f_j para as variáveis $j \in N\backslash S$ de modo a que a desigualdade $\sum_{j\in S} x_j + \sum_{j\in N\backslash S} f_j x_j \le |S| - 1$ seja válida para X.

Inicialmente considera-se f_j=1 para todo o $j \in S$, $f_0 = |S| - 1$ e S^{ext}=S. O processo de *extensão* de S^{ext} a um conjunto N é o seguinte:

Para cada $k \in N\backslash S^{ext}$, pretende-se determinar

$$z_k = \max\left\{ \sum_{j\in S^{ext}} f_j x_j \;\middle|\; \sum_{j\in S^{ext}} a_{ij} x_j \le b_i - a_{ik},\ x_j \in \{0,1\}, \forall j \in S^{ext} \right\} \qquad \textit{(Knap2)}$$

Tal como acontece para *(Knap1)*, também se começa por resolver a relaxação linear de *(Knap2)*. Seja $\bar{z}_k$ o valor óptimo da relaxação linear e $z_k^* = \lfloor \bar{z}_k \rfloor$. Como o valor óptimo de z_k em *(Knap2)* satisfaz $z_k \le z_k^*$, então esta aproximação é possível.

Define-se $f_k = f_0 - z_k^*$. Se $f_k > 0$, então k é inserido em S^{ext}. Se $f_k = 0$, a variável associada não é necessária no processo de *extensão*.

Após ter processado todas as variáveis de $N_1\backslash S$ no processo de *extensão*, testa-se se a desigualdade de cobertura mínima estendida corta $\bar{y}$. Se não for o caso, avança-se para a restrição seguinte de $Ax \le b$. Se cortar $\bar{y}$, então tenta-se estender a desigualdade a variáveis de N_0, usando o processo de *extensão* na forma relaxada como foi descrito no parágrafo anterior.

Depois de se obter uma desigualdade $\sum_{j\in S^{ext}} f_j x_j \le f_0$ (sendo $f_0 = |S| - 1$) violada por $\bar{y}$, escreve-se a desigualdade em função das variáveis originais (para o caso de ter havido alguma mudança de variável do tipo $x_j \leftarrow 1\text{-}x_j$). Guarda-se a desigualdade e avança-se para a restrição seguinte de $Ax \le b$.

ANEXO III.B

Regras usadas na geração dos problemas aleatórios

PROBLEMAS DE MOCHILA (*KNAPSACK*) SIMPLES

$$\max z_i = \sum_{j=1}^{n} c_{ij} x_j \qquad i=1,..,k$$

$$\text{s.a:} \quad \sum_{j=1}^{n} a_j x_j \le b$$

$$x_j \ge 0 \text{ e inteiro, } j=1,..,n \qquad (x_j = 0 \text{ ou } 1 \text{ no } \textit{Knpasack 0-1})$$

Todos os problemas deste tipo têm constantes c_{ij}, a_j e b inteiras geradas aleatoriamente nos seguintes intervalos: $0 \le c_{ij} \le 100$, $1 \le a_j \le 50$ e $\max_{j=1,..,n} a_j \le b \le \sum_{j=1}^{n} a_j - 1$.

Nos problemas inteiros 'KnapI15', 'KnapI20' e 'KnapI30' impôs-se o limite superior de 10 em cada variável.

PROBLEMAS DE MOCHILA (*KNAPSACK*) 0-1 MULTIDIMENSIONAL

$$\max z_i = \sum_{j=1}^{n} c_{ij} x_j \qquad i=1,..,k$$

$$\text{s.a:} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad i=1,...,m$$

$$x_j = 0 \text{ ou } 1 \qquad j=1,..,n$$

Os problemas 'Pet1', 'Pet4' e 'Pet5' são provenientes de uma colectânea de problemas monocritério (BEASLEY, 1990), em que a função objectivo original foi substituída por duas geradas aleatoriamente. Os coeficientes dos outros problemas foram gerados aleatoriamente considerando distribuições uniformes com as seguintes probabilidades (p):

$$\begin{cases} 1 \le c_{ij} \le 100 & p = 0.8 \\ c_{ij} = 0 & p = 0.2 \end{cases}; \qquad \begin{cases} 1 \le a_{ij} \le 50 & p = 0.8 \\ a_{ij} = 0 & p = 0.2 \end{cases}; \qquad 50 \le b_i \le 200.$$

PROBLEMAS DE COBERTURA

$$\min z_i = \sum_{j=1}^{n} c_{ij} x_j \qquad i=1,..,k$$

$$\text{s.a:} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i=1,...,m$$

$$x_j = 0 \text{ ou } 1 \qquad j=1,..,n$$

em que todos os a_{ij} são 0 ou 1.

Os coeficientes c_{ij} são inteiros e $0 \leq c_{ij} \leq 30$; $a_{ij} = \begin{cases} 1 & p = 0.5 \\ 0 & p = 0.5 \end{cases}$.

PROBLEMAS DE EMBALAGEM (*PACKING*)

$$\max z_i = \sum_{j=1}^{n} c_{ij} x_j \qquad i=1,..,k$$

$$\text{s.a:} \quad \sum_{j=1}^{n} a_{ij} x_j \leq 1 \qquad i=1,...,m$$

$$x_j = 0 \text{ ou } 1 \qquad j=1,..,n$$

em que todos os a_{ij} são 0 ou 1.

Os coeficientes c_{ij} são inteiros e $0 \leq c_{ij} \leq 30$.

Em 'Pack20_10' e 'Pack40_10ª', $a_{ij} = \begin{cases} 1 & p = 0.5 \\ 0 & p = 0.5 \end{cases}$.

Em 'Pack40_10b', 'Pack50_10', 'Pack60_10', 'Pack70_10', 'Pack80_10' e 'Pack100_10', $a_{ij} = \begin{cases} 1 & p = 0.45 \\ 0 & p = 0.55 \end{cases}$.

Em 'Pack40_20' e 'Pack100_20', $a_{ij} = \begin{cases} 1 & p = 0.4 \\ 0 & p = 0.6 \end{cases}$.

ANEXO III.C

Resultados de pesquisas direccionais

O ponto de referência inicial para uma pesquisa direccional é dado pelo utilizador, sendo os seguintes ajustados automaticamente pelo método. Na tabela que se segue, a primeira interacção – com a designação de 'Início' – refere-se a uma resolução independente do programa escalarizante. Cada uma das interacções seguintes – com a designação de 'melhorar...' – refere-se ao processo (iterativo) de análise de sensibilidade e re-optimização que termina quando é atingida uma nova solução não dominada.

Todos os tempos referem-se a um computador PENTIUM II a 350MHz.

Problemas de *Knapsack* simples (funções objectivo a maximizar)

(a) Com variáveis 0-1

Problema	***Estratégia***	***P^{to} de Referência***	***Solução (z_1,z_2...)***	***tempo (seg)***
KnapS10	Início	(451, 553)	(456, 471)	0,11
	melhorar z_2		igual à anterior; óptimo z_2	0
KnapS10_3FO	Início	(491, 513, 427)	(469, 468, 363)	2.91
	melhorar z_2	(491, 546, 427)	(423, 511, 350)	3.85
	melhorar z_3	(491, 546, 429)	(469, 468, 363)	0.16
KnapS15 *	Início	(653, 610)	(604, 576)	8.95
	novo início	(700, 700)	(602, 606)	0.39
	melhorar z_1	(727, 700)	(604, 576)	6.87
KnapS20 *	Início	(864, 695)	"*erro*"	
	novo início	(950, 700)	(851, 618)	0.72
	melhorar z_2	(950, 724)	(846, 636)	4.61
(b) Com variáveis inteiras				
KnapI10	Início	(3270, 2644)	(2498, 1878)	0.38
	melhorar z_1	(3323, 2644)	(2548, 1820)	5.44
	melhorar z_1	(3431, 2644)	(2598, 1762)	5.77
KnapI10_3FO	Início	(3306, 2751, 2425)	(2083, 1452, 2034)	2.8
	melhorar z_2	(3306, 2787, 2425)	(1972, 1528, 2000)	4.45
	melhorar z_2	(3306, 2947, 2425)	(1861, 1604, 1966)	12.47
KnapI15	Início	(2005, 1475)	(1796, 1285)	2.63
	melhorar z_1	(2017, 1475)	(1798, 1255)	4.45
	melhorar z_1	(2031, 1475)	(1821, 1243)	4.83
KnapI20	Início	(2615, 1688)	(2566, 1688)	1.26
	melhorar z_1	(2637, 1688)	(2575, 1618)	4.78
	melhorar z_1	(2715, 1688)	(2580, 1549)	10.55
KnapI30	Início	(2929, 2084)	(2847, 1946)	1.43
	melhorar z_2	(2929, 2160)	(2716, 1975)	14.22
	melhorar z_2	(2929, 2340)	(2565, 1996)	27.24
KnapI100	Início	(48022, 55118)	(42430, 49529)	1.97
	melhorar z_1	(48031, 55118)	42451, 49518)	2.96
	melhorar z_1	(48090, 55118)	(42462, 49480)	10.71

Problemas de *Knapsack* 0-1 multidimensional (funções objectivo a maximizar)

Problema	*Estratégia*	*P^to de Referência*	*Solução (z_1,z_2...)*	*tempo (seg)*
Pet1	Início	(41, 19)	(37, 15)	0.16
	melhorar z_1	(51, 19)	(38, 6)	0.11
	melhorar z_1	(61, 19)	igual à anterior; óptimo z_1	0.05
	melhorar z_2	(61, 31)	(37,15)	0.11
	melhorar z_2	(61, 53)	(24, 18)	1.21
Pet4 *	Início	(1000, 900)	(913, 764)	2.04
	melhorar z_1	(1071, 900)	*"erro"*	
Pet5	Início	(604, 673)	(591, 662)	0.82
	melhorar z_1	(609, 673)	(593, 656)	2.53
KnapM10	Início	(244, 333)	(174, 269)	0.33
	melhorar z_2	(244, 370)	(144, 283)	3.4
	melhorar z_2	igual ao anterior	igual à anterior; óptimo z_2	0
KnapM20 *	Início	(231, 275)	*"erro"*	
	novo início	(300, 300)	(199, 185)	2.85
	melhorar z_2	(300, 302)	*"erro"*	

Problemas de cobertura (funções objectivo a minimizar)

Problema	*Estratégia*	*P^to de Referência*	*Solução (z_1,z_2...)*	*tempo (seg)*
Cover9_7	Início	(15, 18)	(21, 25)	0.16
	melhorar z_1	(12, 18)	(19, 26)	0.11
	melhorar z_1	(5,18)	(18, 31)	0.11
Cover20_20	Início	(16, 20)	(19, 21)	0.11
	melhorar z_1	(7, 20)	(17, 31)	0.39
	melhorar z_1	(-12, 20)	(16, 48)	0.06
Cover20_20_3	Início	(16, 20, 11)	(22, 25, 13)	0.17
	melhorar z_1	(3, 20, 11)	(21, 38, 20)	2.91
	melhorar z_1	(-1, 20, 11)	(20, 42, 14)	3.9
Cover30_10	Início	(4, 1)	(18, 17)	0.17
	melhorar z_2	(4, -2)	(22, 15)	0.28
	melhorar z_2	(4, -21)	(39, 10)	3.68
Cover30_20	Início	(11, 10)	(25, 17)	11.2
	melhorar z_1	(10, 10)	(21, 24)	0.65
	melhorar z_1	(4, 10)	(19, 26)	0.55
Cover40_10	Início	(12, 0)	(17, 1)	0.11
	melhorar z_1	(4,0)	(15, 12)	1.37
	melhorar z_1	(-3, 0)	(14, 17)	0.22
Cover40_20 *	Início	(7, 13)	(13, 22)	0.27
	melhorar z_1	(-9,13)	(12, 34)	3.08
	melhorar z_1	(-26,13)	(9, 50)	11.53
Cover50_20 *	Início	(0, 3)	(17, 22)	0.38
	melhorar z_2	(0, -3)	(24, 21)	3.02
	melhorar z_2	(0, -6)	(26, 14)	1.43
Cover60_10	Início	(8, 0)	(10, 11)	0.43
	melhorar z_2	(8, -5)	(23, 6)	1.43
	melhorar z_2	(8,-13)	(26, 4)	0.27
Cover60_20a	Início	(0, 0)	(10,9)	0.28
	melhorar z_1	(-2, 0)	(4, 11)	0.16
	melhorar z_1	(-16,0)	(0, 19)	0.72
Cover60_20b	Início	(5, 9)	(14, 20)	0.16
	melhorar z_2	(5, 7)	(17, 17)	0.17
	melhorar z_2	(5, 1)	(20, 12)	0.83

Cover60_30a	Início	(1,0)	(8, 13)	5.17
	melhorar z_2	(1, -2)	(15, 4)	2.37
	melhorar z_2	(1, -13)	(17, 1)	0.44
Cover60_30b *	Início	(4, 14)	(13, 22)	0.6
	melhorar z_2	(4, -4)	(29, 20)	9.5
	melhorar z_2	(4, -17)	"*erro*"	
Cover70_20a	Início	(0,0)	(3, 2)	0.05
	melhorar z_2	(0, -24)	(25, 0)	0.87
	melhorar z_2	igual ao anterior	igual à anterior; óptimo z_2	0
Cover70_20b *	Início	(6,5)	(19,18)	0.87
	Novo início	(4, 4)	(19, 18)	1.21
	melhorar z_2	(4, 0)	(21, 16)	0.77
	melhorar z_2	(4, -6)	(25, 14)	0.93
Cover70_30a	Início	(4, 0)	(15, 14)	2.03
	melhorar z_1	(-7, 0)	(10, 21)	11.04
	melhorar z_1	(-14, 0)	(9, 23)	0.33
Cover70_30c *	Início	(2, 3)	(12, 19)	0.38
	melhorar z_2	(2, 1)	(19, 18)	0.5
	melhorar z_2	(2, -1)	"*erro*"	

Problemas de embalagem (*packing*) (funções objectivo a maximizar)

Problema	*Estratégia*	*P^{to} de Referência*	*Solução (z_1,z_2,...)*	*tempo (seg)*
Pack20_20	Início	(40, 38)	(26, 29)	0.55
	melhorar z_1	(51, 38)	(29, 24)	0.16
	melhorar z_1	(53, 38)	igual à anterior; óptimo z_1	0.06
Pack40_10a *	Início	(55, 56)	(41, 37)	1.6
	novo início	(70, 70)	(41, 37)	1.43
	melhorar z_2	(70, 75)	*"erro"*	
Pack40_10b	Início	(52, 60)	(37, 60)	0.99
	melhorar z_1	(53, 60)	(45, 45)	0.11
	melhorar z_1	(69, 60)	(48, 37)	0.6
Pack40_20	Início	(68, 67)	(66, 59)	0.44
	melhorar z_2	(68, 70)	(58, 60)	0.28
	melhorar z_2	igual ao anterior	igual à anterior; óptimo z_2	
Pack50_10	Início	(61, 65)	(56, 58)	0.11
	melhorar z_1	(82, 65)	igual à anterior; óptimo z_1	1.32
	melhorar z_2	(82, 113)	(28, 62)	2.75
Pack60_10 *	Início	(98, 85)	(74, 55)	0.6
	melhorar z_2	(98, 92)	*"erro"*	
Pack70_10	Início	(92, 109)	(74, 92)	0.11
	melhorar z_1	(95, 109)	(76, 89)	0.11
	melhorar z_1	(101, 109)	(77, 85)	0.55
Pack80_10	Início	(107, 88)	(89, 80)	1.37
	melhorar z_1	(115, 88)	(93, 63)	4.95
	melhorar z_1	(121, 88)	(100, 61)	1.42
Pack100_10	Início	(116, 101)	(107, 91)	0.06
	melhorar z_2	(116, 116)	(92, 95)	1.54
	melhorar z_2	(116, 129)	(83, 101)	1.21
Pack100_20 *	Início	(146, 168)	(119, 153)	7.75
	melhorar z_2	(146, 197)	*"erro"*	

* Ocorreu "erro" numérico em pelo menos uma experiência (teste individual ou durante uma pesquisa direccional

Capítulo IV

Método interactivo para PLIMO e PLIMMO baseado em *branch-and-bound*

Este capítulo é parcialmente baseado em ALVES E CLÍMACO (1998[B]) e ALVES E CLÍMACO (2000[A]), e apresenta um novo método interactivo para problemas de programação linear inteira (pura) multiobjectivo (PLIMO) e problemas de programação linear inteira-mista multiobjectivo (PLIMMO). Este método baseia-se na técnica de *branch-and-bound*.

Após o desenvolvimento do método interactivo para PLIMO baseado em planos de corte (apresentado no capítulo anterior), fomos confrontados com as suas limitações práticas. Trata-se de um método dedicado exclusivamente a problemas de PLIMO, e muitos problemas práticos são inteiros-mistos. A principal razão desta limitação reside no facto da análise de sensibilidade assentar em pressupostos válidos para o caso inteiro puro, mas que não se verificam no caso inteiro-misto. Além disso, mesmo no tratamento de problemas de PLIMO, o método está limitado a problemas de pequenas dimensões ou com estruturas matriciais particulares, em virtude de as fases de optimização dos programas escalarizantes recorrerem unicamente a técnicas de planos de corte. Todavia, as experiências efectuadas com a abordagem de planos de corte revelaram, em nosso entender, que a filosofia subjacente ao método pode ser interessante para o apoio à decisão em problemas com variáveis inteiras. Consequentemente, julgámos importante poder dispor de uma ferramenta mais robusta e genérica que permitisse lidar, quer com problemas de PLIMO, quer com problemas de PLIMMO.

Decidimos então desenvolver uma nova abordagem, tecnicamente diferente da anterior, mas tendo por base as mesmas intenções: (1) oferecer um meio de interacção simples, em que o AD

especifica níveis de aspiração (pontos de referência) do espaço dos critérios que gostaria de atingir, ou escolhe apenas uma função objectivo que deseja melhorar relativamente a uma solução já conhecida; neste último caso, desenrola-se uma *pesquisa direccional*, em que trajectórias particulares de pontos de referência são projectadas no conjunto das soluções não dominadas; (2) diminuir o esforço computacional envolvido no cálculo de soluções não dominadas. As soluções não dominadas são obtidas pela optimização de programas escalarizantes de Tchebycheff ($P_{z^+}^{\infty,\rho}$), e optámos por usar técnicas de *branch-and-bound* para o fazer.

Pretendíamos que esta abordagem fosse particularmente eficaz nas *pesquisas direccionais*, sendo para tal importante que o ponto de referência fosse ajustado automaticamente (como já tínhamos verificado antes), e que houvesse uma redução do esforço computacional relativamente ao da resolução independente dos sucessivos programas escalarizantes. Nessa perspectiva, desenvolvemos uma técnica de análise de sensibilidade que utiliza informação proveniente da árvore de *branch-and-bound* que resolveu o último programa escalarizante. A análise de sensibilidade determina pontos de referência que conduzem à solução não dominada anterior, ou a soluções diferentes da anterior mas que se obtêm sem alterar a estrutura actual da árvore de enumeração. Além disso, os programas escalarizantes seguintes são resolvidos de forma "encadeada", ou seja, o método usa a árvore de *branch-and-bound* anterior como ponto de partida para a resolução do programa escalarizante seguinte de uma pesquisa direccional. Se forem necessárias novas ramificações da árvore, tenta-se fazer em primeiro lugar uma simplificação da árvore, para que esta não esteja em crescimento permanente. A simplificação consiste em cortar ramos associados a limitações de variáveis (impostas anteriormente pelo processo de *branch-and-bound*) que entretanto deixaram de estar activas. Este processo elimina partes da árvore, para depois acrescentar outras provenientes das ramificações necessárias. Este modo de proceder revelou-se eficaz como veremos adiante nos resultados computacionais.

Este capítulo está organizado da seguinte forma:

Na secção 1, começamos por introduzir genericamente a técnica de *branch-and-bound* para problemas de programação inteira ou inteira-mista (secção 1.1). Fazemos em seguida (secção 1.2) uma breve referência à investigação anterior na área da análise de sensibilidade/paramétrica aos termos independentes de restrições de um programa inteiro (misto) no contexto de *branch-and-bound*.

A secção 2 é dedicada à abordagem interactiva que desenvolvemos para PLIMO e PLIMMO. Começamos por descrever o método do ponto de vista técnico, dando particular destaque ao processo de análise de sensibilidade e de actualização da árvore de *branch-and-bound* de uma pesquisa direccional (secção 2.1). Apresentamos em seguida um exemplo ilustrativo (secção 2.2) e resultados de experiências computacionais (secção 2.3).

Na secção 3 propomos um algoritmo combinado do método baseado em planos de corte (apresentado no capítulo III) e do método baseado em *branch-and-bound*. Tiramos algumas ilações acerca do seu comportamento referindo, em particular, os casos em que o algoritmo combinado é melhor do que as versões isoladas e os casos em que é pior.

Na secção 4 apresentamos algumas reflexões sobre o esquema de diálogo com o AD, propondo um protocolo interactivo que possibilite, no contexto das abordagens propostas, um melhor apoio à decisão.

A secção 5 encerra este capítulo com algumas conclusões.

IV.1 BRANCH-AND-BOUND E ANÁLISE DE SENSIBILIDADE EM PROGRAMAÇÃO INTEIRA E INTEIRA-MISTA

IV.1.1 Algoritmo de *branch-and-bound*

Consideremos o problema genérico de programação linear inteira (mista):[1]

$$z = \min \{cx \mid x \in X\} \qquad \text{(PLIM)}$$

$$\text{com } X=\{x \in \Re^n \mid Ax \geq b,\ x \geq 0,\ x_j \text{ inteiro para } j \in I\},$$

em que I é o conjunto dos índices das variáveis inteiras ($I \neq \varnothing$).

O *branch-and-bound* é um algoritmo de ramificação e limitação que se enquadra na classe dos métodos de enumeração. É aplicável quer a problemas inteiros (puros) quer a problemas inteiros-mistos, ou seja, ao problema PLIM em geral.

O primeiro passo consiste na resolução da *relaxação linear* do problema PLIM, i.e. o problema linear resultante de PLIM pela eliminação das condições de integralidade. Se a solução óptima da relaxação linear for admissível para o problema PLIM, então é essa a solução óptima de PLIM. Caso contrário, o valor da função objectivo dessa solução constitui um limite inferior para o mínimo de PLIM. No caso em que a solução óptima do problema relaxado não satisfaz as condições de integralidade, escolhe-se uma variável x_j, $j \in I$, com valor não inteiro x_j^0, e constroem-se dois subproblemas lineares através da introdução da restrição $x_j \leq \lfloor x_j^0 \rfloor$ num deles e $x_j \geq \lfloor x_j^0 \rfloor + 1$ no outro. Este procedimento pode ser ilustrado através de uma árvore binária como a da figura IV.1.

[1] Dado que a função objectivo do programa escalarizante que usaremos na abordagem multiobjectivo é a minimizar, optamos por estabelecer os conceitos básicos do *branch-and-bound* para a minimização.

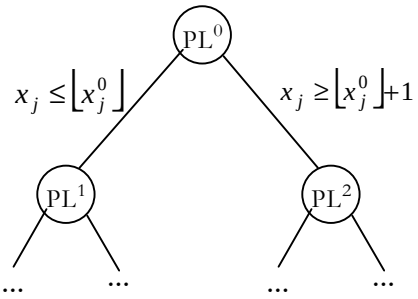


Fig. IV.1 – Esquema da árvore de branch-and-bound.

Designando por PL^i o problema resultante da relaxação linear da região admissível X^i, e considerando $X^0 = X$, no caso da figura IV.1 temos:

$$X^1 = X \cap \{x \mid x_j \le \lfloor x_j^0 \rfloor\}$$

$$X^2 = X \cap \{x \mid x_j \ge \lfloor x_j^0 \rfloor + 1\}$$

É claro que $X = X^1 \cup X^2$ e $X^1 \cap X^2 = \varnothing$. A solução x^0 não é admissível nem em PL^1 nem em PL^2. Um problema PL^i resulta, pois, da adição de uma restrição, de limite inferior ou superior numa variável, ao problema linear seu ascendente. A base óptima anterior deixa de ser admissível para o primal mas permanece admissível para o dual, sendo então natural que se re-optimize a partir dessa base usando o algoritmo dual-simplex. Tipicamente serão necessárias apenas algumas pivotações para atingir a nova solução óptima (caso exista).

Durante a pesquisa na árvore de *branch-and-bound*, utiliza-se habitualmente uma lista de problemas/nodos activos que serão examinados. A lista começa por conter PL^0, que será depois retirado para dar entrada a PL^1 e PL^2, e assim sucessivamente. Pode, contudo, não haver razão para ramificar um problema PL^i, dizendo-se que a árvore de enumeração é *podada* por esse nodo. A lista de nodos activos contém os nodos que ainda não foram ramificados ou podados. A poda pode ser por *inadmissibilidade*, se PL^i for impossível, por *optimalidade*, se a solução óptima de PL^i for admissível para PLIM e a melhor encontrada até ao momento ($\bar{z} \leftarrow z^i$), ou por *limite*, se o limite inferior dado por PL^i para esse ramo da árvore for superior ao valor da função objectivo de alguma solução admissível de PLIM já conhecida ($z^i \ge \bar{z}$). Assim, de cada vez que se encontra uma *solução inteira*[2], verifica-se se é possível actualizar a *solução incumbente* – melhor solução inteira encontrada até ao momento – e o limite superior $\bar{z}$ para o valor de z.

[2] Por simplicidade de linguagem, designamos por *solução inteira* uma solução admissível para o problema PLIM.

A figura IV.2 apresenta o fluxograma de um algoritmo simples de *branch-and-bound* para um problema de minimização.

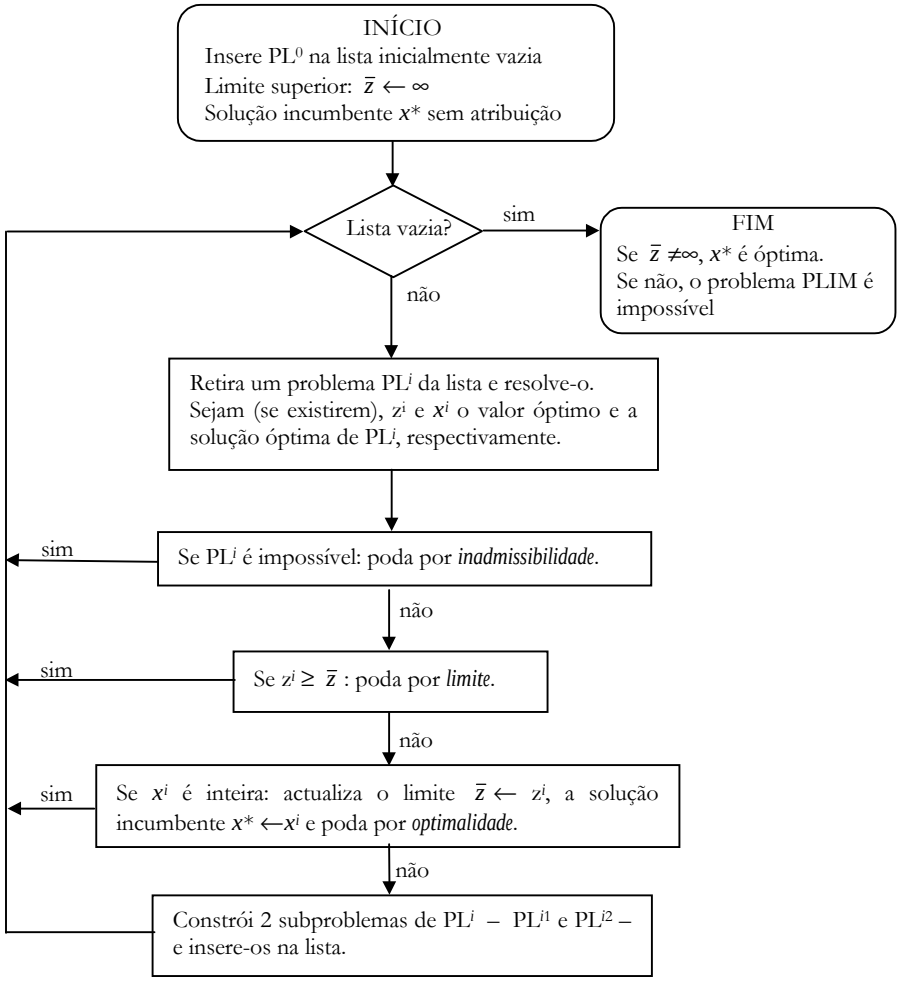


Fig. IV.2 – Fluxograma de branch-and-bound para minimização (adaptado de WOLSEY, 1998).

Há várias opções que se têm de fazer numa pesquisa de *branch-and-bound*, nomeadamente no que se refere ao critério de selecção da variável para proceder a uma ramificação ou na escolha do nodo a examinar. Quanto ao primeiro, uma estratégia vulgar é escolher a variável cuja parte fraccionária mais se aproxima de ½. Quanto à escolha do nodo a examinar, há vários argumentos contraditórios que podem ser invocados (WOLSEY, 1998). Uns defendem que só é possível podar significativamente a árvore a partir de uma solução inteira que, de preferência, dê um bom limite superior. Para tal, deve-se descer o mais rapidamente possível na árvore para encontrar uma primeira solução inteira, seguindo-se desta forma uma pesquisa em *profundidade*. Um outro argumento a favor desta estratégia é o facto de ser mais fácil resolver um subprograma linear, que se obtém do anterior introduzindo uma nova restrição, se estiver disponível a base óptima anterior, o que encoraja a passagem de um nodo para o seu descendente imediato. Por outro lado, para minimizar o número total de nodos examinados, a

melhor estratégia é escolher sempre o nodo activo com o melhor limite inferior. Isto leva a uma estratégia de pesquisa pelo *melhor nodo*. Na prática, pode ainda haver lugar a uma combinação entre os dois tipos de estratégias.

Na implementação que fizemos do *branch-and-bound*, no contexto da metodologia multiobjectivo, tomámos algumas opções que serão indicadas mais à frente.

IV.1.2 Análise de sensibilidade / paramétrica no contexto de *branch-and-bound*

Terá sido ROODMAN (1972) um dos primeiros investigadores a desenvolver trabalho computacional na área da análise de sensibilidade em algoritmos de enumeração. O seu trabalho consistiu basicamente em estender o algoritmo aditivo de Balas (BALAS, 1965) para uma análise de sensibilidade em problemas de programação 0-1, considerando um domínio estreito de valores dos parâmetros.

Da investigação mais recente de análise de sensibilidade/paramétrica em programação inteira ou inteira-mista, num contexto específico de *branch-and-bound*, e para variações dos termos independentes das restrições, podemos referir os trabalhos de ROUNTREE E GILLETT (1982), OHTAKE E NISHIDA (1985), SCHRAGE E WOLSEY (1985) e ACEVEDO E PISTIKOPOULOS (1999).

Nenhum destes trabalhos permite, contudo, responder à nossa questão de análise de sensibilidade. Pretendemos fazer uma análise de sensibilidade ao termo independente de uma restrição do tipo '≥' do programa escalarizante, que é um problema de programação inteira ou inteira-mista a minimizar. O termo independente ('RHS') a analisar refere-se à componente do ponto de referência que desejamos incrementar, e colocam-se-nos questões como: "qual o máximo incremento no 'RHS' que mantém óptima do programa escalarizante a solução eficiente actual?", ou "qual o máximo incremento no 'RHS' que mantém óptimo o actual nodo da árvore de *branch-and-bound*", ou ainda, "qual o máximo incremento no 'RHS' tal que a solução óptima é obtida a partir de algum nodo da actual árvore de *branch-and-bound?*".

Os desenvolvimentos teóricos que mais se aproximam dos nossos interesses (apesar de não responderem às questões que se nos colocam) são os de SCHRAGE E WOLSEY (1985). Os autores analisam o efeito de pequenas alterações nos 'RHS' de programas inteiros-mistos. Sem modificar o algoritmo de *branch-and-bound*, mostram como a recolha de dados adicionais durante a resolução do problema original pode dar limites úteis para o problema modificado. Usando o conceito de função de preços duais, discutida por WOLSEY (1981), são definidos limites superiores para o valor da função objectivo de problemas a maximizar.

Todos os outros trabalhos que referimos analisam processos de resolução *completa* de um programa paramétrico de 'RHS', utilizando *branch-and-bound*. A resolução paramétrica *completa*

consiste em calcular as soluções óptimas de todos os programas que resultam da alteração de um ou mais parâmetros, dentro de um determinado domínio.

Apresentamos, em seguida, um breve resumo de cada um destes trabalhos.

Comecemos por definir o seguinte programa paramétrico, inteiro ou inteiro-misto, com um parâmetro $\theta \in [0,1]$:

$$z(\theta) = \min\{cx \mid Ax \geq b+\theta r,\ x \geq 0,\ x_j \text{ inteiro para } j \in I\} \qquad (P_\theta)$$

O algoritmo de ROUNTREE E GILLETT (1982) trata o problema P_θ inteiro puro, em que r é um vector de componentes inteiras não negativas. Começa por determinar todos os vectores $b+\theta r$ que são inteiros e acrescenta-os em colunas à direita de um quadro inicial, formando um problema expandido, ou melhor, uma família com um número finito de problemas. Como $r \geq 0$, a região admissível reduz-se conforme θ varia de 0 a 1. Aproveitando este facto, as relaxações lineares dos vários problemas da família são resolvidas em simultâneo pelo dual-simplex, fazendo-se pivotações até se atingir a optimalidade para $\theta=1$. Emprega-se depois um esquema de *branch-and-bound* com planos de corte para resolver simultaneamente toda a família de problemas. As restrições de ramificação impõem limites nas variáveis que podem diferir dentro dos vários elementos da família. A opção de planos de corte aplica um número finito de cortes fraccionários de Gomory a cada nodo antes de este ser ramificado.

OHTAKE E NISHIDA (1985) apresentam um algoritmo de *branch-and-bound* para programação paramétrica 0-1 mista considerando P_θ com $r \geq 0$. Tal como na caso anterior, a região admissível reduz-se com o aumento de θ, sendo $z(\theta)$ uma função não decrescente. A cada nodo i da árvore de *branch-and-bound* corresponde um subconjunto de variáveis 0-1 com valor atribuído, e um intervalo $T_i \subseteq [0,1]$ para o parâmetro θ (o intervalo é [0,1] na raiz da árvore). Seja $\underline{z}_i(\theta)$, $\theta \in T_i$, a função dos limites inferiores dada pela relaxação linear do problema do nodo i, e $\overline{z}(\theta)$ uma função geral de limites superiores que se calcula para o problema original. Se para algum $\theta \in T_i$, $\underline{z}_i(\theta) \geq \overline{z}(\theta)$, então esse valor de θ pode ser eliminado de T_i. Os intervalos T_i vão assim diminuindo de pai para filho ao longo da árvore de *branch-and-bound*. Um nodo é *podado* se $T_i=\varnothing$ ou se a sua solução é inteira para $\forall\theta \in T_i$. Procedendo deste modo, obtém-se uma parametrização completa de P_θ.

Utilizando um processo de enumeração semelhante ao anterior, ACEVEDO E PISTIKOPOULOS (1999) consideram o problema paramétrico de programação linear 0-1 mista, mas com vários parâmetros nos termos independentes das restrições a variarem independentemente (vector θ). O algoritmo baseia-se no cálculo das sucessivas soluções dos

programas lineares multiparamétricos associados aos nodos da árvore de *branch-and-bound*. Para cada nodo i definem-se k regiões críticas $CR_{i,k} \subseteq T_i$ (em que T_i é a região de θ a considerar no nodo) correspondentes às k bases óptimas diferentes. Uma região crítica $CR_{i,k}$ pode ser eliminada se se verificar algum dos critérios habituais de *poda* do *branch-and-bound* para todo o $\theta \in CR_{i,k}$; o nodo i é *podado* se todos as suas regiões críticas $CR_{i,k}$ forem eliminadas.

IV.2 MÉTODO DE PONTOS DE REFERÊNCIA PARA PLIMO E PLIMMO QUE USA BRANCH-AND-BOUND

Consideremos o modelo de programação linear inteira-mista multiobjectivo (PLIMMO):

$$\begin{aligned} &\max \quad z_1 = c^1 x \qquad\qquad (\text{PLIMMO})\\ &\ldots\\ &\max \quad z_k = c^k x\\ &\text{s.a:} \quad x \in X\\ &\qquad X = \left\{ x \in \Re^n \mid Ax = b, x \geq 0, x_j \text{ inteiro}, j \in I \right\}, \end{aligned}$$

em que $I \neq \varnothing$ é o conjunto dos índices das variáveis inteiras.

Consideremos o programa escalarizante $P_{z^+}^{\infty,\rho}$ que determina a solução $x \in X$ cujo ponto do espaço dos objectivos, $z = Cx$ ($z_i = c^i x$, $i=1,...,k$) minimiza a distância ao ponto de referência $z^+ \geq z^*$ segundo a métrica aumentada de Tchebycheff:

$$\begin{aligned} &\min \quad \alpha - \rho \sum_{i=1}^{k} c^i x \qquad\qquad (P_{z^+}^{\infty,\rho})\\ &\text{s.a:} \quad c^i x + \alpha \geq z_i^+ \qquad i=1,...,k\\ &\qquad x \in X\\ &\qquad \alpha \geq 0 \end{aligned}$$

em que ρ é uma constante positiva suficientemente pequena.

Uma solução óptima de $P_{z^+}^{\infty,\rho}$ é uma solução eficiente para PLIMMO e, a menos de soluções contínuas muito próximas de soluções fracamente eficientes, qualquer solução eficiente pode ser gerada a partir de $P_{z^+}^{\infty,\rho}$ (para detalhes, ver secções II.1.3 e II.1.4). O método multiobjectivo baseado em *branch-and-bound* que iremos apresentar usa $P_{z^+}^{\infty,\rho}$ para gerar as soluções eficientes/não dominadas (tal como o método anterior baseado em planos de corte).

O resultado estabelecido na proposição III.4 para PLIMO é também válido para PLIMMO, pelo que podemos calcular soluções não dominadas que melhorem sucessivamente uma dada

função objectivo z_p (*pesquisa direccional*) através da projecção de pontos de referência com valor crescente na *p*-ésima componente e valores constantes nas outras componentes.

Começamos por estabelecer um algoritmo de base para a abordagem interactiva (semelhante ao apresentado no fluxograma da figura III.1 para a abordagem de planos de corte), remetendo para mais tarde uma discussão mais detalhada sobre o protocolo de diálogo com o AD.

O algoritmo resume-se nos seguintes passos:

1. O AD deve indicar um ponto de referência.

   Na primeira interacção, é proposto, por omissão, o ponto ideal do problema PLIMMO ou da sua relaxação linear.

   Seja o ponto de referência $z^+ = (z_1^+, ..., z_k^+)$.

   (Se necessário, e apenas por razões técnicas, adiciona-se uma constante a todas as componentes de z^+ de modo a que este satisfaça $z^+ \geq z^*$).

2. Resolve-se o programa escalarizante de Tchebycheff, $P_{z^+}^{\infty,\rho}$, usando *branch-and-bound*, para calcular a solução não dominada mais próxima de z^+ segundo a métrica aumentada de Tchebycheff (L_∞). Esta solução é apresentada ao AD.
3. Se o AD considerar a solução satisfatória, termina; caso contrário, o AD pode fazer uma pesquisa livre de soluções não dominadas através da indicação de novos pontos de referência, regressando a 1, ou uma pesquisa direccional/local, seguindo para 4.
4. O AD deve indicar um objectivo que pretende melhorar relativamente à solução não dominada anterior; seja z_p a função objectivo escolhida pelo AD.
5. Faz-se uma pesquisa direccional considerando pontos de referência do tipo $z^+ + \theta_p = (z_1^+, ..., z_p^+ + \theta_p, ..., z_k^+)$, em $P_{z^+ + \theta_p}^{\infty,\rho}$, de modo a calcular sucessivas soluções não dominadas que melhorem z_p. Sempre que o AD quiser alterar a direcção de pesquisa, regressa a 4. Se o AD quiser recomeçar a pesquisa a partir de outro ponto, regressa a 1.

A parte principal deste algoritmo é a forma como são feitas as *pesquisa direccionais* (passo 5). Trata-se de um processo de optimização de sucessivos programas escalarizantes do tipo $P_{z^+ + \theta_p}^{\infty,\rho}$, em que cada um deles apenas difere do anterior no valor do parâmetro θ_p, ou seja, no termo independente de uma restrição. Desenvolvemos uma técnica de análise de sensibilidade e pós-optimização em *branch-and-bound* para efectuar o ajuste automático do valor do parâmetro e para resolver, de forma "encadeada", os sucessivos programas escalarizantes. Este é um processo iterativo com duas fases principais: *(i) análise de sensibilidade* e *(ii) actualização da árvore de branch-and-bound*.

A análise de sensibilidade *(i)* calcula um valor do parâmetro, θ_p^{max}, que mantém inalterada a estrutura da árvore de *branch-and-bound*. Isso significa que os pontos de referência correspondentes a $0 \leq \theta_p \leq \theta_p^{max}$ conduzem à mesma solução não dominada ou a soluções não dominadas diferentes produzidas pelo mesmo nodo da árvore. Logo, estas soluções são facilmente obtidas por análise de sensibilidade clássica ao subproblema linear desse nodo. Supondo que o AD quer continuar a pesquisa na mesma direcção, eleva-se então θ_p a um valor ligeiramente superior a θ_p^{max} – seja esse valor $\hat{\theta}_p$. É necessário depois proceder à *(ii)* actualização da árvore de *branch-and-bound* no sentido de determinar a solução não dominada seguinte, que optimiza $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$. Devemos, no entanto, referir que o valor de θ_p^{max} calculado em *(i)* é apenas uma aproximação (minorante) do verdadeiro máximo, o que pode implicar que resulte de *(ii)* a solução não dominada anterior. Neste caso, o algoritmo regressa automaticamente a *(i)*, repetindo-se o processo.

Vejamos então como se processam as *pesquisas direccionais* do ponto de vista técnico.

IV.2.1 Pesquisas direccionais no contexto de *branch-and-bound*

Consideremos que o programa escalarizante de Tchebycheff $P^{\infty,\rho}_{z^+}$ foi resolvido para $z^+ = (z_1^+, ..., z_k^+)$, produzindo uma solução eficiente x^0 do problema PLIMMO cujo ponto dos critérios não dominado é z^0. Consideremos também que o AD pretende efectuar uma *pesquisa direccional* e escolheu a função objectivo z_p para melhorar relativamente à solução z^0. As soluções eficientes seguintes serão obtidas pela resolução do programa de Tchebycheff paramétrico $P^{\infty,\rho}_{z^+ + \theta_p}$ com $\theta_p \geq 0$:

$$f(\theta_p) = \min \left\{ \alpha - \rho \sum_{i=1}^{k} c^i x \right\} \qquad (P^{\infty,\rho}_{z^+ + \theta_p})$$

s.a: $c^i x + \alpha - s_i = z_i^+ \qquad i=1,\ldots,k,\ i \neq p$

$c^p x + \alpha - s_p = z_p^+ + \theta_p$

$x \in X$

$\alpha \geq 0,\ s_i \geq 0,\ i = 1,...,k$

em que s_i, $i=1,..,k$, são as variáveis desvio das primeiras k restrições, que correspondem às funções objectivo do problema multiobjectivo.

O programa escalarizante de Tchebycheff inicial $P_{z^+}^{\infty,\rho}$ foi resolvido usando o algoritmo de *branch-and-bound*. A estratégia adoptada para a escolha do nodo a ramificar no algoritmo de *branch-and-bound* é a do *melhor nodo* (melhor limite inferior).

A cada nodo da árvore de *branch-and-bound* de $P_{z^+}^{\infty,\rho}$ está associado um subproblema de $P_{z^+}^{\infty,\rho}$ relaxado linearmente. Relativamente ao programa paramétrico $P_{z^+ + \theta_p}^{\infty,\rho}$, o subproblema linear associado ao nodo Q^r pode ser escrito da seguinte forma (com $\theta_p \geq 0$) :

$$f^r(\theta_p) = \min \left\{ \alpha - \rho \sum_{i=1}^{k} c^i x \right\} \qquad (Q_{\theta_p}^r)$$

$$\text{s.a:} \quad c^i x + \alpha - s_i = z_i^+ \qquad i=1,\ldots,k,\ i \neq p$$

$$c^p x + \alpha - s_p = z_p^+ + \theta_p$$

$$x \in X_{(PL)}^r$$

$$\alpha \geq 0,\ s_i \geq 0,\ i = 1,\ldots,k$$

em que $X_{(PL)}^r = \left\{ x \in \Re^n \middle| Ax = b,\ x \geq 0,\ L_i^r \leq x_i \leq U_i^r,\ i \in I \right\}$, podendo existir alguns $L_i^r = 0$ e $U_i^r = \infty$.

Abordemos agora a questão de como aproveitar informação da resolução de $P_{z^+}^{\infty,\rho}$ para variar o parâmetro θ_p e resolver $P_{z^+ + \theta_p}^{\infty,\rho}$. Esta questão prende-se naturalmente com a análise de sensibilidade à variação do termo independente de uma restrição de $P_{z^+}^{\infty,\rho}$, sendo essa restrição particular porque advém duma função objectivo do problema multiobjectivo.

A variação positiva de θ_p pode conduzir, basicamente, a uma das seguintes situações: (a) a solução eficiente x^0 mantém-se; (b) a solução altera-se mas é obtida a partir do mesmo nodo da árvore, mantendo-se a estrutura da árvore de *branch-and-bound*; (c) a estrutura da árvore de *branch-and-bound* pode alterar-se.

O processo de *análise de sensibilidade* pretende identificar o intervalo $[0,\ \theta_p^{max}]$ de valores de θ_p que implicam situações dos tipos (a) ou (b). Na situação (a), para $\theta_p \leq \theta_p^{max}$ a solução do programa escalarizante pode alterar-se, no sentido em que pode haver variação das variáveis s_i e α, mas a solução eficiente do problema multiobjectivo (x^0) mantém-se. Relativamente à situação (b), é computacionalmente simples calcular as soluções eficientes correspondentes aos valores de

$\theta_p \leq \theta_p^{max}$. Para continuar, a partir de (a) ou (b), avançamos para um valor de $\theta_p > \theta_p^{max}$ que conduzirá à situação (c).

A situação (c) significa que a solução óptima do programa escalarizante será obtida através de uma árvore possivelmente diferente da que dispomos. Tendo como intenção resolver este programa de uma forma mais eficaz do que recomeçar do início um processo de optimização, partimos da árvore anterior e procedemos à sua *actualização*.

Para a análise de sensibilidade, é necessário analisar alguns dos subproblemas associados aos nodos terminais da anterior árvore de *branch-and-bound*. Veremos a seguir o caso geral e os casos particulares. Indicaremos também a informação da árvore de *branch-and-bound* que é importante preservar para que possa ser feita esta análise de sensibilidade e a actualização da árvore. No final, apresentaremos um esquema geral que estrutura estas duas fases principais, a *análise de sensibilidade* e a *actualização da árvore de branch-and-bound*.

IV.2.1.1 Subproblemas impossíveis

Se um subproblema $Q^r_{\theta_p}$ é impossível para $\theta_p = 0$, então ele é também impossível para todo o $\theta_p > 0$, visto que a região admissível de $Q^r_{\theta_p}$ se reduz com o aumento de θ_p, estando sempre contida na de Q^r_0 (qualquer que seja $\theta_p > 0$). Assim, os nodos (subproblemas) impossíveis da árvore actual podem ser excluídos de qualquer análise posterior.

IV.2.1.2 Comportamento dos outros subproblemas lineares

Analisemos agora o comportamento do subproblema (não impossível) de um qualquer nodo terminal Q^r da árvore de *branch-and-bound* actual. Esse nodo pode ou não corresponder a uma solução inteira (solução que satisfaça as condições de integralidade) ou óptima de $P^{\infty,\rho}_{z^+}$.

Um resultado bem conhecido da programação linear indica que o valor mínimo da função objectivo de um programa paramétrico nos termos independentes das restrições é uma função (do parâmetro) convexa e linear por troços. Os programas $Q^r_{\theta_p}$ são casos particulares do caso geral, em que $f^r(\theta_p)$, além de convexa, é não decrescente e o último troço tem declive 1 (ver exemplo na figura IV.3). A justificação é a seguinte:

Sejam π_i, $i=1,...,k$, as variáveis duais associadas às primeiras k restrições de $Q^r_{\theta_p}$.

Estas variáveis satisfazem $\pi_i \geq 0$, $i=1,...,k$ e $\sum_{i=1}^{k} \pi_i \leq 1$. Mas como z^+ é

inatingível, em qualquer solução admissível de $Q^r_{\theta_p}$ se verifica $\alpha > 0$, o que implica que $\sum_{i=1}^{k} \pi_i = 1$.

A função $f^r(\theta_p)$, $\theta_p \geq 0$, é não decrescente porque os troços têm declives não negativos – o declive de um troço é dado pelo valor de π_p no respectivo intervalo (base) de θ_p.

Conforme θ_p aumenta a partir de 0, $Q^r_{\theta_p}$ gera soluções com valor superior (ou pelo menos igual) para $z_p = c^p x$ até ser alcançada a solução que optimiza z_p em $X^r_{(PL)}$. Se θ_p continuar a aumentar, $Q^r_{\theta_p}$ determinará iguais valores para x, seja $\hat{x}$, podendo variar apenas os valores das variáveis s_i, $i=1,...,k$ e da variável α. Existe, portanto, uma valor particular de θ_p a partir do qual a distância de Tchebycheff entre o ponto de referência e $\hat{z} = C\hat{x}$ é dada exclusivamente pela diferença entre as componentes p dos dois pontos. Ou seja, $s_p=0$ e $s_i>0$, $i \in \{1,\ldots,k\} \setminus \{p\}$. Logo, para θ_p acima desse valor particular, $\pi_i = 0$, $i \in \{1,\ldots,k\} \setminus \{p\}$ e $\pi_p = 1$.

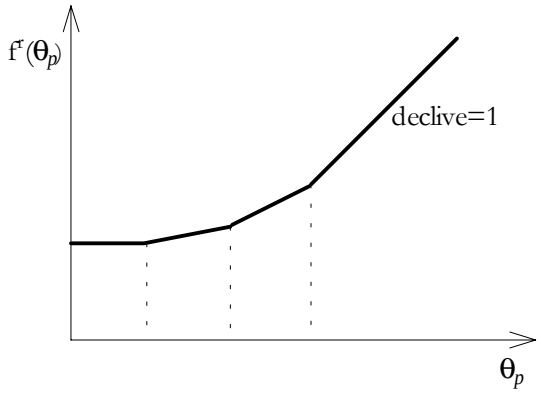


Fig. IV.3 – Exemplo do comportamento da função objectivo de $Q^r_{\theta_p}$.

IV.2.1.3 Análise de sensibilidade

Seja Q^0 o nodo (subproblema) da árvore de *branch-and-bound* a partir do qual se obteve a solução eficiente x^0 que optimiza $P^{\infty,\rho}_{z^+}$. O procedimento de *análise de sensibilidade* que iremos descrever determina um intervalo $[0, \theta_p^{max}]$ (não necessariamente o mais amplo possível) para o parâmetro θ_p que garante que a solução óptima de $P^{\infty,\rho}_{z^+ + \theta_p}$ ainda será dada pelo nodo Q^0.

Consideremos separadamente duas situações, a primeira em que s_p é básica em Q^0, e a segunda em que s_p é não básica em Q^0.

IV.2.1.3.(a) s_p é básica em Q^0

Se s_p é básica em Q^0, então para $\theta_p \leq s_p^0$ (valor actual de s_p) a solução x^0 não se altera e o valor de $f^0(\theta_p)$ também não. Consequentemente, Q^0 continua a ser o nodo óptimo de $P^{\infty,\rho}_{z^+ + \theta_p}$ para $\theta_p \leq s_p^0$, fornecendo a mesma solução eficiente. Por isso, consideramos $\theta_p^{max} = s_p^0$. Não é, pois, necessário explorar quaisquer variações de θ_p abaixo deste valor, sendo z^0 o ponto não dominado mais próximo (segundo a métrica L_∞ aumentada) de todos os pontos de referência desde z^+ a $\left(z_1^+,...,z_p^+ + \theta_p^{max},...,z_k^+\right)$.

Para $\hat{\theta}_p = \theta_p^{max} + \varepsilon$ com $\varepsilon > 0$, a base actual de Q^0 deixa de ser admissível e é necessário proceder à actualização do respectivo quadro simplex. Será necessário também actualizar a informação de outros nodos da árvore para determinar o novo óptimo – explicaremos o processo de o fazer no ponto IV.2.1.7 'ACTUALIZAÇÃO DA ÁRVORE DE *BRANCH-AND-BOUND*'. Conhecemos desde já um limite superior para o óptimo da função de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$, i.e. $f(\hat{\theta}_p)$, que é $f(0) + \varepsilon$ sendo $\alpha^0 + \varepsilon$ a distância de Tchebycheff de z^0 ao ponto de referência $\left(z_1^+,...,z_p^+ + \hat{\theta}_p,...,z_k^+\right)$.

IV.2.1.3.(b) s_p é não básica em Q^0

Seja $\theta_p \in [0,\ \theta_p^{0^{max}}]$ o intervalo positivo de optimalidade para a base actual de Q^0. Analisemos apenas o caso em que a variação de θ_p neste intervalo continua a conduzir a soluções inteiras – situação **(b).1**. Caso contrário, consideramos $\theta_p^{max} = 0$ – situação **(b).2**.

Nas condições de **(b).1**, para $\theta_p \leq \theta_p^{0^{max}}$, a parte x da solução óptima de $Q^0_{\theta_p}$ é admissível para $P^{\infty,\rho}_{z^+ + \theta_p}$ e $f^0(\theta_p) = f^0(0) + \pi_p^0 \theta_p$, em que π_p^0 é o valor actual de π_p no nodo Q^0. Analogamente, seja π_p^r o valor actual de π_p no nodo Q^r. O "desempenho" de Q^0 deve ser comparado com o de outros nodos terminais que sejam *potenciais candidatos* a gerar a nova solução óptima (eles próprios ou seus futuros descendentes). Os nodos *potenciais candidatos* são aqueles que, contendo ou não uma solução inteira, satisfazem $\pi_p^r < \pi_p^0$. Não é necessário examinar os nodos Q^t para os quais $\pi_p^t \geq \pi_p^0$, porque, para $\theta_p \in [0,\ \theta_p^{0^{max}}]$, estes nodos ou os seus descendentes não poderão fornecer

soluções admissíveis para $P^{\infty,\rho}_{z^+ + \theta_p}$ melhores do que a dada por Q^0 (para ilustração, ver a figura IV.4). De facto, mesmo que a base óptima de $Q^t_{\theta_p}$ se altere para $\theta_p < \theta_p^{0^{max}}$, $f^t(\theta_p)$ será sempre superior (pior, visto que $P^{\infty,\rho}_{z^+ + \theta_p}$ é a minimizar) a $f^0(\theta_p)$, porque π^t_p nunca diminuirá com a mudança de base (π^t_p pode ir até 1, o que corresponde à linha superior a tracejado na figura IV.4).

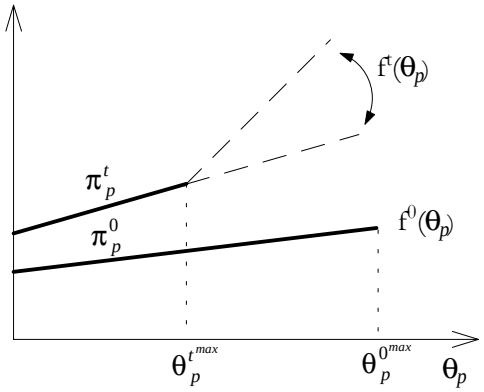


Fig. IV.4 – Exemplo de uma situação em que $\pi^t_p \geq \pi^0_p$.

Para cada nodo terminal Q^r que seja *potencial candidato*, é calculado um valor do parâmetro, $\theta_p^{0,r}$, que designamos por valor de *intersecção*. Utilizando apenas a informação proveniente da base actual óptima associada a cada nodo, $\theta_p^{0,r}$ representa o ponto onde $f^r(\theta_p)$ "intersecta" $f^0(\theta_p)$, podendo tratar-se de uma "intersecção real" (como na figura IV.5a) ou de uma "intersecção virtual" (como na figura IV.5b, c e d):

$$\theta_p^{0,r} = \frac{f^r(0) - f^0(0)}{\pi^0_p - \pi^r_p}$$

Se $\theta_p^{0,r} < \theta_p^{0^{max}}$, então Q^0 dará uma solução melhor do que a de Q^r pelo menos até $\theta_p^{0,r}$ (ver exemplos na figura IV.5a e b). Caso contrário, Q^0 dará uma solução melhor do que a de Q^r pelo menos até $\theta_p^{0^{max}}$ (ver exemplos na figura IV.5c e d). Assim, um valor do parâmetro que assegura que as soluções eficientes seguintes continuam a ser obtidas a partir do nodo Q^0 é $\theta_p^{max} = \min\left\{\theta_p^{0^{max}}, \min_r\left\{\theta_p^{0,r}\right\}\right\}$. As soluções eficientes que optimizam $P^{\infty,\rho}_{z^+ + \theta_p}$ para $0 \leq \theta_p \leq \theta_p^{max}$ podem ser calculadas facilmente através de análise pós-optimização clássica da programação linear a

partir do quadro simplex do nodo Q^0. Consideramos $\hat{\theta}_p = \theta_p^{max} + \varepsilon$ com $\varepsilon > 0$, conhecendo desde já um limite superior para o óptimo da função de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$: $f(\hat{\theta}_p) \leq f(0) + \pi_p^0 \, \theta_p^{max} + \varepsilon$.

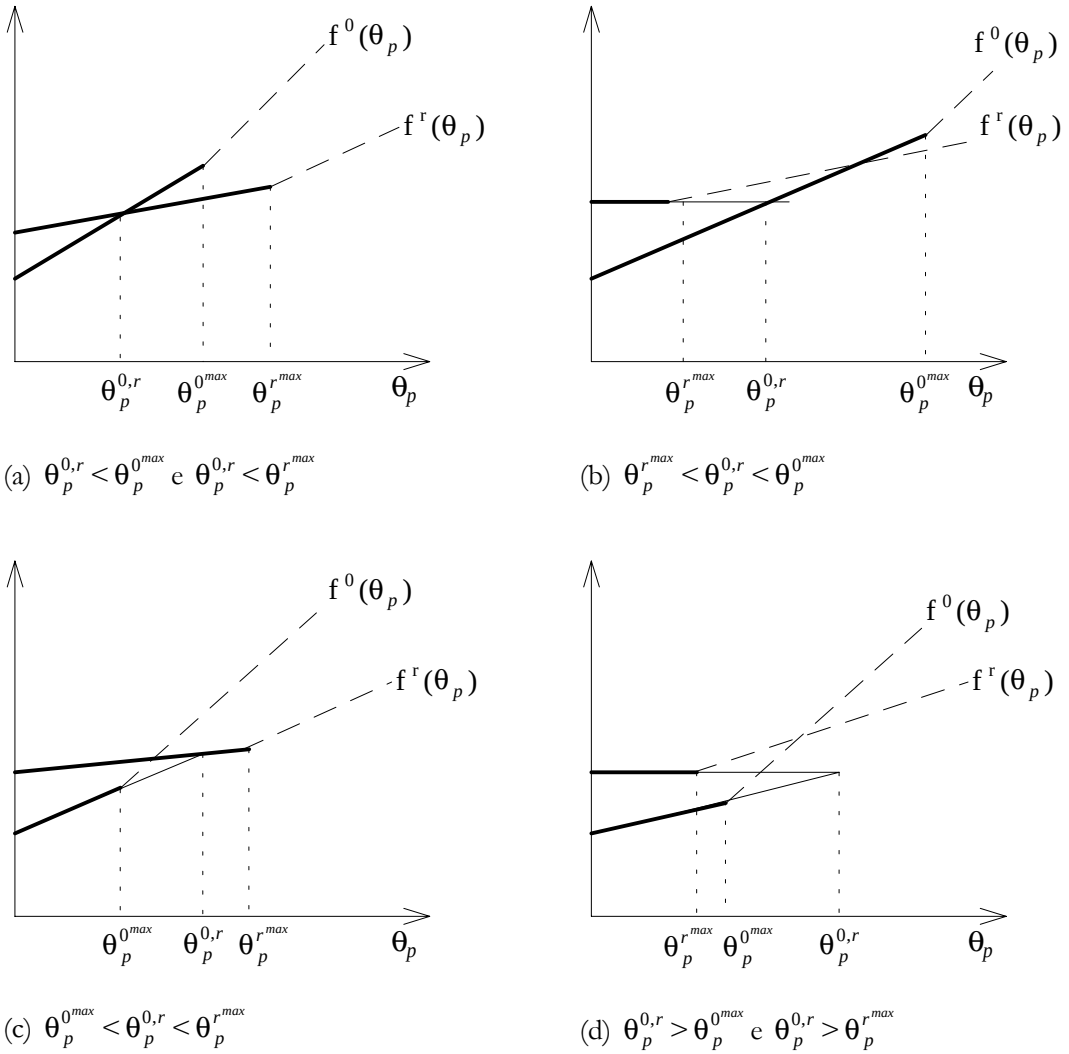


Fig. IV.5 – Exemplos de "intersecções".

Se o problema multiobjectivo for inteiro puro (problema PLIMO definido no capítulo anterior, em que todas as variáveis de decisão e as funções objectivo são inteiras), podemos introduzir melhoramentos na análise desta situação, que se devem ao facto de podermos trabalhar apenas com pontos de referência inteiros. Recordemos que a situação aqui tratada é a **(b).1**, em que a variação contínua de θ_p em $Q^0_{\theta_p}$, no intervalo $[0, \ \theta_p^{0^{max}}]$, continua a produzir soluções inteiras, e s_p é não básica. No caso geral inteiro-misto, podem existir diferentes soluções eficientes vindas deste nodo que diferem nos valores de variáveis contínuas e mantêm os valores das variáveis inteiras. Mas, no caso inteiro puro, a variação contínua do parâmetro mantém a integralidade da solução se todas as variáveis de decisão estiverem num dos seus limites (inferior

ou superior) impostos para este nodo, $\pi_p^0=1$ e $\theta_p^{0^{max}}=+\infty$. Nestas circunstâncias, podemos definir um processo de cálculo diferente para o valor de *intersecção* $\theta_p^{0,r}$ que aqui designaremos por *valor crítico*.

Seja z^+ um ponto de referência inteiro. Consideremos o programa escalarizante $P^{\infty}_{z^+ + \theta_p}$ que apenas difere de $P^{\infty,\rho}_{z^+ + \theta_p}$ pelo facto de a função objectivo ser apenas 'min α'. Como ρ é uma constante positiva muito pequena, assumimos que, se uma solução for óptima para $P^{\infty,\rho}_{z^+ + \theta_p}$, então ela também optimiza $P^{\infty}_{z^+ + \theta_p}$, e esta implicação verifica-se em particular para x^0 (solução eficiente actual). Neste contexto, seja $\alpha^r(\theta_p)$ o mínimo de $Q^r_{\theta_p}$, subproblema de $P^{\infty}_{z^+ + \theta_p}$ associado ao nodo Q^r. Como $\pi_p^0=1$ e $\theta_p^{0^{max}}=+\infty$, então $\alpha^0(\theta_p) = \alpha^0(0) + \theta_p$ para todo o $\theta_p>0$. Qualquer que seja o nodo terminal Q^r com $\pi_p^r<1$, o valor de $\lceil \alpha^r(\theta_p) \rceil$ constitui um limite inferior para α em qualquer solução inteira que possa ser obtida a partir deste ramo da árvore. Considerando apenas valores inteiros de θ_p (aqueles que conduzem a pontos de referência inteiros), e visto que $\lceil \alpha^r(0) + \pi_p^r \theta_p \rceil \le \lceil \alpha^r(\theta_p) \rceil$, $\forall \theta_p>0$, então o *valor crítico* $\theta_p^{0,r}$ pode ser definido por $\theta_p^{0,r} = \max \{\theta_p \text{ inteiro} \mid \alpha^0(0) + \theta_p \le \lceil \alpha^r(0) + \pi_p^r \theta_p \rceil\}$ e $\theta_p^{max} = \min_r \{\theta_p^{0,r}\}$. A figura IV.6 ilustra esta situação, onde os círculos a branco representam $\alpha^0(0) + \theta_p$ e os círculos mais pequenos a cheio representam $\lceil \alpha^r(0) + \pi_p^r \theta_p \rceil$ (que na figura coincide com $\lceil \alpha^r(\theta_p) \rceil$).

A solução x^0 (dada pelo nodo Q^0) mantém-se óptima de $P^{\infty}_{z^+ + \theta_p}$ pelo menos até θ_p^{max}; $\hat{\theta}_p$ será igual a $\theta_p^{max}+1$.

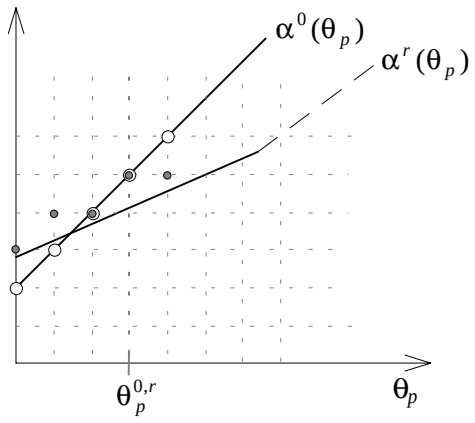


Fig. IV.6 – O 'valor crítico' $\theta_p^{0,r}$ num problema inteiro puro.

Nota: Na situação de óptimos alternativos para $P^{\infty}_{z^+ + \theta_p}$, pode acontecer que x^0 optimize $P^{\infty}_{z^+ + \theta_p}$ e não optimize $P^{\infty,\rho}_{z^+ + \theta_p}$. No entanto, julgamos que esta questão é pouco relevante, porque a única consequência de calcularmos θ_p^{max} (e $\hat{\theta}_p$) desta forma é podermos avançar para um ponto de referência que não é o primeiro ao longo desta direcção a conduzir a uma solução eficiente diferente de x^0 através de $P^{\infty,\rho}_{z^+ + \theta_p}$.

IV.2.1.4 Desactivação de nodos

Um qualquer nodo terminal Q^v diferente de Q^0, para o qual $\pi_p^v = 1$, pode ser considerado temporariamente *inactivo* enquanto a análise paramétrica se referir à restrição p do programa escalarizante. De facto, este nodo ou os seus futuros descendentes não poderão fornecer soluções óptimas do programa escalarizante para valores de $\theta_p > 0$. Q^v será reactivado se o AD alterar a direcção de pesquisa escolhendo uma outra função objectivo para melhorar.

IV.2.1.5 Óptimos individuais

Se $\pi_p^0 = 1$ e todos os outros nodos terminais da árvore de *branch-and-bound* estão *inactivos*, ou os seus programas são impossíveis, então a actual solução eficiente optimiza a função objectivo z_p do problema PLIMMO.

IV.2.1.6 Informação a preservar da árvore de *branch-and-bound*

Para a análise de sensibilidade e posterior actualização da árvore, deve ser preservada a estrutura geral da árvore que inclui os índices dos nodos, as suas ligações e as restrições de ramificação (figura IV.7). Excluem-se os ramos/nodos referentes a subproblemas impossíveis.

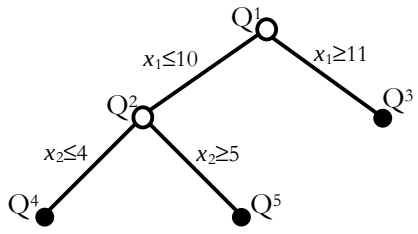


Fig. IV.7 – Exemplo da estrutura da árvore a preservar.

Para cada nodo terminal da árvore Q^r (que na figura IV.7 são Q^3, Q^4 e Q^5) devem ser guardados os seguintes dados:

- uma codificação da base que permita uma reconstrução rápida do quadro simplex – a codificação adoptada é a de potências de 2 para os índices das variáveis básicas (proposta por STEUER (1986) para o cálculo de todos os óptimos alternativos de um programa linear);
- f^r, o valor óptimo actual da função objectivo do subproblema do nodo Q^r;
- π_i^r e $\theta_i^{r^{max}}$ para todo o $i \in \{1,...,k\}$;
- estado actual do nodo (que indica se o nodo contém ou não uma solução inteira ou se está *inactivo*).

IV.2.1.7 Actualização da árvore de *branch-and-bound*

O nodo actualmente óptimo, Q^0, mantém-se óptimo para (pelo menos) $0 \leq \theta_p \leq \theta_p^{max}$, de acordo com o resultado da análise de sensibilidade. A base de Q^0 não se altera e as soluções óptimas são contínuas com a mesma parte inteira (podendo corresponder à mesma solução eficiente – situação IV.2.1.3.(a)). Para estes valores de θ_p a estrutura da árvore não se modifica.

Para $\theta_p = \theta_p^{max}$ estamos perante uma de três situações:

(a) $\theta_p^{max} = 0$ porque a solução proveniente de Q^0 para valores de $\theta_p > 0$ não é inteira (esta situação ocorre no contexto de IV.2.1.3.(b).2);

(b) Q^0 é "interceptado" por outro nodo terminal Q^r (situação IV.2.1.3.(b).1 com $\theta_p^{max} = \theta_p^{0,r}$);

(c) A base óptima de Q^0 muda para $\theta_p > \theta_p^{max}$ (situações IV.2.1.3.(a) ou IV.2.1.3.(b).1 com $\theta_p^{max} = \theta_p^{0^{max}}$).

Para continuar a pesquisa de soluções eficientes ao longo desta direcção, podemos considerar $\hat{\theta}_p = \theta_p^{max} + \varepsilon$, com $\varepsilon > 0$ pequeno, no caso geral de problemas PLIMMO, e ε calculado de modo a que $\hat{\theta}_p = \lfloor \theta_p^{max} \rfloor + 1$ para o caso inteiro puro (em que nos podemos cingir a pontos de referência inteiros sem que haja perda de soluções eficientes intermédias). O novo ponto de referência é então $(z_1^+, ..., z_p^+ + \hat{\theta}_p, ..., z_k^+)$.

Independentemente da situação ser **(a)**, **(b)** ou **(c)**, o processo de actualização da árvore para o novo ponto de referência começa por actualizar a informação do nodo Q^0, ou seja f^0, π_i^0 e $\theta_i^{0^{max}}$, $i = 1,...,k$, alterando a base óptima se for necessário. A informação relativa aos outros nodos terminais será também actualizada mas, nas situações **(b)** e **(c)**, esta tarefa é deixada para mais tarde, uma vez que pode existir uma fase prévia de *simplificação* que elimine partes da árvore.

Nessa altura será actualizada a informação dos nodos terminais que restarem na árvore, procedendo-se do seguinte modo para cada nodo terminal Q^r:

- Se $\theta_p^{r\,max} \leq \hat{\theta}_p$, faz-se a uma mudança de base (aproveitando a codificação da base anterior) e guardam-se os novos valores de f^r, π_i^r e $\theta_i^{r\,max}$, $i=1,...,k$. Se o novo valor de π_p^r for igual a 1 e Q^r não for o novo nodo óptimo, então Q^r ficará *inactivo*.
- Se $\theta_p^{r\,max} > \hat{\theta}_p$, então a base mantém-se e apenas se actualiza f^r e $\theta_p^{r\,max}$: $f^r \leftarrow f^r + \pi_p^r \hat{\theta}_p$ e $\theta_p^{r\,max} \leftarrow \theta_p^{r\,max} - \hat{\theta}_p$. Os valores de π_i^r, $i=1,...,k$, permanecem iguais e os de $\theta_i^{r\,max}$, $i \neq p$, ficam com valor *desconhecido* porque, para além de não serem necessários para a actual direcção de pesquisa, a sua actualização requereria outra informação do quadro simplex que não é explicitamente guardada. Os valores *desconhecidos* só serão calculados quando o AD alterar a direcção de pesquisa.

Analisemos agora as operações específicas de cada situação **(a)**, **(b)** e **(c)** na actualização da árvore para $\theta_p = \hat{\theta}_p$, que tem como intenção calcular a solução óptima de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$.

IV.2.1.7.(a) A solução de Q^0 já não é inteira

Após a actualização da informação de todos os nodos terminais, começa-se por ramificar Q^0 no seguimento de uma estratégia de ramificação pelo *melhor nodo* (melhor limite inferior). O algoritmo de *branch-and-bound* prossegue normalmente até encontrar a solução óptima de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$.

IV.2.1.7.(b) Q^0 foi "interceptado" por outro nodo terminal

Seja Q^r o nodo que "interceptou" Q^0 em $\theta_p^{max} = \theta_p^{0,r}$. É de notar que, se existir mais do que um nodo que "intercepte" Q^0 no intervalo $[\theta_p^{max}, \hat{\theta}_p]$, então todos eles devem ser considerados na análise seguinte.

De acordo com o exposto na situação IV.2.1.3.(b).1, a base óptima de Q^0 mantém-se e fornece uma solução admissível para $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$. Após a actualização da informação dos nodos Q^0 e Q^r, o desempenho de Q^r deve ser novamente comparado com o de Q^0: se $f^0 \leq f^r$, então é porque houve anteriormente uma "intersecção virtual" (como a da figura IV.5b) e Q^0 continua a ser o nodo óptimo; se $f^0 > f^r$ então a solução de Q^r, se for inteira, é a nova solução óptima; caso contrário Q^r é o *melhor nodo* e deverá ser ramificado.

Contudo, se for necessário ramificar Q^r e se já houve mudança(s) de base neste nodo desde a sua criação, podem acontecer situações indesejáveis, como por exemplo a seguinte: suponhamos

que na criação de Q^r foi introduzida a limitação $x_i \leq K_i$ mas, a variação de z^+, fez com que entretanto se alterasse a base óptima de Q^r, passando esta limitação a já não estar activa; a variável inteira x_i tornou-se básica com valor não inteiro inferior a K_i; agora Q^0 será ramificado em dois subproblemas através de limitações em x_i, produzindo uma árvore com ramificações consecutivas na mesma variável! Para que esta situação não aconteça, antes de ramificar Q^r o procedimento começa por examinar a limitação de variável que liga Q^r ao seu pai e, se essa limitação já não estiver activa[3], faz uma simplificação. Dizemos que é uma *simplificação de nível inferior* porque apenas diz respeito ao ramo mais próximo Q^r, aquele que o une ao seu ascendente directo. Isso deve-se ao facto de ser a única limitação de que temos a garantia que estava activa quando o nodo foi criado, já que não é mantida informação histórica. O processo de simplificação consiste no seguinte (figura IV.8): suponhamos que $x_i \leq K_i$ é a limitação não activa em Q^r que liga este nodo ao seu pai; a limitação e o ramo correspondente são eliminados, bem como o pai de Q^r e os seus outros descendentes; Q^r passa a ser um descendente directo do seu antigo avô.

Após a simplificação, actualiza-se a informação dos outros nodos terminais que restaram, ramifica-se Q^r e prossegue-se normalmente com o algoritmo de *branch-and-bound* até se encontrar a solução óptima de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$.

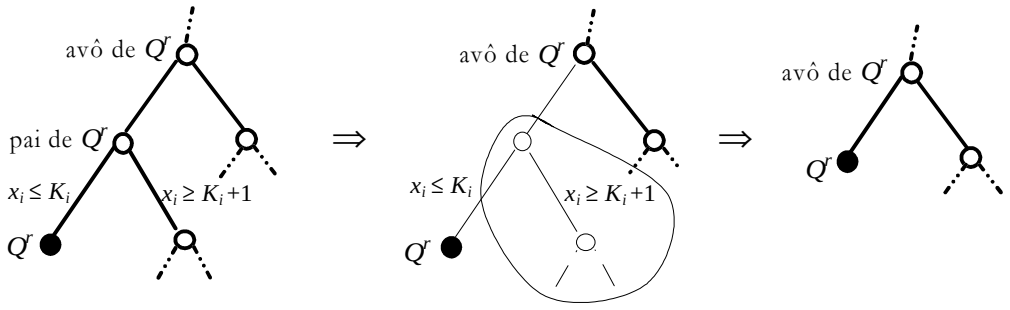


Fig. IV.8 –Simplificação de nível inferior.

IV.2.1.7.(c) A base de Q^0 muda

Se a solução actualizada de Q^0 for inteira, então é esta a solução óptima de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$ e está encontrada a nova solução eficiente. Se não, será necessário continuar com o *branch-and-bound*

[3] Utilizamos a designação de *limitação activa* para uma limitação que é satisfeita como igualdade e não é dispensável. Assim, se existirem limitações $x_i \leq K_i$ e $x_i \geq K_i$, o valor de x_i é forçosamente igual a K_i e as duas limitações são satisfeitas como igualdades, mas apenas uma delas é considerada activa, porque a outra é dispensável (esta informação pode ser obtida através do quadro óptimo do simplex para variáveis limitadas).

começando por ramificar Q^0 (o *melhor nodo*). Contudo, há que prevenir que a árvore esteja sempre a crescer, tentando primeiro simplificá-la. O processo de simplificação refere-se agora a <u>todas</u> as limitações de variáveis que estavam activas na base anterior e que se tornaram redundantes com a actual mudança de base. Faz-se uma simplificação por cada limitação nestas circunstâncias, podendo ser de *nível inferior* se a limitação corresponder ao ramo mais próximo de Q^0 (como em IV.2.1.7.(b)), ou de *nível superior* se corresponder a um ramo superior. Os exemplos IV.1 e IV.2 ilustram vários casos de simplificação.

Exemplo IV.1. Na árvore da figura IV.9, $Q^0 \equiv Q^{15}$ é o nodo óptimo para $\theta_p = 0$ com a solução inteira $x_1=10$, $x_2=4$ e $x_3=2$, estando activas as limitações $x_2 \leq 4$, $x_3 \geq 2$ e $x_1 \geq 10$.

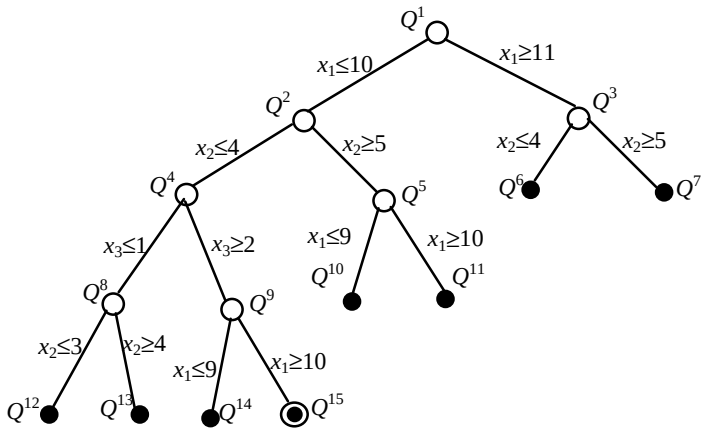


Fig. IV.9 – Árvore inicial do exemplo IV.1.

Para $\theta_p = \hat{\theta}_p$ a base de Q^{15} muda, $x_2 \leq 4$ deixa de estar activa e x_2 toma um valor não inteiro entre 3 e 4. Como na ramificação seguinte Q^{15} será bifurcado em dois subproblemas, um com a limitação $x_2 \leq 3$ e o outro com $x_2 \geq 4$, deve ser feita uma simplificação que elimine a limitação antiga, por forma a que a árvore não seja cada vez mais extensa. Essa simplificação é feita do seguinte modo (figura IV.10):

1. a limitação em causa une Q^4 a Q^2, logo elimina-se esta ligação;
2. liga-se Q^4 directamente ao pai de Q^2, que é Q^1;
3. elimina-se Q^2 e os seus descendentes do outro ramo (direito);
4. quanto a Q^4 e os seus descendentes, deixam-se apenas os nodos intermédios necessários ao alcance de Q^{15} com o cuidado de os manter bifurcados. Assim, mantêm-se Q^4 e Q^9 e uma ligação para a esquerda de Q^4 e uma ligação para a esquerda de Q^9. Q^4 liga-se à esquerda a um "novo Q^8" e Q^9 liga-se à esquerda a um "novo Q^{14}" que, pelo menos temporariamente, são nodos terminais. A figura IV.10 mostra, esquematicamente, estes passos e a figura IV.11 apresenta a árvore simplificada.

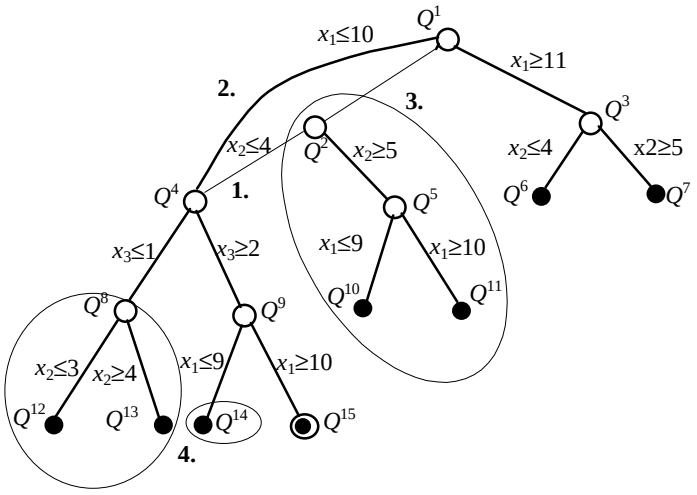


Fig. IV.10 – Simplificação da árvore do exemplo IV.1.

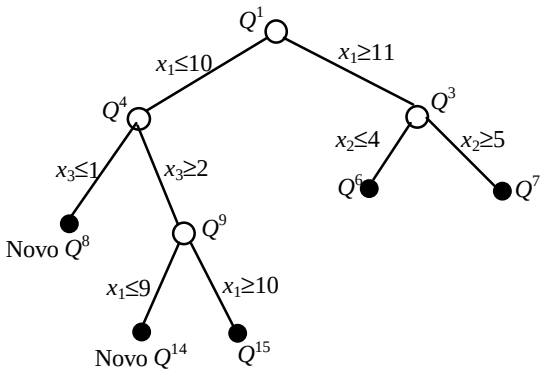


Fig. IV.11 – Árvore do exemplo IV.1 simplificada.

Exemplo IV.2. Na árvore da figura IV.12, $Q^0 \equiv Q^9$ é o nodo óptimo para $\theta_p = 0$ com a solução inteira $x_1=10$ e $x_2=4$, estando activas as limitações $x_2 \le 4$ e $x_1 \ge 10$.

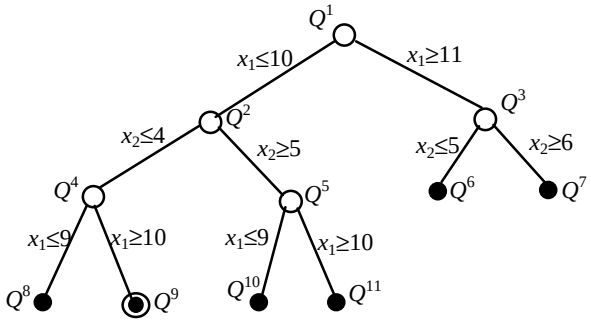


Fig. IV.12 – Árvore inicial do exemplo IV.2.

Para $\theta_p = \hat{\theta}_p$ a base de Q^9 muda e $x_2 \le 4$ deixa de estar activa, mantendo-se activa $x_1 \ge 10$. Seguindo um conjunto de passos semelhantes aos do exemplo IV.1, ilustrados na figura IV.13, a simplificação tem como resultado a árvore da figura IV.14.

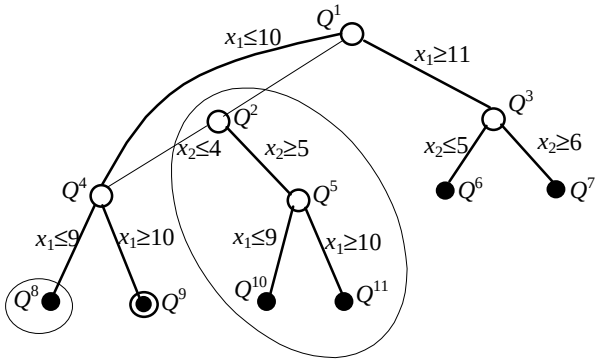


Fig. IV.13 – Simplificação da árvore do exemplo IV.2.

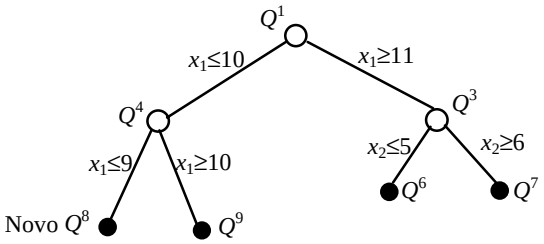


Fig. IV.14 – Árvore do exemplo IV.2 após a simplificação.

Como podemos ver na figura IV.14, esta simplificação conduziu a duas limitações consecutivas segundo a mesma variável (x_1). Neste caso faz-se uma simplificação adicional para eliminar a limitação que não está (e já não estava antes) activa em Q^9 que é $x_1 \leq 10$. A árvore resultante é a da figura IV.15. Notemos que os antigos problemas à direita de Q^1 são agora subproblemas de Q^9.

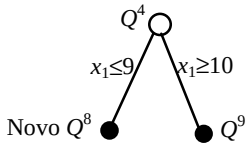


Fig. IV.15 – Árvore do exemplo IV.2 após a simplificação adicional.

Vejamos os passos do processo geral de simplificação.

Para cada limitação que liga Q^w (nodo inferior) a Q^v (nodo superior), que estava activa na base anterior de Q^0 e é agora redundante, faz-se o seguinte (figura IV.16):

1. elimina-se o ramo $Q^w - Q^v$;
2. liga-se Q^w directamente ao pai de Q^v (se Q^v era a raiz da árvore, então Q^w será a nova raiz);
3. elimina-se Q^v, a sua ligação superior e todos os seus descendentes do ramo oposto ao de Q^w;
4. quanto a Q^w e os seus descendentes, deixam-se apenas os nodos intermédios necessários ao alcance de Q^0 com o cuidado de os manter bifurcados – se existirem q nodos intermédios na ligação de Q^w a Q^0, ou seja, $Q^w \equiv Q^{q+1}$, Q^q, Q^{q-1},..., Q^1, Q^0,

substituem-se todos os descendentes de Q^i, $i \in \{1,...,q+1\}$ do ramo oposto ao de Q^{i-1} por um único nodo que, pelo menos temporariamente, é terminal.

Notemos que, se $Q^w \equiv Q^0$, estamos perante uma simplificação de *nível inferior*, apresentada em particular para a situação IV.2.1.7.(b).

Se, após os passos anteriores, surgirem duas ramificações consecutivas segundo a mesma variável, então deve ser feita uma simplificação adicional (seguindo os mesmos passos) para eliminar a limitação que não está activa em Q^0.

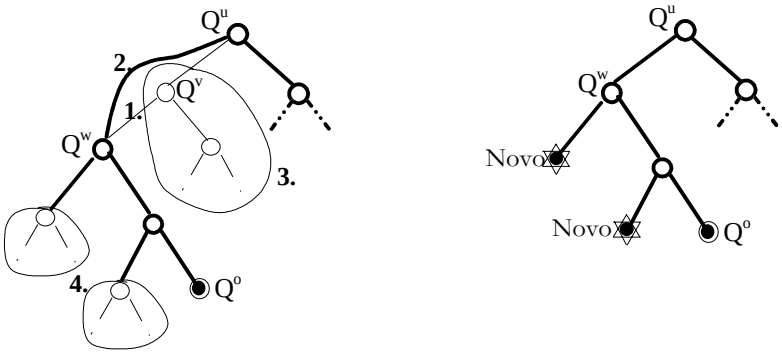


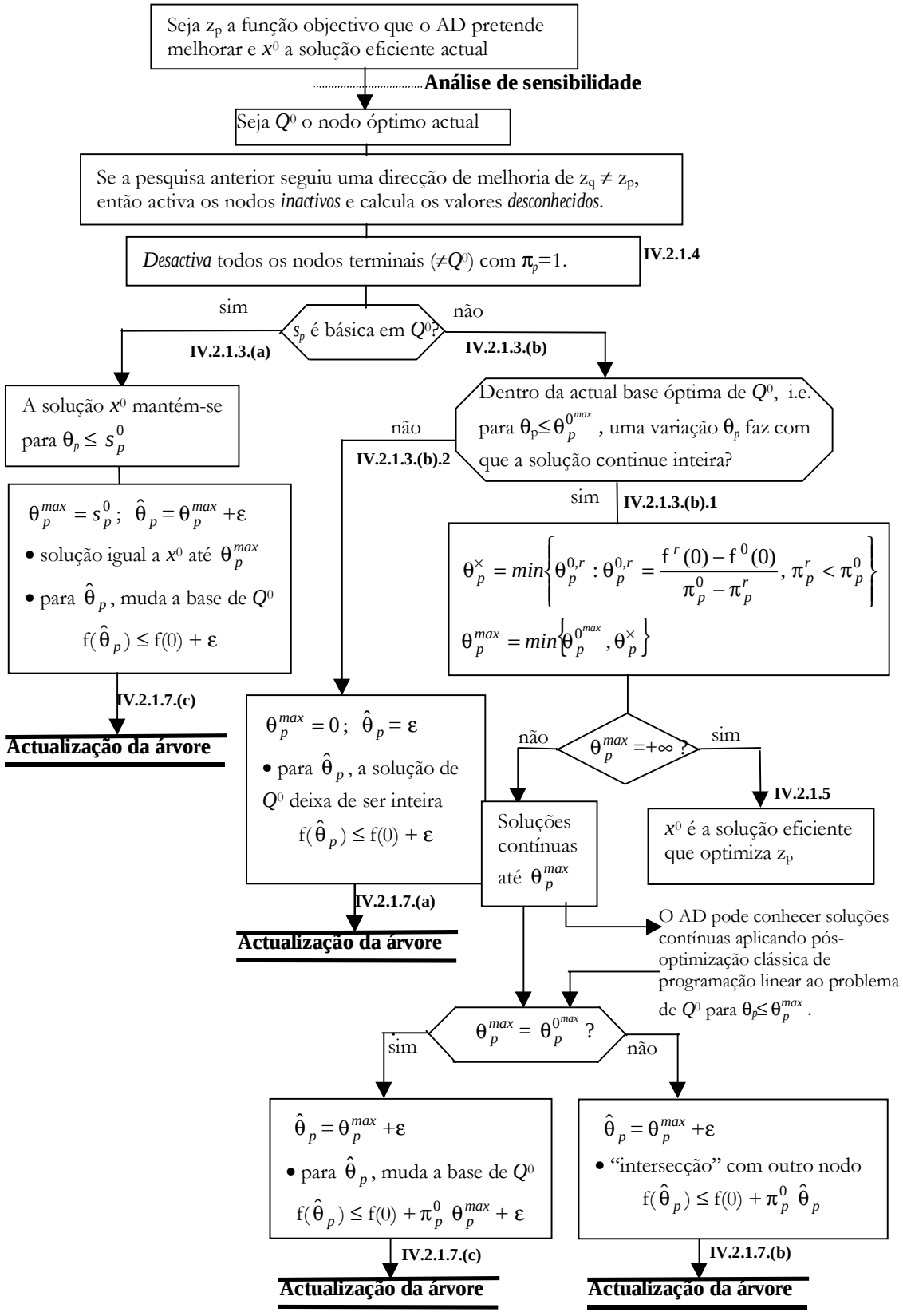
Fig. IV.16 – Ilustração do processo geral de simplificação.

Em suma, se na situação IV.2.1.7.(c) a solução actualizada de Q^0 não for inteira, então o primeiro passo consiste em simplificar a árvore (se possível). Em seguida, actualiza-se a informação nos nodos terminais que restaram e constroem-se os novos nodos terminais que surgiram do processo de simplificação. Por último, expande-se a árvore pelo processo habitual de *branch-and-bound,* começando por ramificar Q^0 e prosseguindo normalmente até ser encontrada a solução óptima de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$.

É de referir que, se ocorrerem ambas as situações IV.2.1.7.(b) e IV.2.1.7.(c) no intervalo $[\theta_p^{max}, \hat{\theta}_p]$, então é necessário testar as condições das duas situações.

IV.2.1.8 Esquema geral

O esquema da figura IV.17 resume o processo de análise de sensibilidade e de actualização da árvore de *branch-and-bound* apresentado nas secções anteriores. Este esquema refere-se ao tratamento de um problema multiobjectivo de programação linear inteira-mista (genérico), não incluindo, portanto, os melhoramentos da situação IV.2.1.3.(b) específicos ao caso inteiro puro.


Seja z_p a função objectivo que o AD pretende melhorar e x^0 a solução eficiente actual
Análise de sensibilidade
Seja Q^0 o nodo óptimo actual
Se a pesquisa anterior seguiu uma direcção de melhoria de $z_q \neq z_p$, então activa os nodos *inactivos* e calcula os valores *desconhecidos*.
Desactiva todos os nodos terminais ($\neq Q^0$) com π_p=1.
IV.2.1.4
sim
s_p é básica em Q^0?
não
IV.2.1.3.(a)
IV.2.1.3.(b)
A solução x^0 mantém-se para $\theta_p \leq s_p^0$
Dentro da actual base óptima de Q^0, i.e. para $\theta_p \leq \theta_p^{0\,max}$, uma variação θ_p faz com que a solução continue inteira?
não
IV.2.1.3.(b).2
sim
IV.2.1.3.(b).1
$\theta_p^{max} = s_p^0$; $\hat{\theta}_p = \theta_p^{max} + \varepsilon$
• solução igual a x^0 até θ_p^{max}
• para $\hat{\theta}_p$, muda a base de Q^0
$f(\hat{\theta}_p) \leq f(0) + \varepsilon$
$\theta_p^{\times} = min\left\{\theta_p^{0,r} : \theta_p^{0,r} = \frac{f^r(0) - f^0(0)}{\pi_p^0 - \pi_p^r}, \pi_p^r < \pi_p^0\right\}$
$\theta_p^{max} = min\left\{\theta_p^{0\,max}, \theta_p^{\times}\right\}$
IV.2.1.7.(c)
Actualização da árvore
$\theta_p^{max} = 0$; $\hat{\theta}_p = \varepsilon$
• para $\hat{\theta}_p$, a solução de Q^0 deixa de ser inteira
$f(\hat{\theta}_p) \leq f(0) + \varepsilon$
não
$\theta_p^{max} = +\infty$?
sim
IV.2.1.5
x^0 é a solução eficiente que optimiza z_p
Soluções contínuas até θ_p^{max}
IV.2.1.7.(a)
Actualização da árvore
O AD pode conhecer soluções contínuas aplicando pós-optimização clássica de programação linear ao problema de Q^0 para $\theta_p \leq \theta_p^{max}$.
$\theta_p^{max} = \theta_p^{0\,max}$?
sim
não
$\hat{\theta}_p = \theta_p^{max} + \varepsilon$
• para $\hat{\theta}_p$, muda a base de Q^0
$f(\hat{\theta}_p) \leq f(0) + \pi_p^0\ \theta_p^{max} + \varepsilon$
IV.2.1.7.(c)
Actualização da árvore
$\hat{\theta}_p = \theta_p^{max} + \varepsilon$
• "intersecção" com outro nodo
$f(\hat{\theta}_p) \leq f(0) + \pi_p^0\ \hat{\theta}_p$
IV.2.1.7.(b)
Actualização da árvore

(Continua...)

Actualização da árvore:

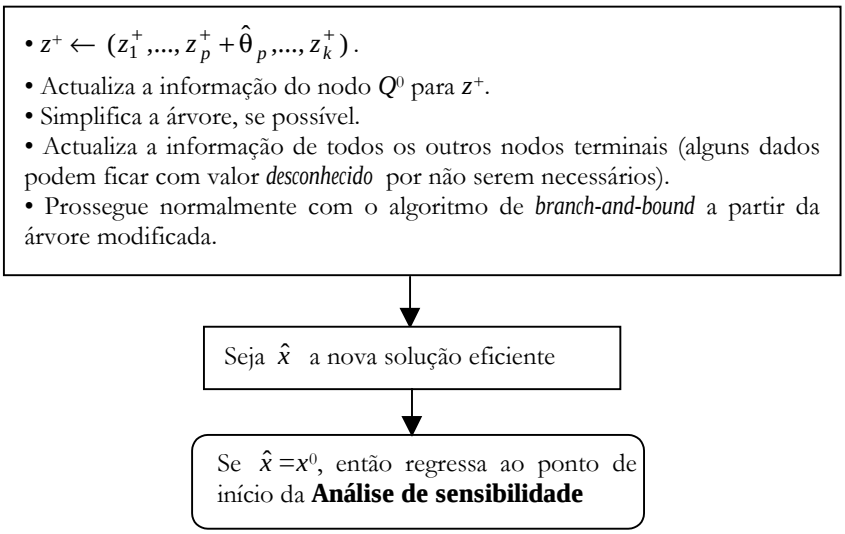


Fig. IV.17 – Esquema geral de análise de sensibilidade e re-optimização numa pesquisa direccional.

IV.2.2 Exemplo ilustrativo

Consideremos o seguinte problema de PLIMMO:

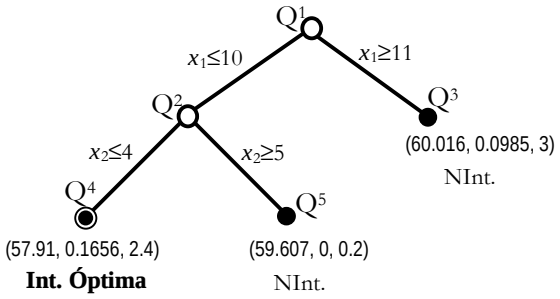
$$\begin{aligned}
\max \quad & z_1 = 3x_1 + x_2 + 2x_3 + x_4 \\
\max \quad & z_2 = x_1 - x_2 + 2x_3 + 4x_4 \\
\max \quad & z_3 = -x_1 + 5x_2 + x_3 + 2x_4 \\
\text{s.a:.} \quad & 2x_1 + x_2 + 4x_3 + 3x_4 \leq 56 \\
& 3x_1 + 4x_2 + x_3 + 2x_4 \leq 55 \\
& x_j \geq 0,\ j=1,..,4 \\
& x_1 \text{ e } x_2 \text{ inteiros}
\end{aligned}$$

Seja $z^+=(108, 80, 75)$ o ponto de referência inicial. O programa escalarizante $P_{z^+}^{\infty,\rho}$ (com $\rho=0.001$) foi resolvido, para o ponto de referência inicial, usando o algoritmo de *branch-and-bound*. A solução eficiente obtida é $x^0=(10, 4, 8, 0)$ cujo ponto dos critérios é $z^0=(50, 22, 18)$.

Suponhamos que o AD pretende fazer uma pesquisa direccional de soluções sucessivamente melhores para a função objectivo z_2. A 2ª componente do ponto de referência é então incrementada e $P_{z^+}^{\infty,\rho}$ dá lugar ao programa de Tchebycheff paramétrico $P_{z^+ + \theta_2}^{\infty,\rho}$. De cada vez que

o AD manifestar a intenção de continuar a pesquisa procede-se a um nova *interacção* e, em cada interacção, há uma ou mais *iterações* do processo de "análise de sensibilidade e actualização da árvore". Há repetição do processo apenas se a solução eficiente encontrada for igual à anterior.

A solução x^0 foi obtida a partir da árvore de *branch-and-bound* da figura IV.18, e esta será a árvore inicial para a próxima interacção. Da informação guardada, a figura IV.18 mostra apenas a que é necessária para a actual pesquisa direccional.



LEGENDA:

$(f^r, \pi_2^r, \theta_2^{r\,max})$ para Q^r

Int.- solução inteira

NInt.- solução não inteira

Fig. IV.18 – A árvore final para o ponto de referência $z^+=(108, 80, 75)$.

Neste exemplo consideramos $0\leq \varepsilon \leq 0.1$ em $\hat{\theta}_2 = \theta_2^{max} + \varepsilon$ e, por razões de simplicidade, ε é escolhido caso a caso de modo a que $\hat{\theta}_2$ tenha apenas um algarismo decimal, i.e. $\hat{\theta}_2 = (\theta_2^{max}$ truncado à 1ª casa decimal) $+0.1$.

ANÁLISE DE SENSIBILIDADE (1ª INTERACÇÃO)

Q^4 (na figura IV.18) é o nodo actualmente óptimo, e a solução deste nodo permanece inteira para variações de θ_2 que não alteram a base, ou seja para $\theta_2 \leq \theta_2^{4\,max} = 2.4$. O desempenho dos outros nodos terminais Q^r tais que $\pi_2^r < \pi_2^4$ é comparado com o de Q^4 calculando-se os respectivos valores de *intersecção* do parâmetro: $\theta_2^{4,3} = 31.386$, $\theta_2^{4,5} = 10.25$; $\theta_2^{max} = \min\{\theta_2^{4\,max}, \theta_2^{4,3}, \theta_2^{4,5}\} = 2.4$.

Q^4 fornece as soluções óptimas de $P^{\infty,\rho}_{z^+ + \theta_2}$ para $0 < \theta_2 \leq 2.4$. Como a base actual de Q^4 permanece admissível nesse intervalo e a variação de θ_2 não destrói a integralidade da solução, então o cálculo das soluções eficientes correspondentes aos pontos de referência desde (108, 80, 75) até (108, 82.4, 75) resulta directamente da aplicação de pós-optimização clássica ao programa linear de Q^4.

Consideremos $\hat{\theta}_2 = 2.4 + \varepsilon_{(1)} = 2.5$ para continuar a pesquisa, sendo o ponto de referência seguinte $z^+ = (108, 82.5, 75)$.

ACTUALIZAÇÃO DA ÁRVORE PARA z^+=(108, 82.5, 75)

Situação **IV.2.1.7.(c)** – A base de Q^4 muda, conduzindo a uma solução não inteira porque a limitação $x_2 \leq 4$ deixou de estar activa. Faz-se uma simplificação de *nível inferior* (figura IV.19), actualizando em seguida a informação do outro nodo terminal que restou, Q^3.

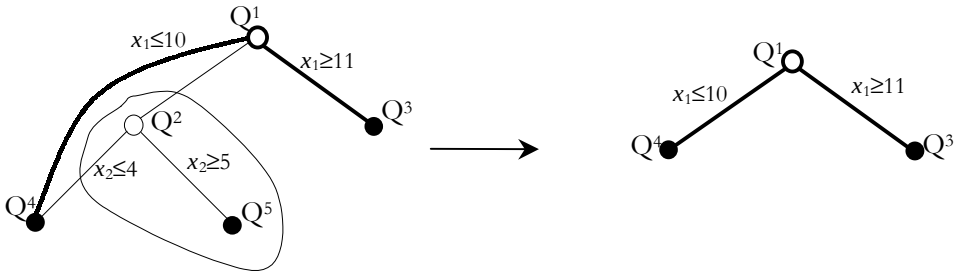


Fig. IV.19 – Simplificação que dá a árvore de partida para z^+=(108, 82.5, 75).

Recomeçando o *branch-and-bound* a partir da árvore simplificada, e continuando até atingir o óptimo do programa escalarizante (figura IV.20), encontra-se uma nova solução eficiente: x^1 =(10, 4, 7.4, 0.8), z^1 =(49.6, 24, 19). Como esperávamos, z_2 melhorou.

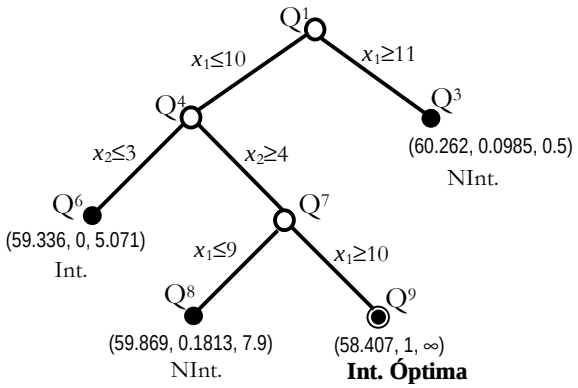


Fig. IV.20 – Árvore final para z^+=(108, 82.5, 75).

Admitindo que o AD pretende continuar a pesquisa nesta direcção, regressa-se à fase de análise de sensibilidade (2ª interacção).

ANÁLISE DE SENSIBILIDADE (2ª INTERACÇÃO)

Q^9 é o novo nodo óptimo e a sua solução permanece inteira para qualquer valor positivo de θ_2; $\theta_2^{max} = \min\{\theta_2^{9^{max}}, \theta_2^{9,6}, \theta_2^{9,8}, \theta_2^{9,3}\} = \min\{\infty, 0.929, *, *\} = 0.929$. Não é necessário calcular os valores assinalados por * porque $f^* > f^6$ e $\pi_2^* > \pi_2^{.6}$, tanto em Q^8, como em Q^3, o que conduz a valores de *intersecção* superiores a 0.929. Assim, para $0 \leq \theta_2 \leq 0.929$, Q^9 continua a ser o nodo óptimo do programa escalarizante.

Consideremos $\hat{\theta}_2 = 0.929 + \varepsilon_{(2)} = 1.0$ para continuar a pesquisa, sendo o ponto de referência seguinte $z^+ = (108, 83.5, 75)$.

ACTUALIZAÇÃO DA ÁRVORE PARA $z^+ = (108, 83.5, 75)$

Situação **IV.2.1.7.(b)** – Q^6 "interceptou" Q^9. A base de Q^6 mantém-se para $\hat{\theta}_2$ e a solução correspondente é inteira o que significa que a estrutura da árvore não se altera, sendo apenas necessário actualizar a informação dos nodos terminais (figura IV.21). A nova solução eficiente é dada por Q^6 e é: $x^2 = (10, 3, 6.857, 1.857)$, $z^2 = (48.571, 28.143, 15.571)$.

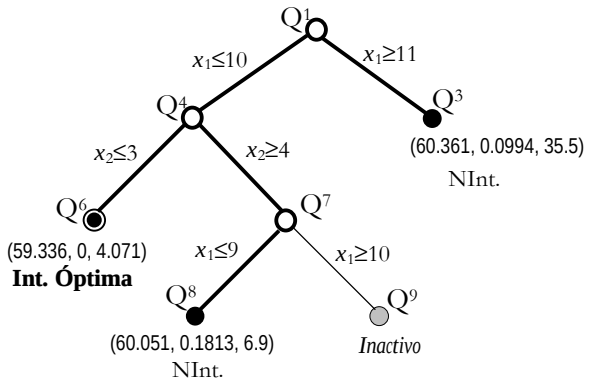


Fig. IV.21 – Árvore final para $z^+ = (108, 83.5, 75)$.

ANÁLISE DE SENSIBILIDADE (3ª INTERACÇÃO)

Q^6 (figura IV.21) é o nodo actualmente óptimo em que a variável s_2 (variável desvio da segunda restrição do programa escalarizante) é básica e $\pi_2^6 = 0$ – situação **IV.2.1.3.(a)**. Consequentemente, o anterior ponto não dominado z^2 é o mais próximo (segundo L_∞) de todos os pontos de referência $(108, 83.5 + \theta_2, 75)$ para $0 < \theta_2 \leq \theta_2^{6^{max}} = s_2^6 = 4.071$, ou seja, a anterior solução eficiente x^2 mantém-se até $\theta_2^{max} = 4.071$. Consideremos então $\hat{\theta}_2 = 4.071 + \varepsilon_{(3)} = 4.1$ para continuar a pesquisa, sendo o ponto de referência seguinte $z^+ = (108, 87.6, 75)$.

ACTUALIZAÇÃO DA ÁRVORE PARA $z^+ = (108, 87.6, 75)$

Situação **IV.2.1.7.(c)** – A base de Q^6 muda, mas Q^6 continua a ser o nodo óptimo porque a sua solução é inteira e apresenta o menor valor de f. A nova solução é: $x^3 = (10, 3, 6.850, 1.867)$, $z^3 = (48.567, 28.167, 15.583)$.

Continuando a pesquisa na mesma direcção, o processo repete-se. De modo a ilustrar situações diferentes das anteriores, evitando repetir as já exemplificadas, iremos avançar várias interacções passando directamente para o ponto de referência z^+=(108, 120.5, 75). A árvore correspondente é da figura IV.22, cuja solução é: x =(10, 0, 0.875, 10.833), z =(42.583, 55.083, 12.542).

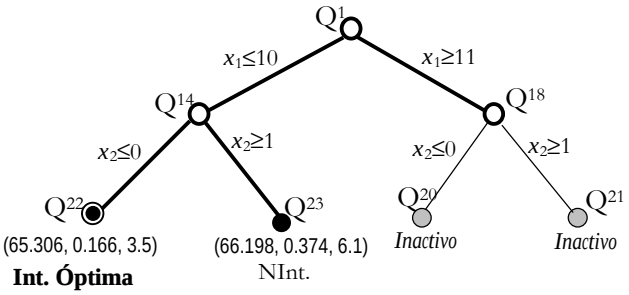


Fig. IV.22 – Árvore final para z^+=(108, 120.5, 75).

ANÁLISE DE SENSIBILIDADE (NOVA INTERACÇÃO)

Q^{22} (figura IV.22) é o nodo actualmente óptimo e a sua solução permanece inteira para variações de θ_2 que não alterem a base, ou seja, para $\theta_2 \leq \theta_2^{22^{max}}$ =3.5. Além disso, nenhum outro nodo satisfaz os requisitos básicos para testar a "intersecção" com Q^{22}. Logo, Q^{22} é o nodo óptimo do programa escalarizante para $0 \leq \theta_2 \leq 3.5$, e o cálculo das soluções não dominadas mais próximas dos pontos de referência desde (108, 120.5, 75) a (108, 124, 75) é directo.

Consideremos $\hat{\theta}_2 = 3.5 + \varepsilon = 3.6$ para continuar a pesquisa, sendo o ponto de referência seguinte z^+=(108, 124.1, 75).

ACTUALIZAÇÃO DA ÁRVORE PARA z^+=(108, 124.1, 75)

Situação **IV.2.1.7.(c)** – A base de Q^{22} muda e as limitações $x_2 \leq 0$ e $x_1 \leq 10$ deixam de estar activas. Apesar de a condição de não-negatividade de x_2 se tornar activa, obrigando a x_2=0, a solução não é inteira porque viola a condição de integralidade de x_1 (variável que assume o valor 9.975). Q^{22} é o *melhor nodo* e deve ser ramificado mas, antes de o fazer, a árvore deve sofrer duas simplificações, uma por cada limitação. Começa-se pela limitação $x_2 \leq 0$, e depois $x_1 \leq 10$, o que resulta em duas simplificações consecutivas de *nível inferior* (figura IV.23).

Recomeçando o *branch-and-bound* a partir da árvore simplificada (apenas o nodo Q^{22}), e continuando até se atingir o óptimo do programa escalarizante (figura IV.24), obtém-se uma nova solução eficiente: x =(10, 0, 0, 12), z =(42, 58, 14).

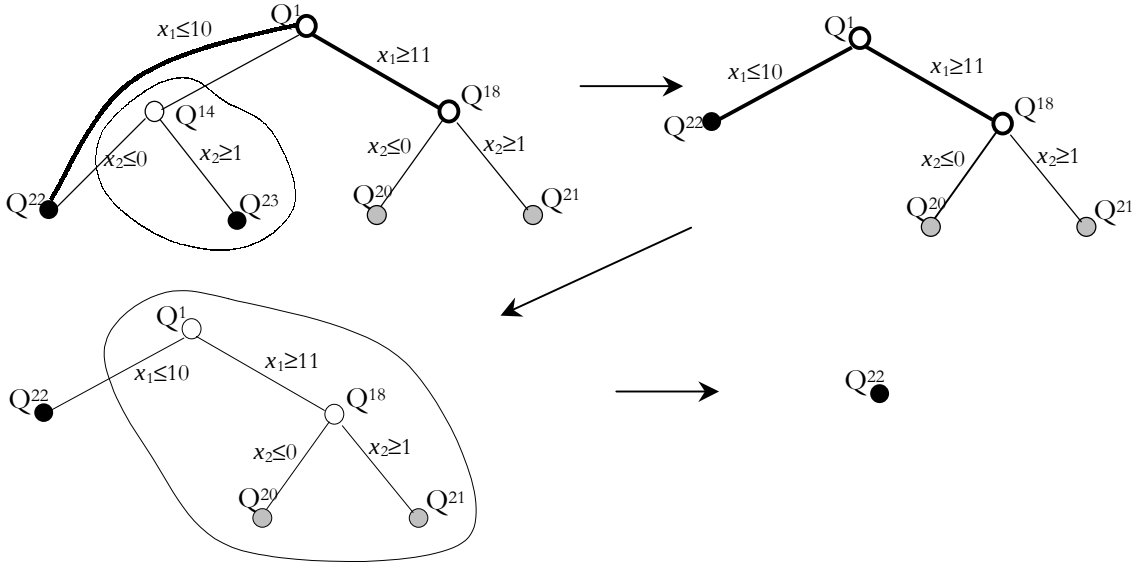


Fig. IV.23 – Simplificação que dá a árvore de partida para $z^+=(108, 124.1, 75)$.

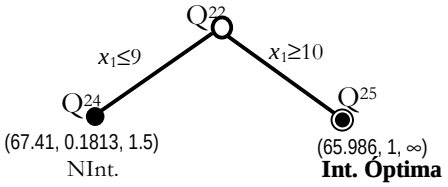


Fig. IV.24 –Árvore final para $z^+=(108, 124.1, 75)$.

Nota: Neste exemplo nunca foi necessário mais do que uma iteração de "análise de sensibilidade e actualização da árvore" por interacção. A necessidade de repetir o processo para abandonar a solução anterior é muito mais frequente em problemas inteiros puros.

IV.2.3 Resultados computacionais

O método interactivo que propusemos foi implementado no ambiente de desenvolvimento DELPHI e integrado no sistema computacional que se havíamos iniciado com o método de planos de corte. Usufrui do mesmo ambiente de interacção com o AD e de iguais ferramentas de edição de problemas, pré-processamento e códigos para programação linear (*LP_SOLVE* e *LP_OPTIMIZER*) – para detalhes, ver o capítulo III.

O método foi testado em vários problemas de PLIMO e de PLIMMO gerados aleatoriamente. Considerámos um conjunto de 100 problemas dos quais 27 são genéricos (15 inteiros puros e 12 inteiros-mistos) e os outros 73 são das classes de problemas testados com o método de planos de corte, mas com um maior número de variáveis: problemas de mochila, *knapsack* 0-1 ('KS_'), *knapsack* com variáveis inteiras ('KI_'), *knapsack* 0-1 multidimensional

('KM_'), problemas de cobertura ('C_'), e problemas de (*packing*) embalagem ('P_'). O anexo IV.A complementa o anexo III.B indicando alterações ou outras regras usadas na geração aleatória destes problemas. Atendendo a que são quase inexistentes as experiências computacionais com problemas de PLIMO ou PLIMMO relatadas na literatura, não é fácil encontrar colectâneas destes problemas. Contudo, na geração aleatória dos problemas de PLIMO genéricos, resolvemos utilizar as mesmas regras de KARAIVANOVA ET AL. (1993), que é um dos poucos artigos onde esta informação é fornecida.

A tabela IV.1 mostra alguns dos resultados de experiências computacionais do método multiobjectivo baseado em *branch-and-bound*. Esta tabela é um extracto da tabela mais completa apresentada no anexo IV.B. Para este extracto ilustrativo, retirámos o primeiro problema de cada tipo com 3 funções objectivo, de entre o subconjunto com maior número de variáveis. Juntámos ainda três outros problemas porque ilustram diferentes comportamentos que comentaremos adiante.

Para cada problema fizemos uma breve pesquisa direccional, em que indicamos o ponto de referência inicial, sendo os seguintes ajustados automaticamente pelo método. Assim, nas tabelas IV.1 e do anexo IV.B, a primeira interacção ('Início') refere-se a uma resolução independente do programa escalarizante, e cada uma das interacções seguintes ('melhorar...') refere-se ao processo iterativo de análise de sensibilidade e actualização da árvore de *branch-and-bound*. As tabelas indicam, para além do ponto de referência e a solução não dominada de cada interacção, o número de iterações (*It.*) e o *tempo* total gasto (em segundos) no processo iterativo de análise de sensibilidade e actualização da árvore de *branch-and-bound*. Estas tabelas apresentam ainda, na coluna intitulada '*Re-início*', o tempo (em segundos) da resolução independente (usando *branch-and-bound*) do programa escalarizante correspondente ao ponto de referência final de cada interacção. Todos os tempos se referem a um computador PENTIUM II a 350MHz.

Problema	*Estratégia*	*P^to de Referência*	*Solução (z_1,z_2...)*	*It.*	*tempo (seg)*	*'Re-início' (seg)*
KS1_3_200	Início	(3708, 3972, 3935)	(3351, 3616, 3581)	–	468.1	
(200 variáveis	melhorar z_3	(3708, 3972, 3939)	(3353, 3614, 3584)	1	144.4	390.8
	melhorar z_3	(3708, 3972, 3943)	(3350, 3614, 3586)	1	64.8	146.0
KI1_3_200	Início	(30496,30124, 31786)	(27684, 27304, 28966)	–	25.1	
(200 variáveis)	melhorar z_1	(30514,30124, 31786)	(27685, 27302, 28957)	6	146.9	153.0
	melhorar z_1	(30515,30124, 31786)	(27692, 27298, 28956)	1	43.7	161.4
KM3_2_50_10	Início	(994, 1060)	(823, 889)	–	16.2	
(50 variáveis	melhorar z_1	(1012, 1060)	(827, 872)	10	33.9	32.4
10 restrições)	melhorar z_1	(1053, 1060)	(837, 835)	20	125.6	94.3
KM3_3_50_10	Início	(985, 898, 824)	(851, 747, 664)	–	28.6	
(50 variáveis	melhorar z_3	(985, 898, 844)	(806, 731, 695)	11	115.5	114.2
10 restrições)	melhorar z_3	(985, 898, 887)	(804, 707, 698)	8	104.6	58.0
KM2_3_60_10	Início	(1111, 947, 986)	(928, 758, 813)	–	70.1	
(60 variáveis	melhorar z_3	(1111, 947, 1013)	(912, 796, 818)	6	67.5	84.3
10 restrições)	melhorar z_3	(1111, 947, 1019)	(911, 770, 841)	2	14.4	76.3
C1_3_250_50	Início	(29, 56, 36)	(191, 219, 193)	–	31.0	
(250 variáveis	melhorar z_2	(29, 54, 36)	(167, 197, 200)	1	3.5	31.2
50 restrições)	melhorar z_2	(29, 21, 36)	(204, 195, 210)	3	18.1	21.7
P2_3_250_50	Início	(726, 812, 813)	(536, 625, 702)	–	39.7	
(250 variáveis	melhorar z_1	(761, 812, 813)	(542, 657, 589)	16	95.5	97.6
50 restrições)	melhorar z_1	(787, 812, 813)	(604, 567, 657)	11	95.3	121.0
I1_2_15_15	Início	(704, 789)	(500, 626)	–	11.6	
(15 variáveis	melhorar z_1	(734, 789)	(508, 556)	14	14.4	23.7
15 restrições)	melhorar z_1	(761, 789)	(523, 537)	11	10.7	23.0
I1_3_30_30	Início	(1576, 1284, 1222)	(1119, 794, 748)	–	926.1	
(30 variáveis	melhorar z_2	(1576, 1286, 1222)	(1085, 808, 1222)	1	64.8	961.1
30 restrições)	melhorar z_2	(1576, 1304, 1222)	(1081, 826, 807)	3	432.6	756.3
IM1_3_100_15	Início	(4724, 3929, 4387)	(3756.4, 2961.4, 3419.4)	–	26.5	
(100 variáveis	melhorar z_2	(4724, 3931, 4387)	[a] (3755.7, 2962.7, 3418.7)	1	2.5	28.6
15 restrições)	melhorar z_2	(4724, 3933.1, 4387)	[b] (3755.1, 2964.2, 3418.1)	1	1.0	28.7

(a) houve alteração nos valores das variáveis inteiras (b) apenas se alteraram valores de variáveis contínuas

Tabela IV.1 – Extracto da tabela do anexo IV.B com resultados de algumas pesquisas direccionais.

A tabela IV.2 sintetiza estas experiências, apresentando os tempos médios para cada tipo de problemas.

Nota: Os tempos de re-início usados para calcular os valores médios da última coluna da tabela IV.2 são os que figuram na coluna '*Re-início*' da tabela do anexo IV.B. São, portanto, aqueles que têm correspondência na coluna anterior da mesma tabela ('melhorar...'), o que exclui os tempos de 'Início' de uma pesquisa direccional. Esta opção teve como intenção possibilitar uma comparação mais fiável das duas colunas de tempos da tabela IV.2.

Tipo de problema	*n*	*m*	Nº de Prob.	Tempos médios	
				Análise de sensibilidade + Actualização da árvore	Resolução independente de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$ (*Re-início*)
Knapsack 0-1	100 B	1	6	**53”**	1’ 13”
	200 B	1	6	**2’ 42”**	4’ 4”
Knapsack inteiro	100 I (*)	1	6	**32”**	33”
	200 I (*)	1	6	**1’ 27”**	2’ 7”
Knapsack 0-1 multidimensional	50 B	10	6	55”	**54”**
	60 B	10	5	**2’ 23”**	3’ 34”
Cobertura	100 B	20-30	5	2.1”	**1.7”**
	200 B	30	7	**6.2”**	11.3”
	250 B	50	8	**24”**	37”
Embalagem	100 B	20	6	1.1”	**1.0”**
	200 B	30	6	**6.5”**	7.3”
	250 B	50	6	**33”**	41”
Inteiros puros genéricos	15 I	15	3	**15”**	19”
	30 I	15	6	**1’ 52”**	2’ 29”
	30 I	30	6	**4’ 48”**	11’ 58”
Inteiros-mistos genéricos	20 I+20 B+40 C	40	6	**0.5”**	5.8”
	35 I+35 B+30C	15	6	**2.8”**	26.2”

B: binária; I: inteira; C: contínua (*) $0 \leq x_j \leq 10$, $j=1,...,n$

Tabela IV.2 – Tempos médios dos testes computacionais num PENTIUM II a 350 MHz.

O método mostrou-se robusto no conjunto de problemas testado, não ocorrendo qualquer erro numérico.

Observámos que, na maior parte dos problemas testados, os tempos de cálculo diminuiram pelo uso da estratégia de simplificação/ramificação a partir da árvore de *branch-and-bound* anterior, relativamente ao re-início. Este facto é particularmente relevante e sistemático em problemas inteiros-mistos e problemas de maior dimensão. Recordamos que os tempos apresentados para as pesquisas direccionais (cujos valores médios estão na penúltima coluna da tabela IV.2) incluem, não só o tempo gasto no cálculo, como também na análise de sensibilidade. É de notar ainda que, sem o processo de análise de sensibilidade, seriam necessárias várias tentativas de incremento de θ_p (talvez através de um passo) para abandonar a solução anterior, não havendo a garantia de se obter a solução mais próxima segundo a direcção especificada.

Se, por um lado, os problemas inteiros-mistos requerem em geral apenas uma iteração de “análise de sensibilidade e actualização da árvore” por interacção, já os problemas inteiros puros precisam muitas vezes de repetir o processo para abandonar a solução eficiente anterior. Não obstante, os tempos totais gastos são em geral bons, se comparados com a optimização independente do programa escalarizante correspondente ao ponto de referência final desse processo. Excluem-se os casos em que o número de iterações é muito elevado. Não nos

atrevemos, contudo, a quantificar um valor crítico genérico para o número de iterações dado depender de problema para problema. A título de exemplo, reparemos que as optimizações independentes foram mais rápidas no problema KM3_2_50_10, em comparação com processos de 10 e 20 iterações de "análise de sensibilidade e actualização da árvore"; o mesmo se passou para KM3_3_50_10 com processos de 11 e 8 iterações; a situação oposta aconteceu para P2_3_250_50 com 16 e 11 iterações, e para I1_2_15_15 com 14 e 11 iterações.

Os tempos médios apresentados na tabela IV.2 mostram que a vantagem computacional da estratégia que propomos ganha relevância conforme se aumentam as dimensões dos problemas. É nos problemas maiores que os tempos são melhores. Como curiosidade referimos que, nas primeiras experiências que fizemos numa máquina mais antiga (computador PENTIUM I a 166MHz, em que os tempos são globalmente maiores), a vantagem da nossa estratégia foi visível logo a partir dos problemas pequenos (em número de variáveis). Apresentamos na tabela IV. 3 o quadro resumo dessas experiências.

(Nota: igual descrição dos problemas – *tipo*, *n* e *m* – nas tabelas IV.2 e IV.3 significa que os problemas são os mesmos).

Tipo de problema	*n*	*m*	Nº de Prob.	Tempos médios	
				Análise de sensibilidade + Actualização da árvore	Resolução independente de $P^{\infty,\rho}_{z^+ + \hat{\theta}_p}$ (*Re-início*)
Knapsack 0-1	100 B	1	6	**2' 30"**	3' 32"
	200 B	1	6	**3' 37"**	6' 08"
Knapsack inteiro	100 I (*)	1	6	**50"**	1" 00"
	200 I (*)	1	6	**2' 56"**	6' 01"
Knapsack 0-1 multidimensional	50 B	10	6	**1' 26"**	2' 05"
	60 B	10	5	**7' 08"**	11' 50"
Cobertura	100 B	20-30	5	19"	**18"**
Embalagem	100 B	20	6	**16"**	22"
	200 B	30	6	**1' 08"**	3' 41"
Inteiros puros genéricos	30 I	15	6	**2' 48"**	4' 30"
Inteiros-mistos genéricos	15 I+15 B+30 C	30	6	**2.8"**	39"
	20 I+20 B+40C	40	6	**3.0"**	1' 04"

B: binária; I: inteira; C: contínua (*) $0 \leq x_j \leq 10$, $j=1,...,n$

Tabela IV.3 – Resumo de testes computacionais anteriores num PENTIUM I a 166 MHz.

IV.3 PROPOSTA DE UMA ABORDAGEM COMBINADA PARA PLIMO

Os anteriores métodos interactivos multiobjectivo – um baseado em *planos de corte* e o outro baseado em *branch-and-bound* – consideram o mesmo tipo de protocolo para interagir com o AD, diferindo apenas na técnica utilizada e pelo facto do primeiro se destinar apenas a problemas de PLIMO. Podemos, pois, considerar uma abordagem que combine os dois tipos de técnicas que, tal como a de *planos de corte*, se limitará a problemas de PLIMO.

IV.3.1 Algoritmo

O algoritmo que propomos em seguida é uma combinação simples das duas abordagens anteriores e divide-se em duas partes principais: o *'Início'*, em que se optimiza o programa escalarizante para um ponto de referência inicial de modo a obter uma primeira solução eficiente; e *'Soluções seguintes duma pesquisa direccional'*, em que se aproveita o cálculo anterior para determinar a solução eficiente seguinte duma pesquisa direccional.

IV.3.1.1 Início

Escolhido o ponto de referência, o algoritmo começa por aplicar *planos de corte* na resolução do programa escalarizante até se verificar uma das seguintes condições de paragem:

P1) foi encontrada uma solução inteira – termina com a abordagem de *planos de corte*;

P2) os cortes de Gomory começam a ser muito próximos uns dos outros, o que indicia que provavelmente ocorrerão erros numéricos;

P3) já foram feitas muitas iterações do simplex (em número considerado demasiado elevado).

Se se verificar uma das condições P2 ou P3, então avança-se para a técnica de *branch-and-bound* incluindo os planos de corte activos no momento da paragem. Termina quando for encontrada a solução óptima do problema escalarizante segundo o *branch-and-bound*.

IV.3.1.2 Soluções seguintes duma pesquisa direccional

IV.3.1.2.(a) Se a última solução eficiente foi obtida por *planos de corte*, então prossegue-se normalmente com esta abordagem alternando fases de análise de sensibilidade e cálculo com planos de corte. O processo termina se se verificar uma das condições de paragem (P1, P2 ou P3), ou se tiver passado um número suficientemente grande de iterações sem que o ponto de referência tenha sido actualizado (condição designada por P4). Em caso de paragem prematura (P2, P3 ou P4), avança-se para o *branch-and-bound*, incluindo os planos de corte activos no momento da paragem. Se a solução obtida por *branch-and-bound* for igual à anterior, então

continua-se com esta estratégia usando o processo iterativo de análise de sensibilidade e actualização da árvore, até que seja encontrada uma solução eficiente diferente da anterior.

IV.3.1.2.(b) Se a última solução eficiente foi obtida por *branch-and-bound*, então faz-se uma análise de sensibilidade para ajustar o ponto de referência (a partir da informação proveniente da árvore). Recordamos que, no método de *branch-and-bound*, a análise de sensibilidade pretende determinar o maior incremento do ponto de referência que mantém o nodo actualmente óptimo Q^0. Mas, como a estratégia combinada se destina unicamente a problemas inteiros puros, modificamos o propósito desta análise para o seguinte: a análise de sensibilidade pretende determinar o maior incremento do ponto de referência que conduz à mesma solução eficiente, independentemente de ser ou não obtida a partir de Q^0 ou de ser necessário re-estruturar a árvore. Isto não altera o processo de análise de sensibilidade das situações IV.2.1.3.(a) ou IV.2.1.3.(b).1, se considerarmos as comparações específicas aos problemas inteiros puros, mas afecta a situação IV.2.1.3.(b).2. O método isolado de *branch-and-bound* (para problemas inteiros ou inteiros-mistos) considera $\theta_p^{max}=0$ em IV.2.1.3.(b).2. Porém, dada uma qualquer variação $\theta_p>0$ no ponto de referência, sabemos que o valor óptimo da variável α no respectivo programa escalarizante é limitado superiormente por $\alpha^0(0) + \theta_p$. A solução de outros nodos Q^r pode então ser comparada com a solução eficiente actual x^0. A comparação é semelhante à de IV.2.1.3.(b).1, em que se determina um *valor crítico* do parâmetro $\theta_p^{0,r}$ para cada Q^r, e $\theta_p^{max} = \min_r \{\theta_p^{0,r}\}$; $\theta_p^{0,r} = \max \{\theta_p \text{ inteiro} \mid \alpha^0(0) + \theta_p \le \lceil \alpha^r(0) + \pi_p^r \theta_p \rceil\}$. A solução x^0 mantém-se para $\theta_p \le \theta_p^{max}$, mas não necessariamente em Q^0.

Considera-se o incremento $\hat{\theta}_p = \theta_p^{max} + 1$ na componente p do ponto de referência. Se a análise de sensibilidade foi do tipo IV.2.1.3.(a) ou IV.2.1.3.(b).1, então prossegue-se normalmente com a abordagem de *branch-and-bound*, avançando para a fase de actualização da árvore. Se a situação foi do tipo IV.2.1.3.(b).2, apaga-se a árvore de *branch-and-bound* anterior e inicia-se uma resolução independente do programa escalarizante correspondente ao novo ponto de referência, começando por usar *planos de corte*. Tirando partido do limite superior de α fornecido pelo *branch-and-bound*, prossegue-se com o processo iterativo de análise de sensibilidade e cálculo com *planos de corte*, até se verificar uma das condições de paragem, P1, P2, P3 ou P4. Se a paragem for prematura (P2, P3 ou P4), o procedimento regressa ao *branch-and-bound*, mas começa de raiz uma nova árvore.

IV.3.2. Implementação computacional e testes

Na implementação computacional da abordagem combinada, as condições estabelecidas em P2, P3 e P4 foram quantificadas da seguinte forma:

P2) um indício de prováveis erros numéricos é quando os cortes de Gomory, traduzidos para função das variáveis principais, apresentam coeficientes muito elevados; considerámos elevado algum desses coeficientes se for, em valor absoluto, superior a 100 vezes o maior dos coeficientes que figura no programa escalarizante;

P3) considerámos $25n$ (sendo n o número de variáveis de decisão) para quantificar um número suficientemente grande de iterações;

P4) considerámos $10n$ para quantificar um número suficientemente grande de iterações sem que haja actualização do ponto de referência.

Nota: estas medidas tiveram apenas como intenção possibilitar algumas experiências computacionais preliminares, não tendo sido feito qualquer estudo para a sua afinação.

A experiência computacional consistiu em fazer pesquisas direccionais iguais às efectuadas com o método (isolado) de *branch-and-bound* para alguns dos problemas. Observámos que a abordagem combinada reduziu muitas vezes o tamanho das árvores de *branch-and-bound* (menor número de nodos terminais) e foram necessárias menos iterações de "análise de sensibilidade e actualização da árvore" em cada interacção. Contudo, os tempos computacionais gastos em cada interacção foram maiores para a maior parte dos problemas, com excepção dos problemas de *knapsack 0-1 multidimensional* em que a abordagem combinada mostrou um desempenho superior ao do método de *branch-and-bound*. Este facto deve-se ao uso das desigualdades de cobertura mínima estendidas (dentro da abordagem de planos de corte), que permite acelerar a resolução dos problemas de *knapsack 0-1 multidimensional*. A tabela IV.4 ilustra este comportamento apresentando os tempos computacionais médios da abordagem combinada em comparação com os do método de *branch-and-bound* (já apresentados anteriormente na tabela IV.2).

Tipo de problema	*n*	*m*	Nº de Prob.	Tempos médios de uma interacção em pesquisas direccionais	
				Método de *branch-and-bound*	Abordagem combinada
Knapsack 0-1	100 B	1	6	**53"**	1' 03"
Knapsack inteiro	100 I (*)	1	6	**32"**	40"
Knapsack 0-1	50 B	10	6	55"	**44"**
multidimensional	60 B	10	5	2' 23"	**1' 52"**
Cobertura	250 B	50	8	**24"**	40"
Embalagem	250 B	50	6	**33"**	34"
Inteiros puros genéricos	30 I	15	6	**1' 52"**	2" 1'

B: binária; I: inteira; C: contínua (*) $0 \leq x_j \leq 10$, $j=1,...,n$

Tabela IV.4 – Tempos médios da abordagem combinada vs. método de branch-and-bound (PENTIUM II a 350 MHz).

IV.4. UM PROTOCOLO DE COMUNICAÇÃO COM O AGENTE DE DECISÃO

As abordagens multicritério desenvolvidas estão especialmente vocacionadas para *pesquisas direccionais*, utilizando direcções particulares de melhoramento de uma função objectivo, e para *pesquisas locais*, uma vez que a solução seguinte é uma solução próxima da anterior. O AD pode também indicar directamente outros pontos de referência/níveis de aspiração para obter um conhecimento de carácter mais global sobre o problema.

Podemos questionar, porém, se não conhecendo nada acerca do seu problema, o AD se sentirá confortável a dar este tipo de informação numa fase inicial do processo de decisão. O AD pode sentir a necessidade de uma pesquisa inicial estratégica que lhe dê um conhecimento global sobre o problema e lhe permita balizar as suas escolhas, definindo pontos de "ancoragem" para a pesquisa que se irá seguir. Em nosso entender, as abordagens anteriores apresentam dois principais pontos fracos. Em primeiro lugar, é difícil efectuar uma pesquisa estratégica inicial. Em segundo lugar, o AD não tem controlo nas outras funções objectivo quando faz uma pesquisa direccional de melhoramento de uma função objectivo.

Um conhecimento global inicial sobre o problema pode ser conseguido através do cálculo das soluções não dominadas que optimizam individualmente cada uma das funções objectivo (que compõem a tabela de *pay-off* e definem o ponto ideal), e/ou de soluções não dominadas dispersas obtidas a partir de somas pesadas das funções objectivo. Apesar de a abordagem das somas pesadas se cingir ao cálculo de soluções suportadas, o facto de o conjunto dos pesos

$W=\left\{w\in\Re^k \,\middle|\, w_i>0,\ \sum_{i=1}^{k} w_i=1\right\}$ ser limitado permite seleccionar vectores de pesos bem distribuídos para calcular um conjunto inicial de soluções com características diferentes.

Durante uma pesquisa direccional, para além da indicação da função objectivo que pretende melhorar nesse instante, o AD pode também desejar controlar a variação de outras funções objectivo, nomeadamente não permitindo que baixem de determinados patamares. Para tal, o método deve permitir que o AD imponha limitações inferiores (em geral temporárias) nos valores das funções objectivo. São restrições adicionais que, juntamente com a informação preferencial anterior, garantem que as soluções não dominadas obtidas durante a pesquisa direccional são progressivamente melhores no objectivo escolhido, sem que nunca sejam inferiores a determinados níveis nos outros critérios.

Propomos, então, um conjunto de ferramentas que o AD pode dispor em qualquer altura do processo de decisão, e que lhe possibilitam uma aprendizagem progressiva do conjunto das soluções não dominadas. Algumas delas, como as somas pesadas das funções objectivo, poderão ser mais úteis numa fase inicial do processo de decisão, em que o AD pretende obter um conhecimento global sobre o problema. A ferramenta de pesquisa direccional, complementada com a introdução de limitações nas funções objectivo, poderá ser usada para fazer um 'varrimento' de soluções numa dada direcção, ou utilizada numa fase final do processo de decisão em que o AD pretende uma pesquisa mais focada (local) para conhecer a 'vizinhança' de uma solução não dominada considerada interessante.

Estas ferramentas foram incorporadas no sistema computacional que desenvolvemos para problemas de PLIMO e de PLIMMO. No que diz respeito à introdução de limitações nos valores das funções objectivo, no caso da abordagem de planos de corte estas restrições são introduzidas no quadro simplex actual prosseguindo-se depois normalmente. No caso da abordagem de *branch-and-bound*, se forem impostas ou retiradas limitações nos objectivos durante uma pesquisa direccional, então refaz-se desde a raiz toda a árvore de *branch-and-bound*, prosseguindo-se depois a partir da nova árvore para o cálculo das soluções seguintes.

O fluxograma da figura IV.25 esquematiza o protocolo geral de interacção com o AD.

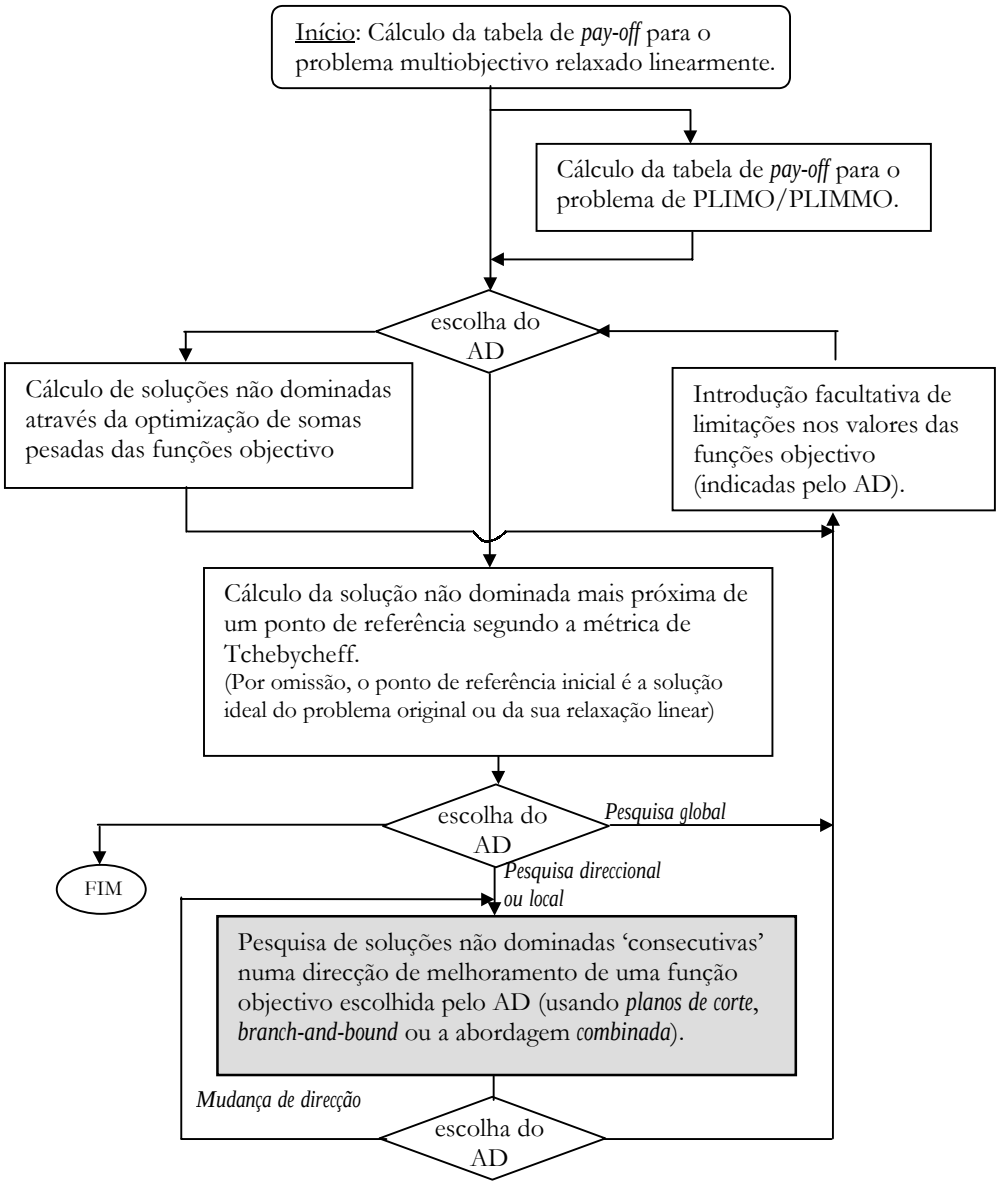


Fig. IV.25 – Esquema do protocolo geral de interacção com o AD.

IV. 5 CONCLUSÕES

Neste capítulo apresentámos um novo método interactivo para problemas de PLIMO e PLIMMO que tira partido da técnica de *branch-and-bound* para resolver sucessivos programas escalarizantes definidos numa pesquisa direccional de soluções eficientes. Este trabalho sucedeu à nossa anterior investigação no contexto das técnicas de planos de corte. Como seria de esperar, o método de *branch-and-bound* é mais eficaz porque é mais robusto e permite tratar uma maior gama de problemas.

A principal potencialidade deste método consiste em usar a anterior árvore de *branch-and-bound* para fazer análise de sensibilidade e prosseguir na busca de outras soluções eficientes. Foram desenvolvidas regras para simplificar a árvore antes de novas ramificações, para que esta

não esteja em constante crescimento, permitindo assim poupar espaço de armazenamento da informação e tempo no seu manuseamento.

As experiências computacionais mostraram que esta abordagem é eficaz em pesquisas direccionais ou locais. Na maior parte dos problemas testados os tempos foram reduzidos relativamente aos tempos necessários para a resolução independente dos programas escalarizantes. Este facto é particularmente relevante em problemas inteiros-mistos. Salientamos o facto de o ponto de referência ser ajustado automaticamente, o que assegura que seja encontrada uma solução eficiente diferente da anterior, e que melhora a função objectivo escolhida pelo AD. Consequentemente, o AD fica desobrigado dessa tarefa, evitando a selecção de pontos de referência que poderiam conduzir à mesma solução. No que diz respeito ao tamanho da árvore de *branch-and-bound* resultante do processo de simplificação/ramificação (medido em número de nodos), verificámos que é comparável ao da árvore que se obtém pela resolução independente do respectivo programa escalarizante, sendo por vezes maior e outras vezes menor.

Experimentámos também uma abordagem que combina o método de planos de corte, apresentado no capítulo anterior, com o de *branch-and-bound*. Começando por aplicar planos de corte de Gomory e desigualdades de cobertura mínima estendidas (para variáveis 0-1) ao programa escalarizante, o procedimento avança automaticamente para *branch-and-bound* no caso de não ter conseguido alcançar uma solução inteira num número predefinido de iterações. Apenas nos problemas de *knapsack* 0-1 multidimensional esta abordagem se revelou superior ao método que usa apenas *branch-and-bound*. De entre o conjunto restrito de problemas testados, é neste tipo de problemas que as desigualdades de cobertura mínima estendidas produzem melhores efeitos. Este facto sugere a importância destas desigualdades na redução de um problema, reforçando a ideia de que a incorporação de outras desigualdades válidas fortes permitiria melhorar o desempenho das abordagens propostas, quer a que usa apenas planos de corte quer a abordagem combinada.

Nestas abordagens concentrámo-nos fundamentalmente na forma de pesquisa direccional – em que o AD apenas especifica o objectivo que pretende melhorar em cada instante – e verificámos a sua eficácia do ponto de vista técnico. Julgamos, contudo, que do ponto de vista interactivo há outras questões relevantes, como a importância de um conhecimento inicial de carácter mais global sobre o problema e a incorporação de outra informação sobre as preferências do AD nas pesquisas direccionais, nomeadamente a introdução de limitações nos valores das funções objectivo. Não sendo este o objecto central da nossa investigação, apresentámos apenas uma proposta de um protocolo interactivo mais flexível que, na nossa opinião, permite um melhor apoio à decisão.

ANEXO IV.A

Regras usadas na geração dos problemas aleatórios

Os problemas de *knapsack*, cobertura e embalagem seguem os modelos apresentados no anexo III.B. Na geração dos respectivos coeficientes, utilizámos algumas regras diferentes das que foram descritas no anexo III.B por forma a testar uma maior diversidade (e eventual dificuldade) de problemas. Esta questão colocou-se após termos testado os problemas gerados segundo as regras anteriores e verificado que as soluções eficientes eram calculadas muito rapidamente o que dificultava a comparação das diferentes estratégias.

Nos problemas seguintes, as constantes são inteiras e foram geradas aleatoriamente num intervalo, ou em mais do que um intervalo com diferentes probabilidades (p), considerando distribuições uniformes.

PROBLEMAS DE MOCHILA (*KNAPSACK*) SIMPLES

$0 \le c_{ij} \le 100$, $\quad 1 \le a_j \le 100$, $\quad 0.1 \sum_{j=1}^{n} a_j \le b \le 0.75 \sum_{j=1}^{n} a_j$.

Nos problemas com variáveis inteiras considerámos $b=0.75 \sum_{j=1}^{n} a_j$ e $0 \le x_j \le 10$, $j=1,...,n$.

PROBLEMAS DE MOCHILA (*KNAPSACK*) 0-1 MULTIDIMENSIONAL

$0 \le c_{ij} \le 100$, $\quad \begin{cases} 1 \le a_{ij} \le 100 & p = 0.8 \\ a_{ij} = 0 & p = 0.2 \end{cases}$, $\quad 101 \le b_i \le 0.7 \sum_{j=1}^{n} a_{ij}$.

PROBLEMAS DE COBERTURA

$0 \le c_{ij} \le 100$, $\quad a_{ij} = \begin{cases} 1 & p = 0.4 \\ 0 & p = 0.6 \end{cases}$ nos problemas com 100 variáveis, $a_{ij} = \begin{cases} 1 & p = 0.2 \\ 0 & p = 0.8 \end{cases}$ nos problemas com 200 variáveis e $a_{ij} = \begin{cases} 1 & p = 0.15 \\ 0 & p = 0.85 \end{cases}$ nos problemas com 250 variáveis.

PROBLEMAS DE EMBALAGEM (*PACKING*)

$0 \le c_{ij} \le 30$, $\quad a_{ij} = \begin{cases} 1 & p = 0.25 \\ 0 & p = 0.75 \end{cases}$ nos problemas com 200 variáveis e $a_{ij} = \begin{cases} 1 & p = 0.2 \\ 0 & p = 0.8 \end{cases}$ nos problemas com 250 variáveis (nos problemas com 100 variáveis não houve uma regra geral seguida para todos).

PROBLEMAS GENÉRICOS INTEIROS E INTEIROS-MISTOS

As funções objectivo são a maximizar e as restrições do tipo '≤';

$$\begin{cases} -100 \le c_{ij} \le -1 & p = 0.2 \\ 0 \le c_{ij} \le 100 & p = 0.8 \end{cases},$$

$$\begin{cases} -100 \le a_{ij} \le -1 & p = 0.1 \\ a_{ij} = 0 & p = 0.1 \\ 1 \le a_{ij} \le 100 & p = 0.8 \end{cases}, \qquad 101 \le b_i \le \sum_{j=1}^{n} a_{ij} .$$

ANEXO IV.B

Resultados de pesquisas direccionais

Este anexo apresenta resultados de pesquisas direccionais efectuadas com o método multiobjectivo interactivo baseado em *branch-and-bound* descrito na secção IV.2.

As colunas *tempo* e '*Re-início*' mostram valores de tempo computacional em segundos (num computador PC PENTIUM II a 350MHz).

Problemas de *Knapsack* simples (funções objectivo a maximizar)

(a) Com variáveis 0-1

Problema	*Estratégia*	*Pto de Referência*	*Solução (z_1,z_2...)*	*It.*	*tempo*	*'Re-início'*
KS1_2_100	Início	(3743, 4070)	(3467, 3791)	–	68.9	
	melhorar z_1	(3749, 4070)	(3471, 3788)	2	49.3	90.2
	melhorar z_1	(3757, 4070)	(3473, 3785)	3	41.8	60.8
KS1_3_100	Início	(4084, 3816, 3966)	(3716, 3448, 3591)	–	98.3	
	melhorar z_2	(4084, 3830, 3966)	(3713, 3455, 3585)	4	251.9	207.0
	melhorar z_2	(4084, 3840, 3966)	(3704, 3458, 3582)	3	122.7	207.2
KS2_2_100	Início	(3257, 3348)	(3078, 3169)	–	48.8	
	melhorar z_2	(3257, 3349)	(3077, 3179)	1	6.7	56.7
	melhorar z_2	(3257, 3362)	(3074, 3193)	2	19.6	25.0
KS2_3_100	Início	(3147, 3022, 3305)	(2836, 2702, 2988)	–	92.1	
	melhorar z_3	(3147, 3022, 3311)	(2830, 2700, 2977)	2	27.1	85.6
	melhorar z_3	(3147, 3022, 3324)	(2821, 2701, 3000)	3	53.5	74.8
KS3_2_100	Início	(3023, 2697)	(2856, 2522)	–	23.0	
	melhorar z_1	(3034, 2697)	(2868, 2519)	2	8.7	11.3
	melhorar z_1	(3053, 2697)	(2874, 2513)	4	11.8	8.1
KS3_3_100	Início	(2454, 2355, 2709)	(2211, 2132, 2474)	–	18.0	
	melhorar z_1	(2459, 2355, 2709)	(2224, 2108, 2473)	3	22.8	27.0
	melhorar z_1	(2478, 2355, 2709)	(2239, 2102, 2461)	4	16.1	18.4
KS1_2_200	Início	(3727, 4049)	(3447, 3768)	–	162.0	
	melhorar z_1	(3729, 4049)	(2451, 3767)	1	31.3	180.4
	melhorar z_1	(3738, 4049)	(3454, 3763)	3	109.7	65.0
KS1_3_200	Início	(3708, 3972, 3935)	(3351, 3616, 3581)	–	468.1	
	melhorar z_3	(3708, 3972, 3939)	(3353, 3614, 3584)	1	144.4	390.8
	melhorar z_3	(3708, 3972, 3943)	(3350, 3614, 3586)	1	64.8	146.0
KS3_2_200	Início	(7868, 7080)	(7487, 6705)	–	803.8	
	melhorar z_2	(7868, 7088)	(7485, 6707)	2	195.7	123.1
	melhorar z_2	(7868, 7091)	(7484, 6709)	1	41.7	125.6
KS3_3_200	Início	(7564, 7314, 6793)	(6785, 6534, 6014)	–	32.5	
	melhorar z_2	(7564, 7321, 6793)	(6783, 6536, 6007)	4	653.7	378.0
	melhorar z_2	(7564, 7323, 6793)	(6783, 6537, 6006)	1	91.6	485.4
	melhorar z_2	(7564, 7327, 6793)	(6775, 6551, 6004)	2	447.6	788.3
KS4_2_200	Início	(8075, 7614)	(7590, 7124)	–	55.2	
	melhorar z_2	(8075, 7618)	(7582, 7131)	2	117.9	129.6
	melhorar z_2	(8075, 7628)	(7579, 7132)	3	34.4	46.8
KS4_3_200	Início	(7586, 7959, 7537)	(6896, 7273,6845)	–	136.3	
	melhorar z_3	(7586, 7959, 7540)	(6892, 7272, 6851)	2	126.3	214.1
	melhorar z_3	(7586, 7959, 7547)	(6891, 7266, 6855)	2	40.3	103.0

KS*i*_*k*_*n*: *i* é o índice do problema, *k* é o nº de objectivos e *n* o nº de variáveis 0-1

(b) Com variáveis inteiras

Problema	*Estratégia*	*Pto de Referência*	*Solução (z_1,z_2...)*	*It.*	*tempo*	*'Re-início'*
KI1_2_100	Início	(19789, 16047)	(18938, 15211)	–	52.5	
	melhorar z_2	(19789, 16071)	(18930, 15212)	6	28.2	22.0
	melhorar z_2	(19789, 16078)	(18924, 15215)	4	39.3	45.3
KI1_3_100	Início	(17775, 18121, 17328)	(16475, 16798, 16000)	–	5.5	
	melhorar z_1	(17808, 18121, 17328)	(16488, 16788, 16011)	3	6.9	6.3
	melhorar z_1	(17826, 18121, 17328)	(16503, 16784, 15993)	3	7.9	6.9
KI2_2_100	Início	(13364, 12180)	(12358, 11190)	–	22.4	
	melhorar z_2	(13364, 12204)	(12351, 11193)	9	57.9	43
	melhorar z_2	(13364, 12213)	(12345, 11196)	4	51.4	72.8
KI2_3_100	Início	(14349,12563,13062)	(12775, 11001, 11487)	–	18.4	
	melhorar z_2	(14349,12586, 13062)	(12777, 11020, 11477)	6	92.3	81.2
	melhorar z_2	(14349,12609, 13062)	(12767, 11021, 11474)	3	34.2	50.5
KI3_2_100	Início	(48022, 55118)	(42430, 49529)	–	4.0	
	melhorar z_1	(48031, 55118)	(42451, 49518)	4	3.8	7.2
	melhorar z_1	(48090, 55118)	(42462, 49480)	17	18.5	8.3
KI3_3_100	Início	(45241,34565,41657)	(41490, 30807, 37919)	–	6.0	
	melhorar z_2	(45241,34577,41657)	(41472, 30810, 37912)	8	37.4	27.2
	melhorar z_2	(45241,34582,41657)	(41470, 30811, 37906)	2	6.1	21.7
KI1_2_200	Início	(27715, 34417)	(25047, 31750)	–	68.7	
	melhorar z_1	(27717, 34417)	(25051, 31747)	1	70.2	120.3
	melhorar z_1	(27727, 34417)	(25052, 31742)	4	83.2	88.7
KI1_3_200	Início	(30496,30124, 31786)	(27684, 27304, 28966)	–	25.1	
	melhorar z_1	(30514,30124, 31786)	(27685, 27302, 28957)	6	146.9	153.0
	melhorar z_1	(30515,30124, 31786)	(27692, 27298, 28956)	1	43.7	161.4
KI2_2_200	Início	(30127, 31614)	(27795, 29273)	–	58.4	
	melhorar z_1	(30149, 31614)	(27796, 29261)	7	92.7	38.2
	melhorar z_1	(30154, 31614)	(27801, 29256)	3	193.3	141.2
KI2_3_200	Início	(29083,31358, 28750)	(25628, 27904, 25300)	–	144.84	
	melhorar z_1	(29085,31358, 28750)	(25629, 27912, 25294)	1	46.2	167.8
	melhorar z_1	(29086,31358, 28750)	(25633, 27911, 25293)	1	132.4	261.6
KI3_2_200	Início	(31481, 33343)	(29945, 31808)	–	2.4	
	melhorar z_2	(31481, 33354)	(29939, 31809)	5	24.5	13.9
	melhorar z_2	(31481, 33358)	(29933, 31810)	2	21.0	24.0
KI3_3_200	Início	(29110,29184, 31894)	(26371, 26446, 29156)	–	178.1	
	melhorar z_1	(29112,29184, 31894)	(26379, 26444, 29166)	1	30.7	180.7
	melhorar z_1	(29125,29184, 31894)	(26381, 26439, 29150)	4	154.3	170.0

KI*i*_*k*_*n*: *i* é o índice do problema, *k* é o nº de objectivos e *n* é o nº de variáveis inteiras

Problemas de *Knapsack* 0-1 multidimensional (funções objectivo a maximizar)

Problema	*Estratégia*	*Pto de Referência*	*Solução (z_1,z_2...)*	*It.*	*tempo*	*'Re-início'*
KM1_2_50_10	Início	(650, 633)	(563, 540)	–	10.2	
	melhorar z_2	(650, 670)	(521, 556)	19	42.0	36.8
	melhorar z_2	(650, 688)	(519, 595)	2	9.3	21.4
KM1_3_50_10	Início	(622, 725, 565)	(520, 618, 470)	–	25.2	
	melhorar z_2	(622, 738, 565)	(527, 619, 446)	7	34.4	65.6
	melhorar z_2	(622, 747, 565)	(535, 638, 437)	5	36.0	65.4
KM2_2_50_10	Início	(947, 1041)	(811, 904)	–	11.2	
	melhorar z_1	(957, 1041)	(818, 896)	5	15.4	18.8
	melhorar z_1	(977, 1041)	(826, 883)	8	30.4	34.6
KM2_3_50_10	Início	(1230, 1103, 959)	(1044, 910, 824)	–	18.0	
	melhorar z_2	(1230, 1109, 959)	(1059, 914, 761)	3	14.1	24.3
	melhorar z_2	(1230, 1134, 959)	(1011, 929, 748)	12	100.3	79.0

KM3_2_50_10	Início	(994, 1060)	(823, 889)	–	16.2	
	melhorar z_1	(1012, 1060)	(827, 872)	10	33.9	32.4
	melhorar z_1	(1053, 1060)	(837, 835)	20	125.6	94.3
KM3_3_50_10	Início	(985, 898, 824)	(851, 747, 664)	–	28.6	
	melhorar z_3	(985, 898, 844)	(806, 731, 695)	11	115.5	114.2
	melhorar z_3	(985, 898, 887)	(804, 707, 698)	8	104.6	58.0
KM1_2_60_10	Início	(2084, 2285)	(1846, 2054)	–	271.3	
	melhorar z_1	(2092, 2285)	(1858, 2040)	4	265.6	392.7
	melhorar z_1	(2115, 2285)	(1864, 2029)	7	309.4	347.7
KM2_2_60_10	Início	(997, 926)	(913, 783)	–	139.1	
	melhorar z_2	(997, 932)	(849, 802)	3	37.5	144.8
	melhorar z_2	(997, 964)	(836, 806)	8	113.8	88.4
	Início	(860, 891)	(791, 824)	–	12.03	
	melhorar z_1	(862, 891)	(796, 821)	1	155.7	182.1
	melhorar z_1	(879,891)	(798, 809)	7	262.8	277.4
KM2_3_60_10	Início	(1111, 947, 986)	(928, 758, 813)	–	70.1	
	melhorar z_3	(1111, 947, 1013)	(912, 796, 818)	6	67.5	84.3
	melhorar z_3	(1111, 947, 1019)	(911, 770, 841)	2	14.4	76.3
KM3_2_60_10	Início	(860, 891)	(690, 739)	–	14.2	
	melhorar z_1	(893, 891)	(699, 689)	17	70.5	67.4
	melhorar z_1	(912, 891)	(708, 679)	6	27.3	61.9
KM3_3_60_10	Início	(746, 803, 857)	(553, 619, 663)	–	154.8	
	melhorar z_3	(746, 803, 862)	(547, 645, 666)	3	128.6	327.7
	melhorar z_3	(746, 803, 868)	(544, 635, 694)	2	105.3	345.8
	melhorar z_3	(746, 803, 903)	(558, 595, 711)	4	299.0	351.3

KM*i*_*k*_*n*_*m*: *i* é o índice do problema, *k* é o nº de objectivos, *n* é o nº de variáveis 0-1 e *m* é o nº de restrições

Nota: Omitimos aqui os resultados dos problemas de cobertura e embalagem de menor dimensão (100 e 200 variáveis), apresentando apenas os de 250 variáveis, uma vez que o comportamento dos primeiros (em termos de tempo) é muito semelhante para todos. Além disso, a comparação dos correspondentes tempos computacionais tem pouco significado físico porque estes são bastante baixos.

Problemas de cobertura (funções objectivo a minimizar)

Problema	***Estratégia***	***P^to de Referência***	***Solução (z_1,z_2...)***	***It.***	***tempo***	***'Re-início'***
C1_2_250_50	Início	(57, 45)	(135, 124)	–	3.8	
	melhorar z_1	(51, 45)	(133, 128)	4	9.3	9.6
	melhorar z_1	(47, 45)	(131, 130)	2	4.4	6.1
C1_3_250_50	Início	(29, 56, 36)	(191, 219, 193)	–	31.0	
	melhorar z_2	(29, 54, 36)	(167, 197, 200)	1	3.5	31.2
	melhorar z_2	(29, 21, 36)	(204, 195, 210)	3	18.1	21.7
C2_2_250_50	Início	(17, 37)	(96, 117)	–	4.0	
	melhorar z_2	(17, 35)	(99, 109)	1	1.3	6.4
	melhorar z_2	(17, 9)	(116, 104)	8	26.9	16.3
	melhorar z_2	(17, 4)	(117, 102)	1	3.2	14.3
C2_3_250_50	Início	(57, 37, 46)	(198, 177, 183)	–	22.1	
	melhorar z_2	(57, 13, 46)	(220, 156, 209)	12	108.3	85.5
	melhorar z_2	(57, -14, 46)	(223, 152, 215)	3	29.5	23.1
C3_2_250_50	Início	(0, 4)	(22, 20)	–	3.0	
	melhorar z_1	(-7, 4)	(18, 32)	2	1.5	2.3
	melhorar z_1	(-15, 4)	(14, 36)	2	0.9	1.9
	melhorar z_1	(-26, 4)	(12, 43)	2	1.6	1.8

C3_3_250_50	Início	(4, 8, 1)	(49, 57, 25)	-	29.5	
	melhorar z_2	(4, 2, 1)	(58, 39, 52)	3	32.5	49.5
	melhorar z_2	(4, -19, 1)	(53, 38, 58)	2	18.2	17.5
	melhorar z_2	(4, -26, 1)	(68, 34, 29)	4	18.5	33.8
C4_2_250_50	Início	(47, 52)	(125, 132)	–	5.0	
	melhorar z_2	(47, 42)	(136, 127)	5	14.0	22.9
	melhorar z_2	(47, 37)	(137, 89)	1	2.8	16.1
C4_3_250_50	Início	(45, 75, 31)	(191, 221, 178)	–	65.3	
	melhorar z_2	(45, 65, 31)	(199, 219, 181)	6	87.3	101.2
	melhorar z_2	(45, 62, 31)	(197, 214, 188)	2	31.0	132.4
	melhorar z_2	(45, 51, 31)	(207, 194, 176)	3	65.0	150.3

C*i_k_n_m*: *i* é o índice do problema, *k* é o nº de objectivos, *n* é o nº de variáveis 0-1 e *m* é o nº de restrições

Problemas de embalagem (*packing*) (funções objectivo a maximizar)

Problema	***Estratégia***	***Pto de Referência***	***Solução (z_1,z_2...)***	***It.***	***tempo***	***'Re-início'***
P1_2_250_50	Início	(559, 483)	(470, 323)	–	30.1	
	melhorar z_2	(559, 490)	(394, 352)	2	5.8	34.3
	melhorar z_2	(559, 597)	(315, 392)	19	72.8	50.6
P1_3_250_50	Início	(588, 519, 511)	(395, 349, 355)	–	30.2	
	melhorar z_3	(588, 519, 592)	(370, 283, 409)	14	56.7	43.1
	melhorar z_3	(588, 519, 822)	igual solução: óptimo z_3	–		
	melhorar z_1	(838, 519, 822)	(395, 349, 355)	8	37.3	41.4
	melhorar z_1	(891, 519, 822)	(405, 272, 327)	8	33.5	46.5
P2_2_250_50	Início	(876, 834)	(771, 729)	–	8.9	
	melhorar z_2	(876, 887)	(719, 739)	14	32.2	22.0
	melhorar z_2	(876, 909)	(707, 765)	3	10.2	22.9
	melhorar z_2	(876, 947)	(696, 766)	3	10.1	10.1
P2_3_250_50	Início	(726, 812, 813)	(536, 625, 702)	–	39.7	
	melhorar z_1	(761, 812, 813)	(542, 657, 589)	16	95.5	97.6
	melhorar z_1	(787, 812, 813)	(604, 567, 657)	11	95.3	121.0
P3_2_250_50	Início	(784, 836)	(638, 697)	–	30.2	
	melhorar z_1	(789, 836)	(647, 686)	2	6.3	29.2
	melhorar z_1	(800, 836)	(662, 684)	2	8.1	34.8
	melhorar z_1	(824, 836)	(663, 675)	3	13.0	24.7
P3_3_250_50	Início	(847, 746, 710)	(750, 690, 581)	–	13.6	
	melhorar z_3	(847, 746, 715)	(731, 613, 606)	2	6.3	17.0
	melhorar z_3	(847, 746, 742)	(720, 611, 623)	2	9.9	13.0

P*i_k_n_m*: *i* é o índice do problema, *k* é o nº de objectivos, *n* é o nº de variáveis 0-1 e *m* é o nº de restrições

Problemas genéricos inteiros puros (funções objectivo a maximizar)

Problema	***Estratégia***	***Pto de Referência***	***Solução (z_1,z_2...)***	***It.***	***tempo***	***'Re-início'***
I1_2_15_15	Início	(704, 789)	(500, 626)	–	11.6	
	melhorar z_1	(734, 789)	(508, 556)	14	14.4	23.7
	melhorar z_1	(761, 789)	(523, 537)	11	10.7	23.0
I1_3_15_15	Início	(649, 804, 667)	(471, 620, 512)	–	6.1	
	melhorar z_2	(649, 827, 667)	(520, 635, 460)	11	8.3	14.3
	melhorar z_2	(649, 847, 667)	(437, 711, 481)	3	3.4	10.8
I1_4_15_15	Início	(562, 716, 694, 619)	(342, 432, 416, 331)	–	10.8	
	melhorar z_1	(663, 716, 694, 619)	(418, 395, 426, 331)	17	38.9	26.7
	melhorar z_1	(749, 716, 694, 619)	(443, 386, 384, 332)	6	13.5	14.2

I1_2_30_15	Início	(888, 826)	(559, 549)	–	17.1	
	melhorar z_1	(892, 826)	(581, 494)	2	1.2	17.1
	melhorar z_1	(1000, 826)	(639, 408)	42	51.3	34.6
I1_3_30_15	Início	(817, 655, 957)	(606, 485, 772)	–	25.1	
	melhorar z_1	(825, 655, 957)	(647, 514, 738)	4	10.4	32.6
	melhorar z_1	(936, 655, 957)	(661, 367, 693)	36	70.6	37.3
I2_2_30_15	Início	(1285, 1166)	(1123, 1026)	–	13.3	
	melhorar z_2	(1285, 1247)	(1065, 1035)	27	41.9	22.9
	melhorar z_2	(1285, 1260)	(1060, 1043)	3	5.3	18.5
I2_3_30_15	Início	(1473, 1231, 1179)	(1209, 897, 856)	–	16.2	
	melhorar z_2	(1473, 1245, 1179)	(1152, 909, 832)	7	17.8	25.9
	melhorar z_2	(1473, 1273, 1179)	(1144, 968, 815)	9	28.4	35.2
I3_2_30_15	Início	(1772, 1738)	(1319, 1284)	–	727.5	
	melhorar z_1	(1782, 1738)	(1342, 1275)	5	273.2	886.9
	melhorar z_1	(1845, 1738)	(1343, 1236)	21	811.8	625.7
I3_3_30_15	Início	(1705, 1732, 2417)	(1219, 1299, 1966)	–	16.8	
	melhorar z_2	(1705, 1790, 2417)	(1215, 1308, 1974)	3	9.2	19.7
	melhorar z_2	(1705, 1810, 2417)	(1279, 1466, 1916)	7	20.9	29.6
I1_2_30_30	Início	(1621, 1502)	(1445, 1330)	–	81.6	
	melhorar z_2	(1621, 1534)	(1418, 1332)	15	199.4	196.3
	melhorar z_2	(1621, 1566)	(1387, 1378)	16	397.9	363.0
I1_3_30_30	Início	(1576, 1284, 1222)	(1119, 794, 748)	–	926.1	
	melhorar z_2	(1576, 1286, 1222)	(1085, 808, 1222)	1	64.8	961.1
	melhorar z_2	(1576, 1304, 1222)	(1081, 826, 807)	3	432.6	756.3
I2_2_30_30	Início	(840, 805)	(706, 667)	–	15.1	
	melhorar z_2	(840, 894)	(614, 681)	45	118.4	75.3
	melhorar z_2	(840, 967)	(555, 705)	31	135.1	111.6
I2_3_30_30	Início	(1556, 1622, 1436)	(1016, 1115, 904)	–	3154.0	
	melhorar z_3	(1556, 1622, 1453)	(1009, 1101, 914)	5	1185.0	2821.9
	melhorar z_3	(1556, 1622, 1466)	(1078, 1071, 921)	3	717.8	3023.4
I3_2_30_30	Início	(693, 660)	(418, 389)	–	28.9	
	melhorar z_2	(693, 718)	(365, 390)	28	75.3	65.2
	melhorar z_2	(693, 723)	(361, 407)	3	7.3	65.4
I3_3_30_30	Início	(566, 872, 666)	(174, 478, 310)	–	109.5	
	melhorar z_1	(575, 872, 666)	(281, 515, 266)	4	91.0	135.2
	melhorar z_1	(684, 872, 666)	(322, 470, 270)	2	37.1	44.8

I*i*_*k*_*n*_*m*: *i* é o índice do problema, *k* o nº é de objectivos, *n* é o nº de variáveis inteiras e *m* é o nº de restrições

Problemas genéricos inteiros-mistos (funções objectivo a maximizar)

(Por razões de espaço das colunas, omitimos os valores das soluções)

Problema	***Estratégia***	***P^to de Referência***	***It.***	***tempo***	***'Re-início'***
IM1_2_80_40	Início	(1843, 2153)	–	4.7	
	melhorar z_1	(1843.9, 2153)	1	0.6	7.7
	melhorar z_1	(1845.1, 2153)	1	0.2	7.3
IM1_3_80_40	Início	(2044, 2194, 1644)	–	7.9	
	melhorar z_3	(2044, 2194, 1646.3)	1	0.3	7.1
	melhorar z_3	(2044, 2194, 1650.1)	1	0.6	6.7
IM2_2_80_40	Início	(1430, 792)	–	3.0	
	melhorar z_2	(1430, 792.7)	1	0.4	3.4
	melhorar z_2	(1430, 794.5)	1	0.3	3.5

IM2_3_80_40	Início	(1003, 1229, 1235)	–	8.9	
	melhorar z_1	(1004.0, 1229, 1235)	1	0.5	9.4
	melhorar z_1	(1004.8, 1229, 1235)	1	0.3	9.5
IM3_2_80_40	Início	(3089, 3387)	–	5.7	
	melhorar z_1	(3090.8, 3387)	1	0.7	6.5
	melhorar z_1	(3092.4, 3387)	1	0.4	6.6
IM3_3_80_40	Início	(3494, 2430, 2812)	–	1.4	
	melhorar z_2	(3494, 2437.6, 2812)	1	0.6	0.9
	melhorar z_2	(3494, 2440.5, 2812)	1	1.2	1.2
IM1_2_100_15	Início	(3938, 4125)	–	12.4	
	melhorar z_1	(3941.1, 4125)	1	0.6	13.7
	melhorar z_1	(3943.8, 4125)	1	0.8	13.2
IM1_3_100_15	Início	(4724, 3929, 4387)	–	26.5	
	melhorar z_2	(4724, 3931, 4387)	1	2.5	28.6
	melhorar z_2	(4724, 3933.1, 4387)	1	1.0	28.7
IM2_2_100_15	Início	(4842, 4520)	–	7.4	
	melhorar z_2	(4842, 4524.5)	1	0.5	6.4
	melhorar z_2	(4842, 4527.6)	1	0.3	6.3
IM2_3_100_15	Início	(3826, 4754, 5153)	–	37.0	
	melhorar z_1	(3828.4, 4754, 5153)	1	3.3	38.5
	melhorar z_1	(3830.3, 4754, 5153)	1	2.8	36.3
IM3_2_100_15	Início	(6158, 5962)	–	22.4	
	melhorar z_2	(6158, 5965.1)	1	3.4	25.3
	melhorar z_2	(6158, 5973.3)	1	4.6	14.0
IM3_3_100_15	Início	(5832, 6184, 5668)	–	44.7	
	melhorar z_3	(5832, 6184, 5670.9)	1	6.6	50.3
	melhorar z_3	(5832, 6184, 5673.8)	1	6.9	53.0

IM*i*_*k*_*n*_*m*: *i* é o índice do problema, *k* é o nº de objectivos, *n* é o nº total de variáveis e *m* é o nº de restrições

Para *n*=80, há 20 variáveis 0-1, 20 inteiras e 40 contínuas.
Para *n*=100, há 35 variáveis 0-1, 35 inteiras e 30 contínuas

Capítulo V

Regiões de indiferença no espaço dos pontos de referência para problemas de PLIMO

Os métodos multiobjectivo descritos nos capítulos III e IV são abordagens interactivas em que cada solução é calculada pela projecção de um ponto de referência no conjunto das soluções não dominadas. Esta projecção é conseguida à custa da optimização de um programa escalarizante *min-max* baseado na métrica (aumentada) de Tchebycheff. Num problema de programação linear inteira multiobjectivo (PLIMO), o conjunto das soluções não dominadas é discreto, existindo múltiplos pontos de referência que conduzem à mesma solução. Esses pontos definem *conjuntos* ou *regiões de indiferença* no espaço dos pontos de referência (que é também o espaço dos objectivos) correspondente a cada solução não dominada do problema de PLIMO. Cada um desses conjuntos representa a região de estabilidade da respectiva solução face à variação do ponto de referência.

O conhecimento das regiões de indiferença pode ser útil para orientar o AD na pesquisa de novas soluções, principalmente se esta informação for integrada num gráfico de fácil interpretação. A representação gráfica das regiões de indiferença de soluções não dominadas já calculadas dá uma indicação da distribuição espacial dessas soluções e da estabilidade de cada uma face à variação do ponto de referência. Permite apoiar o AD na busca de novas soluções, porque ajuda-o a não escolher pontos de referência que conduzem a soluções já conhecidas. O estudo que fizemos sobre a definição de regiões de indiferença no espaço dos pontos de referência de problemas PLIMO – em particular problemas tri-objectivo, caso que permite uma

representação gráfica elucidativa – é o assunto abordado neste capítulo. A apresentação baseia-se em ALVES E CLÍMACO (1999B).

A *região de indiferença* no espaço dos pontos de referência correspondente a uma dada solução não dominada é, em geral, não convexa e desconhecemos um processo computacionalmente simples de a determinar por completo. Todavia, há direcções particulares de pontos de referência onde se verifica convexidade (i.e. os pontos de referência 'intermédios' conduzem à mesma solução). Essa propriedade será apresentada adiante na *proposição V.1*. Além disso, conhecendo-se um ponto de referência que conduz a uma dada solução, é possível definirmos, através de um processo simples, uma sub-região de indiferença convexa para essa solução (*proposição V.2*). Como consequência, se for conhecida uma trajectória de pontos de referência que conduz a uma solução não dominada – o que acontece durante as *pesquisas direccionais* apresentadas nos capítulos anteriores – então podemos definir uma sub-região de indiferença mais alargada. O processo de cálculo das sub-regiões de indiferença é o assunto que abordamos na secção 1 deste capítulo. Propomos uma construção iterativa de sub-regiões de indiferença que tira partido das *pesquisas direccionais* anteriores. Este processo de cálculo baseia-se nas *proposições V.1* e *V.2*.

Nas abordagens interactivas anteriores, existe uma fase de cálculo de cada vez que o AD indica um novo ponto de referência ou escolhe uma função objectivo que pretende melhorar relativamente à solução não dominada anterior, desenrolando-se, neste último caso, uma pesquisa direccional. O AD pode ainda introduzir limitações adicionais nos valores das funções objectivo (para ter um maior controlo na pesquisa direccional), que restringem temporariamente a região admissível do problema multiobjectivo e, consequentemente, o conjunto das soluções não dominadas. Um dado ponto de referência pode, então, conduzir a uma ou a outra solução diferente, conforme é utilizada uma pesquisa com ou sem limitações adicionais.

Podemos definir, de igual forma, regiões de indiferença para o problema restrito mas, atendendo ao carácter temporário das limitações adicionais, julgamos mais interessante reportar essa informação ao problema original. Estamos sobretudo interessados em obter uma informação coerente ao longo de todo o processo de pesquisa de soluções e, por conseguinte, a definição de regiões de indiferença deve referir-se sempre ao problema original. Para tal é necessário fazer uma correspondência entre os resultados produzidos por pontos de referência em pesquisas "restritas" e por pontos de referência em pesquisas "livres" (não restritas). Na secção 2 deste capítulo apresentamos um estudo sobre o assunto, que se baseia na *proposição V.4*. A *proposição V.4* estabelece uma função que transforma um ponto de referência do problema restrito noutro que conduz à mesma solução não dominada, se utilizado no problema original. A

partir desta função podemos transformar regiões de indiferença do problema restrito para o problema original.

A representação gráfica de regiões de indiferença do espaço dos pontos de referência constitui um meio integrado de apresentação de resultados. As regiões de indiferença podem ser representadas graficamente quando os problemas têm duas ou três funções objectivo (o gráfico é unidimensional em problemas bi-objectivo e é bidimensional em problemas tri-objectivo). Essa forma de representação gráfica foi incluída no sistema computacional desenvolvido, apresentando informação sobre as soluções produzidas tanto por pesquisas "livres", como por pesquisas "restritas" (i.e. quando são introduzidas limitações adicionais nos valores das funções objectivo). As regiões de indiferença representadas referem-se sempre ao problema original, pelo que são utilizados os resultados da secção 2 para fazer a transformação da informação quando esta provém de uma pesquisa "restrita". Atendendo a que a representação gráfica só é viável para problemas com duas ou três funções objectivo, sendo o caso tri-objectivo mais elucidativo do que o bi-objectivo, o estudo apresentado nas secções seguintes está orientado fundamentalmente para três funções objectivo.

Por fim, notemos que o processo de cálculo e representação de regiões de indiferença que abordamos neste capítulo poderia ser usado por outro método de pontos de referência para problemas de PLIMO – por exemplo pelos métodos de DURSO (1992) ou de KARAIVANOVA ET AL. (1995), entre outros. Muitos desses métodos consideram restrições temporárias nos valores das funções objectivo para o cálculo de soluções não dominadas, pelo que o estudo da secção 2 (sobre regiões de indiferença na presença de limitações adicionais nos objectivos) seria útil também para esses métodos.

V.1 DECOMPOSIÇÃO DO ESPAÇO DOS PONTOS DE REFERÊNCIA EM PLIMO

V.1.1 Conceitos gerais

Recordemos o programa escalarizante $P_{z^+}^{\infty,\rho}$, considerando z^+ um ponto de referência qualquer pertencente a $\Re^k$ (por não se impor qualquer condição a z^+, a variável α surge sem restrição de sinal):

$$\begin{aligned} \min \quad & \alpha - \rho \sum_{i=1}^{k} c^i x && (P_{z^+}^{\infty,\rho}) \\ \text{s.a:} \quad & z_i^+ - c^i x \le \alpha \quad i=1,...,k \\ & x \in X \end{aligned}$$

A solução eficiente x que optimiza $P_{z^+}^{\infty,\rho}$, para um dado ponto de referência z^+, também optimiza $P_{z^{++}}^{\infty,\rho}$ para z^{++} resultante da adição de uma constante a todas as componentes de z^+. Significa, então, que todos os pontos de referência do tipo $\left(z_1^+ + \delta, z_2^+ + \delta, ..., z_k^+ + \delta\right)$, com $\delta \in \Re$, conduzem à mesma solução eficiente, variando apenas o valor de α em $P_{z^+}^{\infty,\rho}$. Consequentemente, qualquer ponto de referência $z^+ \in \Re^k$ pode ser convertido para um outro pertencente a um hiperplano definido por $\sum_{i=1}^{k} z_i^+ = Q$ com $Q \in \Re$ constante. Designamos esse hiperplano por Z_Q^+. Dado $z^+ = \left(z_1^+, z_2^+, ..., z_k^+\right)$ e Q, z^+ é convertido em $z^{++} = \left(z_1^+ + \overline{\delta}, z_2^+ + \overline{\delta}, ..., z_k^+ + \overline{\delta}\right) \in Z_Q^+$, fazendo-se $\overline{\delta} = \left(Q - \sum_{i=1}^{k} z_i^+\right) \Big/ k$. Desta forma, qualquer ponto de referência de $\Re^k$ pode ser representado num hiperplano de dimensão $(k-1)$. Em particular para problemas tri-objectivo (bi-objectivo), o espaço dos pontos de referência e as suas *regiões de indiferença* – conjuntos de pontos de referência que conduzem à mesma solução – podem ser visualizados graficamente num plano (numa recta). Podemos ainda limitar cada componente z_i^+ (i=1,...,k) a um intervalo $[\, z_i^{+min}, z_i^{+max}\,]$ que permita caracterizar por completo o conjunto das soluções eficientes. Assim, se Q e z_i^{+min} (i=1,...,k) forem definidos apropriadamente, todas as regiões de indiferença do espaço dos pontos de referência têm representação em $\overline{Z}_Q^+ = \left\{ z^+ \in \Re^k \,\middle|\, z_i^+ \in \left[z_i^{+min}, z_i^{+max}\right] \forall i,\ \sum_{i=1}^{k} z_i^+ = Q \right\} \subset Z_Q^+$, onde $z_i^{+max} = Q - \sum_{j \neq i} z_j^{+min}$. Não é fácil, contudo, definir $\overline{Z}_Q^+$ *a priori*. Por um lado, os intervalos $[\, z_i^{+min}, z_i^{+max}\,]$ devem ser suficientemente abrangentes para caracterizarem por completo o conjunto eficiente e, por outro, não devem ser demasiado grandes, porque isso leva a que as regiões de indiferença "interiores" percam expressão face às da fronteira. Podemos usar um procedimento heurístico para atribuir valores iniciais a Q e z_i^{+min} (i=1,...,k) baseado, por exemplo, nas componentes da tabela de *pay-off*, e manter a possibilidade de ajustar posteriormente os valores de z_i^{+min} se se vier a manifestar necessário. Incluímos, no anexo V.A, uma nota sobre esta questão.

A figura V.1(a) mostra a forma de $\overline{Z}_Q^+$ para o caso tri-objectivo. Na decomposição deste triângulo equilátero, as regiões de indiferença das soluções não dominadas que optimizam individualmente cada função objectivo ocupam as áreas junto a cada um dos vértices do triângulo, o do eixo z_1^+, o do eixo z_2^+ e o do eixo z_3^+, respectivamente para z_1, z_2 e z_3. A

projecção do triângulo equilátero no plano 'z_1^+, z_2^+' é o triângulo rectângulo da figura V.1(b). Por questões de simplicidade na representação gráfica, adoptaremos no resto do texto a projecção em 'z_1^+, z_2^+'.

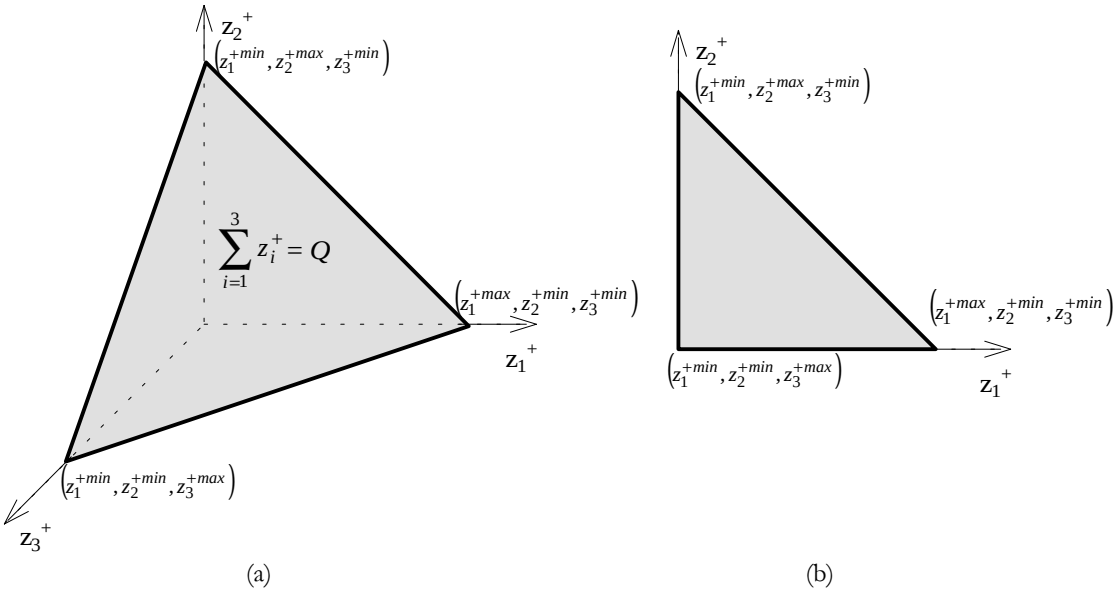


Fig. V.1 – $\overline{Z}_Q^+$ para problemas tri-objectivo e a sua projecção em 'z_1^+, z_2^+'.

Vejamos agora algumas propriedades relacionadas com o comportamento de determinados pontos de referência, quando introduzidos em $P_{z^+}^{\infty,\rho}$.

Proposição V.1: Se a solução eficiente x^a do problema PLIMO optimiza tanto $P_{z^{a+}}^{\infty,\rho}$ como $P_{z^{b+}}^{\infty,\rho}$ com $z^{a+} = \left(z_1^{a+},...,z_p^{a+},...,z_k^{a+}\right)$ e $z^{b+} = \left(z_1^{a+},...,z_p^{a+} + \breve{\theta}_p,...,z_k^{a+}\right)$, $\breve{\theta}_p > 0$, então x^a também optimiza $P_{z^+}^{\infty,\rho}$ para z^+ entre z^{a+} e z^{b+}.

A *proposição V.1* prova-se facilmente por considerações análogas à utilizadas em demonstrações anteriores (como nas proposições III.4 e III.5). Por essa razão omitimos aqui a prova, remetendo-a para o anexo V.B. Esta proposição apenas formaliza o resultado de uma *pesquisa direccional* tal como foi desenvolvida e apresentada nos capítulos III e IV: dado um ponto de referência $z^{a+}=(z_1^{a+},...,z_p^{a+},...,z_k^{a+})$ que conduz à solução eficiente x^a, se o AD pretender melhorar a função objectivo z_p, então incrementa-se a componente p do ponto de referência de uma quantidade $\hat{\theta}_p$ mantendo as outras todas iguais; $\hat{\theta}_p$ é o menor valor inteiro (apenas se consideram pontos de referência inteiros porque os problemas são inteiros puros) que permite abandonar a solução anterior, x^a, conduzindo a outra solução eficiente, x^b. Significa, pois, que

$\breve{\theta}_p = \hat{\theta}_p$ -1 ainda conduz a x^a, o mesmo acontecendo para valores positivos de θ_p menores do que este. Assim, a veracidade do resultado estabelecido na *proposição V.1* é desde logo garantida pela forma como as abordagens de pesquisa operam e calculam o valor de $\hat{\theta}_p$ (tanto com os planos de corte como com *branch-and-bound*). Podemos ainda acrescentar que qualquer ponto de referência não inteiro entre $(z_1^{a+},...,z_p^{a+}+\breve{\theta}_p,...,z_k^{a+})$ e $(z_1^{a+},...,z_p^{a+}+\hat{\theta}_p,...,z_k^{a+})$ conduz ou a x^a ou a x^b (proposição III.5).

No caso tri-objectivo, quando se incrementa de uma quantidade θ uma componente de $z^+=(z_1^+, z_2^+, z_3^+)\in Z_Q^+$, o ponto de referência resultante pode ser convertido para Z_Q^+ subtraindo $\theta/3$ a todas as componentes. Por exemplo, $(z_1^+ + \theta, z_2^+, z_3^+)$ seria convertido em $(z_1^+ + \tfrac{2}{3}\theta, z_2^+ - \tfrac{1}{3}\theta, z_3^+ - \tfrac{1}{3}\theta)$. A figura V.2 mostra as 3 direcções (m_1, m_2 e m_3) de incremento de cada uma das componentes do ponto de referência na projecção 'z_1^+, z_2^+' de $\overline{Z}_Q^+$.

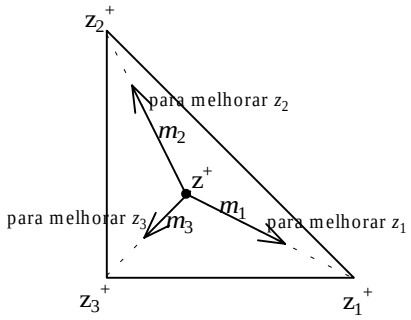


Fig. V.2 – As direcções de z^+ usadas nas pesquisas direccionais.

A *proposição V.1* evidencia que, se z^{a+} e z^{b+} pertencem à mesma região de indiferença e podem ser ligados por um segmento de recta de declive m_1, m_2 ou m_3, então todos os outros pontos do segmento de recta pertencem à mesma região de indiferença, ou seja, há convexidade segundo estas direcções. Como corolário da *proposição V.1* (*CASO 2.1* da prova em anexo) podemos estabelecer o seguinte resultado:

Proposição V.2: Seja x^a a solução eficiente do problema PLIMO que optimiza $P_{z^{a+}}^{\infty,\rho}$ para $z^{a+} = \left(z_1^{a+},...,z_p^{a+},...,z_k^{a+}\right)$, e s_i^a (i=1,...,k) o valor óptimo da variável desvio da restrição i de $P_{z^{a+}}^{\infty,\rho}$. Então, todos os pontos de referência $z^+ = \left(z_1^{a+}+\delta_1,...,z_i^{a+}+\delta_i,...,z_k^{a+}+\delta_k\right)$, $0<\delta_i \le s_i^a$, i=1,...,k, quando introduzidos em $P_{z^+}^{\infty,\rho}$, conduzem à mesma solução x^a.

Seja J $\subseteq \{1,...,k\}$ o conjunto dos índices das restrições activas que definem o valor óptimo de α em $P^{\infty,\rho}_{z^{a+}}$, ou seja, $\alpha^a = z_j^{a+} - c^j x^a$, $j \in$ J. Então, $s_j^a = 0$ para todo o $j \in$ J e $s_i^a > 0$ para todo o $i \in \{1,...,k\} \setminus$ J. A adição de $\delta_i \leq s_i^a$ a uma ou mais componentes i de z^{a+} não altera a distância de Tchebycheff (α) entre esse ponto de referência e o ponto dos critérios $z^a = Cx^a$. Por outras palavras, a solução (x^a, α^a), óptima de $P^{\infty,\rho}_{z^{a+}}$, é admissível para o problema $P^{\infty,\rho}_{z^{+}}$ cuja região admissível é restrita relativamente à de $P^{\infty,\rho}_{z^{a+}}$. Logo, (x^a, α^a) é também óptima para $P^{\infty,\rho}_{z^{+}}$. Podemos então afirmar que todos os pontos de referência $z^+ = \left(z_1^{a+} + \delta_1,..., z_i^{a+} + \delta_i,..., z_k^{a+} + \delta_k\right)$ com $0 \leq \delta_i \leq s_i^a$, i=1,...,k, pertencem à *região de indiferença* da solução x^a – ver exemplo V.1.

> **Exemplo V.1.** Consideremos que $z^{a+} = (z_1^{a+}, z_2^{a+}, z_3^{a+})$ conduz a x^a com $s_1^a > 0$, $s_2^a > 0$ e $s_3^a = 0$. Os pontos $P_0 = z^{a+}$, $P_1 = (z_1^{a+} + s_1^a, z_2^{a+}, z_3^{a+})$, $P_2 = (z_1^{a+}, z_2^{a+} + s_2^{a+}, z_3^{a+})$, $P_{1,2} = (z_1^{a+} + s_1^a, z_2^{a+} + s_2^a, z_3^{a+})$ e todos os pontos $(z_1^{a+} + \delta_1, z_2^{a+} + \delta_2, z_3^{a+})$ com $0 \leq \delta_i \leq s_i^a$, i=1,2, pertencem à região de indiferença do espaço dos pontos de referência correspondente a x^a. A figura V.3 mostra uma representação gráfica desses pontos convertidos para Z_Q^+ e projectados em 'z_1^+, z_2^+'.

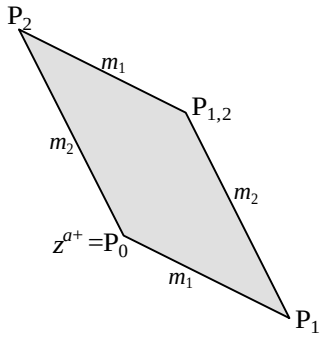


Fig. V.3 – Exemplo de uma sub-região de indiferença.

Em problemas tri-objectivo, cada região de indiferença de Z_Q^+ projectada em 'z_1^+, z_2^+' resulta da união de paralelogramos (como o da figura V.3) em que os lados têm inclinações m_1, m_2 ou m_3. Para cada z^{a+}, existe pelo menos um $s_j^a = 0$ na solução óptima de $P^{\infty,\rho}_{z^{a+}}$, uma vez que J nunca é vazio. Se apenas um s_j^a for nulo, a sub-região de indiferença é um paralelogramo, mas se dois s_j^a forem nulos, então o paralelogramo reduz-se a um segmento de recta e, se $s_j^a = 0$ para todo o j=1,..,3, então apenas se representa um ponto (correspondente a $P_{1,2}$ na figura V.3).

No caso geral, o ponto de referência $N^a = (z_1^{a+} + s_1^a, \ldots, z_k^{a+} + s_k^a)$ – correspondente a $P_{1,2}$ no exemplo V.1 e na figura V.3 – tem um papel central na região de indiferença de x^a. As suas características são as seguintes: (i) o valor óptimo de α no programa escalarizante $P_{z^+}^{\infty,\rho}$ com $z^+ = N^a$ é dado pela diferença entre qualquer componente de N^a e a correspondente componente do ponto dos critérios z^a (imagem de x^a); (ii) a conversão de N^a e de z^a para o plano Z_Q^+ resulta no mesmo ponto; (iii) qualquer sub-região de indiferença de x^a (definida a partir da proposição V.2) inclui N^a. Designamos estes pontos de referência por *NÚCLEOS* das regiões de indiferença.

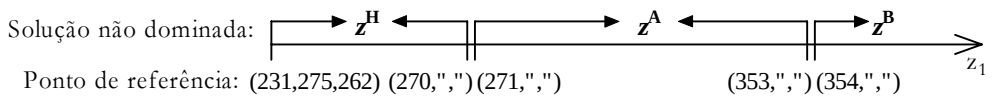
V.1.2 Análise de sub-regiões de indiferença em problemas tri-objectivo

Vimos como um dado ponto de referência z^{a+} pode ser usado para definir uma sub-região de indiferença de uma solução eficiente x^a. Se conhecermos vários pontos de referência (por exemplo, sobre uma determinada trajectória) que conduzem a x^a, então seremos capazes de definir várias sub-regiões de indiferença cuja união poderá definir uma sub-região maior.

Vejamos, então, como se podem traçar sucessivas sub-regiões de indiferença de um problema tri-objectivo aproveitando os resultados das *pesquisas direccionais*. Começaremos com um exemplo e, seguidamente, analisaremos o caso geral.

Consideremos um problema de mochila (*knapsack*) multidimensional com 20 variáveis binárias, 10 restrições e 3 funções objectivo a maximizar, que designamos por KNAPM20_3FO. Como todos os coeficientes do problema são inteiros, trata-se de um problema inteiro puro.

Procedemos a uma pesquisa direccional (de acordo com a descrição nos capítulos III e IV), começando por escolher o ponto de referência (231, 275, 262). Obtivemos a solução não dominada z^H=(117, 162, 170). Escolhemos então a 1ª função objectivo (z_1) para melhorar. O ponto de referência foi automaticamente alterado para (271, 275, 262), tendo como resultado a solução não dominada z^A=(173, 160, 109). O ponto de referência (271, 275, 262) é o primeiro ponto inteiro da trajectória (231+θ, 275, 262), $\theta > 0$, que é capaz de abandonar z^H. Consequentemente, (270, 275, 262) é o maior ponto inteiro desta trajectória que ainda conduz a z^H. Continuando a pesquisa na mesma direcção, obtivemos em seguida a solução não dominada z^B=(199, 185, 81) a partir do ponto de referência (354, 275, 262). O diagrama seguinte resume esta pesquisa:



A informação relevante para o cálculo das sub-regiões de indiferença é a seguinte: os pontos de referência (231+θ, 275, 262), 0≤ θ ≤39, conduzem a z^H e os pontos de referência (271+θ, 275, 262), 0≤ θ ≤82, conduzem a z^A.

Consideremos apenas a solução z^A para ilustrar o cálculo de sub-regiões de indiferença.

De acordo com a proposição V.2, cada ponto de referência de (271, 275, 262) até (353, 275, 262) pode definir uma sub-região de indiferença de z^A. Nas primeiras k (k=3) restrições de $P_{z^+}^{\infty,\rho}$, ou seja, em $z_i^+ - c^i x \le \alpha$ (i=1,...,3), z^+ é o ponto de referência paramétrico (271+θ, 275, 262), 0≤ θ ≤82, e $(c^1x, c^2x, c^3x) = z^A$ = (173, 160, 109). Logo, na solução óptima de $P_{z^+}^{\infty,\rho}$ as restrições tomam a forma 98+θ ≤ α, 115 ≤ α e 153 ≤ α, respectivamente para i =1,2,3, em que α é o menor valor que satisfaz estas condições. O conjunto das restrições activas J ⊆{1,...,3} varia no intervalo 0≤ θ ≤82: para 0 ≤ θ <55, J={3} e o valor óptimo de α é dado pela 3ª restrição (α=153); mas, para 55< θ ≤82, J={1} e o valor óptimo de α é dado pela 1ª restrição (α=98+θ); para θ =55, J={1,3}. Em resumo, as variáveis desvio destas restrições variam qualitativamente da seguinte forma:

0≤ θ <55	⇒	s_1 >0, s_2 >0, s_3=0
θ =55	⇒	s_1 =0, s_2 >0, s_3=0
55< θ ≤82	⇒	s_1 =0, s_2 >0, s_3>0

Para 0 ≤ θ <55 e 55< θ ≤82, cada ponto de referência define um paralelogramo; para θ =55, o paralelogramo reduz-se a um segmento de recta. Analisemos, em primeiro lugar, os paralelogramos definidos por 0 ≤ θ <55.

1ª PARTE: 0≤ θ ≤55, s_1 =55–θ, s_2 =38, s_3=0

Para θ' particular, os vértices da correspondente sub-região de indiferença são (de acordo com a proposição V.2):

$P_0^{\theta'}$: o original	(271+ θ', 275, 262)
$P_1^{\theta'}$: adição de s_1	(271+55, 275, 262)
$P_2^{\theta'}$: adição de s_2	(271+ θ', 275+38, 262)
$P_{1,2}^{\theta'}$: adição de s_1 e s_2	(271+55, 275+38, 262)

Tal como mencionamos atrás, um ponto de referência z^+ pode ser convertido para um plano Z_Q^+, adicionando-se $\left(Q - \sum_{i=1}^{k} z_i^+\right) \Big/ k$ a todas as componentes de z^+. Considerando Q=500, os pontos anteriores transformam-se em:

$P_0^{\theta'}$: (168.33+ ⅔ θ', 172.33- ⅓ θ', 159.33- ⅓ θ')

$P_1^{\theta'}$: (205, 154, 141)

$P_2^{\theta'}$: (155.67+ ⅔ θ', 197.67- ⅓ θ', 146.67- ⅓ θ')

$P_{1,2}^{\theta'}$: (192.33, 179.33, 128.33) (N^A)

Observamos que $P_{1,2}^{\theta'}$ é independente de θ' porque é o *NÚCLEO* (N^A) da região de indiferença de z^A. A conversão anterior possibilita a representação gráfica no plano das sub-regiões de indiferença (paralelogramos). A figura V.4 mostra um esboço de vários paralelogramos sobrepostos, associados aos valores extremos do parâmetro, θ=0 e θ=55 (para este último, apenas um segmento de recta), e a dois valores intermédios, θ=10 e θ=20. Conforme se avança desde θ=0 até θ=55 os paralelogramos são sucessivamente menores, de tal modo que o paralelogramo definido por θ=0 contém todos os outros. Assim, bastaria traçar o paralelogramo definido a partir do ponto de referência (271, 275, 262) para conhecer a maior sub-região de indiferença possível de ser definida a partir desta direcção e utilizando o resultado da proposição V.2.

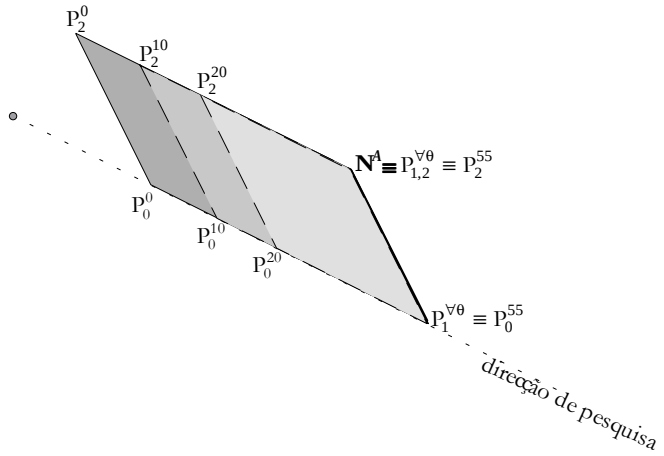


Fig. V.4 – Esboço de sub-regiões de indiferença de z^A definidas na 1ª parte da análise.

Analisemos agora os paralelogramos definidos por 55≤ θ ≤82.

2ª PARTE: 55≤ θ ≤82, s_1 =0, s_2 =θ–17, s_3=θ–55

Consideremos a mudança de variável δ=θ-55, o que implica que 0≤ δ ≤27, s_1 =0, s_2 =δ+38 e s_3=δ. Para um δ' particular, os vértices da correspondente sub-região de indiferença são:

$P_0^{\delta'}$: o original $(326+\delta', 275, 262) \xrightarrow{\bar{Z}_Q^+ (Q=500)} (205+\frac{2}{3}\delta', 154-\frac{1}{3}\delta', 141-\frac{1}{3}\delta')$

$P_2^{\delta'}$: adição de s_2 $(326+\delta', 275+38+\delta', 262)$ $(192.33+\frac{1}{3}\delta', 179.33+\frac{1}{3}\delta', 128.33-\frac{2}{3}\delta')$

$P_3^{\delta'}$: adição de s_3 $(326+\delta', 275, 262+\delta')$ $(205+\frac{1}{3}\delta', 154-\frac{2}{3}\delta', 141+\frac{1}{3}\delta')$

$P_{2,3}^{\delta'}$: adição de s_2 e s_3 $(326+\delta', 275+38+\delta', 262+\delta')$ $(192.33, 179.33, 128.33)$ (N^A)

Os paralelogramos aumentam de tamanho conforme δ varia de 0 a 27, estando todos contidos no paralelogramo definido pelo último ponto de referência, (353, 275, 262), correspondente a δ=27. Esta sub-região de indiferença e a (maior) definida na 1ª parte são apresentadas na figura V.5.

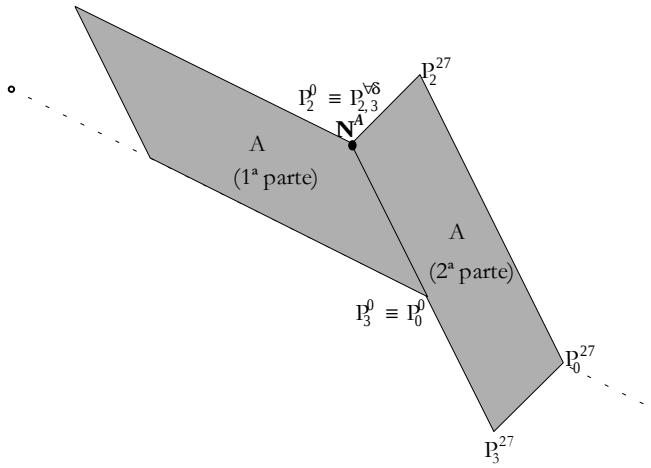


Fig. V.5 – Esboço das sub-regiões de indiferença de z^A definidas nas duas partes da análise.

Podemos estabelecer um processo sistemático para definir e representar graficamente sub-regiões de indiferença de problemas tri-objectivo.

1. Suponhamos que se conhece **um ponto** de referência z^+ que conduz à solução não dominada z^a através da optimização de $P_{z^+}^{\infty,\rho}$. Se apenas uma variável s_i (i=1,...,3) for nula na solução óptima de $P_{z^+}^{\infty,\rho}$, então pode definir-se uma sub-região de indiferença que é um paralelogramo. Contudo, se duas ou três variáveis s_i forem nulas, então o paralelogramo reduz-se a um segmento de recta ou a um ponto, respectivamente. Consideremos que se trata de um paralelogramo, já que um segmento de recta ou um ponto são casos particulares deste, e que as variáveis desvio s_i (i=1,...,3) satisfazem $s_p>0$, $s_q>0$ e $s_r=0$, $p,q,r \in \{1,2,3\}$, $p\neq q\neq r$ na solução óptima de $P_{z^+}^{\infty,\rho}$. Para representar graficamente o paralelogramo, começa-se por converter z^+ num ponto z^{++} do plano Z_Q^+ (como foi indicado atrás), e determina-se o núcleo N^a. O núcleo é o único ponto da região de indiferença de z^a em Z_Q^+ que verifica

s_i=0, i=1,...,3, na solução óptima do programa escalarizante. Obtém-se N^a pela conversão de z^a para Z_Q^+. O paralelogramo pode ser desenhado directamente da seguinte forma: traçam-se duas semi-rectas a partir de z^{++}, uma com direcção m_p e outra com direcção m_q; em seguida, traçam-se outras duas semi-rectas a partir de N^a paralelas a cada uma das semi-rectas que partem de z^{++}. Os pontos z^{++}, N^a e os pontos de intersecção das semi-rectas anteriores são os vértices do paralelogramo. (Nota: se a sub-região de indiferença for um segmento de recta, então este é o segmento que une z^{++} a N^a e, se for apenas um ponto, trata-se do núcleo N^a).

2. Suponhamos agora que se conhece um **intervalo** $[z^{a+},z^{b+}]$ de pontos de referência que apenas diferem numa componente (i.e. formam um segmento de recta numa das direcções m_1, m_2 ou m_3), e que conduzem a z^a. Se o conjunto J de restrições activas de $P_{z^+}^{\infty,\rho}$ variar em $[z^{a+},z^{b+}]$, então o intervalo $[z^{a+},z^{b+}]$ deve ser partido em subintervalos. Nesse caso, existe um ponto z^{c+} que divide $[z^{a+},z^{b+}]$ em $[z^{a+},z^{c+}]$ e $[z^{c+},z^{b+}]$, para o qual $J^c = J^a \cup J^b$. O número de variáveis s_i (i=1,...,3) iguais a zero em $P_{z^{c+}}^{\infty,\rho}$, i.e. o cardinal de J^c, é superior em uma unidade ao cardinal de J^a, que por sua vez é igual ao de J^b. Cada ponto de referência de $[z^{a+},z^{b+}]$ permite definir uma sub-região de indiferença de z^a mas, dentro de cada subintervalo, há um ponto de referência cuja sub-região abrange todas as outras: a de z^{a+} abrange as de $[z^{a+},z^{c+}]$, a de z^{b+} abrange as de $[z^{c+},z^{b+}]$ e, se não for necessário partir o intervalo, então a sub-região definida por um dos pontos extremos (z^{a+} ou z^{b+}) abrange as de $[z^{a+},z^{b+}]$. Desta forma, basta seguir os passos de **1.** para z^{a+} e/ou z^{b+} para traçar a maior sub-região de indiferença de z^a definida a partir de $[z^{a+},z^{b+}]$.

Relembramos que este processo não define toda a região de indiferença de uma dada solução, mas apenas sub-regiões convexas. Este processo pode ser integrado num método interactivo de ponto de referência, em que as sub-regiões são progressivamente calculadas e apresentadas graficamente ao AD. A intenção não será certamente o cálculo de todas as regiões de indiferença no espaço dos pontos de referência, mas apenas apoiar o AD na selecção de novos pontos de referência e/ou direcções de pesquisa.

Apesar de a intenção do processo não ser o cálculo exaustivo, resolvemos calcular todas as regiões de indiferença do problema anterior para ilustrar a configuração geral dessas regiões. A decomposição completa do triângulo $\overline{Z}_Q^+$ definido por Q=500 e $z_i^{+\min}$=50, i=1,...,3, para o problema KNAPM20_3FO, que inclui a representação das regiões de indiferença de todas as soluções não dominadas do problema, é apresentada na figura V.6.

Como podemos observar na figura V.6, as regiões de indiferença são, em geral, não convexas, mas as linhas que as delimitam têm direcções particulares, que são m_1, m_2 ou m_3. Isto acontece em todos os problemas de PLIMO tri-objectivo, e a extensão para mais do que três funções objectivo é directa, apesar de não ser fácil a sua visualização gráfica.

As sub-regiões definidas durante a pesquisa direccional anterior (pelo processo descrito acima) para as soluções 'H' e 'A' estão delimitadas no gráfico da figura V.6 por linhas a cinzento mais escuro. Os NÚCLEOS estão também assinalados (a preto). A tabela V.1 mostra os valores das funções objectivo de todas as soluções não dominadas do problema, onde podemos observar que as soluções 'B', 'D' e 'E' optimizam z_1, z_2 e z_3, respectivamente. Como seria de esperar, as regiões de indiferença destas soluções ocupam áreas junto aos respectivos vértices do triângulo.

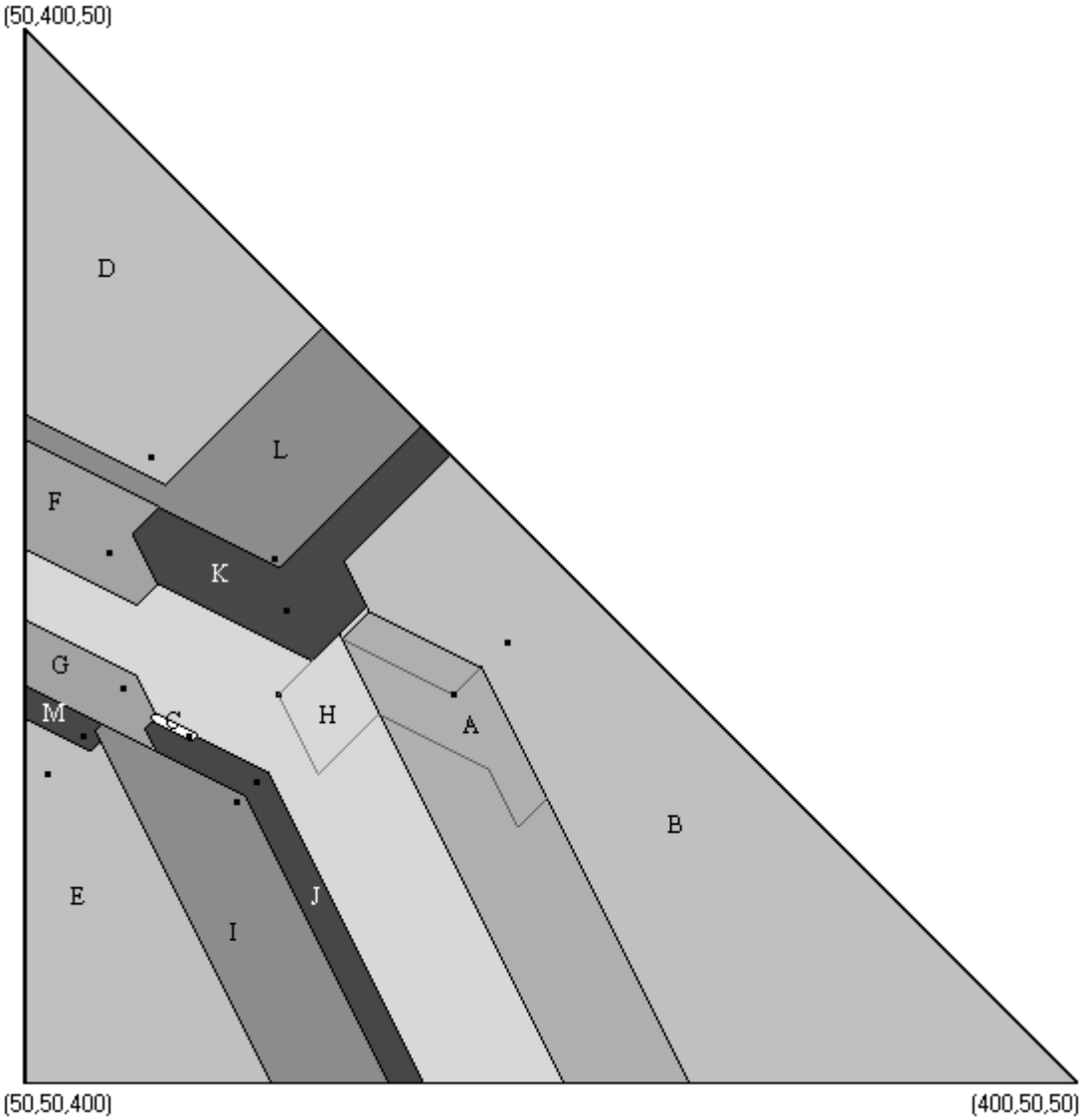


Fig. V.6 – Regiões de indiferença de todas as soluções não dominadas do problema KNAPM20_3FO.

Solução	z_1	z_2	z_3
A	173	160	109
B	199	185	81
C	48	109	174
D	39	206	97
E	0	89	221
F	39	188	157
G	28	127	182
H	117	162	170
I	72	95	188
J	84	107	181
K	117	187	136
L	100	192	110
M	0	96	198

Tabela V.1 – Todas as soluções não dominadas do problema KNAPM20_3FO.

No estudo preliminar de um problema de selecção de estratégias de controlo remoto de cargas numa rede de distribuição de energia eléctrica, apercebemo-nos de algumas potencialidades da visualização gráfica de (sub-)regiões de indiferença, nomeadamente quando combinadas com *pesquisas direccionais*. O problema em causa envolve diferentes aspectos conflituosos entre si, como a minimização do pico da procura, a maximização do lucro e a minimização do desconforto para os consumidores. O modelo foi proposto e estudado por JORGE ET AL. (2000) usando o método STEM (BENAYOUN ET AL., 1971). O problema que testámos é semelhante ao original, com a única diferença de que os coeficientes foram arredondados para que o problema se tornasse inteiro puro (PLIMO) e pudéssemos, assim, utilizar a ferramenta de visualização das regiões de indiferença. As implicações nas soluções não dominadas são apenas ao nível dos valores dos critérios, não diferindo nos valores das variáveis.

Para este estudo (preliminar), determinámos 30 soluções não dominadas através de várias *pesquisas direccionais*. O triângulo da figura V.7 mostra as sub-regiões de indiferença no espaço dos pontos de referência obtidas para cada uma das soluções calculadas. Relembramos que cada uma destas regiões pode não estar completa (por exemplo, as soluções de 4 a 18 poderão corresponder a áreas maiores que se estendem para a parte de baixo do triângulo, formando regiões não convexas). A análise do triângulo leva-nos a admitir a existência muitas outras soluções não dominadas, uma vez que ainda restam grandes espaços em branco. Se, por um lado, alguns desses espaços poderão corresponder a soluções já conhecidas, também há outros espaços – como por exemplo entre as soluções 18 e 29 – que correspondem a soluções não calculadas. Admitimos que possam ser bastantes, na medida em que cada solução anterior corresponde a uma área muito pequena do triângulo, o mesmo podendo acontecer com as desconhecidas. As áreas das soluções 1, 2 e 3 são muito superiores às outras porque considerámos amplos intervalos $[z_i^{+min}, z_i^{+max}]$ para a definição do triângulo, de modo a permitir que todas as soluções não dominadas tivessem representação neste triângulo.

Em resumo, julgamos que uma abordagem como a que utilizámos traz algumas vantagens relativamente ao método STEM. O STEM não dá qualquer indicação acerca da quantidade de soluções não dominadas, ou da estabilidade dessas soluções relativamente aos parâmetros do programa escalarizante. A abordagem que propomos tem a desvantagem de obrigar a arredondamentos prévios dos coeficientes do modelo, mas permite uma visualização gráfica dos resultados que dá uma ideia da quantidade e do grau de estabilidade das soluções face à intenção de melhorar uma função objectivo.

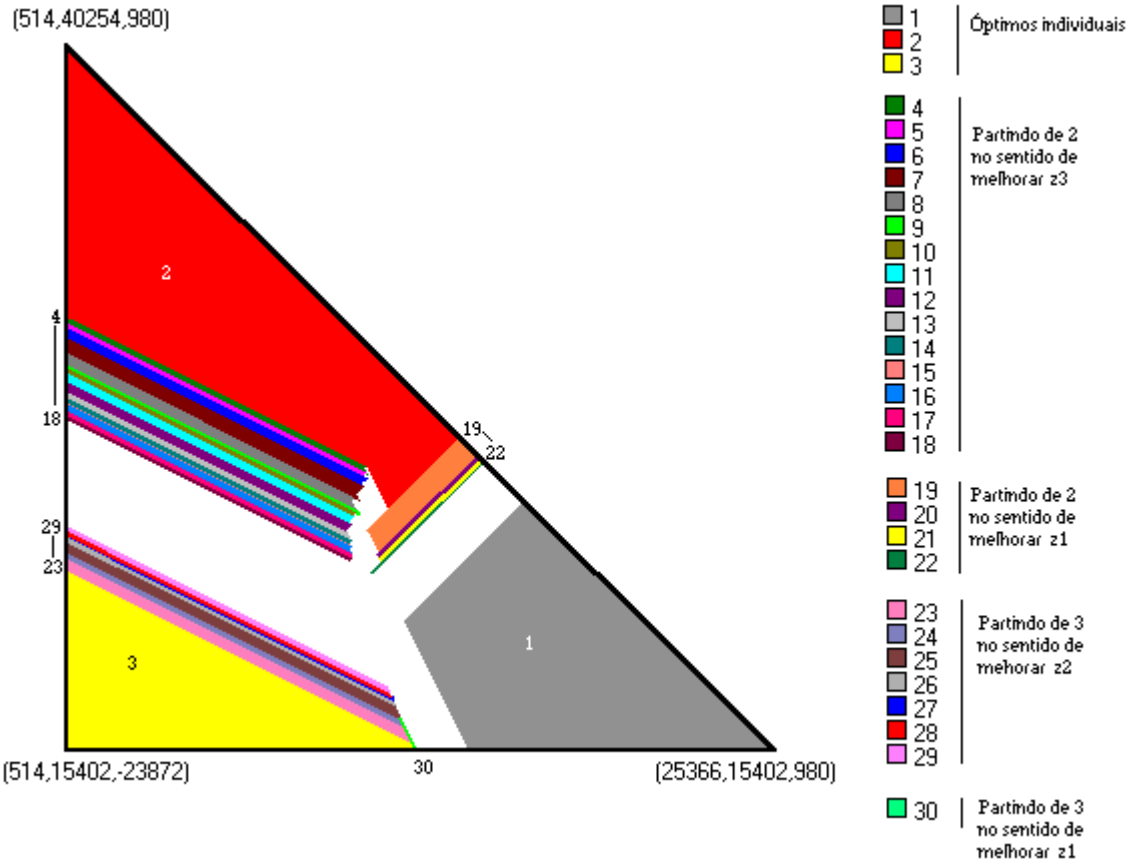


Fig. V.7 – Sub-regiões de indiferença de algumas soluções não dominadas de um problema de selecção de estratégias de controlo remoto de cargas numa rede de distribuição de energia eléctrica .

V.2 COMPORTAMENTO DOS PONTOS DE REFERÊNCIA NA PRESENÇA DE LIMITAÇÕES NAS FUNÇÕES OBJECTIVO

Durante a pesquisa de soluções não dominadas, o AD pode desejar impor limitações adicionais nos valores das funções objectivo. No caso das abordagens com pontos de referência, como as que temos vindo a considerar, essas restrições são incluídas no programa escalarizante $P_{z^+}^{\infty,\rho}$.

Vejamos como podemos continuar a representar regiões de indiferença do espaço dos pontos de referência do problema *original* quando as soluções são obtidas através da resolução de programas escalarizantes que incluem limitações adicionais do tipo $c^j x \geq L_j$.

Consideremos que são introduzidas em $P_{z^+}^{\infty,\rho}$ as limitações adicionais $c^j x \geq L_j$, $j \in \overline{K} \subseteq \{1,...,k\}$. Seja este programa escalarizante denotado por $P_{z^+}^{\infty,\rho}(L)$ em que L representa o vector (de dimensão $\#\overline{K}$) dos limites inferiores para alguns dos critérios.

$$\begin{aligned} \min \quad & \alpha - \rho \sum_{i=1}^{k} c^i x && P_{z^+}^{\infty,\rho}(L) \\ \text{s.a:} \quad & z_i^+ - c^i x \leq \alpha \qquad i=1,...,k \\ & c^j x \geq L_j \qquad \forall j \in \overline{K} \\ & x \in X \end{aligned}$$

A introdução de limitações adicionais restringe em geral o conjunto das soluções eficientes, pelo que, para um dado ponto de referência z^+, a solução eficiente que optimiza $P_{z^+}^{\infty,\rho}(L)$ pode não ser óptima de $P_{z^+}^{\infty,\rho}$. Consequentemente, apenas algumas soluções eficientes têm representação no espaço dos pontos de referência do problema restrito e a decomposição deste espaço difere da decomposição do espaço do problema original. Tendo como intenção uma representação coerente, que possa apresentar qualquer solução eficiente do problema original, independentemente da forma como foi calculada, torna-se indispensável a tradução de pontos de referência z^+ para z^{++} de modo a que $P_{z^+}^{\infty,\rho}(L)$ e $P_{z^{++}}^{\infty,\rho}$ produzam o mesmo resultado.

No lema e proposição que se seguem consideramos ρ suficientemente pequeno tal que, se uma solução $\bar{x}$ optimiza um programa escalarizante de Tchebycheff aumentado, $P_{z^+}^{\infty,\rho}$ ou $P_{z^+}^{\infty,\rho}(L)$, então $\bar{x}$ também optimiza o correspondente programa escalarizante de Tchebycheff simples ($P_{z^+}^{\infty}$ ou $P_{z^+}^{\infty}(L)$) cuja função objectivo é apenas a minimização de α.

O *lema V.3* estabelece uma situação em que o resultado de $P_{z^+}^{\infty,\rho}(L)$ é igual ao de $P_{z^+}^{\infty,\rho}$.

Lema V.3: Seja $(\tilde{x}, \tilde{\alpha})$ a solução óptima de $P_{z^+}^{\infty,\rho}(L)$ para um dado z^+ e um dado vector L. Se $z_j^+ > L_j + \tilde{\alpha}$ para todo o $j \in \overline{K}$, então $(\tilde{x}, \tilde{\alpha})$ também optimiza $P_{z^+}^{\infty,\rho}$ (o que significa que as restrições $c^j x \geq L_j$, $\forall j \in \overline{K}$ poderiam ser eliminadas porque não condicionam o resultado).

Prova: Suponhamos que existe pelo menos uma limitação $c^r x \geq L_r$, $r \in \overline{K}$, que condiciona o resultado do programa escalarizante, ou seja $(\tilde{x}, \tilde{\alpha})$ optimiza

$P_{z^+}^{\infty,\rho}(L)$ mas não optimiza $P_{z^+}^{\infty,\rho}$. Seja então $(\hat{x}, \hat{\alpha})$ a solução óptima de $P_{z^+}^{\infty,\rho}$ (e de $P_{z^+}^{\infty}$), que verifica necessariamente $c^r\hat{x} < L_r$. A restrição $z_r^+ - c^r x \le \alpha$ de $P_{z^+}^{\infty,\rho}$ impõe que $c^r\hat{x} + \hat{\alpha} \ge z_r^+$. Como $z_r^+ > L_r + \tilde{\alpha}$, então $c^r\hat{x} + \hat{\alpha} > L_r + \tilde{\alpha} \Leftrightarrow c^r\hat{x} > L_r + \tilde{\alpha} - \hat{\alpha}$. Sabendo que $c^r\hat{x} < L_r$, temos que $L_r + \tilde{\alpha} - \hat{\alpha} < L_r \Leftrightarrow \tilde{\alpha} < \hat{\alpha}$ o que contraria a hipótese de que $(\hat{x}, \hat{\alpha})$ optimiza $P_{z^+}^{\infty}$ e, consequentemente, $P_{z^+}^{\infty,\rho}$.

A *proposição V.4* estabelece agora uma forma sistemática de se obter um ponto de referência z^{++} que conduz à mesma solução que um outro ponto z^+, se o primeiro for projectado no conjunto total das soluções não dominadas, e o segundo num subconjunto de soluções não dominadas restrito por limitações nos critérios. Os pontos de referência são iguais se as limitações não condicionarem o resultado de $P_{z^+}^{\infty,\rho}(L)$ – *lema V.3* – mas são diferentes no caso contrário.

Proposição V.4: Se $(\tilde{x}, \tilde{\alpha})$ optimiza $P_{z^+}^{\infty,\rho}(L)$, então $(\tilde{x}, \tilde{\alpha})$ também optimiza $P_{z^{++}}^{\infty,\rho}$ para $z_i^{++} = z_i^+, i \notin \overline{K}$ e $z_j^{++} = \max\left\{z_j^+, L_j + \tilde{\alpha}\right\}, j \in \overline{K}$.

Prova: Como $c^j\tilde{x} \ge L_j \Leftrightarrow c^j\tilde{x} + \tilde{\alpha} \ge L_j + \tilde{\alpha}$, $\forall j \in \overline{K}$, e $c^i\tilde{x} + \tilde{\alpha} \ge z_i^+$, $\forall i \in \{1,...,k\}$, então $(\tilde{x}, \tilde{\alpha})$ é uma solução admissível para $P_{z^{++}}^{\infty,\rho}$ (ou $P_{z^{++}}^{\infty}$). Suponhamos que ela não optimiza $P_{z^{++}}^{\infty,\rho}$ e seja $(\hat{x}, \hat{\alpha})$ a respectiva solução óptima. Então, $\hat{\alpha} - \rho\sum_{i=1}^{k} c^i\hat{x} < \tilde{\alpha} - \rho\sum_{i=1}^{k} c^i\tilde{x}$ (1)

Sendo ρ suficientemente pequeno, $(\hat{x}, \hat{\alpha})$ também optimiza $P_{z^{++}}^{\infty}$, o que significa que $\hat{\alpha} \le \tilde{\alpha}$. (2)

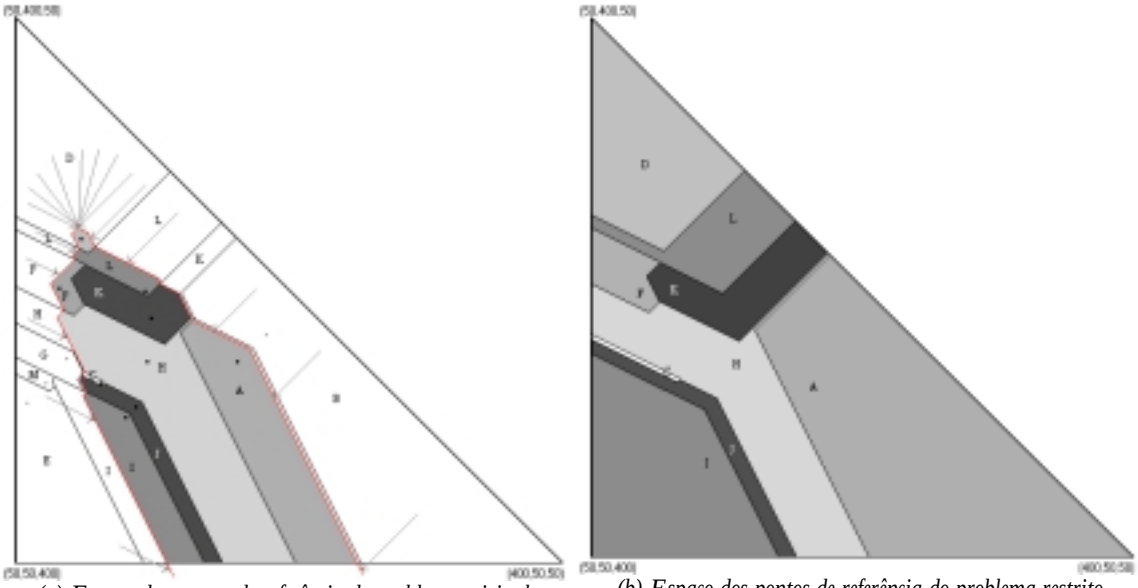
A solução $(\hat{x}, \hat{\alpha})$ não é admissível para $P_{z^+}^{\infty,\rho}(L)$ porque, se o fosse, seria esta a solução óptima já que, por (1), ela é melhor do que $(\tilde{x}, \tilde{\alpha})$.

Mas, $(\hat{x}, \hat{\alpha})$ verifica as k primeiras restrições de $P_{z^+}^{\infty,\rho}(L)$, uma vez que $\begin{cases} c^j\hat{x} + \hat{\alpha} \ge \max\left\{z_j^+, L_j + \tilde{\alpha}\right\} & j \in \overline{K} \\ c^i\hat{x} + \hat{\alpha} \ge z_i^+ & i \notin \overline{K} \end{cases} \Rightarrow c^i\hat{x} + \hat{\alpha} \ge z_i^+, \ \forall i \in \{1,...,k\}$. Então $(\hat{x}, \hat{\alpha})$ não será admissível para $P_{z^+}^{\infty,\rho}(L)$ apenas se existir algum $r \in \overline{K}$ para o qual $c^r\hat{x} < L_r$. (3)

Mas, $c^j \hat{x} + \hat{\alpha} \geq \max\left\{z_j^+, \mathrm{L}_j + \tilde{\alpha}\right\}, \forall j \in \overline{K}, \Rightarrow c^r \hat{x} + \hat{\alpha} \geq \mathrm{L}_r + \tilde{\alpha}$. (4)

Por (3) e (4), $\mathrm{L}_r + \tilde{\alpha} - \hat{\alpha} \leq c^r \hat{x} < \mathrm{L}_r \Rightarrow \tilde{\alpha} < \hat{\alpha}$ o que contradiz (2). Logo, $(\hat{x}, \hat{\alpha})$ não optimiza $P^{\infty}_{z^{++}}$ e, consequentemente, não optimiza $P^{\infty,\rho}_{z^{++}}$.

Para ilustrar a aplicação da *proposição V.4*, consideremos o problema tri-objectivo KNAPM20_3FO introduzido na secção anterior. Suponhamos que, durante o processo de decisão, foram impostas as limitações adicionais $z_1 \geq 30$ e $z_3 \geq 85$ para restringir o âmbito da pesquisa de soluções não dominadas. A figura V.8(a) mostra as regiões de indiferença (produzidas pelo sistema computacional) definidas a partir da transformação de pontos de referência estabelecida na *proposição V.4*. As soluções não dominadas 'B', 'E', 'G' e 'M' não têm representação porque elas não satisfazem as limitações adicionais. Os pontos de referência do interior das áreas sombreadas não são alterados pela função de transformação. Os pontos de referência fora destas áreas são transformados em pontos da fronteira. Este "mapeamento" – ilustrado por setas na figura V.8(a) – é feito segundo uma das direcções m_1 ou m_3 dado que as limitações são em z_1 e z_3 (excepto os pontos da parte superior do triângulo, na transição da parte à esquerda para a parte à direita da zona sombreada, que são todos transformados no vértice superior da região de 'D'). A representação para o problema original (figura V.8a) reserva a branco as áreas correspondentes às soluções não dominadas que não são agora alcançáveis. Assim, essas áreas poderiam ser preenchidas se as limitações fossem alteradas ou eliminadas. O gráfico da figura V.8(a) pode ser comparado com o da figura V.8(b) que mostra a decomposição completa do triângulo para o problema restrito. Como seria de esperar, as soluções não dominadas que satisfazem as limitações adicionais preenchem por completo o triângulo.



(a) Espaço dos pontos de referência do problema original *(b) Espaço dos pontos de referência do problema restrito*

Fig. V.8 –Regiões de indiferença de KNAPM20_3FO considerando limites inferiores nos critérios z_1 e z_3.

V.3 CONSIDERAÇÕES FINAIS

Neste capítulo apresentámos algumas propriedades relativas ao comportamento dos pontos de referência em programas escalarizantes de Tchebycheff para problemas de PLIMO. Analisámos a configuração das regiões de indiferença no espaço dos pontos de referência, em que *região de indiferença* foi a designação adoptada para um conjunto de pontos de referência que conduz à mesma solução não dominada. Apesar de não estabelecermos um procedimento adequado ao cálculo da região de indiferença completa de uma dada solução não dominada, propomos uma forma de definir sub-regiões desta. Investigámos ainda a decomposição do espaço dos pontos de referência quando se incluem limitações adicionais nos valores das funções objectivo.

O espaço dos pontos de referência (coincidente com o dos objectivos) de problemas de PLIMO com duas ou três funções objectivo pode ser representado num gráfico unidimensional ou bidimensional, respectivamente. É, portanto, fácil visualizarmos graficamente as regiões de indiferença destes problemas. Este tipo de representação gráfica foi integrado no sistema computacional multiobjectivo, na perspectiva de que possa ser um meio de apresentação de resultados útil para o AD. Em nosso entender, este tipo de informação é útil porque:

- dá uma ideia da distribuição espacial das soluções, permitindo uma melhor percepção da dimensão do problema e das diferenças de valores entre uma solução não dominada e as suas "vizinhas";
- apoia o AD na busca de novas soluções não dominadas, evitando que o AD escolha alguns pontos de referência que conduzem a soluções não dominadas já conhecidas;
- sugere um grau de estabilidade de cada solução relativamente à mudança do ponto de referência.

Contudo, não podemos deixar de referir, mais uma vez, que a informação de que dispomos é parcial e que seria manifestamente mais interessante se conseguíssemos definir, de uma só vez, uma região de indiferença completa.

Numa perspectiva mais teórica, podemos ainda acrescentar que esta ferramenta de representação gráfica é útil para o estudo das características dos problemas de PLIMO em geral, tendo em conta a utilização de pontos de referência.

ANEXO VII.A

Cálculo dos limites do triângulo

Para representar graficamente (num triângulo) o espaço dos pontos de referência de um problema tri-objectivo, devemos estabelecer valores para Q e $z_i^{+\min}$, i=1,...,3 que definem $\overline{Z}_Q^+ = \left\{ z^+ \in \Re^3 \,\middle|\, z_i^+ \in \left[z_i^{+\min}, Q - \sum_{j \neq i} z_j^{+\min} \right], i = 1,...,3 \right\}$, um triângulo do plano $Z_Q^+ = \left\{ z^+ \in \Re^3 \,\middle|\, \sum_{i=1}^{3} z_i^+ = Q \right\}$. Para um dado Q, os valores de $z_i^{+\min}$ devem ser tais que a escolha de pontos de referência pertencentes a $\overline{Z}_Q^+$ permita alcançar todas as soluções não dominadas. Para tal, bastaria assegurar que os *núcleos* das soluções não dominadas tivessem representação em $\overline{Z}_Q^+$. Ou seja, se z for um ponto não dominado, então a sua conversão para Z_Q^+ resulta no ponto (*núcleo*) z^Q=(z_1+δ, z_2+δ, z_3+δ), δ= $\frac{1}{3}\left(Q - \sum_{j=1}^{3} z_j \right)$, que deverá pertencer a $\overline{Z}_Q^+$. Isto verifica-se se $(z_i + \delta) \in \left[z_i^{+\min}, Q - \sum_{j \neq i} z_j^{+\min} \right]$, i =1,...,3. No entanto, como não conhecemos à partida os pontos não dominados, não podemos definir os núcleos e torna-se difícil estabelecer *a priori* os melhores limites para o triângulo.

Se todas as funções objectivo assumirem apenas valores positivos, podemos considerar, por exemplo, o zero como o mínimo de cada função objectivo (e consequentemente $z_i^{+\min}$) e o plano Z_Q^+ que passa pelo ponto ideal, em que $Q = \sum_{i=1}^{3} z_i^*$. No caso de o zero não ser adequado como mínimo, poderíamos considerar o mesmo plano (igual valor de Q), mas seria necessário conhecer o mínimo de cada função objectivo, ou um limite inferior deste, para definir os $z_i^{+\min}$. Contudo, o cálculo dos mínimos acarreta um esforço computacional adicional, mesmo se considerarmos os mínimos admissíveis, já que as dificuldades são muito maiores para calcular os mínimos eficientes (como se pode ver, por exemplo, em KORHONEN ET AL., 1997). Além disso, como os mínimos admissíveis podem ser muito inferiores aos mínimos eficientes, a sua consideração pode implicar uma tendência para que o triângulo apresente grandes áreas de indiferença nos extremos, concentrando todas as outras no centro.

Face a estes inconvenientes, optámos por utilizar uma regra heurística para especificar os valores de Q e $z_i^{+\min}$, mesmo correndo o risco de que os domínios não sejam suficientemente

abrangentes para representar todas as soluções não dominadas e que, por isso, venham a ser ajustados posteriormente.

Começa-se por calcular a tabela de *pay-off* do problema de PLIMO (tabela dos óptimos individuais). Sejam z_i^* e z_i^-, respectivamente, o máximo e o mínimo da tabela de *pay-off* para a função *i*. Optando por um triângulo em que os pontos de referência são todos superiores à solução ideal, consideramos $z_i^{+\min} = z_i^*$, i=1,2,3. Escolhidos os $z_i^{+\min}$, deve então atribuir-se a Q um valor suficientemente grande para que o uso de pontos de referência de componentes $z_i^{+\min} \le z_i^+ \le Q - \sum_{j \neq i} z_j^{+\min}$ permita, em geral, alcançar qualquer solução não dominada. Salientamos aqui o interesse de que as funções objectivo sejam previamente normalizadas.

Um ponto dos critérios z pode ser convertido para um ponto de referência maior ou igual do que o ponto ideal pela adição de $\delta = \max_i \{z_i^* - z_i\}$ a todas as componentes. Os maiores valores de δ resultam de pontos z em que há um grande afastamento de uma componente z_i relativamente a z_i^*. Além disso, o somatório das componentes do ponto convertido é tanto maior quanto o forem as outras componentes. Logo, as soluções que resultam em maiores somatórios são constituídas pelo mínimo numa função objectivo e máximo em todas as outras: assumindo z_i^- como mínimo da função objectivo *i*, $z^{(i)} = (z_1^*, ..., z_i^-, ..., z_k^*) \xrightarrow{\text{"conversão"}} (z_1^* + z_i^* - z_i^-, ..., z_i^*, ..., z_k^* + z_i^* - z_i^-)$ cujo somatório é $Q^{(i)} = \sum_{j=1}^{k} z_j^* + (k-1)(z_i^* - z_i^-)$, em que k=3 no caso tri-objectivo. Atribuímos a Q o maior dos somatórios $Q^{(i)}$ ou seja, $Q = \sum_{j=1}^{3} z_j^* + 2\delta^*$ com $\delta^* = \max_{i=1,...,3} \{z_i^* - z_i^-\}$.

Notemos que podem existir soluções não dominadas com valores inferiores aos z_i^-, mas possivelmente essas soluções também não apresentarão os valores máximos (ou próximos destes) simultaneamente nas outras 2 componentes. Podemos falar, então, numa certa "compensação" entre as componentes dos $z^{(i)}$ para que os possamos considerar aproximações de "pontos extremos" do conjunto não dominado, e assim justificar a sua adequação na especificação de Q. Mais uma vez, surge a necessidade da normalização das funções objectivo para que haja essa "compensação".

No caso do problema KNAPM20_3FO), a tabela de *pay-off* tem por linhas as soluções não dominadas 'B', 'D' e 'E' (ver valores na tabela V.1). O ponto ideal é z^*=(199, 206, 221), z^-=(0, 89, 81) e $\delta^* = \max_{i=1,...,3} \{z_i^* - z_i^-\}$=199. Então, os limites do triângulo calculados desta forma

são dados por $\left(z_1^{+\min}, z_2^{+\min}, z_3^{+\min}\right)$= (199, 206, 221) e Q= 199 +206 +221 +2×199= 1024. A fig V.9 mostra (em tamanho reduzido) a decomposição do triângulo para estes limites que, como podemos observar, não difere muito da que foi apresentada na figura V.6 para $\left(z_1^{+\min}, z_2^{+\min}, z_3^{+\min}\right)$= (50, 50, 50) e Q=500.

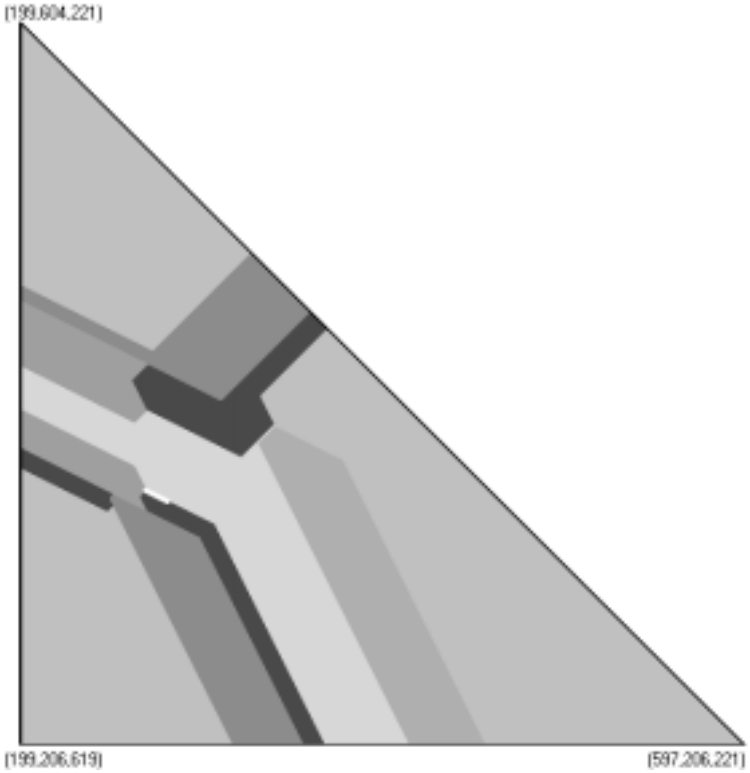


Fig. V.9 – Outros limites do triângulo para o problema KNAPM20_3FO.

ANEXO VII.B

Prova da proposição V.1

Proposição V.1: Se a solução eficiente x^a do problema PLIMO optimiza tanto $P_{z^{a+}}^{\infty,\rho}$ como $P_{z^{b+}}^{\infty,\rho}$ com $z^{a+}=\left(z_1^{a+},...,z_p^{a+},...,z_k^{a+}\right)$ e $z^{b+}=\left(z_1^{a+},...,z_p^{a+}+\breve{\theta}_p,...,z_k^{a+}\right)$, $\breve{\theta}_p>0$, então x^a também optimiza $P_{z^+}^{\infty,\rho}$ para z^+ entre z^{a+} e z^{b+}.

Prova: seja α^a o valor óptimo de α em $P_{z^{a+}}^{\infty,\rho}$.

CASO 1: se $z_p^{a+}-c^p x^a=\alpha^a$ então,

o valor óptimo de α em $P_{z^{b+}}^{\infty,\rho}$ (dado pela solução x^a) é $\alpha^b=\alpha^a+\breve{\theta}_p$ e o mínimo da função objectivo é $\alpha^a+\breve{\theta}_p-\rho\sum_{i=1}^{k}c^i x^a$.

O valor de α dado pela solução x^a em $P_{z^+}^{\infty,\rho}$, com $z^+=\left(z_1^{a+},...,z_p^{a+}+\theta_p,...,z_k^{a+}\right)$, $0<\theta_p<\breve{\theta}_p$, é $\alpha^a+\theta_p$. Suponhamos que $(x^a,\alpha^a+\theta_p)$ não optimiza $P_{z^+}^{\infty,\rho}$ e que x' é uma solução eficiente que optimiza $P_{z^+}^{\infty,\rho}$ com α' o valor óptimo de α. Então,

$$\alpha'-\rho\sum_{i=1}^{k}c^i x'<\alpha^a+\theta_p-\rho\sum_{i=1}^{k}c^i x^a \qquad (1)$$

A solução $(x',\alpha'+\breve{\theta}_p-\theta_p)$ é admissível para $P_{z^{b+}}^{\infty,\rho}$ e o valor da função objectivo nesta solução é $\alpha'+\breve{\theta}_p-\theta_p-\rho\sum_{i=1}^{k}c^i x'$. Mas, por (1), $\alpha'+\breve{\theta}_p-\theta_p-\rho\sum_{i=1}^{k}c^i x'<\alpha^a+\breve{\theta}_p-\rho\sum_{i=1}^{k}c^i x^a$, ou seja x' é melhor do que x^a para $P_{z^{b+}}^{\infty,\rho}$ o que contraria o facto de x^a optimizar $P_{z^{b+}}^{\infty,\rho}$.

CASO 2: se $z_p^{a+}-c^p x^a<\alpha^a$ então,

seja $s_p^a=\alpha^a-z_p^{a+}+c^p x^a$.

(2.1) Se $\breve{\theta}_p\le s_p^a$, então (x^a,α^a) é admissível em todo o problema $P_{z^+}^{\infty,\rho}$ com $z^{a+}\le z^+\le z^{b+}$ em que a região admissível se reduz sucessivamente. Logo, x^a optimiza qualquer um destes problemas.

(2.2) Se $\breve{\theta}_p>s_p^a$, então seja $z^{c+}=\left(z_1^{a+},...,z_p^{a+}+s_p^a,...,z_k^{a+}\right)$. Para $z^{a+}\le z^+\le z^{c+}$, estamos perante o caso *(2.1)*. Para $z^{c+}\le z^+\le z^{b+}$ a análise de incrementos relativamente a z^{c+} cai no *CASO 1* porque o valor óptimo de α para z^{c+} é dado pela diferença na componente *p*.

■

Capítulo VI

Estudo de um problema de localização - transporte

Este capítulo é baseado em ALVES E CLÍMACO (1999c) e pretende mostrar a aplicação do método interactivo de PLIMMO descrito no capítulo IV no estudo de um problema de localização-transporte. O problema segue um modelo multiobjectivo de programação linear inteira-mista e consiste em seleccionar locais para abrir estações de tratamento de materiais tóxicos, bem como estabelecer as rotas de transporte dos materiais (e respectivas quantidades) desde os locais onde são gerados até às estações de tratamento.

Um dos factores relevantes nas decisões que envolvem localização de estações de serviço e transporte de material é, obviamente, o custo. Contudo, quando se trata de material tóxico, o risco envolvido no transporte e tratamento do material são também factores importantes a considerar. CURRENT E RATICK (1995) propuseram um modelo multicritério para este tipo de problemas que inclui, para além da minimização do custo, outros quatro critérios que visam a minimização de diferentes riscos e a equidade na sua distribuição espacial. Como os autores referem, estes cinco critérios estão geralmente em conflito: "Por exemplo, uma solução que minimiza o custo irá enviar grandes quantidades de materiais através de vias de menor custo, o que implicará que as populações ao longo dessas vias fiquem expostas a elevados riscos. Por outro lado, optimizar o critério da equidade do transporte conduz a que pequenas quantidades sejam transportadas por um elevado número de vias, o que tende a incrementar o custo e o risco total envolvido, mas reduz a exposição máxima a que um indivíduo está sujeito". O modelo proposto por Current e Ratick assume que os materiais não podem ser transportados através dos locais onde são gerados ou dos locais potenciais para as estações de tratamento. A razão desta opção foi fundamentalmente a simplificação da rede, mas ela pode justificar-se, em certos casos, por questões de segurança. Baseando-se no modelo de Current e Ratick, COUTINHO-RODRIGUES ET AL. (1995, 1997) propõem um outro modelo que ultrapassa esta questão, sem ter

necessidade de modificar a rede que representa espacialmente o problema. É esse o modelo que adoptamos neste trabalho.

Os problemas de localização-transporte deparam-se, em geral, com três tipos de dificuldades (COUTINHO-RODRIGUES ET AL., 1995): são problemas inteiros ou inteiros-mistos, muitas vezes difíceis de resolver, mesmo em casos monocritério; por outro lado, o número de soluções não dominadas em problemas multiobjectivo pode crescer exponencialmente com o tamanho do problema; e, por último, quando o número de funções objectivo aumenta, a análise de compromissos entre as várias alternativas (soluções) torna-se cada vez mais difícil. COUTINHO-RODRIGUES ET AL. (1995, 1997) desenvolveram um sistema de apoio à decisão interactivo dedicado a este tipo de problemas. As soluções não dominadas são geradas através da abordagem das *somas pesadas*, com a possibilidade de se introduzir restrições nos valores das funções objectivo de modo a poder alcançar soluções não suportadas. Os programas escalarizantes das somas pesadas são optimizados usando 'sofware standard' de programação linear inteira-mista. Segundo os autores, a apresentação das soluções ao AD são os "olhos" do sistema. É usada uma técnica de representação gráfica, designada por BAGAL ('Best Againt Least'), em que cada solução não dominada é representada sob a forma de teia-de-aranha para poder ser facilmente comparada com a solução ideal e a solução anti-ideal (constituída pelo pior valor de cada objectivo na tabela de *pay-off*). As soluções ideal e anti-ideal definem os contornos interno e externo do BAGAL, respectivamente.

A abordagem que usámos para estudar o problema é a descrita no capítulo IV, que consiste fundamentalmente em *pesquisas direccionais*, com possível inserção de limitações adicionais nos valores das funções objectivo. Numa pesquisa direccional em que o AD manifestou a vontade de incrementar um critério z_j face a uma solução prévia, as soluções seguintes são obtidas pela projecção de pontos de referência z^+, de valores sucessivamente crescentes na componente z_j^+. A variação de z_j^+ é automática e é feita de modo a garantir que a solução não dominada seguinte seja próxima, mas diferente da anterior. Em problemas multiobjectivo inteiros-mistos, como o caso aqui tratado, existem troços de soluções contínuas em que as soluções são obtidas através de variações muito pequenas do ponto de referência z^+. Mas, ao longo de uma mesma direcção de pesquisa, pode também ser necessária uma grande variação de z^+ para poder "escapar" da solução anterior e "saltar" uma descontinuidade nas soluções não dominadas. A pesquisa direccional é um processo de varrimento de soluções, ao longo de uma dada direcção, que produz soluções contínuas (a menos de um passo controlado pelo AD) enquanto for possível, e salta automaticamente as descontinuidades, quando tal for imperativo. Esta ferramenta de pesquisa de soluções não dominadas é bastante diferente da utilizada por COUTINHO-RODRIGUES ET AL. (1995, 1997), o que torna difícil qualquer comparação de

resultados. Em nosso entender, estas duas tácticas de abordagem podem ser consideradas complementares.

Este capítulo está organizado da seguinte forma:

Na secção 1 apresentamos a formulação do problema. Na secção 2 ilustramos a aplicação da abordagem interactiva ao problema, mostrando e interpretando alguns resultados. Ao longo dessa secção apresentamos também algumas das características do sistema computacional, designadamente no que diz respeito à interacção com o utilizador. Por fim, apresentamos na secção 4 algumas conclusões do estudo efectuado.

A instância do problema que estudámos usa dados inspirados num problema real que nos foram facultados pelos autores do modelo. Os dados encontram-se no anexo VI.A e correspondem à rede usada por COUTINHO-RODRIGUES ET AL. (1997). No trabalho de COUTINHO-RODRIGUES ET AL. (1995) é adoptada uma simplificação desta rede.

VI.1 FORMULAÇÃO DO PROBLEMA

O problema consiste em seleccionar locais para instalar (abrir) estações de tratamento de material tóxico e estabelecer as rotas de transporte (e respectivas quantidades) desde os locais geradores até às estações de tratamento. Podemos distinguir três tipos de locais: os locais potenciais para instalar as estações de tratamento, os locais geradores de resíduo/material tóxico e os outros locais que são apenas pontos de passagem durante o transporte. O número de estações a instalar não é pré-definido, podendo variar de 1 até n_l, o número de locais potenciais.

As variáveis principais do problema dividem-se em dois grupos: variáveis contínuas X_{ij}, que representam a quantidade de resíduo a transportar de um local i para um local j, e variáveis binárias Y_j (em número n_l), que indicam se deve ou não ser instalada uma estação no local potencial j.

Os seguintes requisitos têm que ser cumpridos: todo o resíduo dos locais geradores tem que ser tratado; nenhuma estação pode tratar mais resíduo do que o permitido pela sua capacidade máxima; os pontos de passagem não geram nem absorvem resíduo. Estes requisitos traduzem-se em restrições do problema. É de notar que o modelo admite que os locais geradores de resíduo possam também funcionar com pontos de passagem. Logo, as restrições do problema também contemplam esses casos.

O problema admite ainda duas outras variáveis e restrições de carácter definidor para essas variáveis. São variáveis auxiliares que quantificam riscos e são usadas nas funções objectivo. Uma das variáveis, designada por P, representa o máximo risco imposto a um indivíduo devido ao processamento nas estações. Este índice de risco é quantificado pela maior quantidade de resíduo que é processada em alguma estação. A outra variável, designada por M, representa o

máximo risco imposto a um indivíduo devido ao transporte de resíduo. Este índice de risco é quantificado pela maior quantidade de resíduo que atravessa algum ponto de passagem, ou uma ligação directa entre um local gerador e uma estação de tratamento.

O problema tem cinco funções objectivo. As funções objectivo Z_1 e Z_2 são critérios de minimização de risco, em que Z_1 representa o risco total do transporte e Z_2 o risco total do processamento nas estações. As funções objectivo Z_3 e Z_4 representam critérios de equidade, em que se minimiza a máxima exposição a que um indivíduo está sujeito devido ao transporte de resíduo ($Z_3 = M$) e devido ao processamento nas estações ($Z_4 = P$). A função objectivo Z_5 minimiza o custo total, o que inclui os custos de transporte e os custos de operação do sistema, fixos e variáveis.

Na representação em rede do problema, os nodos simbolizam os locais (locais potenciais para as estações, locais geradores e locais de passagem), e os arcos que unem os nodos representam as ligações entre os diversos locais. Assim, seja G=(N,A) uma rede composta por um conjunto de nodos N e um conjunto A de arcos dirigidos, em que (i,j) é o arco que liga o nodo i ao nodo j. Seja N=(F, W, T), em que F é o conjunto das estações de tratamento potenciais, W é o conjunto das fontes de resíduo (locais geradores) e $T=N\setminus\{F\cup W\}$ é o conjunto dos outros nodos, nodos de transporte que são apenas pontos de passagem. Consideremos ainda as seguintes definições:

X_{ij} – (variável) quantidade de resíduo enviada do nodo i para o nodo j;

Y_j – (variável binária) $Y_j = 1$ se a estação de tratamento $j \in F$ é aberta e $Y_j = 0$ no caso contrário;

w_i – quantidade de resíduo gerada em $i \in W$ num dado período de tempo;

a_{ij} – população total a menos de uma distância D do arco (i,j);

a_j – factor de densidade populacional junto à estação $j \in F$;

c_{ij} – custo, por unidade de resíduo, para atravessar o arco (i,j);

f_j – custo fixo de abertura da estação $j \in F$;

h_j – custo de operação, por unidade de resíduo, da estação $j \in F$;

k_j – capacidade máxima da estação $j \in F$;

A formulação do problema é a seguinte:

$$\min \quad Z_1 = \sum_i \sum_j a_{ij} X_{ij}$$

$$\min \quad Z_2 = \sum_{j \in F} a_j \left(\sum_i X_{ij} - \sum_i X_{ji} \right)$$

$$\min \quad Z_3 = M$$

$$\min \quad Z_4 = P$$

$$\min \quad Z_5 = \sum_i \sum_j c_{ij} X_{ij} + \sum_{j \in F} \left(f_j Y_j + h_j \left(\sum_i X_{ij} - \sum_i X_{ji} \right) \right)$$

sujeito a:

$$\sum_j X_{ij} - \sum_j X_{ji} = w_i \qquad \forall i \in W \qquad (1)$$

$$\sum_i X_{ij} - \sum_i X_{ji} \leq k_j Y_j \qquad \forall j \in F \qquad (2)$$

$$\sum_i X_{il} - \sum_i X_{li} = 0 \qquad \forall l \in T \qquad (3)$$

$$\sum_i X_{ij} - \sum_i X_{ji} \geq 0 \qquad \forall j \in F \qquad (4)$$

$$\sum_i X_{il} \leq M \qquad \forall l \in T \qquad (5)$$

$$X_{ij} \leq M \qquad \forall (i,j) \text{ com } i \text{ e } j \in W \cup F \qquad (6)$$

$$\sum_i X_{ij} - \sum_i X_{ji} \leq P \qquad \forall j \in F \qquad (7)$$

$$X_{ij} \geq 0 \qquad \forall (i,j) \in A \qquad (8)$$

$$Y_j \in \{0,1\} \qquad \forall j \in F \qquad (9)$$

O conjunto de restrições (1) assegura que todo o resíduo gerado é enviado das fontes para outros nodos, podendo os nodos geradores funcionar também como pontos de passagem; (2) proíbe que seja retido resíduo numa estação se ela não for aberta (i.e. Y_j=0) e, caso seja aberta, restringe a retenção de resíduo à sua capacidade máxima; (3) são equações de equilíbrio que impõem que todos os nodos de transporte, que não são nem fontes nem estações de tratamento, funcionem apenas como pontos de passagem; (4) complementa (2) para estabelecer as condições de equilíbrio nos nodos de estações potenciais, quando estes funcionam apenas como pontos de passagem; as restrições (5) e (6) atribuem à variável *M* (máximo risco imposto a um indivíduo devido ao transporte de resíduo) o valor da maior quantidade de resíduo que atravessa algum nodo de transporte (5), ou uma ligação directa entre uma fonte e uma estação (6); as restrições (7) atribuem à variável *P* (máximo risco imposto a um indivíduo devido ao processamento nas estações) o valor da maior quantidade de resíduo que é processado em alguma estação; (8) assegura o transporte de quantidades não negativas (não necessariamente inteiras) e (9) estabelece a natureza binária das decisões de localização.

Os dados do problema que estudámos encontram-se no anexo VI.A. A rede tem 50 nodos em que 4 são estações de tratamento potenciais (assinalados na rede como 'FAC' de 1 a 4), 15 são

fontes de resíduos tóxicos ('WASTE' de 1 a 15) e 31 são nodos de transporte ('TN' de 1 a 31). A rede tem 113 arcos não dirigidos que podem ser atravessados em ambos os sentidos, o que corresponde a 226 arcos dirigidos. Este problema tem 232 variáveis, das quais 226 são variáveis contínuas X_{ij}, 4 são variáveis binárias Y_j, e as outras duas são as variáveis auxiliares M e P. Tem 107 restrições (excluindo as de não negatividade): 15 restrições do tipo (1), 4 do tipo (2), 31 do tipo (3), 4 do tipo (4), 31 do tipo (5), 18 do tipo (6) e 4 do tipo (7).

Adoptaremos, ao longo deste capítulo, as designações F_j, W_i e T_l para $j \in F$, $i \in W$ e $l \in T$, respectivamente.

VI.2 ESTUDO DO PROBLEMA E ILUSTRAÇÃO DO FUNCIONAMENTO DO SISTEMA COMPUTACIONAL

A pesquisa interactiva de soluções não dominadas que apresentaremos em seguida pretende ilustrar o funcionamento do sistema computacional em geral, e do algoritmo de pesquisa direccional em particular, não se alienando da interpretação física das soluções obtidas para o problema.

O sistema computacional inclui diferentes meios gráficos e numéricos de apresentação de resultados. Os valores das funções objectivo das soluções não dominadas calculadas podem ser visualizados em vários formatos gráficos – barras, pontos ou linhas – ou sob a forma numérica, com várias opções de ordenação. A apresentação habitual dos valores das variáveis é a numérica, mas o sistema inclui também uma representação em rede para problemas com estrutura de rede semelhante àquela que abordamos neste trabalho. Ao longo deste estudo apresentaremos algumas destas formas de representação da informação. Existem ainda outros gráficos que não se adequam ao problema em causa por se limitarem a um máximo de 3 funções e/ou problemas inteiros puros.

Iniciámos o processo de pesquisa com o cálculo das 5 soluções eficientes que minimizam individualmente cada uma das funções objectivo. Para o efeito, foram usadas somas pesadas das funções objectivo, tomando peso 0.9999 para a função objectivo que pretendíamos minimizar e 2.5×10^{-5} para cada uma das outras. Estas soluções dão-nos um conhecimento geral de estratégias de decisão extremas, e por isso faremos uma análise mais detalhada das suas características. Elas dão uma percepção inicial dos domínios dos valores dos critérios envolvidos no problema, o que

pode ajudar o AD na detecção posterior de soluções de compromisso. A tabela VI.1 (tabela de *pay-off*) mostra os valores das funções objectivo destas soluções.[1]

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 1	**1 467 210**	*25 955 000*	8 040	15 630	*3 701 069*
Solução 2	2 939 590	**15 592 500**	*14 460*	*34 650*	3 223 054
Solução 3	*3 799 851*	21 770 318	**1 682.7**	13 462	3 579 522
Solução 4	2 137 958	23 388 750	6 370	**8 662.5**	3 367 100
Solução 5	3 324 450	23 677 500	9 410	25 410	**2 597 166**

Tabela VI.1 – Soluções não dominadas que minimizam cada uma das funções objectivo.

Solução 1 – solução eficiente que minimiza Z_1 (risco total do transporte).

A solução 1 considera a abertura de todas as estações de tratamento. Isso acarreta um elevado risco total nas estações (Z_2) e um custo (Z_5) também muito elevado (é de salientar que existem custos fixos para abertura das estações). As maiores exposições individuais ao material tóxico (riscos individuais Z_3 e Z_4) são moderados. Este plano contempla uma rede pouco "densa" em termos de utilização de vias de transporte (como podemos ver na figura VI.1). As estações F_1, F_2, F_3 e F_4 processam 640, 5780, 15630 e 12600 unidades, respectivamente. Notemos que a estação com maior processamento é F_3 (15630=Z_4), que é também aquela que se situa no local com maior densidade populacional (daí que Z_2 seja elevado). A maior exposição individual durante o transporte (Z_3) é 8040 na ligação W_2–F_4.

A figura VI.1 é cópia da janela do sistema que mostra as características da rede da solução. As vias de comunicação usadas estão a traço mais grosso com indicação do respectivo sentido. As estações de tratamento ('FAC') fechadas têm o formato ■ e as abertas têm formato ▣ (que nesta solução são todas). Durante a utilização do sistema, o analista pode visualizar na rede o resíduo que atravessa cada arco ou cada nodo. No caso dos nodos, é indicado o total de resíduo que entra (+), o que sai (-) e a respectiva diferença (+/-). Esta característica é visível na figura VI.1: posicionando o cursor num qualquer nodo, neste caso o nodo 'FAC' 2, a barra de estado (na parte inferior da janela) mostra a respectiva informação.

[1] Durante este estudo apercebemo-nos de que as funções objectivo Z_2, Z_3 e Z_4 admitem várias soluções óptimas alternativas. Estas soluções são eficientes ou fracamente eficientes do problema multiobjectivo. A tabela de *pay-off* não é, portanto, única. Para além disso, devido ao elevado número de soluções eficientes do problema, os resultados são muito sensíveis aos pesos usados nas somas pesadas. A título de exemplo, referimos que, se considerarmos o peso 0.999 para Z_3 e 0.00025 para todas as outras funções, a solução eficiente obtida não minimiza Z_3. Nessa solução, Z_3=1820 (sendo 1682.7 o respectivo mínimo). Estes factos devem explicar as diferenças das 5 soluções da tabela VI.1 relativamente aos valores obtidos por COUTINHO-RODRIGUES ET AL. (1997).

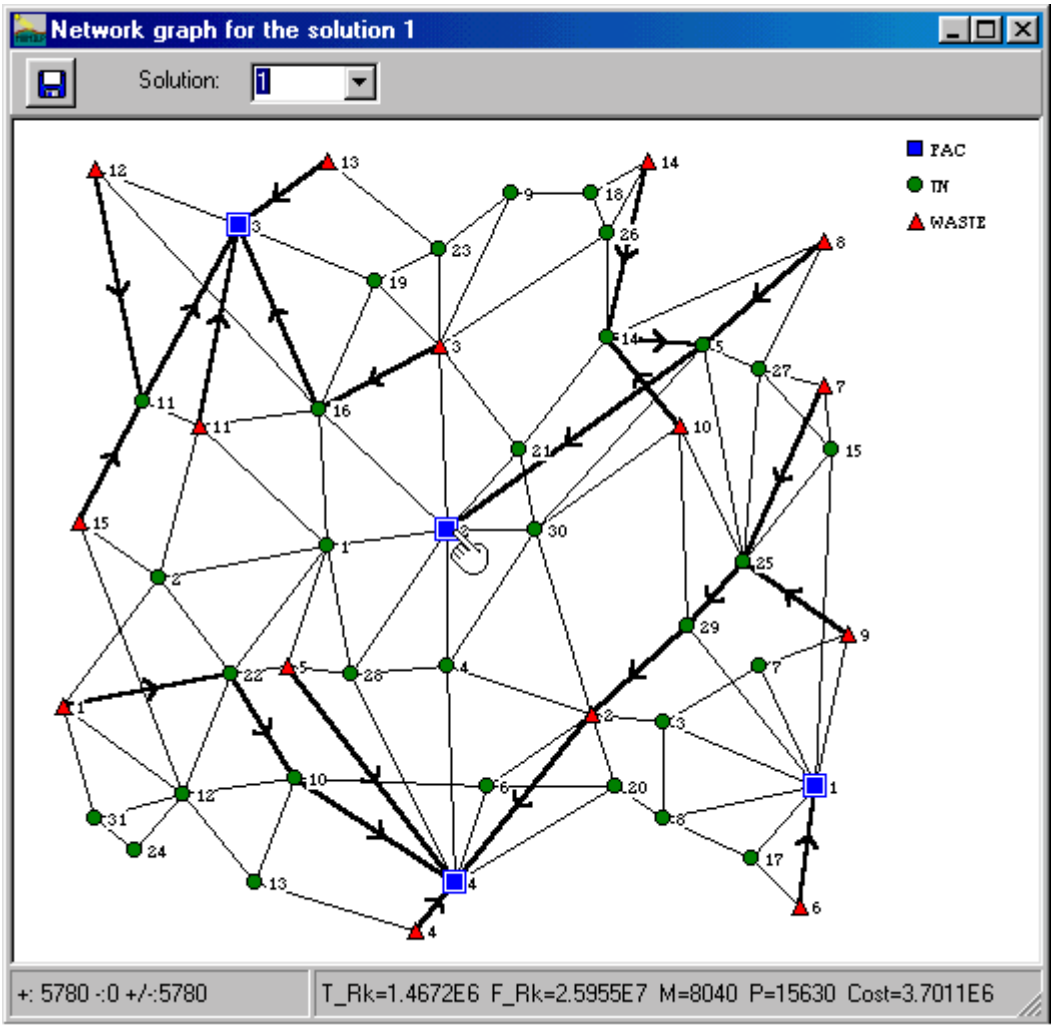


Fig. VI.1 – Rede da solução 1.

Observamos na figura VI.1 que todo o resíduo gerado em W_{12} (880) é enviado pelo percurso W_{12}–T_{11}–F_3, quando existe uma via de ligação directa W_{12}–F_3. Apesar de ser contra-intuitivo, este facto é explicado pela existência de um elevado número de população (a_{ij}) em W_{12}–F_3 o que condicionaria o valor da 1ª função objectivo, aquela que se pretendia minimizar nesta fase. A população em W_{12}–F_3 é representada pelo coeficiente 70 (ver dados no anexo VI.A) enquanto que em W_{12}–T_{11}–F_3 é de apenas 11+6=17, o que diminui Z_1 de 46 640 face à hipótese W_{12}–F_3. Contudo, o percurso adoptado sacrifica o custo em 31 116.8 relativamente a W_{12}–F_3.

Solução 2 – solução eficiente que minimiza Z_2 (risco total nas estações de tratamento).

Em contraste com a solução 1, a solução 2 considera a abertura de apenas uma estação de tratamento, F_4, que é a que se situa no local com menor densidade populacional. Minimiza assim o risco total nas estações (Z_2). Contudo, como é a única estação, os riscos individuais são muito elevados. O valor de Z_4 (maior exposição nalguma estação) é máximo (34650) visto que todo o resíduo é processado na mesma estação. Há, portanto, um grande desequilíbrio entre as populações sujeitas a risco. A população de F_4 é pouca, mas muito sacrificada. O custo total é menor do que na solução 1. A maior exposição individual durante o transporte (Z_3) é 14460 na ligação W_2-F_4. Nessa via passa o resíduo proveniente de 7 fontes (quase metade do total das fontes), o que representa cerca de 42% do total do resíduo. A figura VI.2 mostra as características espaciais desta solução.

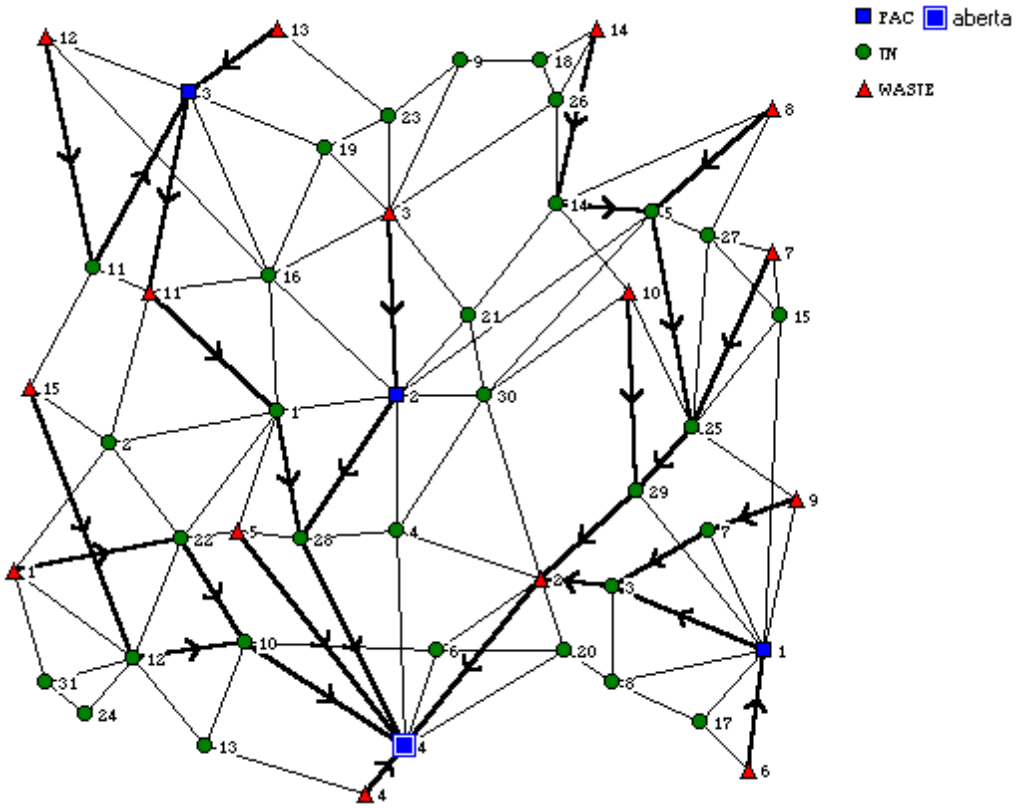


Fig. VI.2 – Rede da solução 2.

Solução 3 – solução eficiente que minimiza Z_3 (maior exposição individual durante o transporte).

Tal como a solução 1, também a solução 3 considera a abertura de todas as estações de tratamento, mas a rede de utilização de vias de transporte é agora muito densa (figura VI.3). Há muitas vias e nodos de passagem que são atravessados por baixas quantidades de resíduo. A maior quantidade é 1682.7 e atravessa 70% dos nodos de transporte activos (que, por sua vez, são 84% do número total dos nodos de transporte 'TN'), atravessando também todas as ligações W-F excepto W_{12}–F_3. A maior quantidade processada (Z_4) é 13462 em F_4 e a menor é 5252 em F_2 (para além do processamento, este nodo funciona também como ponto de passagem). O custo total (Z_5) é elevado.

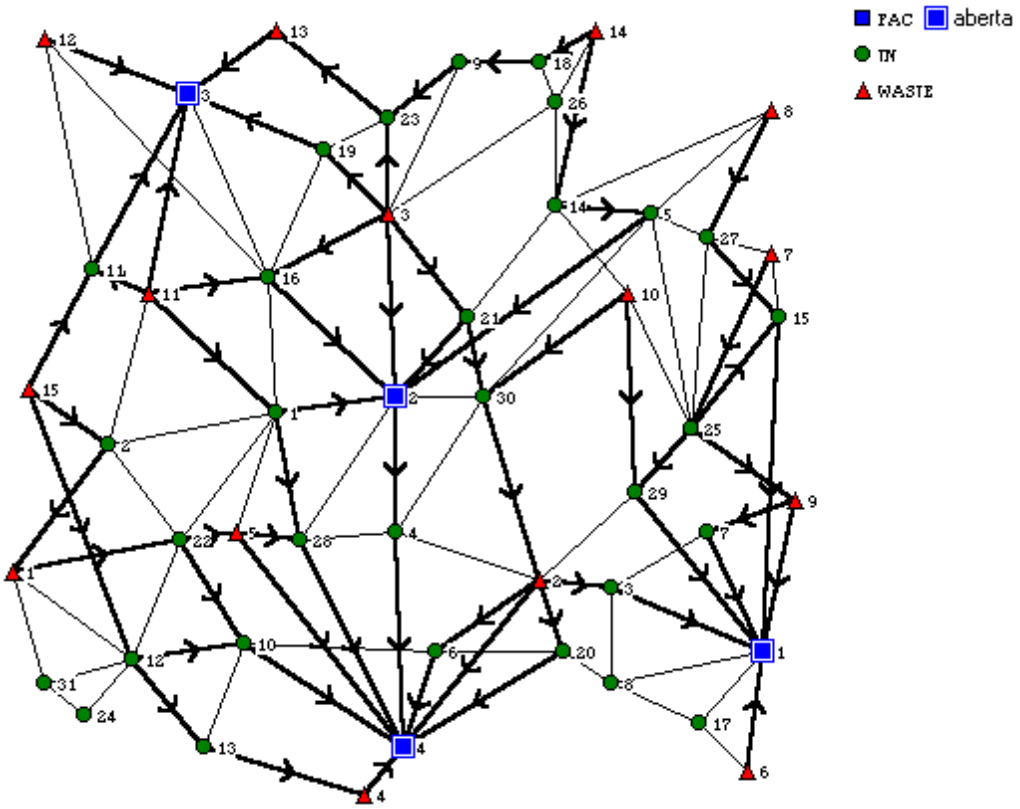


Fig. VI.3 – Rede da solução 3.

Solução 4 – solução eficiente que minimiza Z_4 (maior exposição individual devido ao processamento nas estações).

Tal como as soluções 1 e 3, também a solução 4 considera a abertura de todas as estações de tratamento (figura VI.4). A intenção é minimizar a maior quantidade processada em alguma estação e, portanto, a quantidade total de resíduo é dividida em partes iguais (8662.5) pelas 4 estações. Os valores resultantes para o risco total das populações nas estações (Z_2) e para o custo total (Z_5) são relativamente elevados. A maior exposição durante o transporte (Z_3) é 6370 e refere-se à ligação W_3–F_2. Observamos na rede da figura VI.4 que, tal como na solução 1, o resíduo de W_{12} é enviado primeiro para T_{11}, e deste para F_3, quando existe uma ligação directa W_{12}–F_3. Com esta configuração privilegia-se Z_1 em detrimento de Z_5 (custo), porque estas vias são menos populosas, mas têm custos de transporte mais elevados. A solução que apenas difere desta na utilização de W_{12}–F_3, em vez de W_{12}–T_{11}–F_3, é também eficiente e óptima alternativa (de entre muitas outras) para Z_4. Os valores de Z_2, Z_3 e Z_4 nessa solução são iguais aos da solução 4, mas Z_1=2 184 598 e Z_5=3 335 983.

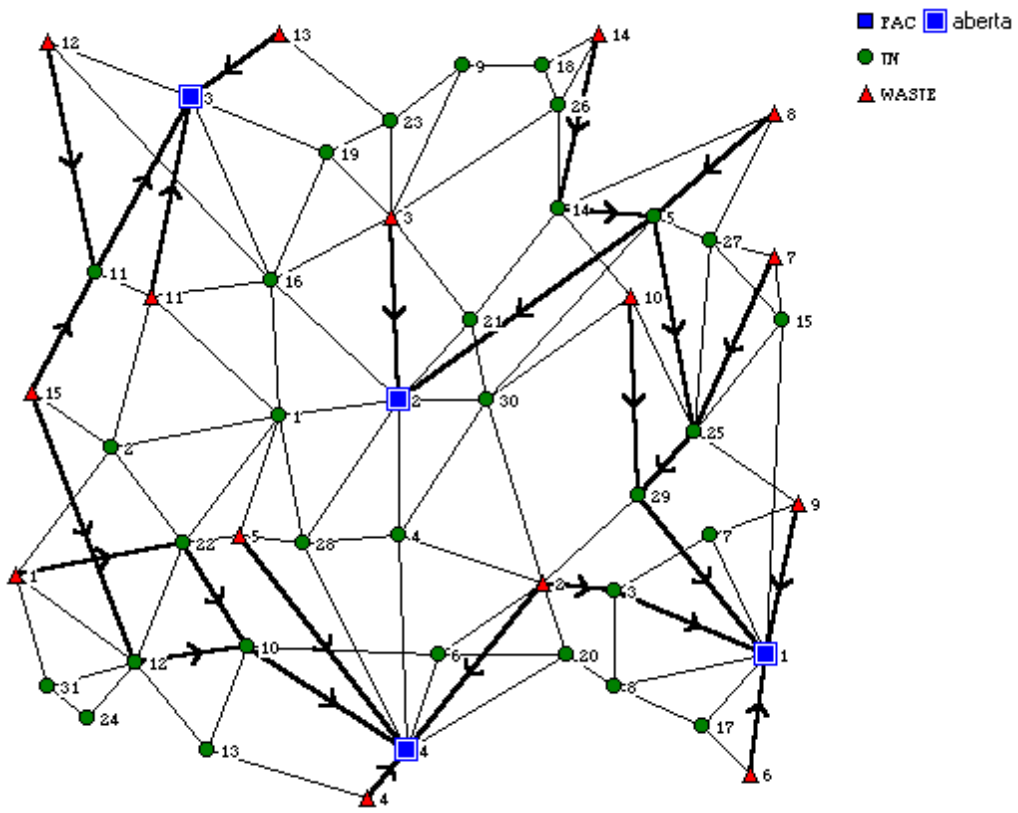


Fig. VI.4 – Rede da solução 4.

Solução 5 – solução eficiente que minimiza Z_5 (custo total).

O menor custo total é conseguido à custa da abertura de 2 estações de tratamento, F_1 e F_2 (figura VI.5). Todas as outras funções objectivo têm valores relativamente elevados, mas nenhuma delas atinge o maior valor conhecido até ao momento. A maior quantidade processada (Z_4) é 25410 em F_2 e a estação F_1 processa 9240 unidades. A maior exposição durante o transporte (Z_3) é 9410 na ligação W_3–F_2.

Observamos que, apesar de F_4 apresentar o menor custo de abertura, ela não é aberta. Um estudo mais aprofundado que fizemos revelou ainda que esta é a única solução eficiente que minimiza Z_5, pelo que não existe nenhuma outra configuração de custo mínimo que considere a abertura da estação F_4.

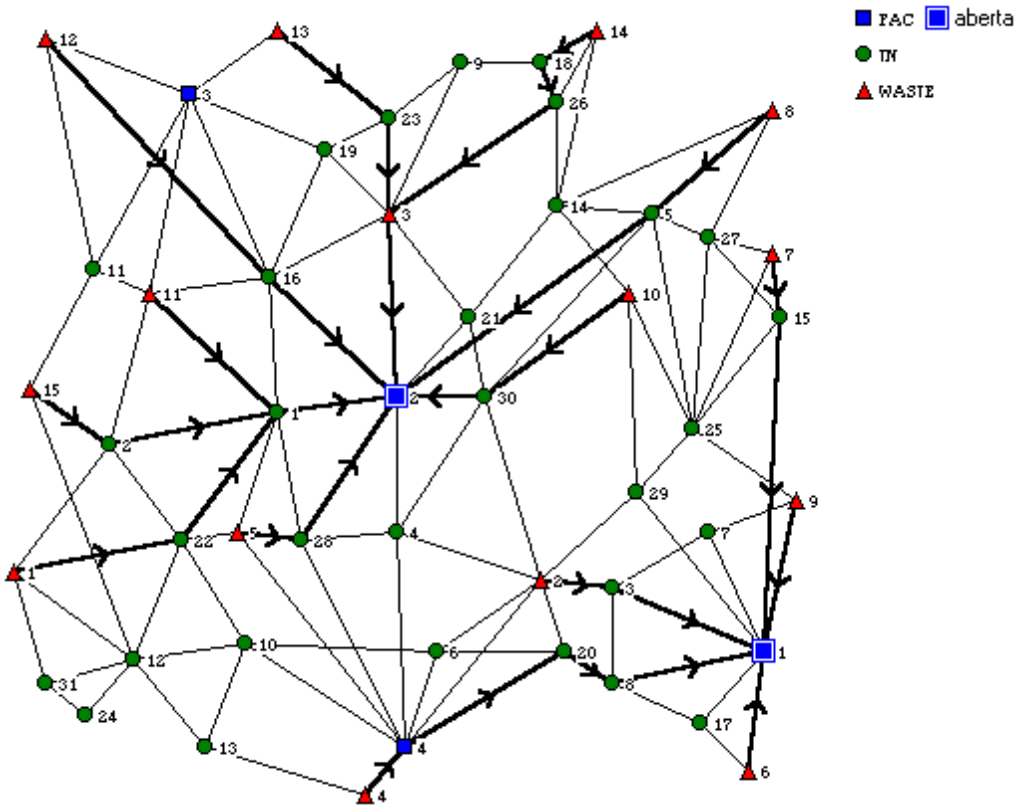


Fig. VI.5 – Rede da solução 5.

Conhecidas as 5 primeiras soluções, decidimos então calcular a solução não dominada que minimiza a distância (aumentada) de Tchebycheff ao ponto ideal z^*=(1 467 210, 15 592 500, 1 682.7, 8 662.5, 2 597 166), através da resolução do programa escalarizante $P_{z^+}^{\infty,\rho}$ com $z^+=z^*$. Obtivemos a solução 6 (ao fim de um tempo de cálculo de 1.15 seg num computador Pentium II a 350MHz). Os valores das funções objectivo da solução 6 estão na tabela VI.2.

A solução 6 é relativamente desequilibrada nos valores das funções objectivo; Z_1 é moderado e Z_2 bastante baixo, mas o valor de Z_3 é o maior encontrado até ao momento, e é quase o dobro

do maior das soluções anteriores; também Z_4 e Z_5 são próximos dos máximos conhecidos. Esta solução considera a abertura das estações F_2 e F_4. Em F_4 são processadas 31085 unidades (Z_4), das quais 24275 (Z_3) passam na ligação W_2–F_4.

Devemos referir que não houve qualquer normalização prévia das funções objectivo e, como estas têm escalas diferentes, houve nesta solução um "privilégio" de Z_2 em detrimento das outras funções objectivo, visto que é Z_2 a função objectivo com maior ordem de grandeza de valores.

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 6	2 536 719	16 662 009	*24 275*	31 085	3 666 675

Tabela VI.2 – Solução que minimiza $P_{z^+}^{\infty,\rho}$ *com* z^+*=(1 467 210, 15 592 500, 1 682.7, 8 662.5, 2 597 166).*

A figura VI.6 mostra um gráfico de barras com as 6 soluções calculadas. O sistema usa a mesma cor para as barras das soluções com partes inteiras iguais. Neste problema, são as soluções que consideram a abertura das mesmas estações de tratamento, formando os seguintes grupos: (sol. 1, sol. 3, sol. 4), (sol. 2), (sol. 5) e (sol. 6). As opções de visualização disponíveis nesta janela são:  – gráfico de barras;  – gráfico de linhas;  – pontos no espaço dos critérios (apenas disponível para 2 ou 3 funções objectivo);  –opção disponível para o gráfico de barras, e que permite ordenar por ordem crescente ou decrescente do índice da solução (como na figura VI.6) ou do valor de uma das funções objectivo.

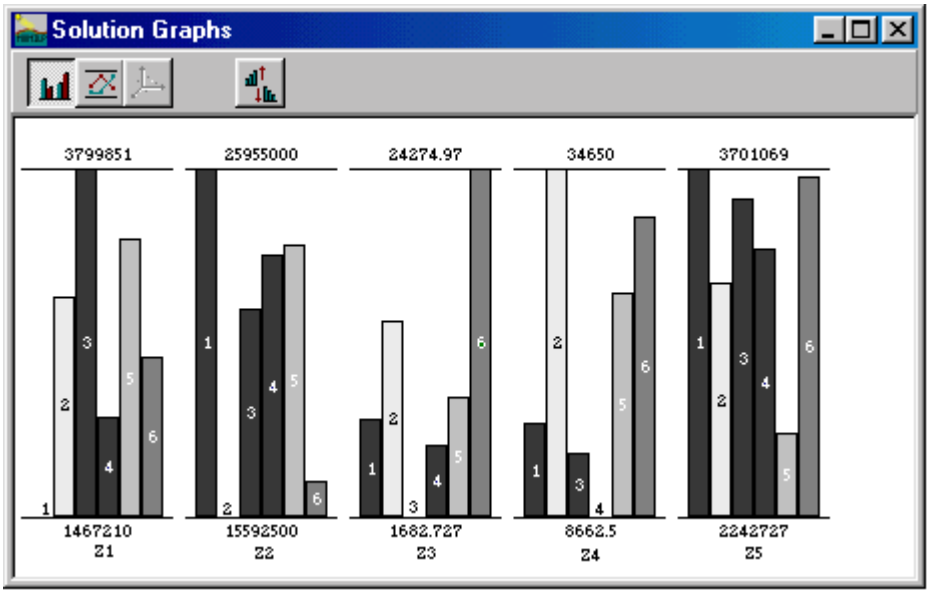


Fig. VI.6 – Gráfico de barras das primeiras 6 soluções.

Sempre que é calculada uma nova solução eficiente, ela é apresentada ao utilizador na janela principal do sistema (como, por exemplo, a da figura VI.9). Esta janela inclui um gráfico de

barras com os valores dos critérios da(s) última(s) solução(ões) calculada(s), informação numérica e um painel com as principais opções. As opções dependem do tipo de cálculo efectuado (com pontos de referência ou pesos).

Os diagramas das figuras VI.7 e VI.8 fazem uma breve descrição das opções de interacção com o utilizador disponíveis no método de pontos de referência para PLIMMO. Estas opções distribuem-se em 2 separadores do painel: '*Standard*' (figura VI.7) e '*View Options*' (figura VI.8). As opções assinaladas com * chamam uma caixa de diálogo para a entrada de parâmetros.

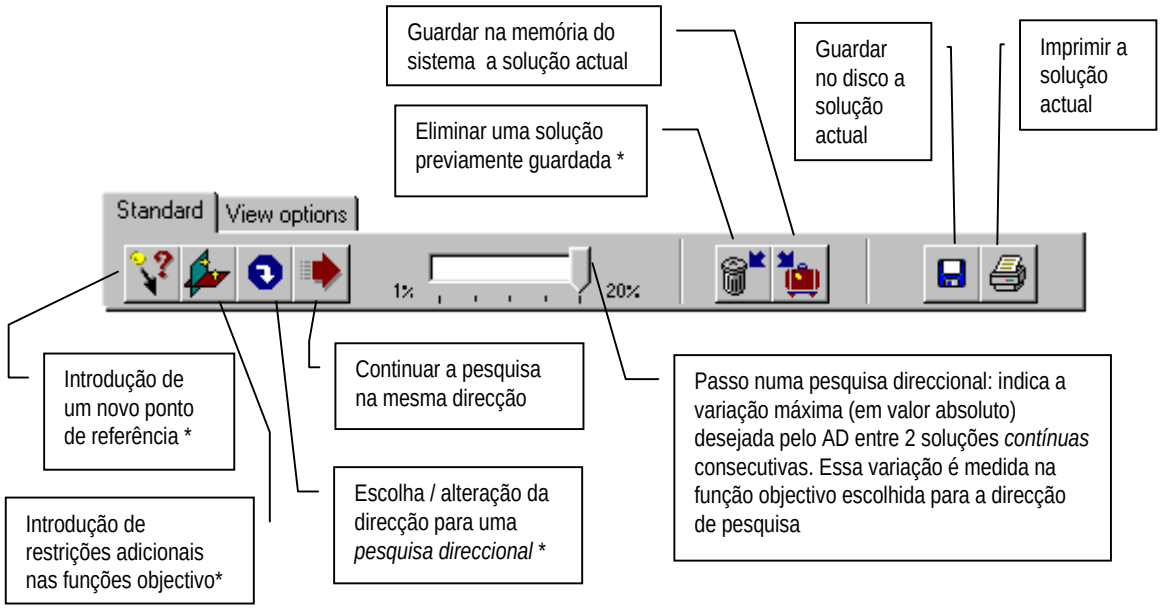


Fig. VI.7 – Painel com as opções principais de pesquisa.

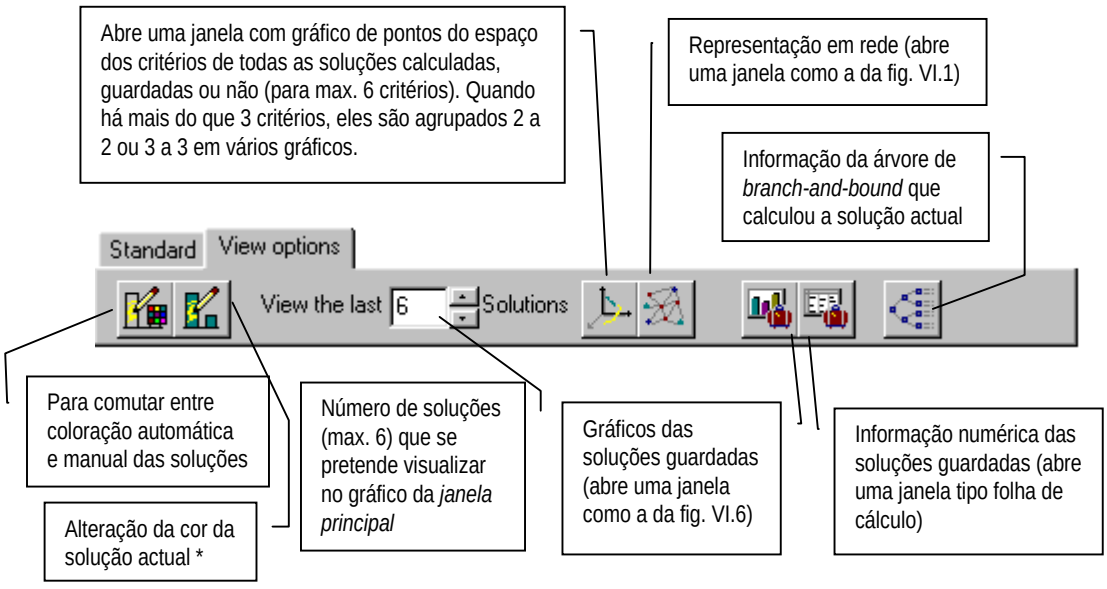


Fig. VI.8 – Painel com as opções principais de visualização.

Continuando com o estudo do problema, decidimos fazer uma *pesquisa direccional* a partir do ponto de referência ideal z^* (que conduziu à solução 6), escolhendo a 3ª função objectivo para melhorar (aquela que julgamos ser a mais sacrificada na solução 6).

A escolha de uma *pesquisa direccional* faz-se através da selecção da opção  do painel de pesquisa que, por sua vez, abre uma caixa de diálogo para introdução da função objectivo que se pretende melhorar. O sistema procede ao cálculo de uma nova solução eficiente ao longo da mesma direcção sempre que for premido o botão  (nova interacção). Ao longo de uma pesquisa direccional, o método actualiza sucessivamente o ponto de referência z^+ e calcula a solução não dominada mais próxima de cada ponto de referência dessa trajectória (segundo a métrica aumentada de Tchebycheff).

No nosso caso, o ponto de referência era inicialmente igual a z^* e a sua alteração consistiu em diminuir a 3ª componente (porque Z_3 é a minimizar), mantendo iguais as outras componentes. O método começou por calcular soluções com valores sucessivamente maiores em Z_1, Z_2 e Z_5 e menores em Z_3 e Z_4. As primeiras soluções são 'contínuas' e consideram a abertura das mesmas estações (F_2 e F_4). A dado momento (6ª interacção), houve alteração das estações abertas, que passaram a ser F_1, F_2 e F_4, mas mantiveram-se as tendências de variação das funções objectivo. A última solução (a mais escuro no gráfico da figura VI.9) é Z=(2 573 156, 16 698 446, 19 284, 30 430, 3 703 112).

A ferramenta de pesquisa aqui utilizada evidencia que, grandes variações do ponto de referência z^+, nem sempre correspondem a grandes variações nos valores das funções objectivo. Apesar de o AD se poder alhear, se assim o desejar, das variações automáticas de z^+, uma análise técnica deste processo permite também aumentar o conhecimento geral do problema. A análise de sensibilidade ao ponto de referência tenta dar "pequenos passos" de modo a que haja um varrimento total ao longo da direcção de pesquisa. Mas, por vezes é necessário um grande avanço do ponto de referência para "escapar" da solução anterior, o que não implica necessariamente variações bruscas nas funções objectivo. Foi o que aconteceu na 1ª interacção desta pesquisa em que a 3ª componente de z^+ (z_3^+) baixou logo de 1682.7 para –1 046 281 (variação de –62 279 %). Em cada uma das interacções seguintes, a variação de z_3^+ situou-se entre -9×10^{-5} % e -2.2 %. As variações dos valores das funções objectivo podem ser avaliadas pelo gráfico e valores finais na figura VI.9.

Nesta pesquisa calcularam-se 6 soluções em apenas 0.81 seg, o que representa 70% do tempo de cálculo da solução de partida. O tempo despendido no cálculo de cada solução variou entre 0.06 e 0.27 seg. Cada um destes valores inclui o tempo da análise de sensibilidade e o da

actualização da árvore de *branch-and-bound* que determina cada nova solução. A figura VI.9 mostra o tempo relativo à última solução (0.16 seg).

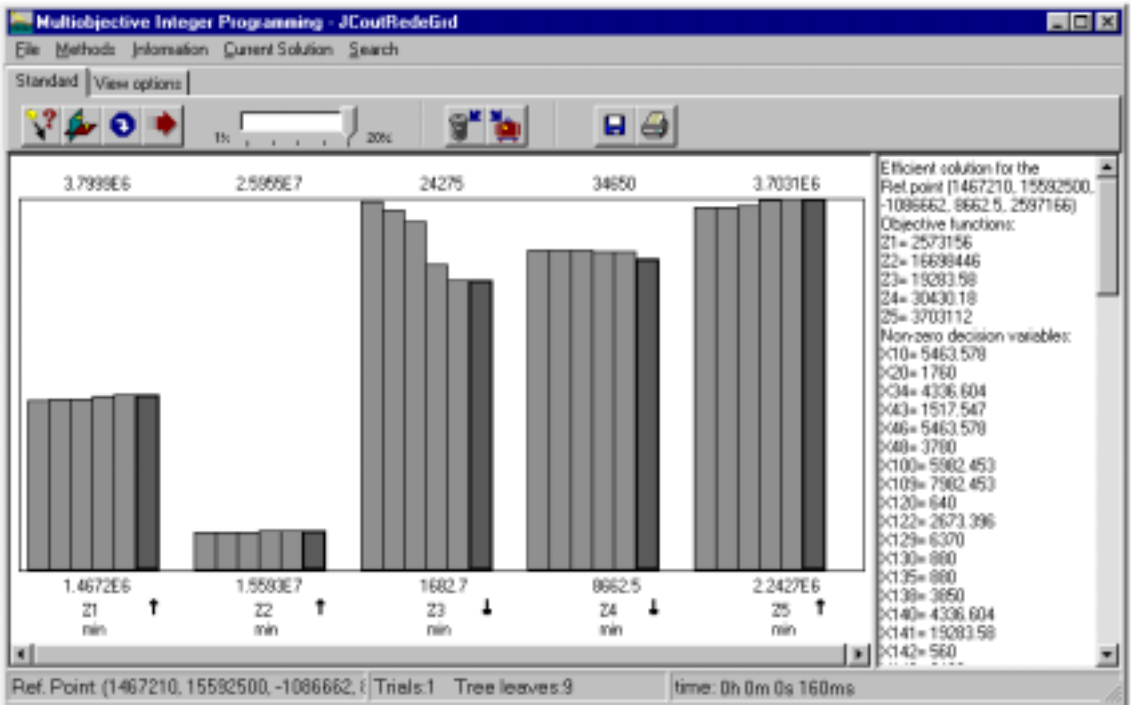


Fig. VI.9 – Primeira pesquisa direccional (diminuição da 3ª função objectivo).

Como o custo (Z_5) das soluções obtidas era muito elevado, optámos por alterar a direcção de pesquisa com vista à diminuição do custo. Como consequência, inverteu-se inicialmente o sentido da variação de Z_3 que começou a aumentar, retomando a situação de abertura das estações F_2 e F_4. Passadas algumas soluções (6ª interacção nesta direcção), alterou-se a tendência de variação de Z_3, que diminuiu bruscamente. Simultaneamente, Z_5 diminuiu também bastante e Z_4 diminuiu ligeiramente. Apenas Z_1 e Z_2 aumentaram ligeiramente (figura VI.10). Manteve-se a situação de abertura de F_2 e F_4. Decidimos guardar esta solução – solução 7 (tabela VI.3).

Durante esta pesquisa direccional, a maior variação do ponto de referência ocorreu na última interacção, em que z_5^+ baixou de 2 458 002 para 2 175 695 (variação de –11.5%).

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 7	2 661 806	16 787 096	16 837	30 668	3 370 291

Tabela VI.3 – Solução que minimiza $P_{z^+}^{\infty,\rho}$ com z^+=(1 467 210, 15 592 500, –1 086 662, 8 662.5, 2 175 695).

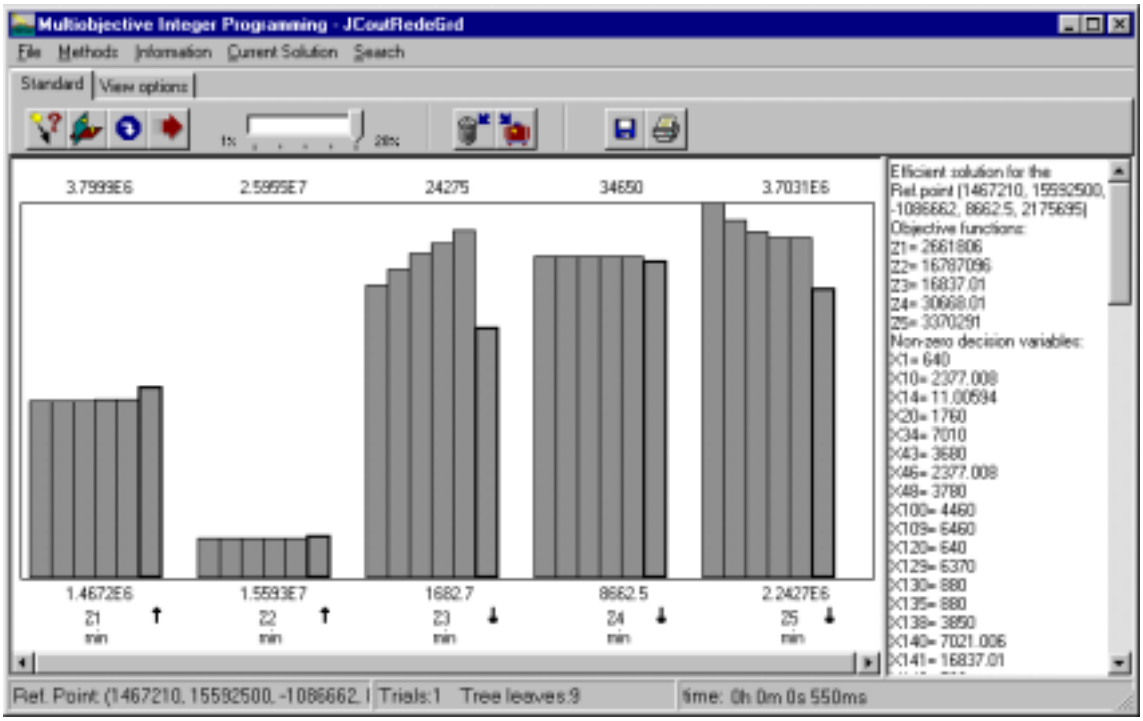


Fig. VI.10 – Segunda pesquisa direccional (diminuição da 5ª função objectivo).

Continuando a pesquisa segundo a mesma direcção, Z_3 diminuiu logo a seguir para 14 460 e Z_5 para 3 337 144. Mantiveram-se os sentidos da variação dos outros objectivos que, durante algumas interacções, sofreram variações muito ligeiras: Z_1 e Z_2 aumentaram; Z_4 e Z_5 diminuíram; Z_3 manteve-se a 14460 durante 5 interacções, descendo depois para 13820, onde se manteve durante mais 5 interacções. Na interacção seguinte, houve um "salto abrupto" para uma solução com valores "extremos". A figura VI.11 mostra esta solução e as 5 anteriores. A última solução é semelhante à solução 2, em que se considera a abertura de apenas a estação F_4. Esta solução foi obtida a partir de uma variação de apenas -0.5 % em z_5^+, relativamente ao respectivo valor da interacção anterior. Z_4 atingiu o máximo conhecido até ao momento e Z_2 o mínimo. Esta solução optimiza Z_2, só diferindo da solução 2 em Z_1 (ligeiramente maior) e Z_5 (ligeiramente menor). Resolvemos abandonar esta direcção de pesquisa.

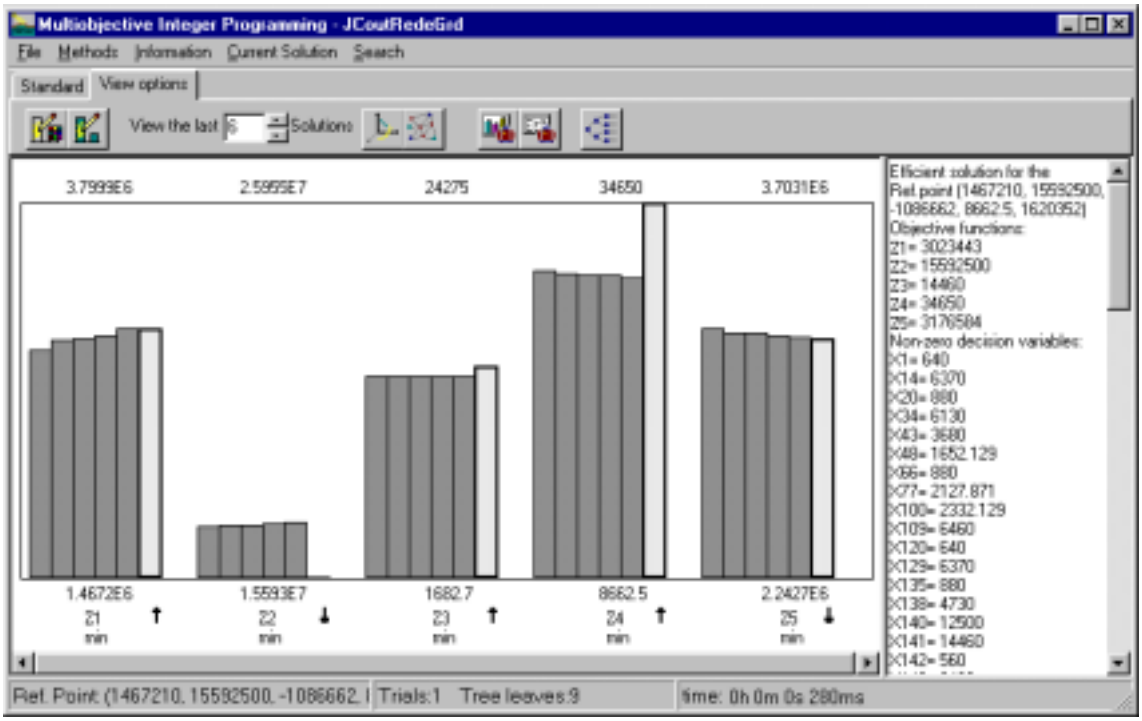


Fig. VI.11 – Continuação da pesquisa direccional anterior (diminuição da 5ª função objectivo).

Face ao desagrado pela última solução, e admitindo que Z_2 poderia ser sacrificada de modo a melhorar as outras funções objectivo, decidimos indicar explicitamente um novo ponto de referência: $z^+=(2\times10^6,\ 20\times10^6,\ 10\ 000,\ 15\ 000,\ 3\times10^6)$. Obtivemos uma solução relativamente equilibrada, designada por solução 8 na tabela VI.4. Comparando-a com a última solução da pesquisa anterior, é melhor em todas as funções objectivo excepto em Z_2 (ver figura VI.12). Considera a abertura das estações F_2 e F_4. O tempo de cálculo desta solução foi de 1.16 seg.

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 8	2 028 630	20 028 630	13 053	19 863	3 028 630

Tabela VI.4 – Solução que minimiza $P_{z^+}^{\infty,\rho}$ *com* $z^+=(2\times10^6, 20\times10^6, 10\ 000, 15\ 000, 3\times10^6)$.

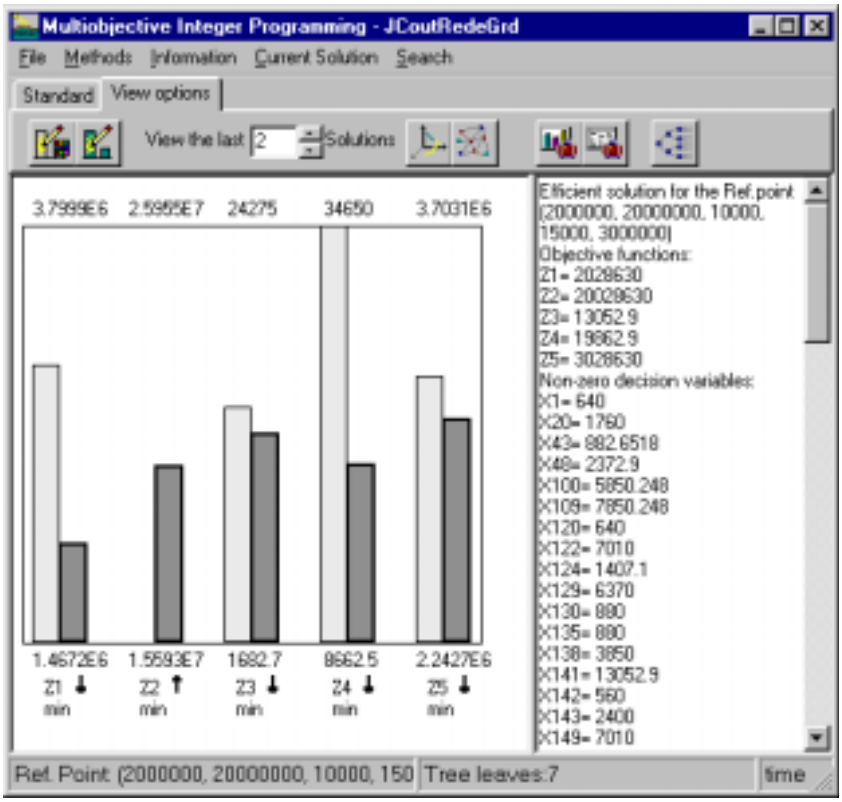


Fig. VI.12 – Solução 8 (a mais escuro) e a solução anterior.

Partindo desta solução, procedemos a uma nova pesquisa direccional no sentido de melhorar Z_2. As soluções seguintes sacrificam todas as outras funções objectivo para melhorar Z_2. Perante Z =(2 183 084, 18 944 650, 16 666, 23 476, 3 183 084), 4ª solução desta pesquisa, resolvemos impor a limitação adicional $Z_2 \leq 19\times10^6$ (através da opção ) e escolher novamente Z_3 para melhorar: Z_2 atingiu exactamente o valor de 19×10^6, Z_3 diminuiu e Z_1 e Z_5 aumentaram; Z_4 diminuiu ligeiramente. Passadas algumas soluções com variações muito pequenas, foi encontrada uma solução que diminuiu significativamente o valor de Z_4, com pequenas variações nos outros objectivos relativamente à solução imediatamente anterior. Esta solução considera a abertura das estações F_1, F_2 e F_4 – solução 9 na tabela VI.5.

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 9	2 253 931	19 000 000	12 321	21 509	3 253 931

Tabela VI.5 – Solução que minimiza $P_{z^+}^{\infty,p}$ *com* $z^+=(2\times10^6, 18\,761\,566, -241\,610, 15\,000, 3\times10^6)$.

Continuando a pesquisa na mesma direcção (de melhoramento de Z_3), observámos que Z_2 se manteve igual, Z_1 e Z_5 aumentaram e Z_3 e Z_4 diminuíram. Em virtude de o custo total se tornar bastante elevado, resolvemos escolher Z_5 para melhorar, adicionando a limitação $Z_3 \leq 9000$ para obrigar Z_3 a não se tornar demasiado elevado. Partimos do ponto de referência da interacção

anterior, ou seja, (2×10^6, 18 761 566, -334 898.3, 15 000, 3×10^6). A solução não dominada mais próxima desse ponto de referência e que satisfaz as duas limitações adicionais é a solução 10 (tabela VI.6).

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 10	2 346 793	19 000 000	9 000	21 892	3 346 793

Tabela VI.6 – Solução que minimiza $P_{z^+}^{\infty,\rho}$ *com* $z^+=(2\times10^6, 18\ 761\ 566, -334\ 898.3, 15\ 000, 3\times10^6)$.

Prosseguindo então no sentido de melhorar Z_5, obtivemos inicialmente soluções com valores constantes em Z_2, Z_3 (os das respectivas limitações) e Z_4 (20 225), crescentes em Z_1 e decrescentes em Z_5. Passadas algumas soluções (7 interacções), houve um aumento significativo em Z_4 para 23 291, sem grandes contrapartidas nas outras funções objectivo. Já não se propunha a abertura da estação F_1, mas apenas de F_2 e F_4. Esta solução pareceu-nos desinteressante face às anteriores (penúltima solução na figura VI.13).

Face a esta solução, decidimos impor uma terceira limitação adicional, desta vez em Z_4: $Z_4 \leq 20\ 000$. A solução não dominada mais próxima do ponto de referência anterior, (2×10^6, 18 761 566, -334 898.3, 15 000, 2 767 193), que satisfaz as três limitações adicionais, é a solução 11 (tabela VI.7 e figura VI.13).

Esta solução (que julgamos apresentar um compromisso razoável entre os vários critérios), tem as seguintes características (ver rede na figura VI.14): são abertas as estações de tratamento F_1, F_2 e F_4 que processam, respectivamente, 3950, 10700 e 20000 unidades; a quantidade máxima processada é, pois, de 20000, indicador do maior risco individual em estações de tratamento (Z_4); a maior exposição ao resíduo durante o transporte (Z_3) é 9000 na ligação W_2–F_4; o local da estação F_3 funciona como ponto de passagem de 1008 unidades. Observamos, nesta rede, que é utilizado o percurso T_{11}–F_3–W_{11}, quando existe uma ligação directa T_{11}–W_{11}. É uma situação contra-intuitiva que, tal como outras já observadas anteriormente (por exemplo nas soluções 1 e 4), relaciona-se com o compromisso *risco total do transporte* (Z_1)/*custo total* (Z_5). O custo de transporte por unidade de resíduo em T_{11}–F_3–W_{11} é 50.56 (25.06+25.5), enquanto que em T_{11}–W_{11} é 7.62. Por outro lado, a população total sujeita a risco em T_{11}–F_3–W_{11} é representada pelo coeficiente 14, enquanto que em T_{11}–W_{11} é 90.

	Z_1 (min)	Z_2 (min)	Z_3 (min)	Z_4 (min)	Z_5 (min)
Solução 11	2 415 817	19 000 000	9 000	20 000	3 183 010

Tabela VI.7 – Solução que minimiza $P_{z^+}^{\infty,\rho}$ *com* $z^+=(2\times10^6, 18\ 761\ 566, -334\ 898.3, 15\ 000, 2\ 767\ 193)$.

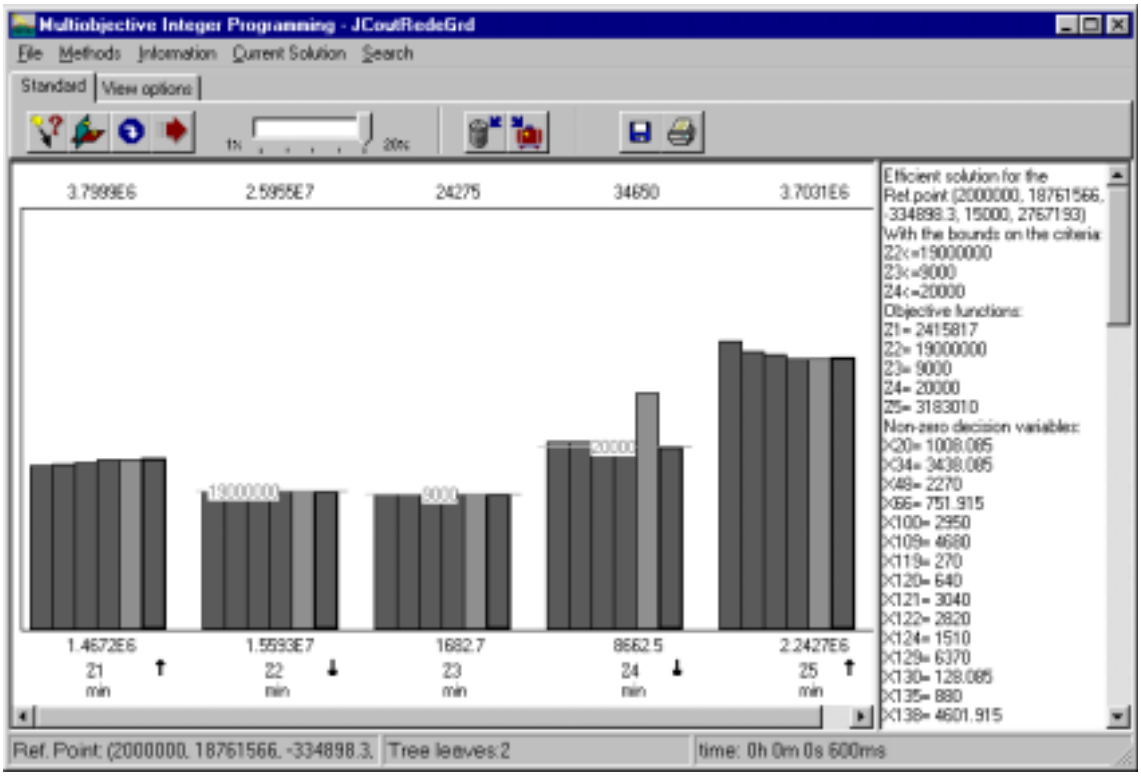


Fig. VI.13 – Solução 11 (última barra de cada coluna) e a pesquisa que antecedeu o cálculo desta solução.

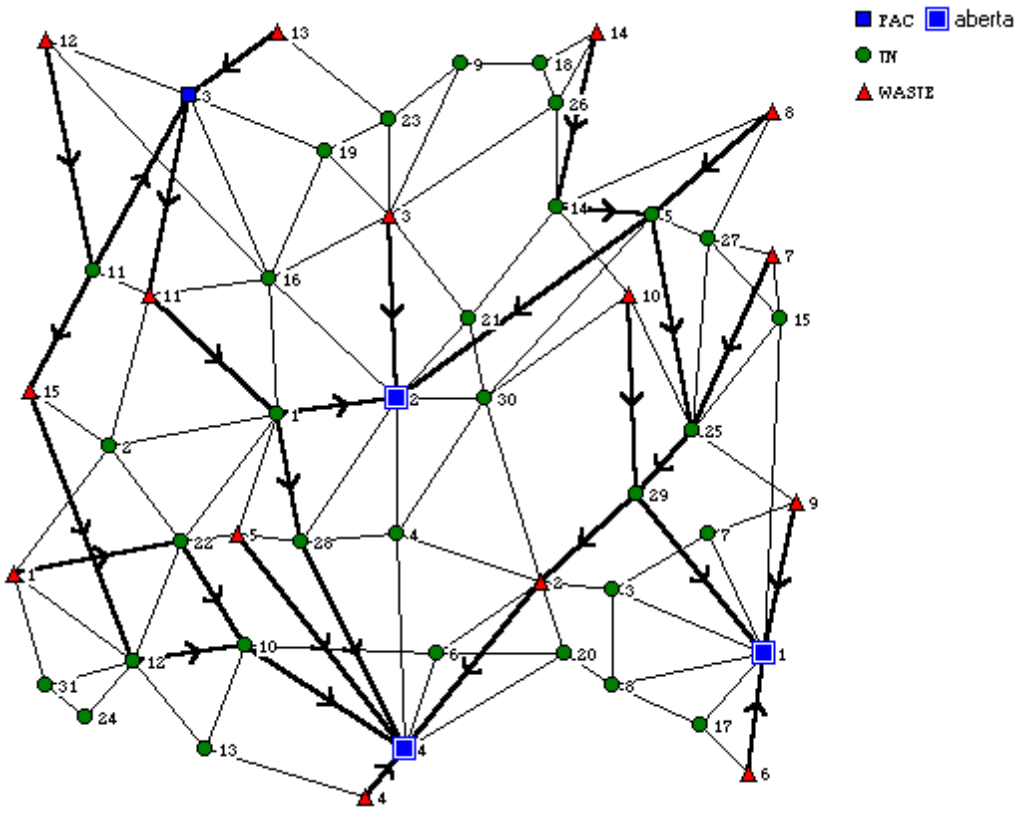


Fig. VI.14 – Rede da solução 11.

A tabela VI.8 e a figura VI.15 resumem o estudo efectuado para este problema, mostrando os valores e um gráfico de barras das 11 soluções não dominadas guardadas.

	Estações abertas	**Z_1 (min)**	**Z_2 (min)**	**Z_3 (min)**	**Z_4 (min)**	**Z_5 (min)**
Solução 1	**1, 2, 3, 4**	**1 467 210**	*25 955 000*	8 040	15 630	*3 701 069*
Solução 2	**4**	2 939 590	**15 592 500**	14 460	*34 650*	3 223 054
Solução 3	**1, 2, 3, 4**	*3 799 851*	21 770 318	**1 682.7**	13 462	3 579 522
Solução 4	**1, 2, 3, 4**	2 137 958	23 388 750	6 370	**8 662.5**	3 367 100
Solução 5	**1, 2**	3 324 450	23 677 500	9 410	25 410	**2 597 166**
Solução 6	**2, 4**	2 536 719	16 662 009	*24 275*	31 085	3 666 675
Solução 7	**2, 4**	2 661 806	16 787 096	16 837	30 668	3 370 291
Solução 8	**2, 4**	2 028 630	20 028 630	13 053	19 863	3 028 630
Solução 9	**1, 2, 4**	2 253 931	19 000 000	12 321	21 509	3 253 931
Solução 10	**1, 2, 4**	2 346 793	19 000 000	9 000	21 892	3 346 793
Solução 11	**1, 2, 4**	2 415 817	19 000 000	9 000	20 000	3 183 010

Tabela VI.8 – Tabela de valores das 11 soluções não dominadas guardadas.

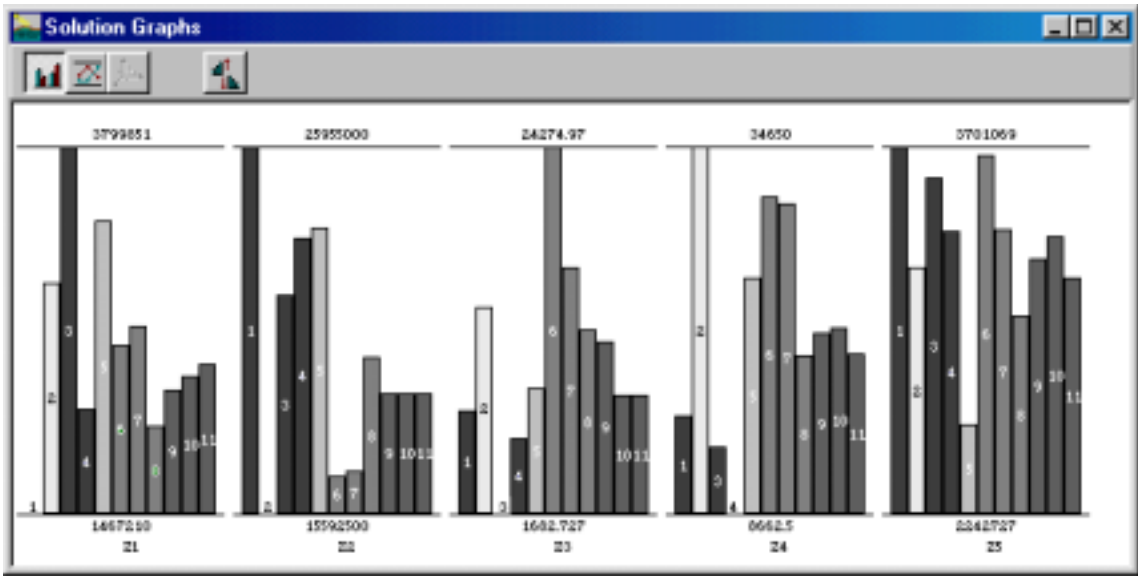


Fig. VI.15 – Gráfico de barras das 11 soluções não dominadas guardadas.

Fixando as funções objectivo de equidade de risco (Z_3 e Z_4) em 9000 e 20000, respectivamente, fizemos ainda uma pesquisa local na vizinhança da solução 11, considerando apenas os critérios Z_1, Z_2 e Z_5. A figura VI.16 mostra os pontos no espaço dos objectivos referentes às soluções pesquisadas. A solução de partida (solução 11) está representada por 'A'. Primeiro explorámos soluções que melhorassem Z_5, até à solução 'B' (Z_1=2 882 798, Z_2=19 466 981 e Z_5=2 989 890) e, em seguida, soluções que melhorassem Z_1, até à solução 'C' (Z_1=2 274 689, Z_2=19 606 441, Z_5=3 129 351). É de notar que o gráfico da figura VI.16 utiliza

escalas semelhantes às dos gráficos anteriores o que, apesar de dificultar a visualização, permite situar melhor no espaço esta pesquisa local. A figura VI.17 esclarece a evolução dos valores destes 3 objectivos através de um gráfico de barras. Todas estas soluções consideram a abertura das mesmas estações de tratamento de resíduo, F_1, F_2 e F_4.

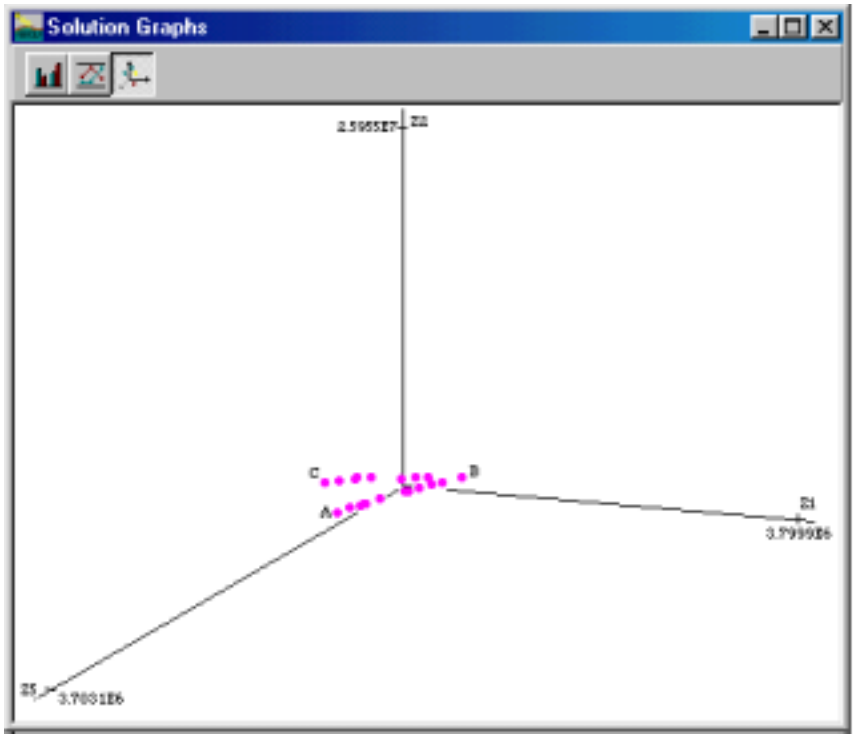


Fig. VI.16 – Espaço dos objectivos numa pesquisa local com Z_3 e Z_4 fixos.

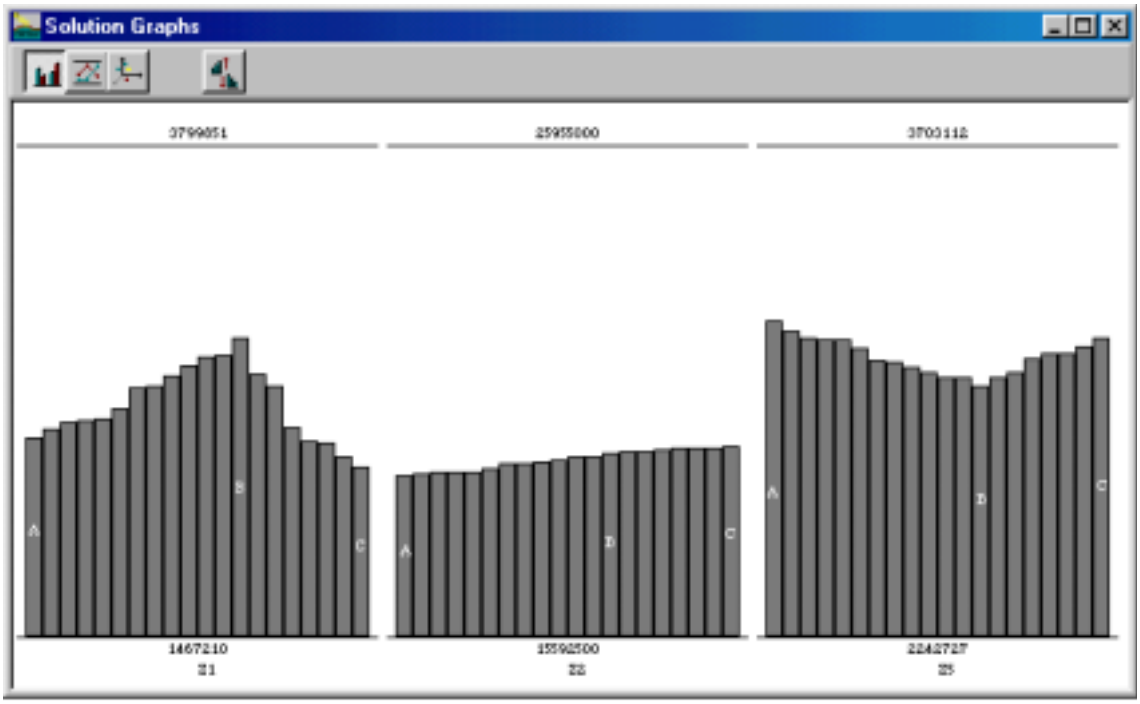


Fig. VI.17 – Gráfico de barras duma pesquisa local com Z_3 e Z_4 fixos.

VI.3 CONCLUSÕES

Neste capítulo descrevemos uma aplicação do método interactivo que desenvolvemos para PLIMMO a um problema multiobjectivo de localização-transporte. Apresentámos alguns resultados (parciais) ilustrativos do estudo efectuado para o problema. Desse estudo, ressaltam as seguintes características do problema.

As funções objectivo Z_2, Z_3 e Z_4 admitem várias soluções óptimas alternativas (eficientes e também fracamente eficientes) quando optimizadas individualmente. Assim, as soluções 2, 3 e 4 não são as únicas soluções eficientes que atingem o valor mínimo para a respectiva função objectivo. O valor máximo de Z_3 da tabela de *pay-off* obtida (tabela VI.1) está muito aquém do máximo eficiente desse critério (vejamos, por exemplo, o valor de Z_3 na solução 6), pelo que é grosseira a percepção inicial da gama de valores eficientes dada por esta tabela. Apercebemo-nos de que são muitas as soluções eficientes junto aos óptimos individuais de Z_2, Z_3 e Z_4, não só as que atingem o mínimo nessas funções objectivo, como também soluções próximas destas. Isso implica que seja necessário um cuidado especial na selecção de pesos para calcular os óptimos individuais. Por um lado, não devem ser usados pesos nulos, porque podem conduzir a soluções fracamente eficientes (que também são muitas); por outro lado, quando se pretende optimizar Z_3 e se lhe atribui um peso de 0.999, basta a consideração de pesos 0.00025 nas outras funções para que não seja atingido o mínimo de Z_3. Estas dificuldades podem ser atenuadas por uma normalização prévia das funções objectivo, o que não aconteceu neste estudo. O efeito da normalização faz-se também sentir no cálculo da solução mais próxima de um ponto de referência segundo a métrica de Tchebycheff. Podemos recordar, por exemplo, o caso da solução 6, que é a solução mais próxima do ponto ideal, mas que tem poucas características de solução "central". A normalização já não é tão relevante para a continuação do cálculo de soluções durante uma pesquisa direccional, visto que o método determina automaticamente a variação necessária do ponto de referência.

As pesquisas direccionais evidenciaram que há soluções eficientes com variações muito pequenas em 4 das 5 funções objectivo e uma variação significativa na restante (positiva ou negativa). Comparemos, por exemplo, a solução 11 com a última solução da pesquisa direccional anterior (figura VI.13). Embora estas soluções sejam todas eficientes, haverá certamente algumas delas que apresentam um compromisso entre os critérios mais facilmente aceitável.

Na nossa opinião, a formulação do problema (apresentada na secção 1) poderia incluir o grupo de restrições $\sum_i X_{ji} \leq M,\ \forall j \in \mathrm{F}$, já que os nodos de estações de tratamento são também pontos de passagem. $\sum_i X_{ji}$ representa todo o resíduo que passa no nodo $j \in \mathrm{F}$ que não é lá

processado. As soluções eficientes apresentadas na secção anterior não seriam afectadas por estas restrições porque o maior valor de resíduo transportado (M) atravessa sempre outros nodos ou arcos. Contudo, existem outras soluções eficientes do problema onde estas restrições têm influência. A título de exemplo, referimos que o problema admite uma solução eficiente em que M=9817, mas no nodo relativo à estação 2 (fechada) passam 10860 unidades que não são contabilizadas para M.

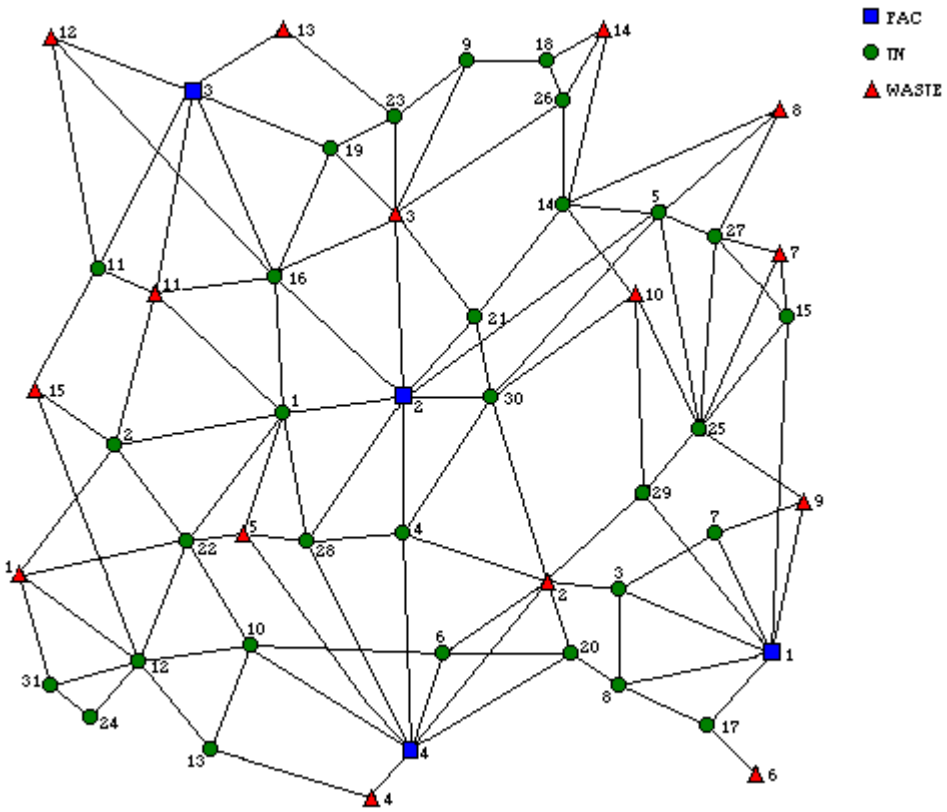
A abordagem interactiva usada neste trabalho consiste fundamentalmente em pesquisas direccionais, para as quais se podem considerar limitações adicionais nos valores das funções objectivo. As principais vantagens desta abordagem residem no cálculo sucessivo de soluções não dominadas – o que dá uma percepção das tendências de variação dos critérios ao longo de determinadas trajectórias – e na rapidez desse cálculo. Estas vantagens resultam do facto de se aproveitar a informação da fase de cálculo anterior para obter a solução seguinte. Um outro processo implicaria a resolução independente de um problema escalarizante para determinar uma nova solução. Salientamos ainda o interesse da actualização automática do ponto de referência, que permite que o AD se possa demitir desta tarefa. Julgamos que a indicação explícita de pontos de referência não é, em geral, uma opção fácil, sobretudo porque pode conduzir a resultados muito diferentes das expectativas do AD – sentimos essa dificuldade em particular neste problema.

A abordagem que utilizámos é substancialmente diferente da proposta por COUTINHO-RODRIGUES ET AL. (1995, 1997), e que foi experimentada anteriormente no tratamento do mesmo problema. As diferenças tornam difícil qualquer comparação de tácticas ou resultados. Julgamos, porém, que as duas abordagens podem ser consideradas complementares.

ANEXO VI.A

Dados do problema de localização-transporte

```
#N= 50       (FAC 1...FAC 4; WASTE 1...WASTE 15; TN 1...TN 31)
#A= 113x2 =226
```



Junto a cada nodo da rede é indicado o respectivo índice, dentro da sua categoria. FAC – estações de tratamento; TN – nodos de transporte (passagem); WASTE – fontes de resíduo.

c_{ij}	a_{ij}				
16.12	72	FAC 1-TN 3	13.6	59	FAC 3-WASTE 13
15.13	78	FAC 1-TN 7	27.02	78	FAC 4-TN 4
14.56	65	FAC 1-TN 8	15	34	FAC 4-TN 6
42.05	78	FAC 1-TN 15	23.85	7	FAC 4-TN 10
9.49	27	FAC 1-TN 17	23.32	60	FAC 4-TN 20
22.83	94	FAC 1-TN 29	29.07	28	FAC 4-TN 28
15.13	30	FAC 1-WASTE 6	27.02	0	FAC 4-WASTE 2
19.1	87	FAC 1-WASTE 9	7.81	81	FAC 4-WASTE 4
15.13	0	FAC 2-TN 1	34.21	69	FAC 4-WASTE 5
17	48	FAC 2-TN 4	21.38	89	TN 1-TN 2
39.41	32	FAC 2-TN 5	17.03	95	TN 1-TN 16
21.93	92	FAC 2-TN 16	20	59	TN 1-TN 22
13.45	92	FAC 2-TN 21	16.28	39	TN 1-TN 28
21.63	34	FAC 2-TN 28	15.81	86	TN 1-WASTE 5
11	90	FAC 2-TN 30	21.93	39	TN 1-WASTE 11
23.02	58	FAC 2-WASTE 3	15	74	TN 2-TN 22
25.06	6	FAC 3-TN 11	20	14	TN 2-WASTE 1
25.08	42	FAC 3-TN 16	19.65	52	TN 2-WASTE 11
18.38	63	FAC 3-TN 19	12.21	19	TN 2-WASTE 15
25.5	8	FAC 3-WASTE 11	13.89	21	TN 3-TN 7
19.31	70	FAC 3-WASTE 12	12	72	TN 3-TN 8

9.06	10	TN 3-WASTE 2	15.23	15	TN 15-TN 25
12.04	94	TN 4-TN 28	10.77	80	TN 15-TN 27
20.25	45	TN 4-TN 30	8.94	61	TN 15-WASTE 7
18.97	1	TN 4-WASTE 2	17.46	55	TN 16-TN 19
12.04	8	TN 5-TN 14	17	2	TN 16-WASTE 3
27.46	26	TN 5-TN 25	15.13	80	TN 16-WASTE 11
7.62	40	TN 5-TN 27	41.04	55	TN 16-WASTE 12
31.14	82	TN 5-TN 30	6.08	56	TN 17-WASTE 6
19.85	44	TN 5-WASTE 8	5.39	67	TN 18-TN 26
29.02	40	TN 6-TN 10	8.06	30	TN 18-WASTE 14
11	21	TN 6-TN 20	8.94	18	TN 19-TN 23
12.04	68	TN 6-WASTE 2	11.31	51	TN 19-WASTE 3
5.66	21	TN 7-WASTE 9	9.49	31	TN 20-WASTE 2
12.08	8	TN 8-TN 17	10.2	58	TN 21-TN 30
7.21	68	TN 8-TN 20	16.4	19	TN 21-WASTE 3
10	33	TN 9-TN 18	21.38	27	TN 22-WASTE 1
11.4	25	TN 9-TN 23	7.07	70	TN 22-WASTE 5
21.02	91	TN 9-WASTE 3	12	81	TN 23-WASTE 3
14.14	19	TN 10-TN 12	17.8	81	TN 23-WASTE 13
13.93	78	TN 10-TN 13	6.4	8	TN 24-TN 31
15.26	19	TN 10-TN 22	24.08	63	TN 25-TN 27
7.62	90	TN 11-WASTE 11	10.63	3	TN 25-TN 29
29.61	11	TN 11-WASTE 12	24.17	0	TN 25-WASTE 7
17	59	TN 11-WASTE 15	10.82	18	TN 25-WASTE 9
14.21	24	TN 12-TN 13	18.79	59	TN 25-WASTE 10
16.16	75	TN 12-TN 22	25.24	1	TN 26-WASTE 3
9.22	26	TN 12-TN 24	10.3	98	TN 26-WASTE 14
11.4	15	TN 12-TN 31	8.25	90	TN 27-WASTE 7
18.6	92	TN 12-WASTE 1	17.89	51	TN 27-WASTE 8
36.4	65	TN 12-WASTE 15	8.06	49	TN 28-WASTE 5
20.88	96	TN 13-WASTE 4	16.28	26	TN 29-WASTE 2
17.8	57	TN 14-TN 21	25.02	54	TN 29-WASTE 10
13	78	TN 14-TN 26	24.04	77	TN 30-WASTE 2
29.55	37	TN 14-WASTE 8	22.2	56	TN 30-WASTE 10
14.21	10	TN 14-WASTE 10	14.56	52	TN 31-WASTE 1
22.56	67	TN 14-WASTE 14			

a_j	k_j	f_j	h_j	
500	34650	240000	22	FAC 1
750	34650	300000	5	FAC 2
1000	34650	500000	35	FAC 3
450	34650	200000	20	FAC 4

w_i	
1600	WASTE 1
4320	WASTE 2
6370	WASTE 3
560	WASTE 4
2400	WASTE 5
640	WASTE 6
680	WASTE 7
1620	WASTE 8
3040	WASTE 9
2000	WASTE 10
5250	WASTE 11
880	WASTE 12
880	WASTE 13
2160	WASTE 14
2250	WASTE 15

Capítulo VII

Abordagem interactiva com meta-heurísticas para problemas de PLIMO 0-1

Nos capítulos anteriores a nossa atenção centrou-se no desenvolvimento de métodos interactivos "exactos" para problemas de programação inteira (mista) multiobjectivo. Designamo-los por métodos "exactos" porque calculam soluções eficientes do problema, e não apenas aproximações destas. Durante o desenvolvimento desses métodos, foi nossa preocupação que o esforço computacional envolvido nas fases de cálculo não fosse excessivo mas, como é do conhecimento geral, as dificuldades inerentes à complexidade da programação inteira, agravadas pela existência de múltiplos objectivos, podem tornar os problemas de PLIMO de maiores dimensões intratáveis pelos métodos "exactos". Nesses casos, as técnicas *heurísticas* – que se caracterizam pela busca de soluções *boas* num tempo computacional razoável – podem assumir um papel importante para ultrapassar as dificuldades do cálculo. Tratando-se de problemas multiobjectivo, as soluções *boas* traduzir-se-ão por soluções aproximadas (i.e. próximas) das soluções não dominadas.

Um método multiobjectivo que usa técnicas heurísticas pode ser útil como abordagem isolada ou como um módulo de pesquisa inicial que antecede um método "exacto". Nesta segunda situação, a intenção do módulo heurístico é dar ao AD um conhecimento inicial do problema que o ajude a estabelecer e rever as suas preferências, a um baixo custo computacional. O AD avançará depois para uma pesquisa mais focada utilizando, por exemplo, um tipo de abordagem como a que propusemos nos capítulos anteriores, que está vocacionada para pesquisas direccionais e locais.

Começando por observar o caso dos problemas monocritério, podemos afirmar que uma solução *boa* para um problema de programação inteira (a maximizar) pode ser interessante para

estabelecer um limite inferior da função objectivo, capaz de reduzir substancialmente o tamanho da árvore de *branch-and-bound*. Pode até ser uma solução aceitável para o utilizador, se o *'gap'* conhecido para a optimalidade for considerado pequeno. Por isso, um procedimento que encontre bons limites inferiores deve fazer parte de qualquer *software* projectado para resolver problemas inteiros. Contudo, o maior inconveniente de muitos procedimentos heurísticos é serem demasiado específicos a certas classes de problemas, deixando de ser aplicáveis quando algum elemento particular perturba o modelo clássico a que se destinavam. Por conseguinte, apesar de muitas heurísticas funcionarem bem na prática, o seu uso é limitado devido à sua especificidade.

Surgem então as *meta-heurísticas* – como o *simulated annealing* (KIRKPATRICK ET AL., 1983), *tabu search* (GLOVER, 1986) ou os *algoritmos genéticos* (GOLDBERG, 1989) – motivadas pela necessidade de procedimentos suficientemente flexíveis que pudessem ser adaptados a um grande conjunto de aplicações. As meta-heurísticas constituem uma nova forma de resolver problemas combinatórios, independentemente da sua estrutura matemática, e sem requerer grande esforço de implementação.

As meta-heurísticas são frequentemente referidas como sendo heurísticas genéricas e este é um dos seus atributos mais elogiados. Na verdade, uma meta-heurística caracteriza-se por uma estrutura geral muito flexível, mas que precisa de ser configurada de acordo com o tipo de problema em causa. Existe um grande número de aplicações de meta-heurísticas publicadas na literatura da área, mas a maior parte dessas aplicações são específicas a determinados problemas e, alterações ao modelo, mesmo que pequenas, poderão implicar a necessidade de uma nova implementação. Uma das razões desta especificidade prende-se com a dificuldade em lidar com as restrições de um problema. A manutenção da admissibilidade em todas as fases de pesquisa tem demonstrado ser vantajosa relativamente à relaxação das restrições com penalização ao nível da função objectivo (ABRAMSON ET AL., 1996) e, por isso, a concepção de um algoritmo genérico para problemas combinatórios passa pela definição de um mecanismo eficaz de reposição da admissibilidade das soluções.

A implementação de uma meta-heurística que é específica a um dado problema particular será certamente mais eficaz na sua resolução do que a utilização de uma implementação genérica. Mas, na nossa opinião, os algoritmos de propósito geral poderão ser muito úteis para tratar diferentes tipos de problemas ou quando estes estão sujeitos a frequentes variações do modelo. Até ao momento, foram desenvolvidos alguns, embora poucos, algoritmos genéricos de meta-heurísticas que se destinam a problemas de programação inteira com variáveis 0-1 (eventualmente 0-1 mistos). Todos os algoritmos genéricos de meta-heurísticas que encontrámos na literatura e que incorporam mecanismos de reposição da admissibilidade são versões de *simulated annealing* ou de *tabu search*. Notemos que estas duas meta-heurísticas utilizam esquemas

de vizinhança com algumas características em comum, sendo mais fácil introduzir um processo de reposição da admissibilidade nestas meta-heurísticas do que em outras (como por exemplo nos *algoritmos genéticos*), sem desvirtuar a filosofia subjacente ao processo de pesquisa.

O nosso interesse nas meta-heurísticas é, sobretudo, para lidar com problemas genéricos de programação inteira multiobjectivo. As soluções não dominadas do problema multiobjectivo são calculadas nas fases de optimização (por exemplo, através da resolução de um programa escalarizante), e a eficácia de uma abordagem baseada em meta-heurísticas depende naturalmente da qualidade das aproximações obtidas para as soluções não dominadas. Por essa razão, a nossa atenção inicial incidiu na concepção de versões genéricas de meta-heurísticas para o caso monocritério. Atendendo às dificuldades, referidas atrás, em criar versões genéricas, cingimos a nossa investigação ao caso da programação inteira com variáveis binárias. Desenvolvemos uma versão de *simulated annealing* (*0-1SA*) e uma de *tabu search* (*0-1TS*) que incorporam um novo mecanismo de reposição da admissibilidade. Estas versões foram depois estendidas e inseridas numa abordagem interactiva multiobjectivo que se destina a problemas de PLIMO com variáveis binárias (PLIMO 0-1).

A abordagem multiobjectivo que propomos tem como intenção fazer uma pesquisa progressiva e selectiva de soluções não dominadas (aproximadas). Em cada interacção, o AD delimita a região que pretende pesquisar através da imposição de limitações nos valores das funções objectivo e, dentro de cada região, o procedimento faz um cálculo selectivo de soluções. Esta abordagem pode ser usada isoladamente, que termina quando o AD encontrar alguma solução que considera satisfatória, ou como módulo de pesquisa estratégica inicial em que, com um baixo esforço computacional, se reduz a região admissível a pesquisar posteriormente por um método "exacto".

Neste capítulo, parcialmente baseado em ALVES E CLÍMACO (2000B), começamos por apresentar as versões monocritério *0-1SA* e *0-1TS*, passando depois à descrição da abordagem interactiva multiobjectivo. O capítulo está organizado da seguinte forma:

Na secção 1 fazemos uma introdução às meta-heurísticas *simulated annealing* (secção 1.1) e *tabu search* (secção 1.2).

Na secção 2, começamos por fazer referência aos algoritmos genéricos de *simulated annealing* e *tabu search* que encontrámos na literatura, descrevendo em particular (na secção 2.1) o algoritmo de CONNOLLY (1992) que constituiu para nós uma base de trabalho, tanto como fonte de inspiração, como por motivo de comparação. Nas secções 2.2 e 2.3 descrevemos as versões de *simulated annealing* e *tabu search* que desenvolvemos para problemas genéricos de programação linear binária. Designamos estas versões por *0-1SA* e *0-1TS*, respectivamente.

A secção 3 é dedicada a abordagens baseadas em meta-heurísticas para problemas multiobjectivo. Na secção 3.1 introduzimos os fundamentos e as características principais das abordagens encontradas na literatura. Fazemos uma avaliação crítica destas abordagens, a partir da qual justificamos algumas das opções que tomámos no nosso trabalho. Descrevemos em seguida (secção 3.2) a nova abordagem que propomos para problemas de PLIMO 0-1.

Na secção 4 apresentamos algumas conclusões deste trabalho.

VII.1 INTRODUÇÃO ÀS META-HEURÍSTICAS *SIMULATED ANNEALING* E *TABU SEARCH*

VII.1.1 *Simulated Annealing*[1]

Os primeiros desenvolvimentos de aplicação da meta-heurística *simulated annealing* (SA) na resolução de problemas de optimização combinatória devem-se a KIRKPATRICK ET AL. (1983). *Simulated annealing* é uma técnica que pode ser vista como uma variante da conhecida heurística de *pesquisa local*, na qual é explorado um subconjunto de soluções admissíveis através de repetidos movimentos da solução actual para uma solução vizinha. Para um problema de minimização, as formas tradicionais de pesquisa local consistem em estratégias descendentes em que apenas existem movimentos na direcção de melhoramento da função objectivo. Contudo, uma estratégia deste tipo resulta normalmente num óptimo local e não global, o que depende fortemente do ponto inicial da pesquisa.

Contrariamente à heurística de pesquisa local, SA permite movimentos que pioram a função objectivo, mas a sua frequência é governada por uma função probabilística que se modifica à medida que o algoritmo prossegue. Esta forma de controlo foi inspirada no trabalho de Metropolis na termodinâmica (METROPOLIS ET AL., 1953), que consistiu no desenvolvimento de um algoritmo para simular o arrefecimento de materiais que tinham sido sujeitos a um "banho quente". O processo físico consiste em elevar inicialmente a temperatura até um ponto em que o material passa ao estado líquido, procedendo depois a um arrefecimento – conhecido por *annealing* – em que a temperatura é cuidadosamente diminuída até que as partículas do sistema (que representa o material) se arranjem convenientemente no estado sólido. O algoritmo de Metropolis pretende simular a alteração energética do sistema de partículas durante o processo de arrefecimento. As leis da termodinâmica estabelecem que a probabilidade de existir um aumento de ΔE na energia de um sistema à temperatura T é dada por $p(\Delta E) = e^{-\Delta E/(KT)}$, onde K é uma constante física conhecida por constante de Boltzmann. A simulação de Metropolis gera

[1] Esta introdução à meta-heurística *simulated annealing* baseia-se nas obras de DOWSLAND (1993) e AARTS E KORST(1989).

uma perturbação e calcula a alteração resultante na energia do sistema. Se a energia baixar, o sistema move-se para o novo estado. Caso contrário, o novo estado é aceite de acordo com a probabilidade $p(\Delta E)$. O processo é repetido para cada valor de temperatura num número pré-definido de iterações, após o qual a temperatura é diminuída até que o sistema atinja um estado estável.

O algoritmo de SA é uma adaptação do algoritmo de Metropolis para gerar uma sequência de soluções de um problema de optimização combinatória em que se pretende minimizar um custo. A analogia entre o problema de optimização e o sistema físico é estabelecida pelo seguinte:

- as soluções do problema de optimização combinatória correspondem aos estados do sistema físico;
- o custo da solução corresponde ao estado energético;
- SA usa um parâmetro de controlo que faz o papel da temperatura.

Consideremos um problema de minimização de uma função custo $c(x)$ num espaço de soluções $x \in X$. Seja N uma estrutura de vizinhança definida em X, tal que $N(x)$ representa as soluções vizinhas de x. O algoritmo de base de SA pode ser descrito da seguinte forma:

Selecciona uma solução inicial $x^0 \in X$;
Atribui x^0 à solução actual: $x \leftarrow x^0$;
Atribui à temperatura T um valor inicial T^0: $T \leftarrow T^0$;
Repete
    Repete
        Escolhe aleatoriamente $y \in N(x)$;
        Seja $\delta = c(y) - c(x)$;
        Se $\delta < 0$ então $x \leftarrow y$ (aceita a solução y);
        Se não
            Considerando uma distribuição uniforme, gera aleatoriamente d no domínio [0,1];
            Se $d < e^{-\delta/T}$ então $x \leftarrow y$ (aceita a solução y);
    Até *contador_iterações* = *Nrep*;
    Diminui a temperatura: $T \leftarrow \alpha(T)$;
Até *condição_de_paragem*;
x é uma aproximação da solução óptima.

T é o parâmetro de controlo que habitualmente se designa por *temperatura*. Inicialmente o valor de T é elevado, o que implica que sejam aceites grandes incrementos no custo, uma vez que a probabilidade de se aceitar um incremento δ no custo é dada por $e^{-\delta/T}$. À medida que T

diminui, as deteriorações aceites no custo são cada vez menores e, quando T atinge 0, não são aceites quaisquer incrementos no custo. Uma solução que diminua o custo é sempre aceite. A probabilidade de aceitação de uma solução y, para substituir a solução actual x, é, pois, dada pela expressão $\min\left\{1,\ e^{-[c(y)-c(x)]/T}\right\}$.

O processo pelo qual se reduz a temperatura é geralmente referido como o 'plano de refrigeração' (*cooling schedule*) e inclui a selecção do valor inicial da temperatura (T^0), a função de redução $\alpha(T)$, o número de iterações em cada patamar de temperatura (*Nrep*) e a *condição_de_paragem* do algoritmo. A *condição_de_paragem* pode ser definida por um valor final de temperatura ou por outra condição que determine a estabilidade do sistema.

A implementação do algoritmo de SA e a sua aplicação a problemas de optimização combinatória requer várias opções, algumas delas mais genéricas e outras mais específicas ao problema em causa.

As *opções genéricas* prendem-se com a definição do 'plano de refrigeração'.

A temperatura inicial deve ser suficientemente elevada para permitir uma troca quase livre de soluções vizinhas de modo a que a solução final seja independente da solução inicial. Assim, a temperatura inicial, para um determinado problema, deveria ser definida com base no conhecimento das diferenças de custo entre soluções vizinhas do problema. Infelizmente, esta informação não se encontra disponível *a priori* na maior parte dos casos e, por isso, a temperatura inicial terá que ser especificada na ausência desse conhecimento ou a partir de experiências preliminares.

A forma como o parâmetro de temperatura é reduzido é também muito importante para o sucesso do processo de *annealing*, assim como o número de repetições que são efectuadas dentro de cada patamar de temperatura. O número de repetições (*Nrep*) pode ser proporcional ao tamanho do problema ou ao número de soluções vizinhas de uma dada solução, podendo variar de temperatura para temperatura. A função de redução de temperatura mais usada é $\alpha(T) = aT$, com $a < 1$ (geralmente $0.8 \leq a \leq 0.99$), embora sejam propostas na literatura outras funções.

Em teoria, a temperatura deve ser diminuída até zero antes de se verificar a condição de paragem. No entanto, na prática não é necessário que T atinja o zero desde que a probabilidade de aceitação de um movimento de maior custo seja próxima de zero. Suponhamos, por hipótese, que $T = 0.1$ e que a diferença mínima de custo entre duas soluções vizinhas é 10. Então, a probabilidade máxima de se aceitar uma solução de custo superior é $e^{-10/0.1} = 3.72\times10^{-44}$, o que já é muito baixa apesar de T ainda estar longe de zero. Há autores que consideram a paragem do algoritmo quando se atinge um número pré-definido de iterações ou uma limitação de tempo de

CPU. Para uma revisão de diferentes 'planos de refrigeração' ver, por exemplo, HAJEK (1988), DOWSLAND (1993) ou PARK E KIM (1998).

Além das opções genéricas, a aplicação de SA a um problema de optimização combinatória requer que sejam também tomadas *opções específicas* ao tipo de problema em causa. Estas decisões estão relacionadas com o espaço de soluções, estrutura de vizinhança e função custo.

A resolução de um problema através de SA requer que seja definida, adequadamente, uma forma de gerar vizinhos $x' \in N(x)$ de uma solução x. Estudos da convergência do método apontam para que a escolha da estrutura de vizinhança deva satisfazer a condição de que uma solução possa ser alcançável a partir de um outra qualquer através de um ou mais *movimentos*. Logo, na escolha da estrutura de vizinhança está implícita a definição de *movimento*.

A definição da estrutura de vizinhança pode tornar-se bastante difícil quando o espaço das soluções é restrito por condições de admissibilidade. A forma mais simples de resolver esta questão é incluir as soluções não admissíveis no espaço das soluções e penalizá-las na função custo. Contudo, os problemas muito restritos por condições de admissibilidade vêem assim os seus espaços de soluções muito alargados, podendo conduzir a que o processo de *annealing* perca demasiado tempo na análise de soluções não admissíveis. Além deste inconveniente, é geralmente difícil tomar uma opção quanto ao tipo de penalização a introduzir na função custo, por exemplo, por cada restrição violada. Se a penalização é muito elevada, facilmente o processo fica "preso" num óptimo local; por outro lado, se é demasiado pequena, as soluções não admissíveis poderão não se diferenciar das admissíveis, permitindo que se obtenha no final uma solução não admissível como a melhor solução encontrada pelo algoritmo. Há, pois, que ponderar bem qual a penalização a introduzir na função custo, o que pode variar de problema para problema.

Simulated annealing tem demonstrado ser uma técnica heurística útil para uma variedade de problemas mas, em algumas instâncias, os resultados podem ser melhorados pela combinação desta com outras abordagens de pré-processamento (antes da aplicação de SA) ou pós-processamento (tentativas de melhorar a solução encontrada por SA). Uma forma de pré-processamento consiste em determinar uma solução de partida de melhor qualidade do que aquelas que em geral se obtêm por uma geração aleatória. Se for adoptada essa estratégia, o processo de *annealing* deverá começar numa temperatura menos elevada porque as altas temperaturas podem destruir as características de um boa solução inicial. No entanto, há que ter especial cuidado para que a temperatura não seja de tal modo baixa que não permita escapar da vizinhança da solução inicial. A forma mais comum de pós-processamento é a aplicação de um algoritmo descendente a partir da solução final de SA, tendo como intenção encontrar, pelo menos, um óptimo local.

Simulated annealing tem sido aplicado a diferentes problemas de optimização combinatória e teoria dos grafos. A maior parte dos estudos e experiências usam diferentes 'planos de refrigeração'. Habitualmente, são versões modificadas relativamente ao algoritmo de base de SA, que melhoram o desempenho do algoritmo para o(s) problema(s) em estudo. Registam-se, até ao momento, um grande número de aplicações em problemas de partição de grafos, coloração de grafos, caixeiro viajante, afectação quadrática, problemas de planeamento de tarefas (*scheduling*) e de horários, entre outros (ver, por exemplo, AARTS E KORST (1989) ou DOWSLAND (1993) para uma revisão e descrição de aplicações de SA). A maior parte das aplicações que se encontram na literatura correspondem a classes específicas de problemas em que as implementações de SA aproveitam as características particulares dos problemas. Em causa estão a definição da estrutura de vizinhança – determinante para o bom desempenho da heurística – e a função custo, assim como o ajuste de parâmetros de controlo. O tratamento dado às restrições e às soluções não admissíveis segue processos particulares, uma vez que o espaço das soluções é definido por conjuntos específicos de condições. Consegue-se, desta forma, tratar adequadamente cada uma das classes de problemas exigindo, porém, que cada um dessas classes possua uma versão específica de SA. Em suma, SA é uma técnica heurística genérica mas há *opções específicas* que têm que ser tomadas na sua implementação, e que podem tornar essa versão dedicada exclusivamente a um tipo de problemas muito particular.

VII.1.2 *Tabu Search*[2]

Tal como *simulated annealing*, a meta-heurística *tabu search* (GLOVER, 1986) caracteriza-se por uma forma de pesquisa de vizinhança, tendo como preocupação evitar que o processo de pesquisa de soluções fique preso em soluções óptimas locais do problema de optimização. Numa pesquisa de vizinhança, a cada solução x do espaço de soluções X está associada uma vizinhança $N(x)$, definida por um subconjunto de soluções de X. *Tabu search* (TS) pode ser usado para conduzir os *movimentos* que transformam uma solução em outras suas vizinhas, utilizando para tal uma *função de avaliação* que mede a atractividade desses movimentos ou das soluções resultantes. Um exemplo de movimento possível, a utilizar em programação inteira-mista, é o incremento ou decremento do valor de uma variável inteira e determinação posterior dos valores das variáveis contínuas pela resolução de um programa linear. Durante o processo de passagem de uma solução a uma vizinha há movimentos considerados proibidos, e daí a designação de *tabu search*, ou seja, pesquisa tabu.

---

[2] Esta introdução à meta-heurística *tabu search* baseia-se nas obras de GLOVER E LAGUNA (1993), GLOVER (1989), GLOVER (1990A) e GLOVER (1990B).

Tabu search emprega uma filosofia diferente da de s*imulated annealing* para ultrapassar o problema de finalização num óptimo local. A aleatoriedade é geralmente desmotivada, assumindo formas mais sistemáticas de condução da pesquisa. Assim, grande parte das implementações de TS são totalmente determinísticas. Existe, no entanto, a excepção da chamada pesquisa tabu probabilística em que os movimentos são seleccionados de acordo com probabilidades baseadas nas avaliações atribuídas a esses movimentos. FAIGLE E KERN (1992) apresentam uma discussão sobre a convergência da pesquisa tabu probabilística. No recente estudo de LØKKETANGEN E GLOVER (1998), os autores mostram como o uso de simples medidas probabilísticas pode melhorar uma primeira fase de TS, porque permite uma diversificação da pesquisa. Os autores deste trabalho referem ainda que a aceitação probabilística de movimentos é especialmente importante quando a função de avaliação dos movimentos é "contaminada" por ruído ou contém elementos que não estão directamente relacionados com a função objectivo do problema.

Os elementos principais que caracterizam TS são os seguintes:

- o uso de estruturas de memória flexíveis que permitam uma exploração de critérios de avaliação e de informação histórica da pesquisa mais completa do que através de estruturas de memória rígidas (como nas pesquisas de *branch-and-bound*) ou por sistemas sem memória (como por exemplo, s*imulated annealing* ou outras abordagens aleatórias);
- a existência de um mecanismo de controlo que permita a interacção entre condições que restringem e condições que libertam o processo de pesquisa, sob a forma de *condições tabu* e *critérios de aspiração* (que explicitaremos mais à frente), respectivamente;
- a incorporação de memórias de diferentes tempos (desde a memória de curta duração até à memória de longa duração) para pôr em prática estratégias de *diversificação* e de *intensificação* da pesquisa. As estratégias de diversificação levam a pesquisa para novas regiões, enquanto que as estratégias de intensificação reforçam combinações de movimentos e características de soluções que historicamente se revelaram com qualidade.

A exploração de estruturas de memória flexíveis para controlar o processo de pesquisa constitui um dos aspectos centrais de TS. Para tal, mantém-se ao longo do processo de pesquisa uma história selectiva H de estados encontrados, e substitui-se a vizinhança simples de uma solução ($N(x)$) por uma vizinhança modificada denotada por $N(H,x)$. A história H condiciona as soluções que podem ser alcançadas quando se efectua um movimento, a partir da solução actual, com vista à obtenção da solução seguinte.

De uma forma geral, as memórias dividem-se em memórias de *curto*, *médio* e de *longo* termo. Relativamente às memórias de *curto* termo, $N(H,x)$ é tipicamente um subconjunto de $N(x)$ e o estado tabu serve para identificar elementos de $N(x)$ que são excluídos de $N(H,x)$. No que diz

respeito às memórias de *médio* e *longo* termo, $N(H,x)$ pode conter soluções que não pertencem a $N(x)$, e que são geralmente soluções de elite (óptimos locais de elevada qualidade) encontradas durante o processo de pesquisa de soluções. Muitas vezes a memória selectiva não contém as soluções propriamente ditas, mas componentes de soluções de elite que são retidas e integradas para constituir informação de entrada no processo de pesquisa.

A retenção de informação histórica em TS pode também ser interessante para modificar a forma como são avaliadas as soluções acessíveis, substituindo a função objectivo $c(x)$ por uma *função de avaliação* $c(H,x)$. TS usa a função de avaliação para seleccionar a solução seguinte, escolhendo a solução $y \in N(H,x^{actual})$ que possui melhor valor para $c(H, y)$.

Nos problemas em que $N(H,x)$ tem muitos elementos, ou é custoso examinar por completo este conjunto, TS pode definir um subconjunto de vizinhos candidatos, $Candidatos\_N(x) \subseteq N(H,x)$, que examinará em vez da vizinhança $N(H,x)$.

Introduzidos os conceitos básicos de TS, passamos a apresentar um algoritmo de base de TS. Consideramos, neste algoritmo, que a função objectivo $c(x)$ é uma função de custo (a minimizar):

Selecciona uma solução inicial $x^0 \in X$;
Atribuir x^0 à solução actual: $x \leftarrow x^0$;
Atribui $x^* \leftarrow x^0$, em que x^* é a melhor solução encontrada até ao momento;
O registo de memória H está vazio;
Repete
    Determina o subconjunto $Candidatos\_N(x)$ de $N(H, x)$;
    Selecciona $y \in Candidatos\_N(x)$ tal que y minimiza $c(H, y)$;
    Se $c(y)<c(x^*)$ então $x^* \leftarrow y$;
    $x \leftarrow y$;
    Actualiza o registo de memória H;
Até *condição_de_paragem*;
x^* é uma aproximação da solução óptima.

A *condição_de_paragem* do algoritmo pode ser estabelecida por um número máximo de iterações após a última actualização da melhor solução x^*, ou por um número total de iterações.

Vejamos agora alguns aspectos relacionados com o uso de registos históricos e a definição de memórias em TS, não só a memória de *curto* termo – a mais utilizada em TS e que consiste, habitualmente, numa *lista tabu* – mas também as memórias de *médio* e *longo* termo.

Quando é efectuado um movimento a partir da solução actual x para uma solução vizinha x', esse movimento pode ser caracterizado por um ou mais *atributos* que descrevem aspectos de mudança como resultado do movimento. Os atributos podem ser, por exemplo, a alteração de

uma variável de 0 para 1 ou vice-versa, a alteração do valor de $c(x)$ para $c(x')$, etc. Um único movimento pode dar origem a múltiplos atributos. Os atributos de um movimento podem ser registados e usados na pesquisa tabu para impor condições. Estas condições previnem a escolha de determinados movimentos que inverteriam alterações recentes dos atributos – por exemplo, a alteração de x_j de 1 para 0 é tabu se x_j foi previamente alterada de 0 para 1. As condições tabu são também usadas para prevenir repetições em vez de inversões – por exemplo, a alteração de x_j de 1 para 0 é tabu se x_j foi previamente alterada de 1 para 0. Define-se habitualmente um conjunto de atributos e um movimento é declarado tabu enquanto esses atributos forem tabu. Nas memórias de *curto* termo é comum especificarem-se vectores que identificam as iterações início e fim que definem o período L durante o qual um dado atributo/movimento é tabu. Define-se assim uma ou mais *listas tabu*. Mas, independentemente das estruturas de dados usadas, o aspecto chave do estado tabu é a especificação de um "bom valor" para L. O valor de L é crítico na medida em que, se for demasiado pequeno, a pesquisa pode ficar "presa" num óptimo local e, se for muito grande, a pesquisa pode ter dificuldade em se focar num óptimo local porque há muitos caminhos para o óptimo que ficam bloqueados pelo estado tabu. Existem, porém, algumas regras para escolher, estática ou dinamicamente, o valor de L. A título ilustrativo, podemos apontar os seguintes exemplos (GLOVER E LAGUNA, 1993): escolher um valor constante para L, como $L = 7$ ou $L = \sqrt{n}$, onde n é uma medida da dimensão do problema (estas são regras estáticas); fazer com que L varie, aleatória ou sistematicamente, entre L_{min} e L_{max} em que, por exemplo, $L_{min} = 5$ e $L_{max} = 11$ ou $L_{min} = 0.9\sqrt{n}$ e $L_{max} = 1.1\sqrt{n}$ (estas são regras dinâmicas).

O estado tabu pode ser muito restritivo porque pode bloquear movimentos "bons". Este inconveniente pode ser ultrapassado através da utilização de *critérios de aspiração* que prevalecem sobre os estados tabu. O *critério de aspiração* mais utilizado consiste em retirar a classificação tabu a um movimento que conduza à melhor solução encontrada até ao momento. Uma discussão detalhada sobre o assunto é apresentada por GLOVER E LAGUNA (1993).

As listas tabu baseiam-se em ocorrências no passado recente, mas podem ser também registadas medidas de *frequência*, um tipo de informação que complementa a memória de curto termo. A frequência é geralmente ponderada ou decomposta em subclasses que têm em conta diferentes aspectos, como a qualidade da solução e a influência dos movimentos. As memórias baseadas em frequência são as memórias de *médio* e *longo* termo.

As medidas de frequência podem consistir em razões cujos numeradores representam contagens de ocorrências de um evento particular (por exemplo, número de vezes que um determinado atributo pertence a uma solução ou movimento) e os denominadores podem representar quantidades tais como: o número total de ocorrência de todos os eventos representados nos numeradores, o somatório dos numeradores, o valor máximo ou o valor

médio dos numeradores. Como exemplos de medidas de frequência (numeradores) podem ser apontados o número de soluções em que a variável x_j tem o valor p, ou o número de soluções que resultaram de um movimento de $x_j = p$ para $x_j = q$.

As memórias baseadas em frequência são importantes para uma *intensificação* ou *diversificação* do processo de pesquisa. As estratégias de intensificação têm como objectivo a incorporação de "bons atributos" nas soluções obtidas. As estratégias de diversificação, por seu lado, tentam gerar soluções que contenham composições de atributos significativamente diferentes daqueles que foram encontrados anteriormente durante a pesquisa.

A interacção entre diversificação e intensificação pode também ser conseguida através de uma abordagem designada por *oscilação estratégica*. A oscilação estratégica começa por efectuar um movimento que atinge a fronteira – geralmente a fronteira da admissibilidade – e, em vez de parar, atravessa-a. Permite-se atravessar a fronteira através da extensão da definição de vizinhança ou modificando-se o critério de avaliação dos movimentos seleccionados. A pesquisa prossegue para além da fronteira, numa dada profundidade, regressando depois. Nessa altura, a fronteira é atravessada na direcção oposta até se atingir um novo ponto de viragem. Este processo, que se caracteriza por uma repetida aproximação e cruzamento da fronteira segundo diferentes direcções, cria uma forma de oscilação que dá o nome ao procedimento. Notemos que a fronteira considerada para a oscilação estratégica não tem que ser definida necessariamente em termos de admissibilidade ou de estrutura, podendo ser definida em termos de uma região onde gravita a pesquisa. Nesse caso, a oscilação consiste em obrigar a pesquisa a movimentar-se fora dessa região permitindo também que regresse.

A meta-heurística TS tem sido aplicada a inúmeros problemas de optimização combinatória, sendo a maior parte das aplicações datada da última década. TS tem demonstrado sucesso em variados problemas com particular incidência na área do *scheduling* que, aliás, constitui uma das áreas mais promissoras para as meta-heurísticas em geral. Registam-se também aplicações em problemas de transporte (incluindo caixeiro viajante), desenho de circuitos electrónicos e telecomunicações (problemas de afectação quadrática, entre outros), problemas de grafos (como a coloração, por exemplo), bem como em lógica probabilística, redes neuronais e outros problemas. Para uma revisão e descrição de aplicações de TS ver, por exemplo, GLOVER E LAGUNA (1993).

Tal como acontece com outras meta-heurísticas, a maior parte das aplicações que se encontram descritas na literatura correspondem a problemas de áreas muito específicas em que as implementações de TS tiram partido das características particulares dos problemas. A dificuldade principal em desenvolver versões genéricas de TS está na definição adequada de uma estrutura geral de vizinhança que se adapte a uma grande classe de problemas.

VII.2 META-HEURÍSTICAS PARA PROBLEMAS GENÉRICOS DE PL 0-1

Como referimos atrás, tem sido escassa a investigação no que diz respeito ao desenvolvimento de sistemas de meta-heurísticas capazes de resolver problemas genéricos. As meta-heurísticas são frequentemente designadas por heurísticas genéricas porque os seus algoritmos de base são suficientemente flexíveis para poderem ser adaptados e aplicados a uma grande diversidade de problemas. Contudo, a quase totalidade das versões desenvolvidas até ao momento estão confinadas a tipos específicos de problemas. Se pretendêssemos aplicá-las a problemas de outros tipos teríamos que, na maior parte dos casos, codificar quase tudo de novo aproveitando apenas a estrutura geral do algoritmo, esta de codificação muito simples. Podemos apontar uma razão principal para que isto aconteça: o sistema genérico de modelação de problemas de optimização combinatória baseia-se na notação da programação linear inteira e é difícil para uma meta-heurística "navegar" num espaço de soluções definido desta forma. A alteração do valor de uma variável pode levar à violação de várias restrições do problema, e a reposição de um estado admissível pode ser custosa.

No caso dos problemas de programação inteira 0-1, podemos definir um movimento natural de vizinhança que consiste na comutação do valor de uma variável de 1 para 0 ou vice-versa. Esta operação não tem em conta as condições de admissibilidade do problema, existindo basicamente duas formas de agir:

Processo 1. Introduzir na função objectivo penalizações para as soluções não admissíveis.

Processo 2. Tentar repor a admissibilidade fazendo alterações na solução obtida.

No *processo 1*, qualquer ponto do espaço de procura, incluindo os inadmissíveis, são admitidos como soluções. Este processo levanta várias dificuldades que se prendem com a determinação de termos adequados de penalização a incluir na função objectivo. Além disso, o aumento do tamanho do problema pode aumentar drasticamente o espaço de procura, o que leva a que o algoritmo perca demasiado tempo a avaliar soluções não admissíveis.

O *processo 2* será, em princípio, mais eficaz do que o primeiro, desde que se consiga repor a admissibilidade num tempo aceitável. É também importante que o processo de restabelecimento da admissibilidade não "polarize" a pesquisa de soluções para determinadas regiões do espaço de soluções em detrimento de outras regiões que nunca são alcançadas.

Em 1992, Connolly desenvolveu uma versão de *simulated annealing* (SA) para problemas genéricos de programação linear 0-1 (CONNOLLY, 1992). Segundo o autor, os responsáveis pela empresa financiadora da sua investigação interessaram-se pelo sucesso da técnica de SA, mas sentiam que a imposição de rescrever um programa para cada novo cliente limitava muito a sua utilidade. Surgiu, assim, a necessidade de desenvolver um algoritmo de utilização genérica capaz de encontrar soluções boas de problemas formulados em programação linear 0-1 (PL 0-1). A

versão de SA desenvolvida por Connolly, e intitulada de GPSIMAN, usa um mecanismo genérico para tentar repor a admissibilidade do sistema após ter efectuado um movimento de troca (comutação) de uma variável de 0 para 1 ou vice-versa. A reposição da admissibilidade é iterativa, escolhendo, em cada iteração, a variável para comutar que é considerada mais "útil" para efectuar um movimento em direcção à admissibilidade. Para o efeito, é usado um esquema de pontuações, em que a variável com maior pontuação é considerada a mais "útil". A pontuação de uma variável depende de quanto essa variável pode ajudar a restabelecer a admissibilidade das restrições violadas. Este procedimento pode precisar de comutar muitas variáveis para atingir a admissibilidade, o que limita muito o sucesso de GPSIMAN. Como o próprio autor afirma, uma condição crítica para o bom desempenho de GPSIMAN é que a admissibilidade seja fácil de alcançar e manter. ABRAMSON E RANDALL (1999) também experimentaram o GPSIMAN e concluíram que este programa passa cerca de 30% do tempo a restabelecer a admissibilidade.

Apesar das suas limitações, o trabalho de Connolly despertou o interesse de outros investigadores para o assunto, motivando-os a fazer estudos de comparação e melhoramento daquele método de SA para algumas classes de problemas 0-1.

De entre esses investigadores estão ABRAMSON ET AL. (1996) que compararam, em problemas de partição de conjuntos, o desempenho de um algoritmo de SA com penalização da inadmissibilidade ao nível da função objectivo e uma adaptação do algoritmo de Connolly. Os algoritmos diferem apenas no esquema usado para as transições de vizinhança. No primeiro, as restrições são relaxadas e incorporadas na função objectivo através de uma função de penalização. O segundo é o algoritmo de Connolly, mas em que são adicionadas algumas técnicas de redução do espaço de procura no decurso do algoritmo. Algumas destas técnicas são genéricas (para problemas de programação inteira 0-1) mas outras são específicas aos problemas de partição de conjuntos. Os autores verificaram que o segundo método foi melhor do que o primeiro em quase todos os casos, porque conseguiu resolver mais problemas e foi mais rápido nos problemas pequenos. Não obstante, assim como o método de penalização da inadmissibilidade, também este falhou em problemas com espaços de soluções muito restritos.

Ainda na continuação do trabalho de Connolly, ABRAMSON E RANDALL (1999) desenvolveram um sistema baseado no GPSIMAN, chamado INTSA. Este novo código tem as mesmas características de estrutura do algoritmo de Connolly, diferindo na forma como é mantida a admissibilidade. Os autores examinaram algumas classes de problemas 0-1 (afectação quadrática, caixeiro viajante, coloração de grafos, entre outros) e reformularam-nas em modelos com variáveis inteiras e restrições dos tipos '$\leq$', '$\geq$', '$=$' e '$\neq$'. A reformulação depende da categorização do problema, o que conduz a processos distintos para a manutenção da admissibilidade. O INTSA teve um melhor desempenho do que o GPSIMAN, e do que um

código comercial para programação inteira (OSL), na maior parte dos problemas testados. No entanto, a maior desvantagem do INTSA é o facto de a técnica de especificação dos problemas não ser convencional, não sendo automática a reformulação. Os autores acrescentam ainda que não está provado que esta técnica permita representar todos os problemas 0-1.

Recentemente foram também desenvolvidas algumas versões de *tabu search* (TS) para problemas genéricos de programação inteira 0-1 (mista), onde se incluem os trabalhos de LØKKETANGEN ET AL. (1994), ABOUDI E JÖRNSTEN (1994) e LØKKETANGEN E GLOVER (1998), que usam mecanismos de alcance/reposição da admissibilidade. Estes mecanismos baseiam-se em 'pivotações' no quadro simplex da relaxação linear do problema. Com efeito, mantém-se a admissibilidade primal e tenta-se encontrar a admissibilidade inteira, aquela que é imposta pelas condições de integralidade das variáveis. Trata-se, portanto, de uma abordagem diferente da abordagem de Connolly (e suas extensões), em que se podia perder a admissibilidade primal, mas mantinha-se sempre a admissibilidade inteira. Vejamos as características principais destas versões de TS.

LØKKETANGEN ET AL. (1994) modificaram a heurística *Pivot and Complement* (P&C, pivotar e complementar), desenvolvida por BALAS E MARTIN (1980) para problemas inteiros 0-1, incorporando-lhe características da pesquisa tabu. A heurística P&C começa por calcular a solução óptima da relaxação linear do problema 0-1, considerando em seguida 2 fases: uma fase de procura, onde se tenta encontrar uma solução inteira através de pivotações que reduzam a inadmissibilidade inteira; e uma segunda fase de melhoramento, onde se complementam variáveis (de 0 para 1 ou vice-versa) para melhorar o valor da função objectivo. A incorporação de características de TS consiste, basicamente, em aceitar movimentos (pivotações) que não reduzam a inadmissibilidade inteira (durante a primeira fase), e a complementar variáveis que não melhorem o valor da função objectivo (durante a segunda fase), em combinação com os mecanismos habituais de condução da pesquisa de TS. Por exemplo, em qualquer uma das fases é usada uma lista tabu com as últimas variáveis que se tornaram básicas ou que foram complementadas. Dos testes computacionais efectuados os autores concluíram que o desempenho de TS é semelhante ao de P&C na fase de procura, mas o uso de TS na fase de melhoramento dá muito bons resultados, geralmente superiores aos de P&C.

ABOUDI E JÖRNSTEN (1994) também recorreram à heurística P&C, mas numa perspectiva diferente da de LØKKETANGEN ET AL. (1994). P&C foi usada como uma subrotina do tipo "caixa-preta" em que entra uma solução básica admissível da relaxação linear do problema (original ou modificado) e sai uma solução inteira admissível. Os princípios de TS são usados para modificar o *input* de P&C através da adição de restrições ao problema linear ou impondo outras modificações que alteram a solução de partida de P&C. Neste trabalho, os testes computacionais tiveram como intenção a comparação das diferentes heurísticas de alteração do

input de P&C. Qualquer uma das abordagens é pelo menos tão boa quanto P&C, já que a primeira iteração consiste na aplicação vulgar de P&C.

LØKKETANGEN E GLOVER (1998) desenvolveram uma abordagem de TS para problemas genéricos 0-1 mistos que usa, como subrotina, o simplex 'standard' para variáveis limitadas. Partindo da solução óptima da relaxação linear do problema, fazem-se sucessivas pivotações (movimentos) para soluções básicas admissíveis (da relaxação linear) adjacentes. O movimento seleccionado, de cada vez, é o que apresenta uma melhor *avaliação* de entre um conjunto *candidato*. O conjunto *candidato* é definido a partir de *regras* de TS que excluem ou penalizam movimentos tabu. São definidas e testadas computacionalmente várias estratégias para criar o conjunto candidato, a função de avaliação e as regras que definem o estado tabu. É na função de avaliação que se dá maior ou menor prioridade aos movimentos que diminuem a inadmissibilidade inteira vs. movimentos que melhoram a função objectivo.

Voltaremos a fazer alusão aos testes e resultados computacionais destas versões de TS quando os compararmos com os das nossas versões.

Todas as abordagens de meta-heurísticas genéricas que acabámos de referir se enquadram naquele que designámos atrás por *processo 2* para o tratamento das condições de admissibilidade do problema. Ou seja, em todas ou nalgumas fases de pesquisa, tenta-se atingir ou repor a admissibilidade da solução. De facto, vários autores constataram experimentalmente que esta forma de proceder é geralmente mais eficaz do que relaxar as condições de admissibilidade e usar uma penalização na função objectivo (*processo 1*). O estudo comparativo de BÄCK ET AL. (1996), usando uma outra meta-heurística, reforça esta impressão geral. Este estudo compara os resultados de um processo (heurístico) de 'reparação' e o de uma função de penalização no contexto dos *algoritmos genéticos*, utilizando, para o efeito, problemas de cobertura. O *algoritmo genético* de base é o GENEsYs[3] que utiliza uma função simples de avaliação com um termo de penalização das soluções inadmissíveis. O processo de 'reparação' de soluções inadmissíveis melhorou significativamente o resultado final do algoritmo nos problemas de cobertura testados.

Também nós experimentámos implementar versões simples de SA e TS com funções de penalização para as soluções não admissíveis, e testámo-las em problemas de programação inteira 0-1. O movimento de vizinhança consistiu na comutação de uma variável (de 0 para 1 e vice-versa).

---

[3] Este software foi também usado por KHURI ET AL. (1994) em problemas de *knapsack* 0-1 multidimensional. Os autores descrevem, neste artigo, as características gerais do GENEsYs.

Os resultados obtidos para um conjunto de problemas de *knapsack* 0-1 multidimensional não foram animadores. A solução final foi, em geral, de fraca qualidade quando comparada com o óptimo do problema que também conhecíamos. Mas a situação revelou-se ainda mais problemática em outros tipos de problemas com espaços de soluções muito restritos. O problema de localização P-MEDIAN, cuja formulação incluímos no anexo VII.B, foi um exemplo de um desses casos. Apesar de ser um problema pequeno (20 variáveis e 9 restrições), não foi encontrada nenhuma solução admissível durante uma execução de SA com 1800 iterações (movimentos).

Face a estes resultados, decidimos desenvolver um mecanismo de reposição da admissibilidade que pudesse ser incorporado nas meta-heurísticas SA e TS. Inspirados na rotina de Connolly, pretendemos construir uma outra que tivesse um melhor desempenho do que a de Connolly.

Nos parágrafos seguintes, começaremos por expor o algoritmo SA de Connolly, dando particular ênfase à rotina de reposição de admissibilidade. Passaremos depois à apresentação das novas propostas de SA e TS.

VII.2.1 Algoritmo de *Simulated Annealing* de Connolly (1992)

O programa de computador desenvolvido por CONNOLLY (1992), e intitulado de GPSIMAN, lê problemas formulados em programação linear inteira com variáveis binárias e aplica-lhes um algoritmo de SA. Nesse algoritmo, a temperatura T é controlada pela função $\alpha(T) = T/(1+\beta T)$, em que β é uma constante calculada por forma a que a temperatura varie, em M iterações, de um valor inicial T^0 até um valor final T^f. O programa sugere valores para T^0, T^f e M que podem ser modificados pelo utilizador.

A construção de uma solução vizinha da solução admissível actual consiste em comutar uma variável de 0 para 1 ou de 1 para 0, tentando em seguida restabelecer a admissibilidade, caso esta se perca. A escolha da primeira variável a comutar não é aleatória, existindo uma lista que é percorrida sequencialmente. Alterada essa variável, se a solução resultante não for admissível, então é chamada a rotina RESTORE que escolhe iterativamente variáveis para comutar. Em cada iteração é escolhida a variável que é considerada mais "útil" para repor a admissibilidade e, se essa variável não existir, o algoritmo retrocede desfazendo o movimento da iteração anterior e retirando-o da lista das variáveis possíveis para comutar. O processo iterativo continua até a solução ser admissível ou se concluir que não é possível restabelecer a admissibilidade. Neste último caso, a variável que iniciou a sequência de movimentos fica fixa no seu valor original e é retirada da lista de movimentos que geram vizinhos. Cada iteração de RESTORE é executada

pela subrotina GETSWOP. Vejamos como GETSWOP selecciona a variável mais 'útil' para comutar.

A subrotina GETSWOP calcula a 'pontuação de ajuda' $SCORE_j$ de cada variável x_j, que depende de quanto essa variável pode ajudar a tornar satisfeitas as restrições violadas e da gravidade da violação dessas restrições. A gravidade K_i da violação de uma restrição i é medida pela razão entre a sua insatisfação, INF_i, e a ajuda total possível para essa restrição, $HELP_i$. A razão $INF_i/HELP_i$ deve ser menor do que 1 porque, caso contrário, significa que não é possível satisfazer essa restrição e a subrotina termina. A variável seleccionada para ser comutada é aquela que tem uma maior 'pontuação de ajuda'. Exceptua-se o caso em que existe uma variável que é vital para que alguma restrição fique satisfeita (i.e. sem a sua comutação, essa restrição nunca ficará satisfeita). Essa variável tem prioridade sobre todas as outras.

Apresentamos em seguida os passos da subrotina GETSWOP. Consideremos, sem perda de generalidade, que o problema tem n variáveis 0-1 e m restrições do tipo $\sum_{j=1}^{n} a_{ij} x_j \le b_i$, $i = 1,...,m$.

Seja $\bar{x}$ a solução não admissível à qual se vai aplicar a subrotina, e seja *Candidatos* o conjunto das variáveis que ainda podem ser comutadas (este subconjunto é definido e actualizado na rotina RESTORE).

Inicia a 0 as 'pontuações de ajuda' das variáveis candidatas: $SCORE_j = 0$, $\forall\ x_j \in$ *Candidatos*

Para cada restrição i não satisfeita

Determina a insatisfação da restrição i: $INF_i = \sum_{j=1}^{n} a_{ij} \bar{x}_j - b_i$;

Define o conjunto H_i dos índices das variáveis de *Candidatos* que, se forem comutadas, reduzem a insatisfação da restrição i:

$H_i = \{j \mid x_j \in$ *Candidatos* e $((a_{ij} < 0$ e $\bar{x}_j = 0)$ ou $(a_{ij} > 0$ e $\bar{x}_j = 1))\}$;

Determina a ajuda total possível para a restrição i: $HELP_i = \sum_{j \in H_i} |a_{ij}|$;

Se $INF_i > HELP_i$ então não é possível repor a admissibilidade da solução – SAI;

Determina a gravidade da restrição i: $K_i = INF_i / HELP_i$:

Para cada variável x_j tal que $j \in H_i$

Se $= INF_i > HELP_i - |a_{ij}|$ então x_j é uma variável vital para a restrição i e a variável a comutar será x_j – SAI;

Actualiza a pontuação de x_j: $SCORE_j \leftarrow SCORE_j + K_i \times \min(|a_{ij}|, INF_i)$;

(Fim de j)

(Fim de i)

A variável a comutar é aquela que tiver a maior pontuação $SCORE_j$.

No anexo VII.A incluímos um exemplo ilustrativo da aplicação da rotina RESTORE.

VII.2.1.1 Experiências com o GPSIMAN

Implementámos o algoritmo proposto por Connolly e fizemos alguns testes com problemas de *Knapsack* multidimensional e problemas de localização *p-median*. Esta experiência, apesar de limitada no tipo e no tamanho dos problemas, permitiu-nos observar que, (*i*) nos problemas em que admissibilidade é facilmente restabelecida, as melhores soluções encontradas não eram muito próximas do óptimo; mas, (*ii*) o ponto crítico do algoritmo residiu no processo de alcance e manutenção da admissibilidade para alguns problemas, como aliás o próprio autor do GPSIMAN o tinha constatado.

Relativamente ao ponto (*i*), julgamos que as dificuldades residem essencialmente no facto de esta forma de reposição da admissibilidade ter tendência a "polarizar" a pesquisa de soluções, não havendo uma suficiente diversificação. Reparemos que a subrotina GETSWOP escolhe de forma determinística a variável a comutar. Em alternativa poderia fazer-se uma selecção probabilística, com probabilidades proporcionais às pontuações das variáveis. Isso viria, no entanto, agravar a reposição da admissibilidade e por isso só deveria ser usado em problemas em que a reposição é fácil. Foi o caso dos problemas de *Knapsack* multidimensional que testámos, em que conseguimos resultados substancialmente melhores com a introdução desta alteração. O mesmo já não se pode dizer dos problemas de localização *p-median*, em que a utilização da subrotina GETSWOP, mesmo na forma determinística, conduziu frequentemente a processos de retrocesso que a tornaram morosa.

Relativamente ao ponto (*ii*), julgamos que uma das dificuldades de GETSWOP está no facto de a escolha da variável se basear apenas nas restrições violadas, sem atender ao caso de a comutação de uma variável poder tornar insatisfeitas restrições actualmente satisfeitas. Ou seja, apenas se avaliam as variáveis do ponto de vista do benefício, sem nunca olhar ao prejuízo que podem causar.

Tendo em conta os ponto críticos apontados ao algoritmo de Connolly – em particular à rotina RESTORE – desenvolvemos uma versão genérica de *simulated annealing* para problemas de programação linear 0-1. Para o efeito, tivemos duas preocupações principais. Em primeiro lugar, aumentar a eficácia do processo de restabelecimento da admissibilidade, escolhendo variáveis para comutar não só pelo seu benefício para restrições violadas, mas também pelo menor prejuízo para as restrições satisfeitas. Em segundo lugar, evitar que a rotina de restabelecimento da admissibilidade "polarize" o processo de *annealing*.

VII.2.2 Uma proposta de *Simulated Annealing* para PL 0-1

Nesta secção apresentaremos a versão de *simulated annealing* que desenvolvemos para problemas de programação linear 0-1, e que designamos por *0–1SA*.

Começaremos por descrever o processo de reposição da admissibilidade de uma solução não admissível. Pode tratar-se da solução inicial ou de uma solução que foi obtida a partir da solução actual pela troca do valor de uma variável (de 0 para 1 ou de 1 para 0). Após a descrição desse processo, passaremos a descrever as outras características do algoritmo de *0–1SA*.

VII.2.2.1 Procedimento de reposição da admissibilidade

A rotina de reposição da admissibilidade que propomos baseia-se numa parte do algoritmo aditivo de Balas para problemas 0-1 (BALAS, 1965). O funcionamento desta rotina, a que chamamos REPOE_ADMISS, é o seguinte:

Seja $\bar{x}$ a solução não admissível que se pretende transformar numa admissível. Enquanto $\bar{x}$ continuar não admissível, e ainda for possível repor a admissibilidade, selecciona-se uma variável para comutar (*variável de reposição*) de entre as variáveis que ainda estão *livres*. Por *livres* entendem-se todas as variáveis que ainda não foram comutadas durante este processo. No início do processo todas as variáveis são *livres*, excepto aquela que iniciou o movimento de vizinhança e deu origem a $\bar{x}$. Se o processo terminar com uma solução não admissível, essa será a solução a avaliar, sendo contudo muito penalizada ao nível da função objectivo. Isto implica que o algoritmo de SA raramente aceite uma solução não admissível como substituta de uma solução actual admissível.

A *variável de reposição* de cada iteração é determinada pela subrotina VAR_A_COMUTAR. Esta subrotina pode operar de modo determinístico, em que a variável seleccionada é a que tem maior pontuação, ou de modo estocástico, em que a probabilidade de seleccionar uma variável é proporcional à respectiva pontuação. A subrotina VAR_A_COMUTAR começa por seleccionar o subconjunto H das variáveis *livres* que diminuem a insatisfação de restrições violadas. Para cada uma dessas variáveis, determina a sua "fraqueza" relativamente a todas as restrições, sejam elas satisfeitas ou não. O termo "fraqueza" é usado para designar o resultado de dois tipos de efeitos: o primeiro é o prejuízo de violar uma restrição satisfeita ou agravar a situação de uma restrição já violada; o segundo é a ausência de benefício total para restrições violadas. Consideramos que há ausência de benefício total (ou falta de ajuda) de uma variável relativamente a uma restrição se a comutação dessa variável diminuir a insatisfação dessa restrição mas não for o suficiente para a tornar satisfeita. A "fraqueza" de uma variável é medida em pontuações negativas. A variável com maior pontuação ($PONT_j$) é a que tem menor "fraqueza". Sendo assim, uma variável com pontuação ("fraqueza") nula não oferece qualquer prejuízo e o seu benefício é total relativamente às restrições violadas, o que significa que a comutação dessa variável é o suficiente para repor a admissibilidade da solução.

Consideremos que o problema tem n variáveis 0-1 e m restrições do tipo $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$, $i = 1,...,m$. A função objectivo (custo) é a minimizar de expressão $\sum_{j=1}^{n} c_j x_j$. Os passos da subrotina VAR_A_COMUTAR são os seguintes:

- Calcula o valor da variável desvio s_i de cada restrição (satisfeita ou não):

  $s_i = b_i - \sum_{j=1}^{n} a_{ij} \bar{x}_j$, $i = 1,..,m$. A restrição i é violada se $s_i < 0$.

- Determina o conjunto H dos índices das variáveis *livres* que, se forem comutadas, reduzem a insatisfação de alguma restrição violada:

  $H = \{j \text{ livre} \mid (a_{ij} < 0 \text{ e } \bar{x}_j = 0) \text{ ou } (a_{ij} > 0 \text{ e } \bar{x}_j = 1) \text{ para algum } i \text{ tal que } s_i < 0\}$;

- Para cada restrição i tal que $s_i < 0$, determina y_i :

  $$y_i = s_i + \sum_{\{j \in H:\, a_{ij} < 0 \text{ e } \bar{x}_j = 0\}} (-a_{ij}) + \sum_{\{j \in H:\, a_{ij} > 0 \text{ e } \bar{x}_j = 1\}} a_{ij}$$

Se $H = \varnothing$ ou existe algum $y_i < 0$, então não é possível restabelecer a admissibilidade – termina com uma solução não admissível;

Se não,

- Calcula a pontuação (≤ 0) de cada variável j de H:

$$PONT_j = \begin{cases} \sum_{\{i:\, s_i - a_{ij} < 0\}} (s_i - a_{ij}) & \text{se } \bar{x}_j = 0 \\ \sum_{\{i:\, s_i + a_{ij} < 0\}} (s_i + a_{ij}) & \text{se } \bar{x}_j = 1 \end{cases}$$

A variável a comutar será aquela que tiver a maior pontuação $PONT_j$. (*)

(*) modo determinístico

A rotina REPOE_ADMISS com VAR_A_COMUTAR no modo determinístico mostrou um bom desempenho em experiências preliminares. No entanto, o resultado de *0–1SA* foi em geral superior para uma selecção probabilística da variável a comutar. Esta modificação diversifica as soluções obtidas, diminuindo o efeito de "polarização", mas não é recomendada para problemas onde a admissibilidade é dificilmente restabelecida. Na nossa implementação optámos por considerar um algoritmo misto. A selecção é em geral probabilística, mas torna-se determinística se as primeiras iterações revelarem dificuldades em atingir a admissibilidade.

O exemplo VII.1 ilustra o funcionamento da rotina REPOE_ADMISS (no modo determinístico).

Exemplo VII.1. Consideremos o seguinte problema:

$$\begin{aligned} \min z = \; & x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \\ \text{s.a:} \quad & x_1 - 10x_2 + 2x_3 + 15x_4 - 7x_5 + x_6 \leq -10 \quad (1) \\ & 3x_2 - x_3 - x_4 + x_5 - 3x_6 \leq 0 \quad (2) \\ & 2x_1 + 3x_2 - 2x_3 - 2.5x_4 - 4x_6 \leq -2 \quad (3) \\ & x_j \in \{0,1\},\ j = 1,...,6 \end{aligned}$$

Consideremos a solução inicial não admissível $\bar{x} = (1, 1, 0, 0, 1, 1)$ que pretendemos transformar em admissível.
Livres $= \{1, 2, 3, 4, 5, 6\}$

1ª iteração
Cálculo do desvio de cada restrição:
$s_1 = 5$ (restrição 1 satisfeita)
$s_2 = -1$ (restrição 2 não satisfeita)
$s_3 = -3$ (restrição 3 não satisfeita)
O conjunto das variáveis ajudantes da restrição (2) é $H_2 = \{2, 3, 4, 5\}$ e da restrição (3) é $H_3 = \{1, 2, 3, 4\}$. Logo, $H = H_2 \cup H_3 = \{1, 2, 3, 4, 5\}$.
É ainda possível repor a admissibilidade.
Cálculo das pontuações (negativas) de cada variável pertencente a H:
$PONT_1 = -2 = -1$ (falta de ajuda na restrição 2) – 1 (falta de ajuda na restrição 3)
$PONT_2 = -5$ (prejuízo para a restrição 1 que é satisfeita)
$PONT_3 = \mathbf{-1}$ (falta de ajuda na restrição 3)
$PONT_4 = -10.5 = -10$ (prejuízo para a restrição 1) – 0.5 (falta de ajuda na restrição 3)
$PONT_5 = -5 = -2$ (prejuízo para a restrição 1) – 3 (falta de ajuda na restrição 3)
A maior pontuação pertence à variável x_3, logo será essa a variável a comutar de 0 para 1.

$\bar{x} = (1, 1, \mathbf{1}, 0, 1, 1)$ (Solução não admissível)
Livres $= \{1, 2, 4, 5, 6\}$

2ª iteração
Cálculo do desvio de cada restrição:
$s_1 = 3$ (restrição 1 satisfeita)
$s_2 = 0$ (restrição 2 satisfeita)
$s_3 = -1$ (restrição 3 não satisfeita)
O conjunto das variáveis ajudantes da restrição (3) é $H_3 = \{1, 2, 4\}$, logo, $H = H_3 = \{1, 2, 4\}$
É ainda possível repor a admissibilidade.
Cálculo das pontuações (negativas) de cada variável pertencente a H:
$PONT_1 = \mathbf{0}$ $PONT_2 = -7$ $PONT_4 = -12$
A maior pontuação pertence à variável x_1, logo será essa a variável a comutar de 1 para 0. Como tem pontuação nula garante que a próxima solução é admissível.

$\bar{x} = (\mathbf{0}, 1, \mathbf{1}, 0, 1, 1)$ (**Solução admissível)**

Nota: No exemplo VII.1 foram necessárias duas iterações para tornar a solução admissível. Este exemplo pode ser comparado com o do anexo VII.A que ilustra o funcionamento da rotina RESTORE de CONNOLLY (1992) para o mesmo problema. A rotina RESTORE teve de retroceder duas vezes, uma para anular a primeira tentativa de movimento, a comutação de x_2, e outra para anular a segunda tentativa de movimento, a comutação de x_4. A rotina REPOE_ADMISS (exemplo VII.1) não escolheu as variáveis x_2 e x_4 pelo prejuízo que iriam causar à restrição 1 (que era satisfeita).

VII.2.2.2 Opções genéricas de *0-1SA*

As opções genéricas de um algoritmo de SA dizem respeito à definição do 'plano de refrigeração', que inclui o valor inicial para a temperatura, T^0, a função de redução da temperatura, $\alpha(T)$, o número de iterações em cada patamar de temperatura, *Nrep*, e a condição de paragem do algoritmo.

Na implementação de *0-1SA*, consideramos a função de redução da temperatura $\alpha(T) = \overline{\alpha}.T$, em que $\overline{\alpha}$ é uma constante positiva inferior a 1. A temperatura T é dada por $T = Kt$, sendo a temperatura inicial $T^0 = Kt^0$ com t^0 próximo de 1. O algoritmo termina quando a temperatura atinge o valor final $T^f = Kt^f$, com t^f próximo de 0. A constante K varia de problema para problema e é calculada automaticamente pelo programa. Após várias experiências preliminares com diferentes valores de K em diferentes problemas, decidimos considerar $K = 0.5\, c^{med}$ com $c^{med} = \sum_{j=1}^{n} |c_j| \Big/ n$. Os valores de $\overline{\alpha}$, t^0, t^f e *Nrep* podem ser escolhidos pelo utilizador, mas os sistema sugere (por omissão) os seguintes valores: $\overline{\alpha} = 0.9$, $t^0 = 0.98$, $t^f = 0.01$ e $Nrep = 10$.

VII.2.2.3 Opções específicas de *0-1SA*

As opções específicas de um algoritmo de SA são as que se relacionam directamente com a estrutura do problema, nomeadamente a definição da estrutura de vizinhança e a função custo, e a construção da solução inicial.

Um movimento em *0-1SA* consiste na troca do valor (comutação) de uma variável binária (de 0 para 1 ou vice-versa). Para obter uma solução vizinha da solução actual, *0-1SA* começa por comutar uma variável escolhida aleatoriamente. Se a solução resultante for inadmissível, então tenta alcançar a admissibilidade através da rotina REPOE_ADMISS. A rotina REPOE_ADMISS não garante a reposição da admissibilidade em todas as circunstâncias, podendo haver necessidade de avaliar soluções inadmissíveis. Por conseguinte, a função custo é constituída pela função objectivo do problema (a minimizar) com um termo adicional de penalização igual a $M \times$ 'nº de restrições violadas'. Consideramos M um número positivo bastante grande[4].

A solução inicial é dada pelo arredondamento dos valores das variáveis na solução óptima da relaxação linear do problema. Se a solução arredondada for inadmissível, então aplica-se a rotina REPOE_ADMISS a essa solução.

---

[4] É de notar que, nossas experiências, nunca foi necessário avaliar uma solução inadmissível.

VII.2.2.4 Resultados computacionais de *0-1SA*

Para testar ***0-1SA*** seleccionámos um conjunto de 10 problemas que inclui 8 problemas de ***knapsack*** multidimensional, um problema de localização ***p-median*** (no anexo VII.B, sob o título P-MEDIAN) e um problema de cobertura. O número de variáveis dos problemas de ***knapsack*** multidimensional varia desde 28 até 105 e o problema de cobertura tem 192 variáveis e 240 restrições. Estes problemas foram obtidos a partir da 'OR-Library' de BEASLEY (1990).

Os problemas de ***knapsack*** multidimensional são os seguintes:

PET5, PET6, PET7	de PETERSEN (1967);
SENTO1, SENTO2	de SENYU E TOYODA (1967);
WEING6, WEING7, WEING8	de WEINGARTER E NESS (1967).

O problema de cobertura intitula-se SCPCYC6 e deve-se a GROSSMAN E WOOL (1997).

Apesar de estes problemas de ***knapsack*** terem dimensões pequenas, a sua escolha deveu-se ao facto de já terem sido testados por abordagens de meta-heurísticas desenvolvidas por outros autores, nomeadamente por DREXL (1988), KHURI ET AL. (1994), ABOUDI E JÖRNSTEN (1994), LOKKETANGEN ET AL. (1994) e LØKKETANGEN E GLOVER (1998). O problema SCPCYC6 foi escolhido por ser um problema difícil de resolver, e o problema P-MEDIAN por ter mostrado dificuldades na reposição de admissibilidade nas abordagens que testámos anteriormente.

Fizemos 20 experiências de ***0-1SA*** para cada problema. Na especificação dos valores dos parâmetros de ***0-1SA*** tivemos em conta a condição de que uma aplicação do algoritmo não deveria exceder um determinado limite de tempo. Considerámos t^0=0.98, t^f=0.01 (t^f=0.001 para SCPCYC6), $\overline{\alpha}$ =0.95 e ajustámos ***Nrep*** entre 1 e 20 de modo a que um limite de tempo estipulado para cada problema nunca fosse excedido. Se, por um lado, sabemos que a solução final tende a melhorar com o aumento do número de iterações (***Nrep***), por outro lado não nos podemos esquecer de que o algoritmo deve ser rápido para que seja competitivo. Tendo em vista uma melhor análise da qualidade da solução vs. tempo, tomámos como limite 50% do tempo despendido pelo software comercial CPLEX, no mesmo computador e para o mesmo problema. Este limite tem pouco significado nos problemas de ***knapsack*** em que o tempo máximo do CPLEX foi 13 segundos (em SENTO1). No entanto, em problemas de difícil resolução, este limite já tem significado. Foi o caso do problema SCPCYC6, em que o CPLEX não conseguiu atingir a solução óptima em 10 horas e o menor valor encontrado para a função objectivo (a minimizar), durante esse tempo, foi 64 (sabemos que o óptimo é 60). Nos 20 testes que fizemos para SCPCYC6, usando ***0-1SA***, o tempo médio de uma execução foi de 2 min e 40 seg, e encontrámos por várias vezes valores melhores do que 64. O computador utilizado para todos os testes (CPLEX e ***0-1SA***) foi um PC Pentium I a 166 MHz.

A tabela VII.1 sumaria os resultados computacionais dos testes de ***0-1SA***. A tabela inclui, para cada problema, o 'número de variáveis * número de restrições' (***n*m***), o valor óptimo da

função objectivo ('z óptimo'), o valor de *Nrep* usado, o 'z melhor' e o 'z médio' obtidos nos 20 testes de *0-1SA*, o tempo médio de teste (em segundos), e a diferença percentual entre o 'z óptimo' e o 'z melhor' de *0-1SA*, dada por $GAP^{opt\text{-}melhor} = 100 \frac{|z\ óptimo - z\ melhor|}{|z\ óptimo|}$.

Problema:	P-MEDIAN	PET5	PET6	PET7	SENTO1	SENTO2	WEING6	WEING7	WEING8	SCPCYC6
	(min)	(max)	(max)	(max)	(max)	(max)	(max)	(max)	(max)	(min)
n * *m*	20*9	28*10	39*5	50*5	60*30	60*30	28*2	105*2	105*2	192*240
z óptimo	**3 700**	**12 400**	**10 618**	**16 537**	**7 772**	**8 722**	**130 623**	**1 095 445**	**624 319**	**60**
Nrep	1	10	10	6	10	5	5	10	20	20
z melhor	**3 700**	**12 400**	10 604	16 524	**7 772**	**8 722**	**130 623**	**1 095 445**	**624 319**	**60**
z médio	3 700	12 386	10 524.4	16 463.8	7 692.8	8 722	130 235	1 095 352	616 287	65.0
tempo médio	0.10 s	0.44 s	0.39 s	0.33 s	3.5 s	1.27 s	0.11s	0.39 s	1.9 s	160 s
$GAP^{opt\text{-}melhor}$	0 %	0 %	0.13 %	0.08 %	0 %	0 %	0 %	0 %	0 %	0 %

Tabela VII.1 – Sumário dos resultados computacionais de 0-1SA.

No que diz respeito aos 8 problemas de *knapsack* multidimensional (PET's, SENTO's e WEING's), podemos fazer algumas comparações com outras abordagens de meta-heurísticas que referimos atrás. Esta comparação terá que se cingir à qualidade das soluções, uma vez que não dispomos de outras medidas de desempenho dessas abordagens (por exemplo tempo, número de iterações, ou outras) que pudessem também ser comparadas:

- *0-1SA* obteve valores médios ('z médio') superiores aos do algoritmo genético de KHURI ET AL. (1994) em todos os problemas excepto num (PET6);
- os valores de 'z melhor' de *0-1SA* foram superiores aos do algoritmo de SA especializado de DREXL (1988) em 7 problemas, e foi igual num (SENTO1);
- *0-1SA* obteve valores de 'z melhor' iguais aos máximos encontrados (de entre as diferentes variantes) pelo TS de ABOUDI E JÖRNSTEN (1994) em 5 problemas; foi melhor em 2 problemas e pior num (PET6);
- *0-1SA* obteve valores de 'z melhor' iguais aos máximos encontrados (de entre as diferentes variantes) pelo TS de LOKKETANGEN ET AL. (1994) em 4 problemas; foi melhor em 3 problemas e pior num (PET7).

Esta comparação de resultados com outras abordagens revelou, em nosso entender, um bom desempenho de *0-1SA* para os problemas em causa. Não nos esquecemos, contudo, que se trata de uma comparação muito limitada porque os problemas têm pequenas dimensões e seguem todos o mesmo modelo. O desempenho de *0-1SA* é muito influenciado pelo comportamento da rotina de reposição de admissibilidade (REPOE_ADMISS). Nos testes que efectuámos, esta

rotina foi chamada em todas as iterações do problema P-MEDIAN (fazendo em média 5 movimentos adicionais por cada vez), ao passo que nos outros problemas foi chamada entre 10% (em WEING7) e 70% (em SENTO1 e WEING8) do total das iterações. O número de movimentos adicionais nesses problemas variou entre 1 e 3 por iteração.

VII.2.3 Uma proposta de *Tabu Search* para PL 0-1

O algoritmo de *tabu search* que apresentamos nesta secção, e que designamos por *0-1TS*, é dedicado a problemas de programação linear 0-1. Este algoritmo resultou de várias experiências com versões preliminares. Começando com uma versão mais simples, a evolução consistiu basicamente na inserção de fases de diversificação e intensificação da pesquisa, introdução de um critério de aspiração e modificação da condição de paragem do algoritmo. Há, no entanto, características comuns a estas versões que começamos por descrever.

Tal como tinha acontecido com o *simulated annealing*, fizemos algumas experiências computacionais iniciais que mostraram a prevalência da recuperação da admissibilidade relativamente a trabalhar num espaço de soluções relaxado, em que as restrições são incluídas na função de avaliação das soluções. Contudo, adoptámos aqui uma estratégia diferente da considerada em *0-1SA*: *0-1TS* comuta o valor de apenas uma variável em cada iteração, o que pode resultar numa solução inadmissível. Mas, se a solução for inadmissível, os movimentos das iterações seguintes serão no sentido da reposição da admissibilidade. O conjunto *Candidatos_N*(x) de soluções y, vizinhas da solução actual x, é então definido de maneira diferente conforme x é admissível ou não. Vejamos como:

Se x é admissível, as soluções de *Candidatos_N*(x) são todas aquelas que se obtêm pela comutação de uma variável não tabu que cumpra as condições estabelecidas por outros registos de memória (que especificaremos mais à frente). A solução vizinha $y \in$ *Candidatos_N*(x) seleccionada será aquela que, podendo ser inadmissível, minimiza a função *avaliação*(y) = $c(y) + Pv$, com P um factor de penalização e v o número de restrições violadas.

Se x é inadmissível, o conjunto *Candidatos_N*(x) inclui apenas as soluções vizinhas que se obtêm pela comutação de uma variável não tabu que diminua a inadmissibilidade da solução. É utilizada a rotina VAR_A_COMUTAR para seleccionar a variável a comutar, considerando-se aqui uma definição diferente da anterior para as variáveis *livres*. Enquanto que em *0-1SA* VAR_A_COMUTAR é uma subrotina de REPOE_ADMISS, em que uma variável deixa de ser *livre* se for comutada, em *0-1TS*, VAR_A_COMUTAR actua isoladamente pelo que as variáveis *livres* são aquelas que não pertencem à lista tabu.

Em resumo, a definição de vizinhança difere da utilizada em *0-1SA*. Em *0-1SA*, uma solução é considerada vizinha e sujeita a avaliação, se for admissível, ou, excepcionalmente, se se concluir

que não é possível recuperar a admissibilidade a partir dessa solução. Logo, o processo de reposição da admissibilidade é completo em cada iteração de *0-1SA*. Há necessidade de agir assim porque cada iteração avalia apenas **uma solução** vizinha, comparando-a com a actual. Mas a filosofia da pesquisa tabu é diferente. Cada iteração avalia **um conjunto de soluções** candidatas e escolhe a melhor para uma dada função de *avaliação*. Se existisse a obrigação de que todas elas fossem admissíveis, o esforço computacional seria certamente muito elevado na reposição da admissibilidade de várias soluções. Por esta razão, adoptámos em *0-1TS* o esquema de vizinhança que acabámos de descrever.

Todas as versões que experimentámos, desde a mais simples até aquela que adoptámos no final, usam uma *lista tabu* (do tipo FIFO – 'first in, first out') de tamanho L (parâmetro de entrada escolhido pelo utilizador). Esta lista contém os índices das últimas L variáveis binárias que trocaram de valor (de 0 para 1 ou de 1 para 0).

No anexo VII.C descrevemos a evolução do algoritmo de *0-1TS*, mencionando as modificações que foram introduzidas ao longo das várias versões e comentando o seu desempenho. Foi a partir de experiências com essas versões que construímos a versão aqui proposta, e que foi designada no anexo VII.C por versão 3. Em traços gerais, esta versão tem as seguintes características (para maiores detalhes e justificação, ver o anexo VII.C):

- A pesquisa divide-se em 3 fases.
  - Uma fase inicial que utiliza, como único registo de memória, a lista tabu.
  - Uma 2ª fase de *diversificação*, em que, além da lista tabu, são também impostas outras condições baseadas na frequência dos valores das variáveis. Estas condições consistem no seguinte: uma variável pode tomar um dado valor (0 ou 1) se o número de vezes que ela tomou esse valor nas iterações passadas foi inferior a um determinado limiar.
  - Uma 3ª fase de *intensificação* que se divide em três estádios semelhantes com soluções de partida diferentes. As soluções de partida são, respectivamente, as três melhores soluções encontradas até ao final da 2ª fase. Além da lista tabu, são também impostas outras condições baseadas na qualidade das soluções passadas. Estas condições consistem, basicamente, no seguinte: uma variável pode tomar um dado valor (0 ou 1) se esse valor teve, nas iterações anteriores, uma maior expressão em soluções consideradas "boas" do que em soluções consideradas "más".
- É usado um *critério de aspiração* (o mais comum em algoritmos de TS) em todas as fases de pesquisa, que consiste no seguinte: se a alteração do valor de uma variável conduzir à melhor solução admissível encontrada até ao momento, então será essa a variável seleccionada, independentemente de satisfazer ou não outras condições.

- A passagem de uma fase à fase seguinte, e a paragem do algoritmo, são estabelecidas por um intervalo máximo de iterações (ΔN) permitido entre duas actualizações consecutivas da melhor solução (ΔN é definido *a priori* como parâmetro de entrada do algoritmo).
- O factor de penalização P varia ao longo da pesquisa, tomando valores menores no início da pesquisa e maiores no final da pesquisa. Os menores valores de P favorecem a diversificação da pesquisa, ao passo que os maiores favorecem uma intensificação.

Discutiremos a seguir os resultados computacionais de *0-1TS*, comparando-os com os de *0-1SA*.

VII.2.3.1 Resultados computacionais de *0-1TS*

0-1TS foi aplicado ao conjunto de problemas usado para testar *0-1SA*. Escolhemos um valor de ΔN para cada problema por forma a que o tempo despendido em cada experiência de *0-1TS* fosse semelhante ao de *0-1SA*. As duas meta-heurísticas podem, portanto, ser comparadas mais facilmente. Fizemos testes com diferentes tamanhos da lista tabu (L), escolhendo valores de L próximos de $\sqrt{n}$ (n é o número de variáveis do problema). O valor de $\sqrt{n}$ tem sido uma referência em outros trabalhos de *tabu search* mas, de acordo com GLOVER E LAGUNA (1993), $\sqrt{n}$ ou o tradicional número 7, são apenas valores sugestivos sem justificação teórica. A tabela VII.2 mostra os valores da função objectivo obtidos em cada caso e a tabela VII.3 sumaria os resultados. Para uma melhor comparação das duas meta-heurísticas, repetimos na tabela VII.4 os 'z melhor' e 'z médio' de cada problema obtidos por *0-1SA* e *0-1TS*.

	P-MEDIAN	PET5	PET6	PET7	SENTO1	SENTO2	WEING6	WEING7	WEING8	SCPCYC6
	(min)	(max)	(max)	(max)	(max)	(max)	(max)	(max)	(max)	(min)
L	ΔN=20	ΔN=70	ΔN=30	ΔN=20	ΔN=50	ΔN=10	ΔN=25	ΔN=20	ΔN=100	ΔN=500
2	**3 700**	12 380	**10 618**				**130 623**			
3	**3 700**	12 390	10 518	16 499	**7 772**	**8 722**	**130 623**			
4	**3 700**	12 390	**10 618**	16 499	**7 772**	**8 722**	130 233			
5	**3 700**	12 380	10 584	16 499	**7 772**	**8 722**	130 233			
6	**3 700**	**12 400**	10 604	16 499	**7 772**	**8 722**	130 233			
7				16 499	**7 772**	**8 722**		**1 095 445**	607 702	62
8								**1 095 445**	607 702	62
9								**1 095 445**	607 702	62
10								1 095 352	612 635	62
11								1 095 357	615 110	62
12										64
13										63

Tabela VII.2 – Resultados computacionais de 0-1TS.

Problema:	P-MEDIAN (min)	PET5 (max)	PET6 (max)	PET7 (max)	SENTO1 (max)	SENTO2 (max)	WEING6 (max)	WEING7 (max)	WEING8 (max)	SCPCYC6 (min)
z óptimo	**3 700**	**12 400**	**10 618**	**16 537**	**7 772**	**8 722**	**130 623**	**1 095 445**	**624 319**	**60**
ΔN	20	70	30	20	50	10	25	20	100	500
L	2 – 6	2 – 6	2 – 6	3 – 7	3 – 7	3 – 7	2 – 6	7 – 11	7 – 11	7 – 13
z melhor	**3 700**	**12 400**	**10 618**	16 499	**7 772**	**8 722**	**130 623**	**1 095 445**	615 110	62
z médio	3 700	12 388	10 588.4	16 499	7 772	8 722	130 389	1 095 409	610 170	62.43
L – melhor	todos	6	2, 4	todos	todos	todos	2, 3	7 – 9	11	7 – 11
tempo médio	0.06 s	0.44 s	0.38 s	0.22 s	1.37 s	0.22 s	0.15 s	0.5 s	2.0 s	158 s
$GAP^{opt\text{-}melhor}$	0 %	0 %	0 %	0.23 %	0 %	0 %	0 %	0 %	1.47 %	3.33 %

Tabela VII.3 – Sumário dos resultados computacionais de 0-1TS.

Problema:	P-MEDIAN (min)	PET5 (max)	PET6 (max)	PET7 (max)	SENTO1 (max)	SENTO2 (max)	WEING6 (max)	WEING7 (max)	WEING8 (max)	SCPCYC6 (min)
z melhor SA	3 700	12 400	10 604	16 524	7 772	8 722	130 623	1 095 445	624 319	60
z médio SA	3 700	12 386	10 524.4	16 463.8	7 692.8	8 722	130 235	1 095 352	616 287	65.0
z melhor TS	3 700	12 400	10 618	16 499	7 772	8 722	130 623	1 095 445	615 110	62
z médio TS	3 700	12 388	10 588.4	16 499	7 772	8 722	130 389	1 095 409	610 170	62.43

Tabela VII.4 – Comparação de 0-1SA e 0-1TS.

Observámos que, para os problemas testados, a variação de L em *0-1TS* não provocou grandes variações na qualidade das soluções finais. Em contraste, várias aplicações de *0-1SA* a um mesmo problema produziram, em geral, uma gama de soluções mais alargada. Como consequência, o valor médio das várias experiências de *0-1SA* com o mesmo problema foi em geral pior que o valor médio obtido por *0-1TS* para os diferentes tamanhos da lista tabu. Todavia, a melhor solução de *0-1SA* foi geralmente melhor (ou igual) do que a de *0-1TS*. Por estas razões, é difícil concluir que uma das meta-heurísticas é superior à outra em todas as circunstâncias.

VII.3 META-HEURÍSTICAS MULTIOBJECTIVO

VII.3.1 Introdução – fundamentos e abordagens propostas na literatura

A aplicação de uma meta-heurística a um problema monocritério tem como intenção encontrar a solução óptima do problema ou uma boa aproximação desta solução. Mas, no caso multiobjectivo, a noção de solução óptima deixa de ter lugar, sendo substituída pela noção de solução eficiente, não dominada ou óptima de Pareto. Consequentemente, a aplicação de uma

meta-heurística a um problema multiobjectivo deve gerar soluções eficientes ou próximas destas. Estas soluções têm sido habitualmente designadas na literatura por soluções *potencialmente eficientes* (potencialmente não dominadas ou potencialmente óptimas de Pareto).

As abordagens multiobjectivo baseadas em meta-heurísticas que encontrámos na literatura da área são, na sua maior parte, *geradoras*, porque pretendem gerar um conjunto de soluções potencialmente eficientes que seja representativo de todo o conjunto eficiente. Contudo, as meta-heurísticas podem também ser usadas em abordagens *interactivas*, que incorporam informação sobre as preferências do AD em cada fase de cálculo de soluções potencialmente eficientes. Qualquer um dos tipos de abordagens (geradora ou interactiva) utiliza habitualmente uma estrutura de dados que armazena e actualiza as soluções potencialmente eficientes encontradas ao longo da pesquisa. Se designarmos esse conjunto de soluções por PE, a "actualização de PE com a solução x" significa introduzir x em PE, se ela não for dominada por alguma solução de PE, e retirar de PE todas as soluções que são dominadas por x. A actualização de PE é independente do mecanismo de avaliação das soluções. Este último tem como intenção actualizar a solução actual (ou soluções actuais) do procedimento.

Recordemos o mecanismo habitual de avaliação usado nas meta-heurísticas SA e TS monocritério. As soluções são julgadas pelo seu valor numa única função, que é a função objectivo do problema ou uma função de avaliação (de valor real) construída a partir desta. Em SA, uma solução y, gerada aleatoriamente dentro da vizinhança da solução actual x, é aceite com probabilidade 1 se for melhor do que x na função de avaliação, e com probabilidade menor do que 1 no caso contrário (probabilidade dada por $e^{-\delta/T}$, em que δ é a diferença positiva dos valores da função nessas duas soluções e T é um parâmetro). Em TS, a solução actual x é sempre substituída por uma solução vizinha y, sendo y a solução que possui melhor valor para a função de avaliação de entre um conjunto candidato. A existência de uma única função objectivo permite, assim, definir uma ordenação das soluções, e determinar a melhor. Contudo, apenas uma ordenação parcial é definida naturalmente na avaliação multiobjectivo – a ordem de *Pareto* – e há que estabelecer um mecanismo de avaliação das soluções de cada iteração para determinar a solução (actual) seguinte. Este é um aspecto fundamental na extensão de uma meta-heurística ao caso multiobjectivo que iremos agora abordar.

O mecanismo mais simples de avaliação das soluções, numa meta-heurística multiobjectivo, consiste em definir uma função *escalarizante* que agregue temporariamente as diferentes funções objectivo numa única função. Segundo HANNE (1999), esta abordagem pode não funcionar bem se a intenção for construir uma aproximação de todo o conjunto eficiente (abordagem geradora). Há autores que consideram diferentes funções escalarizantes em cada fase de avaliação, ou usam simplesmente a ordenação de *Pareto*.

HANNE (1999) aponta um problema ao uso da ordenação simples de *Pareto*: todas as soluções eficientes são tratadas por igual; é, pois, possível que x^2 (eficiente) substitua x^1 (eficiente) para o papel de solução actual e, mais tarde, x^2 venha a ser substituída por x^3, que não é dominada por x^2, mas que é dominada por x^1. Recordemos que o mecanismo de avaliação tem como intenção seleccionar a solução actual, o que é independente da actualização de PE. A figura VII.1 ilustra esta situação. Para a leitura desta figura, e no texto que se segue, consideremos que as funções do problema multiobjectivo são a **maximizar** (em conformidade com as definições dos capítulos anteriores). Definimos genericamente o problema multiobjectivo por:

"max" $f(x) = [f_1(x), \ldots, f_k(x)]$

s.a: $x \in X$

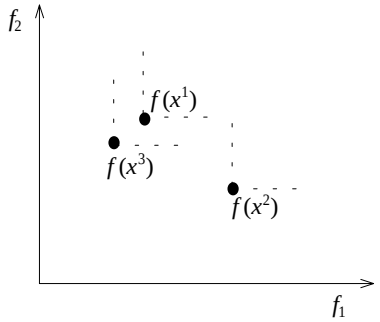


Fig. VII.1 – Deterioração (temporária) no desempenho.

A deterioração (temporária) no desempenho, ilustrada na figura VII.1, pode também acontecer com outros mecanismos de selecção, como por exemplo quando se considera uma função objectivo diferente em cada fase de avaliação. O *algoritmo evolucionário* de HANNE (1999) para problemas multiobjectivo permite ao utilizador optar, durante a sua aplicação, por uma função escalarizante ou por um mecanismo baseado na ordenação de *Pareto*. Os mecanismos disponíveis baseados na ordem de *Pareto* incluem formas secundárias de discriminação. Um deles, por exemplo, considera o número de soluções que dominam uma dada solução como função de desempenho auxiliar. O autor refere que estas formas de discriminação permitem resolver alguns dos problemas causados pela ordenação simples de *Pareto*, mas o grau de discriminação não é muito elevado. Em problemas de grandes dimensões há várias soluções eficientes no mesmo "nível" e a selecção entre elas é aleatória.

Analisemos outros mecanismos de avaliação e selecção da solução actual propostos na literatura. Para tal, comecemos por distinguir três situações na relação de duas soluções, y e x, de um problema multiobjectivo:

(a) *y* domina *x*;

(b) há uma relação de não-dominância entre *y* e *x* (i.e. *y* não domina *x* e *x* não domina *y*);

(c) *y* é dominada por *x*.

Concentrando-nos nas meta-heurísticas de pesquisa em vizinhança, como SA e TS, admitamos que *y* é uma solução vizinha da solução actual *x*. No caso de SA, a probabilidade de se aceitar *y* deve ser 1 em **(a)**, e estritamente menor do que 1 em **(c)**. A situação **(b)** pode ser agrupada à **(c)** – a que SERAFINI (1994) chama de *critério forte* – ou agrupada à **(a)** – *critério fraco*. SERAFINI (1994) examina várias regras alternativas que determinam a probabilidade de aceitação de *y* para uma dada temperatura *T*:

- Usar uma função *escalarizante* *s*(.), em que a probabilidade de aceitação é estabelecida pela expressão usual de SA monocritério para a função *s*(.). Ou seja, $p_{xy}(T) = \min\left\{1\ ,\ e^{[s(f(y))-s(f(x))]/T}\right\}$ (para *s*(.) a maximizar).

  São propostas duas funções escalarizantes em alternativa. Uma delas é a soma pesada das funções objectivo, $s(f) = \sum_i \lambda_i f_i$; a outra é a métrica pesada de Tchebycheff, $s(f) = \max_i \lambda_i (r_i - f_i)$, em que *r* é um ponto de referência (não atingível).

  Notemos que, quando se usa uma função escalarizante, a probabilidade de aceitação na situação **(b)** tanto pode ser 1 como menor do que 1.

- Regras baseadas na ordenação de *Pareto*.

  Uma regra, que se enquadra no *critério forte*, consiste no produto das probabilidades obtidas para cada função objectivo, ou seja, $p_{xy}(T) = \prod_{i=1}^{k} \min\left\{1\ ,e^{(f_i(y)-f_i(x))/T}\right\}$. Uma variante desta regra consiste em usar diferentes temperaturas para cada objectivo, $T_i = T/\lambda_i$, o que equivale a atribuir pesos λ_i às funções objectivo. Ou seja, $p_{xy}(T) = \prod_{i=1}^{k} \min\left\{1\ ,e^{\lambda_i(f_i(y)-f_i(x))/T}\right\}$ – o autor designou-a por regra *P*.

  Uma outra regra – a regra *C* – é, segundo o autor, baseada numa versão local da métrica de Tchebycheff: $p_{xy}(T) = \min\left\{1\ ,\ \min_{i=1,\ldots,k}\left\{e^{\lambda_i(f_i(y)-f_i(x))/T}\right\}\right\}$. As probabilidades resultantes desta regra são sempre maiores ou iguais do que as da regra *P*, mas o critério seguido é ainda o *critério forte*.

  O enfraquecimento desta regra de modo a seguir o *critério fraco* é considerar $p_{xy}(T) = \min\left\{1\ ,\ \max_{i=1,\ldots,k}\left\{e^{\lambda_i(f_i(y)-f_i(x))/T}\right\}\right\}$ – regra *W*.

Segundo SERAFINI (1994), uma combinação entre a regra *P* e a *W*, do tipo $p_{xy}(T)=\alpha\, p^{P}_{xy}(T) + (1-\alpha)p^{W}_{xy}(T)$, com $0< \alpha <1$, parece ser a que apresenta melhores resultados. A justificação é a

seguinte: na regra P (ou na C), a probabilidade de aceitação de uma nova solução é muito baixa, mesmo que ela seja eficiente; esta propriedade não é desejável se quisermos que as soluções eficientes tenham uma probabilidade relevante face às não eficientes; por outro lado, na regra W, todas as soluções têm uma probabilidade de aceitação elevada, mesmo as não eficientes, o que também não é satisfatório. Daí que se proponha uma combinação das duas.

O trabalho de investigação de SERAFINI (1994) centrou-se na análise da probabilidade de aceitação de soluções em algoritmos de SA multiobjectivo. Mas Serafini não propõe um algoritmo de SA multiobjectivo. Embora a literatura sobre o assunto não seja vasta, encontramos algumas propostas de abordagens com meta-heurísticas para problemas de optimização combinatória multiobjectivo. Conforme referimos atrás, todas elas trabalham com um conjunto de soluções potencialmente eficientes, PE. Este conjunto é actualizado de cada vez que se encontra uma solução não dominada por alguma de PE, independentemente da solução ser ou não aceite no passo de selecção.

Vejamos agora as características principais de algumas dessas abordagens. Cingimo-nos neste estudo às abordagens de SA e TS.

ULUNGU ET AL. (1995) desenvolveram um procedimento de SA multiobjectivo, designado por MOSA ('Multi-Objective Simulated Annealing'), que tem como intenção gerar um conjunto aproximado de todas as soluções eficientes. O procedimento começa por gerar um conjunto de q pesos λ^i (i =1,...,q) bem distribuídos. Depois, corre q vezes um algoritmo 'standard' de SA, considerando o seguinte critério (*critério fraco*) de aceitação: y é aceite incondicionalmente nas situações **(a)** e **(b),** e é usada uma função escalarizante $s(f, \lambda)$ para a situação **(c)**. A corrida i (i =1,...,q) de SA utiliza $s(f, \lambda^i)$ e gera PE(λ^i). No final, filtra o conjunto reunião de todos os PE(λ^i), eliminando as soluções dominadas, construindo desta forma uma aproximação da fronteira eficiente. Os autores testaram duas funções escalarizantes – a soma pesada das funções objectivo e a distância pesada de Tchebycheff à solução ideal – concluindo que o tipo de função escalarizante não tem muita influência nos resultados. O mesmo já não acontece com os pesos escolhidos, que têm uma grande influência nos resultados. Segundo os autores, a questão mais delicada do MOSA está na escolha de q e dos pesos, já que o método tem uma missão geradora e deve, portanto, apresentar uma boa aproximação de todo o conjunto eficiente. Além disso, o procedimento consome muito tempo, pelo que os autores concluíram que a integração desta abordagem numa forma interactiva seria uma via a explorar.

É nesta perspectiva que ULUNGU ET AL. (1998) propõem posteriormente o MOSA interactivo. Este procedimento tem uma fase inicial de cálculo semelhante ao MOSA, mas com uma diferença: para cada λ^i, uma solução x só é inserida em PE(λ^i) se satisfizer $f_j(x) \geq g_j \ \forall j$, em

que os g_j ($j=1,...,k$) são indicados pelo AD. Segue-se um processo interactivo em que o AD avalia o conjunto PE, deixando apenas as soluções preferidas, modifica os g_j e propõe um novo vector λ. Após uma fase de diálogo segue-se uma fase de cálculo que consiste em correr uma só vez SA para o novo λ. Actualiza-se PE com PE(λ). O procedimento termina quando o AD o desejar.

Um outro procedimento de SA para problemas de optimização combinatória multiobjectivo deve-se a CZYŻAK E JASZKIEWICZ (1996, 1998). Chama-se 'Pareto Simulated Annealing' (PSA) e tem como intenção gerar uma aproximação de todo o conjunto das soluções eficientes. PSA usa um conjunto de 'soluções geradoras' (S) para cada iteração, em que S assume o papel da solução actual de um SA monocritério. Cada iteração gera uma solução vizinha y para cada solução $x \in S$, que será aceite para substituir x em S com a probabilidade $p_{xy}(T,\lambda) = \min\left\{1\ ,\ e^{\max\limits_{i=1,...,k}\{\lambda_i(f_i(y)-f_i(x))/T\}}\right\}$. PSA pretende obter soluções que se aproximem cada vez mais da fronteira eficiente, ao mesmo tempo que tenta dispersar as soluções. Utiliza o vector de pesos λ para tentar dispersar as soluções, fazendo-o depender de x. A intenção é aumentar a probabilidade de um movimento que se afaste da solução $\tilde{x} \in S$ que está mais próxima de x e é não-dominada em relação a x. Para tal, parte do vector de pesos usado na iteração anterior, λ^x (correspondente à solução x), e aumenta o peso nos objectivos em que x é melhor do que $\tilde{x}$ e diminui nos que é pior. Ou seja, $\lambda_j = \alpha\lambda_j^x$ se $f_j(x) \geq f_j(\tilde{x})$ e $\lambda_j = \lambda_j^x/\alpha$ se $f_j(x) < f_j(\tilde{x})$ com $\alpha>1$. Normaliza λ antes de o introduzir em $p_{xy}(T,\lambda)$.

> Nota: O mecanismo de aceitação usado neste método merece, contudo, alguns comentários da nossa parte. Observamos que a probabilidade de aceitação de y é 1 nas situações definidas atrás como **(a)** e **(b)** – *critério fraco* – sendo $0< p_{xy}(T,\lambda) <1$ na situação **(c).** Em **(c)**, como y é dominada por x, temos que $f_i(y) \leq f_i(x)$, $\forall i$, o que implica que $\lambda_i(f_i(y) - f_i(x))/T \leq 0$, $\forall i$. Denotando por $(-\delta) = \max_i \lambda_i(f_i(y) - f_i(x))/T = = - \min_i \lambda_i(f_i(x) - f_i(y))/T$, em que $\delta>0$, então a probabilidade $p_{xy}(T,\lambda)$ aumenta com a diminuição de δ. Por outras palavras, a probabilidade de aceitação de y é maior se y for muito próxima de x em alguma função objectivo que tenha peso pequeno. Questionamos, então, se a forma como os autores calculam λ e a expressão de $p_{xy}(T,\lambda)$ cumprem a intenção de dispersar as soluções.

No que diz respeito a abordagens de TS multiobjectivo, podemos referir o método MOTS de HANSEN (1997). Tem algumas semelhanças com o PSA de CZYŻAK E JASZKIEWICZ (1996, 1998), mas difere no mecanismo de selecção, inerente ao facto de se basear na pesquisa tabu. Tal como PSA, MOTS trabalha com um conjunto de soluções geradoras (S) para cada iteração, e

pretende obter no final uma aproximação de toda a fronteira eficiente. Para cada solução $x \in S$, é tomada uma direcção de optimização que pretende afastar-se dos outros pontos de S no sentido da fronteira eficiente. No passo de selecção, cada solução $x \in S$ é substituída pela solução vizinha y (não tabu) que apresenta o maior valor de $\sum_i \lambda_i f_i(y)$. Cada solução x tem a sua própria lista tabu. O vector de pesos λ resulta da normalização dos λ_i ($i = 1,...,k$) dados por $\lambda_i = \sum_{x' \in S: f_i(x) > f_i(x')} \pi_i / d(f(x), f(x'), \pi)$ considerando $d(f(x), f(x'), \pi) = \sum_{j=1}^{k} \pi_j \left| f_j(x) - f_j(x') \right|$ e $\pi_i = 1/k$, $i=1,...,k$.

Todas as abordagens mencionadas fornecem esquemas gerais de meta-heurísticas multiobjectivo que, para serem aplicados, requerem a configuração ao problema em causa. Como vimos atrás, a definição da estrutura de vizinhança, e a forma como são tratadas as restrições do problema combinatório, têm muita influência no desempenho de uma meta-heurística monocritério. Esta influência repercute-se, naturalmente, no caso multiobjectivo, uma vez que estas abordagens são extensões das primeiras. Todas as aplicações que encontrámos relatadas para as abordagens multiobjectivo anteriores referem-se a problemas muito particulares em que há uma definição natural de vizinhança.

No nosso caso, pretendemos uma especificação mais completa, que não inclua apenas o enquadramento multiobjectivo de uma meta-heurística, mas também as especificações necessárias à sua aplicação a problemas de programação linear 0-1 multiobjectivo.

O procedimento que propomos baseia-se nas versões monocritério *0–1SA* e *0–1TS*. No que diz respeito ao enquadramento multiobjectivo, o procedimento tem as seguintes características principais:

1) É um procedimento *interactivo*.

A incorporação progressiva de preferências do AD nas fases de cálculo permite reduzir o esforço computacional relativamente ao de uma aproximação geradora. Podemos dirigir uma parte desse esforço no sentido de que as meta-heurísticas gerem melhores aproximações das soluções eficientes em regiões de maior interesse para o AD.

2) É usada uma *função escalarizante* para o passo de avaliação e selecção de soluções da meta-heurística. Esta função de avaliação não sofre alterações ao longo de uma corrida da meta-heurística.

Trata-se, portanto, de uma abordagem diferente da considerada por CZYŻAK E JASZKIEWICZ (1996, 1998), ou por HANSEN (1997), onde os pesos da função escalarizante variam em função da solução actual. Também difere das abordagens de ULUNGU ET AL. (1995) e de ULUNGU ET

AL. (1998), onde a função escalarizante é usada apenas para avaliar soluções que sejam dominadas pela solução actual (situação **(c)**), aceitando-se incondicionalmente uma solução vizinha que estabeleça uma relação de não-dominância com a solução actual (situação **(b)**) – *critério fraco* de aceitação. O PSA de CZYŻAK E JASZKIEWICZ (1996, 1998) também considera um *critério fraco* de aceitação.

Mas, reparemos que a aceitação incondicional de uma solução que não domina nem é dominada pela actual pode levar o procedimento a "saltar" constantemente para regiões distintas do espaço das soluções. Este tipo de mecanismo pode pôr em causa a característica de "convergência" das meta-heurísticas (monocritério) para uma solução realmente boa numa dada função. O resultado será um conjunto disperso de soluções – o que é desejável numa abordagem geradora – mas corre o risco de não ser uma boa aproximação de nenhuma região da fronteira eficiente.

Consequentemente, propomos um procedimento em que o AD delimita interactivamente a região de pesquisa através de restrições adicionais nos valores das funções objectivo. Dentro de cada sub-região, são calculadas algumas soluções potencialmente eficientes resultantes do processo heurístico de optimização de funções escalarizantes definidas *a priori*.

VII.3.2 Nova abordagem interactiva para PLIMO 0-1

Nesta secção apresentamos uma nova abordagem para problemas de programação linear inteira multiobjectivo com variáveis binárias (PLIMO 0-1). Esta abordagem tem como módulo principal um procedimento interactivo com meta-heurísticas baseado nas versões de *simulated annealing* e *tabu search* que desenvolvemos previamente para problemas monocritério, *0-1SA* e *0-1TS*, respectivamente.

O procedimento que propomos tem como intenção uma pesquisa selectiva de soluções potencialmente eficientes que progressivamente se situem nas regiões de maior interesse para o AD. Pretendemos ainda que a pesquisa ofereça um bom compromisso entre o tempo gasto e a qualidade das soluções calculadas. Este procedimento pode ser olhado como um módulo de pesquisa estratégica (inicial), que requer um esforço computacional significativamente menor do que o requerido por um método multiobjectivo "exacto". Após uma pesquisa inicial, o AD poderá usar técnicas exactas para proceder a uma pesquisa local nas regiões que considera mais interessantes.

Na pesquisa interactiva de soluções potencialmente eficientes, o AD pode delimitar as regiões de pesquisa através da imposição de limitações (*níveis de reserva*) nos valores de algumas ou todas as funções objectivo. A fase de cálculo seguinte produz um conjunto de soluções potencialmente eficientes, em que as melhores aproximações são em geral as que estão junto aos óptimos das

funções objectivo da região sob exploração. Se o AD desejar explorar mais profundamente sub-regiões desta, nomeadamente do seu interior, deve especificar novos níveis de reserva que definam uma região reduzida da anterior. Este tipo de abordagem evita os principais inconvenientes dos métodos geradores, nomeadamente no tempo computacional necessário (mesmo no caso de abordagens heurísticas) e na excessiva quantidade de informação que os métodos geradores apresentam ao AD.

Simulated annealing e *tabu search* funcionam como duas rotinas alternativas, podendo o utilizador optar por uma ou por outra em cada fase de cálculo. Uma fase de cálculo consiste em correr várias vezes (para diferentes funções de avaliação) *0-1SA* ou *0-1TS*, em que estas meta-heurísticas sofreram algumas adaptações simples para recolherem soluções potencialmente eficientes durante a sua execução. Assim, no início da pesquisa interactiva é criado um conjunto PE vazio. PE é depois actualizado sempre que se encontra uma solução que não é dominada por alguma solução de PE. A solução é introduzida em PE e são eliminadas todas as soluções de PE que são dominadas por esta.

Como já referimos, a fase de cálculo de uma dada interacção restringe-se à região delimitada pelos níveis de reserva impostos pelo AD: $f_j(x) \geq g_j$, $j \in \overline{K} \subseteq \{1,...,k\}$. Do ponto de vista operacional, significa que estas restrições são incluídas no problema e tidas em conta na rotina de reposição da admissibilidade das meta-heurísiticas *0-1SA* e *0-1TS*.

A fase de cálculo de uma interacção consiste em correr k vezes *0-1SA* ou *0-1TS*. A corrida i ($i = 1,...,k$) privilegia a função objectivo i, usando o critério habitual de avaliação e aceitação monocritério para a função objectivo $f_i(x)$. Por exemplo, a probabilidade de substituir a solução admissível x por uma solução admissível y, numa corrida i de *0-1SA*, é dada por $\min\left\{1, e^{(f_i(y)-f_i(x))/T}\right\}$ (recordemos que $f_i(x)$ foi definida a maximizar). Isto implica que, se y domina x, ou se se estabelece uma relação de não-dominância entre x e y, com y melhor do que x no objectivo i, então y é aceite com probabilidade 1. Caso contrário, a probabilidade de aceitação de y é menor do que 1, e depende da magnitude da diferença verificada no objectivo i e do valor da temperatura T. É de notar que a entrada de y em PE é independente de y ser ou não aceite como solução actual da meta-heurística.

Como a meta-heurística é corrida k vezes por interacção, privilegiando individualmente cada uma das funções objectivo, a maior parte das soluções potencialmente eficientes obtidas situam-se próximo dos óptimos das funções na região sob exploração. Apesar da maior concentração nos extremos da região, há também alguma dispersão provocada pela fase de instabilidade de *0-1SA* para altas temperaturas, ou pela fase de diversificação incluída em *0-1TS*. Mas, como seria de esperar, as soluções do "centro" estão em geral mais afastadas da fronteira eficiente do que as soluções "extremas".

O esquema geral do procedimento com meta-heurísticas que propomos é o seguinte:

Cria PE $\leftarrow \varnothing$

Repete

O AD especifica limites inferiores para as funções objectivo:

$f_j(x) \geq g_j,\ j \in \overline{K} \subseteq \{1,...,k\}$

$\overline{X} \leftarrow X \cap \{ f_j(x) \geq g_j,\ j \in \overline{K} \}$

Para $i = 1$ até k

Corre *0-1SA* ou *0-1TS* em $\overline{X}$ considerando $(-f_i(x))$ como função custo e actualizando PE em cada iteração.

(Como resultado, obtêm-se novas soluções potencialmente eficientes em PE; fica-se ainda a conhecer a solução que optimiza $f_i(x)$ na relaxação linear de $\overline{X}$ porque é a solução inicial de *0-1SA* ou *0-1TS*)

(Fim de *i*)

Até o AD considerar que a pesquisa se focou nas regiões do seu maior interesse.

Após uma pesquisa estratégica usando meta-heurísticas, o AD pode desejar proceder a uma pesquisa local que lhe conceda uma melhor avaliação das soluções potencialmente eficientes preferidas. Se o AD achar conveniente, pode calcular soluções eficientes usando técnicas exactas. Como módulo de "pesquisa exacta", a jusante do módulo de meta-heurísticas, incluímos as seguintes opções:

(i) Definição da fronteira eficiente da relaxação linear do problema multiobjectivo, confinada a uma sub-região suficientemente pequena. Estas soluções fornecem limites superiores para os valores dos objectivos nessa sub-região, possibilitando ao AD avaliar melhor as soluções potencialmente eficientes de que dispõe. Como consequência, o AD poderá concluir que não é necessário explorar mais essa sub-região, se as diferenças entre os limites superiores e os valores dos objectivos nas soluções que conhece forem inferiores a determinados limiares que considera aceitáveis.

(ii) Verificação se uma solução potencialmente eficiente x^{pe} que o AD considera interessante é, de facto, eficiente. Para tal, é resolvido um programa escalarizante de realização (*min-max*) em que o ponto dos critérios de x^{pe}, ou seja z^{pe}, toma o papel do ponto de referência:

$$\min \quad \alpha - \rho \sum_{i=1}^{k} f_i(x) \qquad (P^{\rho}_{z^{pe}})$$

$$\text{s.a:} \quad z_i^{pe} - f_i(x) \leq \alpha \qquad i=1,...,k$$

$$x \in X$$

Se z^{pe} for não dominado, então x^{pe} optimiza $P^{\rho}_{z^{pe}}$, confirmando-se de que se trata de uma solução eficiente. Caso contrário, o resultado de $P^{\rho}_{z^{pe}}$ é a solução eficiente cujo ponto dos critérios está mais próximo de z^{pe} segundo a métrica (aumentada) de Tchebycheff (L_∞).

(iii) Cálculo da solução não dominada mais próxima (segundo a métrica de Tchebycheff) de *níveis de aspiração* desejados pelo AD para os objectivos. Estes níveis de aspiração podem ser, por exemplo, pontos não dominados da relaxação linear do problema que foram calculados em (i).

Na implementação computacional desta abordagem interactiva incluímos os dois módulos referidos: o procedimento de meta-heurísticas e o de pesquisa local com as opções que acabámos de mencionar. Estes módulos foram integrados no sistema computacional que tínhamos vindo a desenvolver para os métodos multiobjectivo de planos de corte e de *branch-and-bound* descritos nos capítulos anteriores. Esta integração dos diferentes procedimentos, no mesmo sistema, permite que o AD possa também proceder a pesquisas *direccionais* como outra opção de pesquisa local.

VII.3.2.1 Exemplos computacionais

Nesta secção apresentamos três exemplos com problemas de *knapsack* 0-1 multidimensional. Os dois primeiros problemas, designados por PET7_2 e SENTO1_2, são problemas bi-objectivo que resultam de termos acrescentado uma segunda função objectivo a dois problemas já conhecidos e usados nos testes das meta-heurísticas monocritério, PET7 e SENTO1, respectivamente. Os coeficientes da segunda função objectivo de cada um destes problemas foram gerados aleatoriamente e figuram no anexo VII.B. O terceiro problema, designado por KNAPM20_3FO, é tri-objectivo e foi usado no capítulo V como exemplo ilustrativo (da definição de regiões de indiferença associadas aos pontos de referência).

A intenção do primeiro exemplo (PET7_2) é discutir a qualidade das soluções potencialmente eficientes obtidas por *simulated annealing* (*0-1SA*) e por *tabu search* (*0-1TS*). Para podermos fazer essa avaliação, resolvemos calcular também todas as soluções eficientes do problema através de um procedimento exacto. A nossa abordagem com meta-heurísticas é interactiva e, por isso, não está vocacionada para construir uma aproximação de toda a fronteira eficiente. Todavia, daremos uma indicação da distância entre os pontos verdadeiramente não dominados e os pontos potencialmente não dominados obtidos em apenas 3 interacções. Relembramos que o problema monocritério PET7, que deu origem a PET7_2, foi um dos que obteve piores resultados nos

testes das versões monocritério de *0-1SA* e *0-1TS*. Por isso o escolhemos para testar esta abordagem.

O segundo exemplo (SENTO1_2) pretende ilustrar o processo interactivo.

O facto de estes dois problemas terem apenas 2 funções objectivo permite visualizar melhor os resultados através de uma representação gráfica no espaço dos objectivos. Estes gráficos são fornecidos directamente pelo sistema computacional. Mas o procedimento não é restrito a problemas bi-objectivo, e por isso incluímos um terceiro exemplo em que mostramos os gráficos disponíveis para o caso geral multiobjectivo.

Nota: As figuras apresentadas a seguir são recortes de écrans de computador (eventualmente sobrepostos). Apenas em alguns casos acrescentámos ou retirámos a identificação das soluções.

EXEMPLO 1: PET7_2 (50 variáveis, 5 restrições, 2 funções objectivo)

Fizemos dois testes independentes, escolhendo para as fases de cálculo *0-1SA* e *0-1TS*, respectivamente. Considerámos em ambos os testes a seguinte sequência de limitações nas funções objectivo:

1ª interacção: sem limitações;

2ª interacção: $f_1(x) \geq 14500$; $f_2(x) \geq 31700$;

3ª interacção: $f_1(x) \geq 15700$; $f_2(x) \geq 30000$;

Enquanto que *0-1TS* não mostrou tendência para aumentar o tempo de cálculo quando se restringe a região, os tempos despendidos por *0-1SA* aumentaram significativamente da 1ª para a 2ª ou 3ª interacções. Este facto deve-se à forma como a rotina de reposição de admissibilidade actua em cada um dos casos: em cada iteração, *0–1TS* faz apenas um movimento em direcção à admissibilidade, mas *0-1SA* desenvolve todos os esforços para repor a admissibilidade nessa iteração, o que conduz a um maior consumo de tempo em espaços mais restritos. Como queríamos que o tempo total consumido pelas duas meta-heurísticas não fosse muito diferente, aumentámos o número de iterações de *0-1TS* da 1ª para a 3ª interacção e diminuímos em *0-1SA*. Para tal, considerámos os seguintes valores dos parâmetros. *0-1SA*: t^0=0.98, t^f=0.005, $\overline{\alpha}$ =0.97 em todas as interacções, *Nrep* =30 na 1ª interacção e *Nrep*=5 nas 2ª e 3ª interacções; *0-1TS*: L =7, ΔN = 250 na 1ª interacção, ΔN = 500 na 2ª interacção e ΔN= 1000 na 3ª interacção.

O gráfico da figura VII.2 mostra os pontos do espaço dos critérios das soluções potencialmente eficientes obtidas: 18 soluções usando *0-1TS* e 14 soluções usando *0-1SA*, calculadas independentemente. Os tempos computacionais gastos por *0-1TS*, num PC PENTIUM II a 350 MHz, foram 2.1, 1.1 e 1.0 segundos na 1ª, 2ª e 3ª interacções, respectivamente; os tempos correspondentes para *0-1SA* foram 1.1, 2.4 e 1.4 segundos. A título de comparação, referimos que o tempo do cálculo exacto, por exemplo, da solução não dominada mais próxima

(segundo a métrica L_∞) da solução ideal foi de 34 segundos (usando a nossa implementação de *branch-and-bound*, e no mesmo computador).

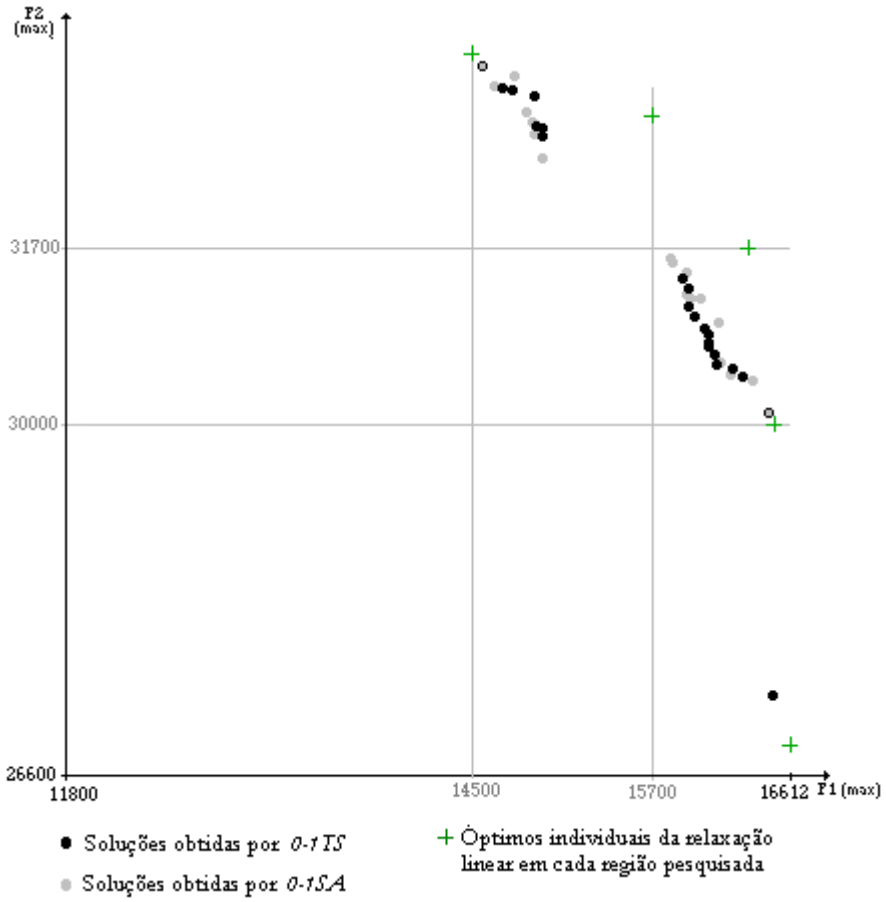


Fig. VII.2 – Soluções potencialmente não dominadas de PET7_2 obtidas independentemente por 0-1TS e 0-1SA.

Foram também calculadas todas as soluções não dominadas do problema[5], que estão representadas na figura VII.3. Os pontos representados por uma cruz (+) nas figuras VII.2 e VII.3 são as soluções não dominadas da relaxação linear do problema que optimizam individualmente cada objectivo na região delimitada em cada interacção.

[5] O cálculo de todas as soluções não dominadas do problema foi feito através de uma *pesquisa direccional* com o método de *branch-and-bound* descrito no capítulo IV. Tratando-se de um problema biobjectivo, conseguimos obter todas as soluções não dominadas se iniciarmos uma *pesquisa direccional* no óptimo da 1ª (ou da 2ª) função objectivo e escolhermos o melhoramento da outra função como direcção de pesquisa.

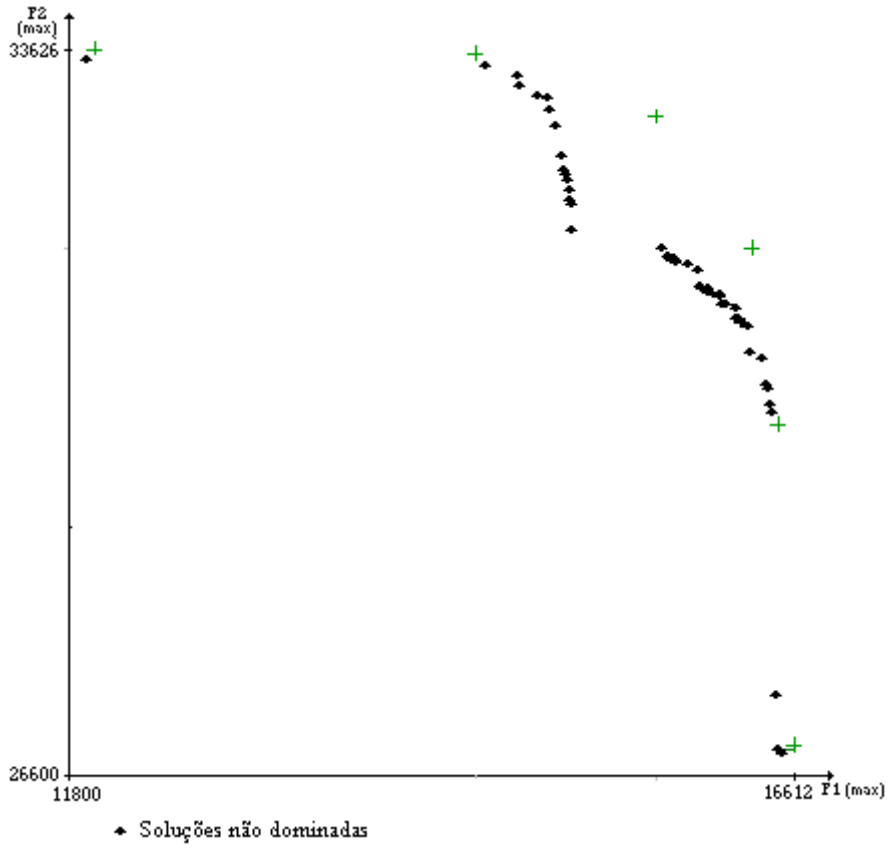


Fig. VII.3 – Todas as soluções não dominadas de PET7_2.

A figura VII.4 mostra um recorte da sobreposição dos gráficos das figuras VII.2 e VII.3 na zona onde se situam a maior parte das soluções não dominadas.

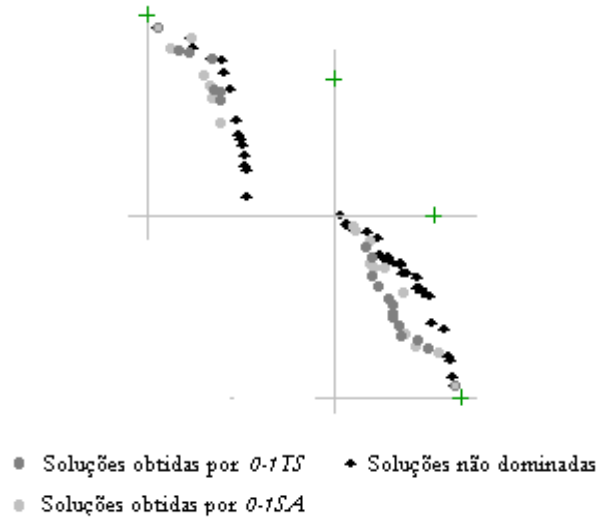


Fig. VII.4 – Sobreposição de soluções potencialmente não dominadas e as soluções não dominadas de PET7_2.

Podemos estabelecer alguma medida da qualidade das soluções potencialmente não dominadas obtidas. Consideremos, por exemplo, a seguinte: para cada solução potencialmente não dominada z^{pnd}, tomemos a solução não dominada z^{nd} que lhe está mais próxima (segundo a métrica L_∞) e calculemos a diferença percentual em cada uma das dimensões, $100\times\left|z_i^{nd} - z_i^{pnd}\right| / z_i^{nd}$, i=1,2. Nesta experiência, as diferenças percentuais médias foram de 0.31% em f_1 e 0.19% em f_2 para as soluções obtidas por *0-1TS* e 0.43% em f_1 e 0.20% em f_2 para as soluções obtidas por *0-1SA*.

Da nossa experiência, com este e outros problemas multiobjectivo, apesar de limitada, pareceu-nos que *0-1TS* oferece um melhor compromisso do que *0-1SA* entre a qualidade das soluções e o tempo computacional requerido.

EXEMPLO 2: SENTO1_2 (60 variáveis, 30 restrições, 2 funções objectivo)

Consideremos que o AD escolheu a heurística *0-1TS* para as fases de cálculo de todas as interacções e atribuiu os seguintes valores aos parâmetros: L=7 e ΔN= 3000.

Na 1ª interacção, para a qual não foram consideradas quaisquer restrições adicionais, foram obtidas 12 soluções potencialmente eficientes (em 14.5 seg num Pentium II a 350 MHz). Os pontos dos critérios destas soluções estão representados no gráfico da figura VII.5.

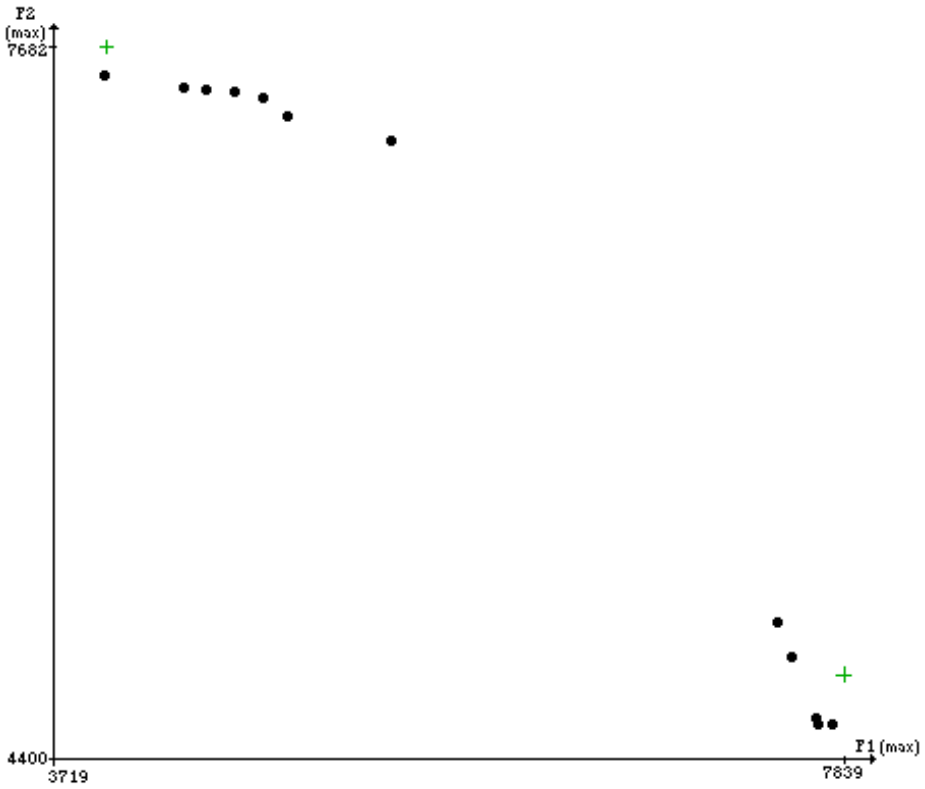


Fig. VII.5 – Soluções potencialmente não dominadas obtidas na 1ª interacção para SENTO1_2.

Suponhamos que foram feitas mais duas interacções em que o AD impôs as limitações $f_1(x) \geq 5300$, $f_2(x) \geq 5000$ na 2ª interacção e $f_1(x) \geq 5400$, $f_2(x) \geq 6300$ na 3ª interacção. O conjunto das soluções potencialmente eficientes (PE) contém 35 soluções no final da 2ª interacção – ver figura VII.6 – e 42 soluções no final da 3ª interacção – ver figura VII.7.

É de notar que, cada interacção começa como o PE final da interacção anterior. Consequentemente, as 42 soluções finais não incluem necessariamente as 35 da 2ª interacção ou as 12 da 1ª interacção. Algumas dessas soluções podem ter sido entretanto eliminadas de PE pela entrada de outras que as dominavam. Os tempos de cálculo das 2ª e 3ª interacções foram de 13.7 e 8.0 seg, respectivamente.

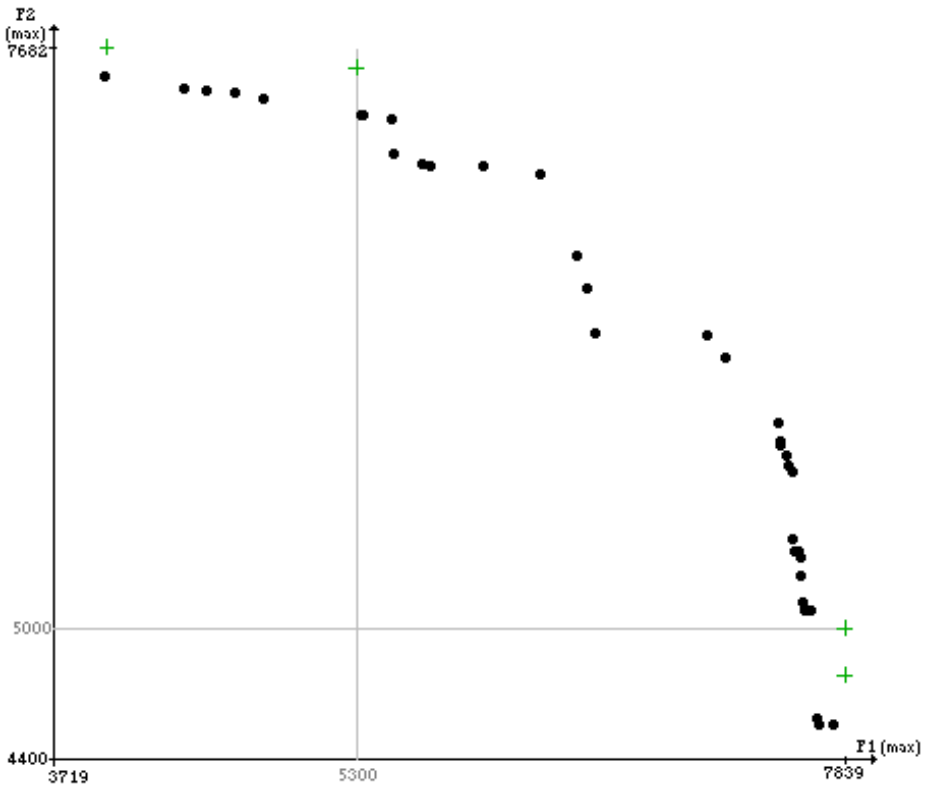


Fig. VII.6 – Soluções potencialmente não dominadas de SENTO1_2 no final da 2ª interacção.

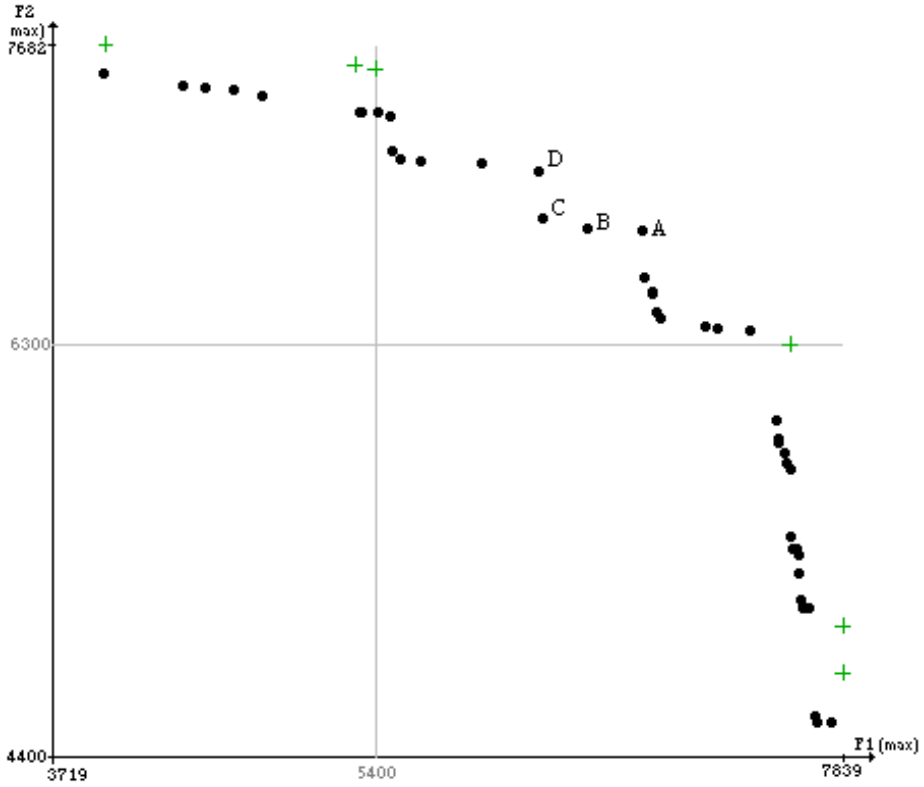


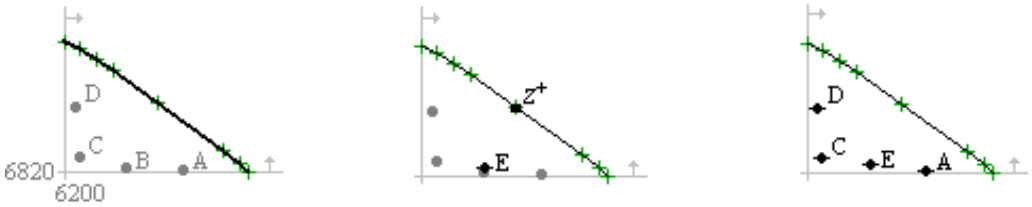
Fig. VII.7 – Soluções potencialmente não dominadas de SENTO1_2 no final da 3ª interacção.

Suponhamos que o AD considera interessante a região onde se situam as soluções A, B, C e D assinaladas na figura VII.7, em que z^A =(6794, 6828), z^B =(6500, 6840), z^C =(6271, 6878) e z^D =(6250, 7099). O AD decide então em fazer uma pesquisa local nesta região, delimitando-a por $f_1(x) \geq 6200$ e $f_2(x) \geq 6820$.

Começa por escolher a opção (i) que determina a fronteira não dominada da relaxação linear do problema na região pré-definida (figura VII.8a, em que as soluções básicas estão representadas por '+'). Isto permite-lhe ter uma melhor percepção dos erros máximos envolvidos nas soluções potencialmente não dominadas que conhece.

Escolhe, em seguida, a opção (iii) considerando o ponto de aspirações z^+ =(6667, 7120) sobre a fronteira não dominada da relaxação linear. O resultado é a solução não dominada E assinalada na figura VII.8b; z^E =(6503, 6842) e domina z^B. O tempo de cálculo desta solução foi de 1 min 11.3 seg.

Suponhamos que o AD escolhia ainda as soluções A, C e D para a opção (ii). Nesse caso chegaria à conclusão que qualquer uma delas é, de facto, uma solução não dominada – figura VII.8c.



(a) (b) (c)

Fig. VII.8 – Pesquisa local de soluções não dominadas de SENTO1_2.

EXEMPLO 3: KNAPM20_3FO (20 variáveis, 10 restrições, 3 funções objectivo)

Fizemos apenas uma interacção, não impondo quaisquer limitações adicionais. Escolhemos a heurística *0-1TS* para o cálculo, com L=7 e ΔN= 250. Dessa interacção resultaram 13 soluções potencialmente eficientes (ao fim de 0.82 seg). Pudemos concluir que este é precisamente o conjunto de todas as soluções eficientes do problema, uma vez que o conhecíamos da análise feita no capítulo V (os valores dos critérios dessas soluções encontram-se na tabela V.1).

As figuras VII.9 e VII.10 mostram os resultados e os tipos de interface disponíveis para problemas com mais do que duas funções objectivo, como é o caso deste. A figura VII.9 apresenta um gráfico de barras simples para os valores das soluções potencialmente não dominadas (PE) calculadas. As soluções podem ser ordenadas pelo índice ou pelo valor de uma das funções objectivo. No caso das figuras VII.9 e VII.10, escolhemos a visualização por ordem crescente dos valores da 1ª função objectivo (F_1). Observamos que as soluções não estão numeradas de 1 a 13 porque o seu índice é dado pela ordem de entrada em PE (apercebemo-nos, por exemplo, que as primeiras 5 soluções que entraram em PE foram depois eliminadas, uma vez que o menor índice é 6).

Os gráficos tridimensionais, como o da figura VII.10, constituem outra forma de visualização disponível no sistema. Apresentam simultaneamente as soluções de PE, soluções (verdadeiramente) não dominadas já conhecidas e guardadas na memória do sistema, e soluções não dominadas para a relaxação linear do problema. Os diferentes tipos de soluções estão a níveis de profundidade diferentes, podendo o utilizador escolher o que fica à frente. A informação numérica do painel da direita refere-se às soluções no nível da frente. Na figura VII.10, as soluções de PE estão à frente e as não dominadas da relaxação linear (já conhecidas) no nível atrás; o nível intermédio é reservado para as soluções (verdadeiramente) não dominadas, em que neste caso está vazio porque não utilizámos nenhuma técnica exacta. No nível mais atrás figuram apenas 3 soluções não dominadas da relaxação linear que são as que optimizam individualmente cada função objectivo.

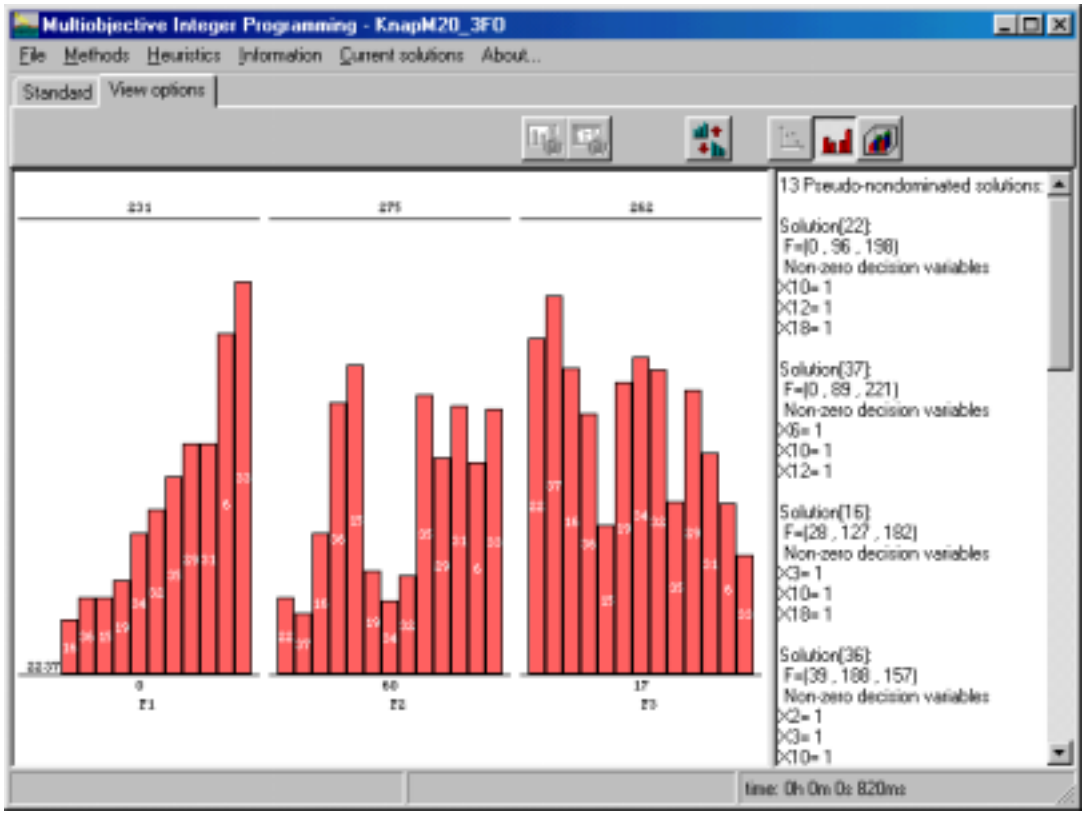


Fig. VII.9 – Soluções (potencialmente) não dominadas de KNAPM20_3FO.

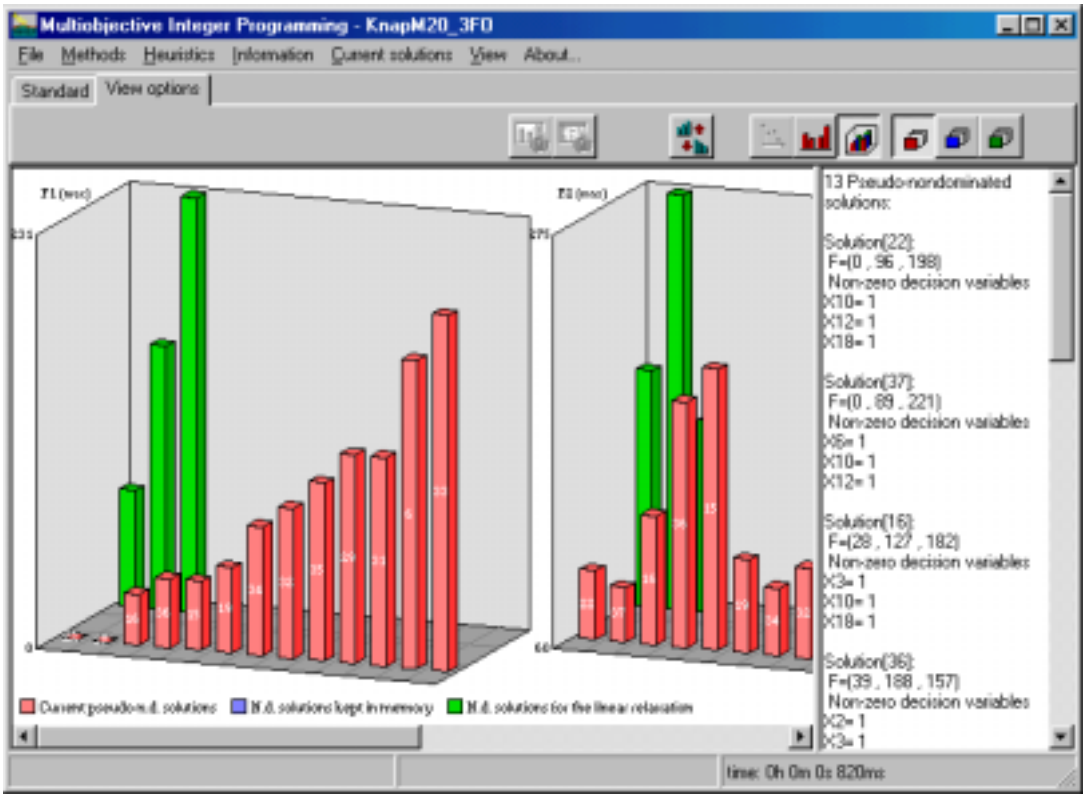


Fig. VII.10 – Outro tipo de interface disponível na abordagem multiobjectivo com meta-heurísticas.

VII.4 CONCLUSÕES

Este trabalho teve como principal intenção desenvolver uma abordagem interactiva baseada em meta-heurísticas para tratar problemas de PLIMO 0-1. As meta-heurísticas são usadas nas fases de cálculo de soluções potencialmente eficientes, o que permite lidar com problemas de dimensões significativamente maiores do que os tratados por métodos multiobjectivo "exactos", e em tempos computacionais razoáveis. Este benefício tem como contrapartida o facto de as soluções encontradas poderem não ser eficientes, e por isso lhes chamamos soluções potencialmente eficientes (ou potencialmente não dominadas). Assim, para que a abordagem multiobjectivo seja eficaz, é muito importante que as meta-heurísticas gerem soluções que, se não forem eficientes, sejam boas aproximações das soluções eficientes.

Nessa perspectiva, preocupámo-nos inicialmente em construir e testar versões monocritério de meta-heurísticas que pudessem depois ser incorporadas nas fases de cálculo da abordagem multiobjectivo. Desenvolvemos versões de *simulated annealing* e *tabu search* para problemas genéricos de PL 0-1, cuja principal inovação está na forma como se mantém a admissibilidade (primal) das soluções inteiras geradas. Das experiências computacionais que fizemos, obtivemos resultados que nos pareceram ser em geral tão bons, e por vezes melhores, do que os de outras meta-heurísticas genéricas existentes na literatura. O facto de as soluções obtidas para os problemas monocritério serem boas aproximações das respectivas soluções óptimas oferece alguma garantia de que as soluções geradas no contexto multiobjectivo sejam boas aproximações das correspondentes soluções eficientes, já que resultam da "optimização" de programas escalarizantes.

Julgamos que talvez sejam demasiado "ambiciosas" as meta-heurísticas multiobjectivo que pretendem gerar uma boa aproximação de todo o conjunto das soluções eficientes. Atendendo a que um problema combinatório multiobjectivo pode ter um número muito elevado de soluções eficientes, mesmo as abordagens heurísticas podem requerer um excessivo esforço computacional para construir um boa aproximação de todo este conjunto. Ou, se esse esforço não for despendido, podemos correr o risco de as soluções encontradas não serem próximas das soluções eficientes.

Assim, propusemos uma abordagem interactiva em que a pesquisa de soluções potencialmente eficientes se restringe, em cada interacção, a uma sub-região da região admissível. O protocolo interactivo consiste em pedir ao AD que especifique limitações inferiores para os valores das funções objectivo. Em nosso entender, esta forma de interagir com o AD tem duas vantagens principais: este é um tipo de informação que o AD está geralmente mais habilitado a fornecer numa fase inicial de um processo de decisão; além disso, permite uma redução progressiva do âmbito da pesquisa, conduzindo o processo para regiões que correspondem mais de perto às preferências do AD.

Na apresentação ao AD de soluções potencialmente não dominadas, são também fornecidos alguns limites superiores que correspondem a soluções não dominadas do problema multiobjectivo relaxado linearmente. Se estes limites estiverem próximos de soluções potencialmente não dominadas, eles poderão dar argumentos para o AD decidir que aquela sub-região já está suficientemente explorada. Incluímos ainda algumas técnicas exactas para serem usadas, preferencialmente, numa fase final do processo de decisão. Se, por um lado, a fase inicial de pesquisa é conduzida por *níveis de reserva* que o AD impõe nos objectivos, já a fase final está especialmente orientada para o cálculo das soluções mais próximas de *níveis de aspiração* que o AD gostaria de atingir. As técnicas exactas possibilitam também verificar se soluções obtidas pelas meta-heurísticas, que o AD considere potencialmente interessantes, são de facto não dominadas. No caso de não o serem, são calculadas as soluções não dominadas mais próximas destas (segundo a métrica L_∞).

A nossa experiência computacional com esta abordagem não foi extensa, mas revelou que este tipo de abordagens poderá ser útil para tratar problemas combinatórios multiobjectivo de maiores dimensões. Uma fase de cálculo com meta-heurísticas determina várias soluções potencialmente eficientes num tempo computacional significativamente menor do que o necessário para calcular uma só solução eficiente. Além disso, estas soluções pareceram-nos ser, em geral, boas ou razoáveis aproximações das soluções eficientes. O cálculo exacto fica então reservado para o final, confinando-se a um número restrito de soluções.

ANEXO VII.A

Exemplo de aplicação da rotina 'RESTORE' de Connolly (1992)

Consideremos o seguinte problema:

$$\min z = x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

$$\text{s.a:} \quad x_1 - 10x_2 + 2x_3 + 15x_4 - 7x_5 + x_6 \leq -10 \quad (1)$$

$$3x_2 - x_3 - x_4 + x_5 - 3x_6 \leq 0 \quad (2)$$

$$2x_1 + 3x_2 - 2x_3 - 2.5x_4 - 4x_6 \leq -2 \quad (3)$$

$$x_j \in \{0,1\}, j = 1,...,6$$

Consideremos a solução não admissível $\bar{x} = (1, 1, 0, 0, 1, 1)$ que pretendemos transformar em admissível através da rotina RESTORE de CONNOLLY (1992). Cada iteração de RESTORE corresponde à execução da subrotina GETSWOP.

1ª iteração:

Candidatos $= \{ x_1 , x_2 , x_3, x_4, x_5 , x_6\}$
Restrições não satisfeitas: (2) e (3);
Pontuações iniciais das variáveis: $SCORE_1 = SCORE_2 = ... = SCORE_6 = 0$;
Restrição (2):
quantidade não satisfeita $INF_2 = 1$;
conjunto das variáveis ajudantes de (2): $H_2 = \{2, 3, 4, 5\}$;
ajuda possível à restrição (2): $HELP_2 = 6$;
$K_2 = INF_2 / HELP_2 = 1/6 = 0.167$;
não há nenhuma variável vital para a restrição (2);
actualiza pontuações das variáveis ajudantes:
$SCORE_2 = 0.167$ $SCORE_3 = 0.167$ $SCORE_4 = 0.167$ $SCORE_5 = 0.167$
Restrição (3):
quantidade não satisfeita $INF_3 = 3$;
conjunto das variáveis ajudantes de (3): $H_3 = \{1, 2, 3, 4\}$;
ajuda possível à restrição (3): $HELP_3 = 9.5$;
$K_3 = INF_3 / HELP_3 = 3/9.5 = 0.316$;
não há nenhuma variável vital para a restrição (3);
actualiza pontuações das variáveis ajudantes:
$SCORE_1 = 0.632$ $SCORE_2 = 1.115$ $SCORE_3 = 0.799$ $SCORE_4 = 0.957$
($SCORE_5 = 0.167$)
A variável com maior pontuação é x_2 com $SCORE_2 = 1.115$, logo será essa a variável escolhida para comutar.

$\bar{x} = (1, \mathbf{0}, 0, 0, 1, 1)$ (Solução não admissível)

2ª iteração:

Candidatos $= \{x_1, x_3, x_4, x_5, x_6\}$
Restrições não satisfeitas: (1);
Pontuações iniciais das variáveis: $SCORE_1 = SCORE_3 = ... = 0$;
Restrição (1):
quantidade não satisfeita $INF_1 = 5$;

conjunto das variáveis ajudantes de (1): $H_1 = \{1, 6\}$;
ajuda possível à restrição (1): $HELP_1 = 2$;
como $INF_1 > HELP_1$ então

não é possível satisfazer esta restrição sendo impossível repor a admissibilidade;

Há que voltar atrás, desfazer o movimento anterior sem o incluir na lista dos *Candidatos*.

$\bar{x}$ =(1, **1**, 0, 0, 1, 1) (Solução não admissível)

3ª iteração:

Candidatos $=\{x_1, x_3, x_4, x_5, x_6\}$
Restrições não satisfeitas: (2) e (3);
Pontuações iniciais das variáveis: $SCORE_1 = SCORE_3 = ... = 0$;
Restrição (2):
quantidade não satisfeita $INF_2 = 1$;
conjunto das variáveis ajudantes de (2): $H_2 = \{3, 4, 5\}$;
ajuda possível à restrição (2): $HELP_2 = 3$;
$K_2 = INF_2 / HELP_2 = {}^1/_3 = 0.333$;
não há nenhuma variável vital para a restrição (2);
actualiza pontuações das variáveis ajudantes:
$SCORE_3 = 0.333$ $SCORE_4 = 0.333$ $SCORE_5 = 0.333$
Restrição (3):
quantidade não satisfeita $INF_3 = 3$;
conjunto das variáveis ajudantes de (3): $H_3 = \{1, 3, 4\}$;
ajuda possível à restrição (3): $HELP_3 = 6.5$;
$K_3 = INF_3 / HELP_3 = {}^3/_{6.5} = 0.462$;
não há nenhuma variável vital para a restrição (3);
actualiza pontuações das variáveis ajudantes:
$SCORE_1 = 0.924$ $SCORE_3 = 1.257$ $SCORE_4 = 1.488$
($SCORE_5 = 0.333$)
A variável com maior pontuação é x_4 com $SCORE_4 = 1.488$, logo será essa a variável escolhida para comutar.

$\bar{x}$ =(1, **1**, 0, **1**, 1, 1) (Solução não admissível)

4ª iteração:

Candidatos $=\{x_1, x_3, x_5, x_6\}$
Restrições não satisfeitas: (1) e (3);
Pontuações iniciais das variáveis: $SCORE_1 = SCORE_3 = ... = 0$;
Restrição (1):
quantidade não satisfeita $INF_1 = 10$;
conjunto das variáveis ajudantes de (1): $H_1 = \{1, 6\}$;
ajuda possível à restrição (1): $HELP_1 = 2$;
como $INF_1 > HELP_1$ então

não é possível satisfazer esta restrição sendo impossível repor a admissibilidade;

Há que voltar atrás, desfazer o movimento anterior sem o incluir na lista dos candidatos.

$\bar{x}$ =(1, **1**, 0, **0**, 1, 1) (Solução não admissível)

5ª iteração:

Candidatos $=\{x_1, x_3, x_5, x_6\}$
Restrições não satisfeitas: (2) e (3);
Pontuações iniciais das variáveis: $SCORE_1 = SCORE_3 = ... = 0$;
Restrição (2):
quantidade não satisfeita $INF_2 = 1$;
conjunto das variáveis ajudantes de (2): $H_2 = \{3, 5\}$;
ajuda possível à restrição (2): $HELP_2 = 2$;

K_2:= INF_2/ $HELP_2$ = ½ = 0.5;
não há nenhuma variável vital para a restrição (2);
actualiza pontuações das variáveis ajudantes:
$SCORE_3$ =0.5 $SCORE_5$ =0.5

Restrição (3):
quantidade não satisfeita INF_3=3 ;
conjunto das variáveis ajudantes de (3): H_3 = {1, 3};
ajuda à restrição (3): $HELP_3$ = 4;
K_3:= INF_3/ $HELP_3$ = ¾ =0.75;
Analisando as variáveis por ordem dos seus índices, verificamos que x_1 é uma variável vital para a restrição (3) uma vez que $INF_3 > HELP_3 - |a_{31}|$;
Logo, x_1 será comutada de 1 para 0.

$\bar{x}$ =(**0**, **1**, 0, **0**, 1, 1) (Solução não admissível)

6ª iteração:

Candidatos = {x_3, x_5, x_6}
Restrições não satisfeitas: (2) e (3);
Pontuações iniciais das variáveis: $SCORE_3$ = $SCORE_5$ =.$SCORE_6$ = 0;
Restrição (2):
quantidade não satisfeita INF_2 =1;
conjunto das variáveis ajudantes de (2): H_2 = {3, 5};
ajuda possível à restrição (2): $HELP_2$ = 2;
K_2 = INF_2/ $HELP_2$ = ½ = 0.5;
não há nenhuma variável vital para a restrição (2);
actualiza pontuações das variáveis ajudantes:
$SCORE_3$ =0.5 = $SCORE_5$ =0.5

Restrição (3):
quantidade não satisfeita INF_3 =1;
conjunto das variáveis ajudantes de (3): H_3 = {3};
ajuda possível à restrição (3): $HELP_3$ = 2;
x_3 é uma variável vital para a restrição (3) uma vez que $INF_3 > HELP_3 - |a_{33}|$;
Logo, x_3 será comutada de 0 para 1.

$\bar{x}$ =(**0**, **1**, **1**, **0**, 1, 1) (Solução admissível)
Variáveis comutadas = {x_1, x_3}

Foram necessários dois movimentos efectivos para tornar a solução admissível (comutação de x_1 e x_3), mas o processo iterou seis vezes, sendo obrigado a voltar atrás por duas vezes.

ANEXO VII.B

Dados de problemas

Problema P-MEDIAN:

$$\min z=1100x_2 + 1760x_3 + 2200x_4 + 1000x_5 + 1500\ x_7 + 1000x_8 + 2080x_9 + 1950x_{10} + 650x_{12} + 2400x_{13} + 1200x_{14} + 600\ x_{15}$$

s.a:
$$x_1 + x_2 + x_3 + x_4 = 1$$
$$x_5 + x_6 + x_7 + x_8 = 1$$
$$x_9 + x_{10} + x_{11} + x_{12} = 1$$
$$x_{13} + x_{14} + x_{15} + x_{16} = 1$$
$$y_1 + y_2 + y_3 + y_4 \leq 2$$
$$110x_1 + 100x_5 + 130x_9 + 120x_{13} - 230y_1 \leq 0$$
$$110x_2 + 100x_6 + 130x_{10} + 120x_{14} - 230y_2 \leq 0$$
$$110x_3 + 100x_7 + 130x_{11} + 120x_{15} - 230y_3 \leq 0$$
$$110x_4 + 100x_8 + 130x_{12} + 120x_{16} - 230y_4 \leq 0$$
$$x_i \in \{0,1\}\ i = 1,\ldots,16$$
$$y_i \in \{0,1\}\ i = 1,\ldots,4$$

Coeficientes da 2ª função objectivo do problema PET7_2:

468	981	484	71	305	392	105	301	141	332	23
495	444	377	458	798	2840	312	151	364	403	138
334	174	3914	78	159	240	41	123	330	500	207
157	348	407	115	3273	102	810	3951	429	137	196
145	3689	3284	477	427	634					

Coeficientes da 2ª função objectivo do problema SENTO1_2:

127	786	27	53	301	40	57	333	488	91	34
56	51	247	65	35	83	5	47	64	831	175
100	17	681	12	27	5	45	92	67	567	43
492	4	280	15	345	44	66	8	99	42	84
74	12	51	698	5	48	2	45	9	52	720
431	53	92	82	20						

ANEXO VII.C

Evolução das várias versões de *0-1TS*

Neste anexo fazemos referência à evolução do algoritmo de *0-1TS*, mencionando as modificações que foram introduzidas ao longo das várias versões e comentando o seu desempenho.

Todas as alterações que introduzimos na definição de *Candidatos_N*(x) apenas se aplicam ao caso em que ***x* é admissível**. No caso em que x é inadmissível, o conjunto das soluções candidatas é definido sempre da mesma forma – a partir de movimentos que reduzem a inadmissibilidade – e é seleccionado o movimento que melhor contribui para repor a admissibilidade de acordo com a rotina VAR_A_COMUTAR.

O processo evolutivo do algoritmo foi ditado por experiências preliminares das diferentes versões. Para estas experiências considerámos os 8 problemas de *knapsack* multidimensional usados anteriormente para testar *0-1SA*: PET5, PET6, PET7, SENTO1, SENTO2, WEING6, WEING7 e WEING8. Para cada problema fizemos testes com vários tamanhos da lista tabu: L= 6, 7, 8 e 9. Os comentários que fizermos aos resultados computacionais de cada versão referem-se exclusivamente a estas experiências, em que mencionaremos como um 'caso' o teste de um problema para um valor particular de L.

VERSÃO 1

- Tem uma única fase de pesquisa que utiliza como registo de memória apenas a *lista tabu*. O processo de pesquisa termina ao fim de *Nrep* iterações (valor definido *a priori* como parâmetro de entrada do algoritmo).
- Se x é admissível, *Candidatos_N*(x) é o conjunto das soluções obtidas a partir de x pela troca do valor de uma variável que não pertence à *lista tabu*.
- A função *avaliação*(y) = $c(y)$ + Pv é usada para avaliar as soluções $y \in$ *Candidatos_N*(x) (para x admissível). A constante de penalização adoptada foi $P = c^{med}$ com $c^{med} = \sum_{j=1}^{n} |c_j| \Big/ n$.

  Nas experiências que fizemos, começámos por considerar separadamente três valores para P: $P = c^{min}$ com $c^{min} = \min_{j=1,\ldots,n} \{|c_j| : c_j \neq 0\}$, $P = c^{max}$ com $c^{max} = \max_{j=1,\ldots,n} |c_j|$ e $P = c^{med}$.

  Os resultados foram significativamente piores para $P = c^{min}$ do que para qualquer um dos outros valores de P. Isto talvez se deva ao facto de o algoritmo trabalhar com um número muito significativo de soluções inadmissíveis quando $P = c^{min}$ (nos casos testados foi cerca de metade das soluções avaliadas). Pelo contrário, para $P = c^{max}$ o algoritmo não aceita uma solução inadmissível se houver uma solução vizinha

admissível. O algoritmo pode, portanto, correr o risco de não diversificar a pesquisa. Apesar de os resultados das nossas experiências não terem sido em geral piores com $P = c^{max}$ do que com $P = c^{med}$, resolvemos adoptar $P = c^{med}$ para a experiência seguinte por permitir a aceitação de um maior leque de soluções vizinhas.

Considerámos $Nrep$ =200. Experiências com $Nrep$ =300 melhoraram a solução final em apenas 2 dos 32 casos (32 = 8 problemas × 4 valores de L).

VERSÃO 2.0

- Relativamente à versão 1 acrescentámos duas fases, uma de **diversificação** da pesquisa (2ª fase) e uma última de **intensificação** da pesquisa (3ª fase). O número total de iterações Nrep é dividido igualmente pelas três fases.
- A 2ª fase (diversificação) usa um registo de memória de frequência dos valores das variáveis. Este registo, designado por *frequencia1*, é um vector de dimensão n (número de variáveis) que conta o número de vezes que cada variável tomou o valor 1 em soluções admissíveis. *Frequencia1* é usado na definição de *Candidatos_N*(x) da 2ª fase da seguinte forma: uma solução vizinha de x é candidata a avaliação se resulta da comutação de uma variável x_j não tabu que verifica $\begin{cases} frequencia1_j < filtro1 & \text{se } x_j = 0 \to 1 \\ frequencia1_j > filtro0 & \text{se } x_j = 1 \to 0 \end{cases}$. No início da 2ª fase, *filtro1* e *filtro0* são iguais à média dos valores de *frequencia1*. Mas, se em alguma altura, o conjunto *Candidatos_N*(x) for vazio, então os filtros são relaxados: aumenta-se o *filtro1* e diminui-se o *filtro0* de uma unidade de cada vez, até *Candidatos_N*(x) deixar de ser vazio. Começa-se a registar dados em *frequencia1* desde o início da 1ª fase e, tanto este registo como os filtros, são actualizados durante a 2ª fase.
- A 3ª fase (intensificação) inicia-se com a melhor solução encontrada nas fases anteriores. Usa 2 registos de memória baseados na qualidade das soluções obtidas anteriormente. Designados por *registo1_boas* e *registo1_mas*, são vectores de dimensão n que contabilizam o número de vezes que cada variável tomou o valor 1 em soluções admissíveis consideradas historicamente como "boas" e "más", respectivamente. O preenchimento destes registos começa no início da 2ª fase e prolonga-se até ao final da 3ª fase. Para estabelecer o conceito de "boas" e "más", consideramos o seguinte: sejam x^+ e x^-, respectivamente, a melhor e a pior soluções admissíveis para a função custo $c(x)$ obtidas até ao final da 1ª fase; uma solução y é considerada "boa" se $c(y) \le c(x^+) + \delta$ e "má" se $c(y) \ge c(x^-) - 2\delta$, com $\delta = [c(x^-) - c(x^+)]/4$. Entre x^+ e x^- existe, portanto, um intervalo de soluções "boas", um intervalo de soluções "intermédias" (que não são tidas em conta) e um intervalo (maior) de soluções "más". R*egisto1_boas* e *registo1_mas* são usados na definição de *Candidatos_N*(x) da 3ª

fase da seguinte forma: uma solução vizinha de x é candidata a avaliação se resulta da comutação de uma variável x_j não tabu que verifica

$$\begin{cases} (registo1\_boas_j \geq b1 \text{ e } registo1\_mas_j \leq m1) & \text{se } x_j = 0 \rightarrow 1 \\ (registo1\_boas_j \leq b0 \text{ e } registo1\_mas_j \geq m0) & \text{se } x_j = 1 \rightarrow 0 \end{cases}.$$

Inicialmente, $b1=b0$ e $m1=m0$ são os valores médios de *registo1_boas* e *registo1_mas*, respectivamente, mas estes limiares são relaxados se não existirem candidatos. Os registos e os limiares são actualizados durante a 3ª fase sempre que é obtida uma solução admissível.

> Para os testes computacionais considerámos $Nrep$ = 300 (100 iterações em cada fase). Relativamente aos resultados da versão 1 com 300 iterações, a versão 2.0 obteve melhores resultados em 7 dos 32 casos (cerca de 22%) e não piorou nenhum outro.

Como já referimos atrás, a versão 1 não foi conclusiva face ao factor de penalização P. Resolvemos então experimentar uma versão semelhante à versão 2.0, mas que utiliza um factor de penalização dinâmico. O algoritmo começa com valores de P baixos e termina com valores elevados. Os valores baixos favorecem a diversificação, porque permitem uma oscilação entre o espaço admissível e o espaço inadmissível, e os valores elevados favorecem uma intensificação da pesquisa.

VERSÃO 2.1

- Esta versão teve por base a versão 2.0 diferindo apenas no valor de P que varia desde c^{min} até c^{max}: $P_{i+1} = \phi P_i$ em que i designa o índice da iteração e ϕ é uma constante calculada de modo a que $P_0 = c^{min}$ e $P_{Nrep} = c^{max}$.

> A comparação do desempenho da versão 2.1 com o da versão 2.0 não foi conclusiva. A versão 2.1 sugeriu uma maior instabilidade dos resultados relativamente ao tamanho da lista tabu (L) do que a versão 2.0. O melhor resultado para um problema, de entre os diferentes L, foi em geral melhor com a versão 2.1 do que com a versão 2.0. Mas, por outro lado, o pior resultado da versão 2.1 foi muitas vezes pior que o da versão 2.0.

Em nenhuma das experiências anteriores foram usados *critérios de aspiração*. Experimentámos então introduzir nas versões 2.0 e 2.1 o *critério de aspiração* mais comum ('standard') que se traduz pelo seguinte: se existir um movimento que conduza à melhor solução encontrada até ao momento, então é esse o movimento escolhido independentemente de ser tabu ou de não satisfazer condições de baixa frequência (2ª fase) ou outras condições baseadas na qualidade histórica das soluções (3ª fase).

A introdução deste critério de aspiração melhorou os resultados anteriores, fazendo-o de forma mais significativa na versão 2.1 do que na versão 2.0.

Uma questão que ainda não discutimos diz respeito à escolha do número de iterações (*Nrep*) a efectuar para cada problema. Este número não deve ser muito elevado para que a heurística se torne competitiva. Mas, por outro lado, se o número de iterações for muito baixo, pode haver uma paragem prematura do algoritmo com uma solução final de fraca qualidade.

Fizemos testes com as versões 2.0 e 2.1 para 300 e 900 iterações. O aumento do número de iterações na versão 2.0 não melhorou os resultados da maior parte dos problemas (o que já tinha acontecido na passagem de 200 para 300 iterações na versão 1). Na versão 2.1 os resultados melhoraram num número de casos mais significativo. Houve, porém, casos em que a melhor solução foi encontrada ao fim de poucas iterações, efectuando-se a seguir muitas iterações que não melhoraram a solução final. Isto levou-nos a analisar uma condição diferente para a paragem do algoritmo, definida não em termos do número global de iterações (*Nrep*), mas do número de iterações consecutivas sem que haja actualização da melhor solução.

As experiências anteriores mostraram-nos ainda que uma melhor solução de partida para a 3ª fase (por vezes diferente para 300 e 900 iterações) nem sempre conduzia a um melhor resultado final. Por vezes, a utilização de outra solução de partida para a 3ª fase conduzia a um resultado final superior. Assim, julgámos que poderia ser vantajoso repetir mais do que uma vez a fase de intensificação, por exemplo 2 ou 3 vezes, considerando como soluções de partida as 2 ou 3 melhores soluções conhecidas até ao final da 2ª fase.

Por estas razões, testámos uma outra versão de *0-1TS* com as seguintes características.

VERSÃO 3

Esta versão teve por base a versão 2.1 diferindo nos seguintes pontos:

- Considera o *critério de aspiração* 'standard' (em todas as fases da pesquisa).
- A passagem de uma fase à fase seguinte e a paragem do algoritmo são estabelecidas por um intervalo máximo de iterações (ΔN) permitido entre duas actualizações consecutivas da melhor solução (definido *a priori* como parâmetro de entrada do algoritmo).
- O factor de penalização P é actualizado de forma diferente. Atendendo a que não existe um número global de iterações *Nrep* para definir ϕ como na versão 2.1, P é agora crescente por períodos, com valores entre c^{min} e c^{max}. Designando por P_i^j o valor de P na iteração i da fase j (j =1, 2, 3), $P_{i+1}^j = \begin{cases} \phi.P_i^j & \text{se } i \text{ não é divisível por } \Delta N \\ P_0^j & \text{caso contrário} \end{cases}$, em que ϕ é calculada tendo por base um crescimento de P de c^{min} a c^{max} em $3\Delta N$ iterações, ou seja, $\phi = \left(c^{max}/c^{min}\right)^{1/3\Delta N}$ (ver figura VII.11); $P_0^1 = c^{min}$, $P_0^2 = \phi^{\Delta N} c^{min}$ e $P_0^3 = \phi^{2\Delta N} c^{min}$. Desta forma, o valor de P no início da 2ª e da 3ª fase é o valor de P no final da 1ª e da 2ª fase, respectivamente. Dentro de cada

fase, P tem um comportamento periódico de período ΔN, conforme se ilustra na figura VII.12.

- Após a fase de diversificação (2ª fase) seguem-se 3 etapas de intensificação (3ª fase) partindo de cada uma das 3 melhores soluções encontradas até ao final da 2ª fase.

Para testar esta versão em comparação com as anteriores – em particular, com a versão 2.1 que lhe serviu de base – considerámos valores de ΔN que conduzissem a um número total de iterações (*Nrep*) próximo de 300. ΔN variou entre 50 e 70, dependendo do problema, totalizando um *Nrep* entre 283 e 307. Os resultados revelaram-se superiores ou iguais aos da versão 2.1 com critério de aspiração, em quase todos os casos. Exceptua-se o problema WEING8 em que a versão 2.1 foi melhor em 3 dos 4 casos testados (L=7, 8 e 9).

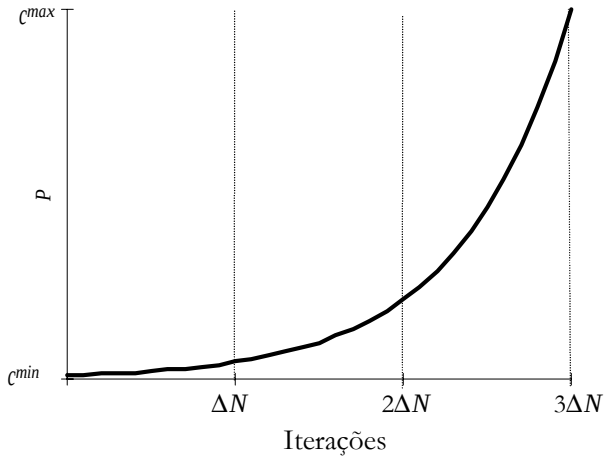


Fig. VII.11 – Evolução de P na versão 3 de 0-1TS se cada fase durasse exactamente ΔN iterações.

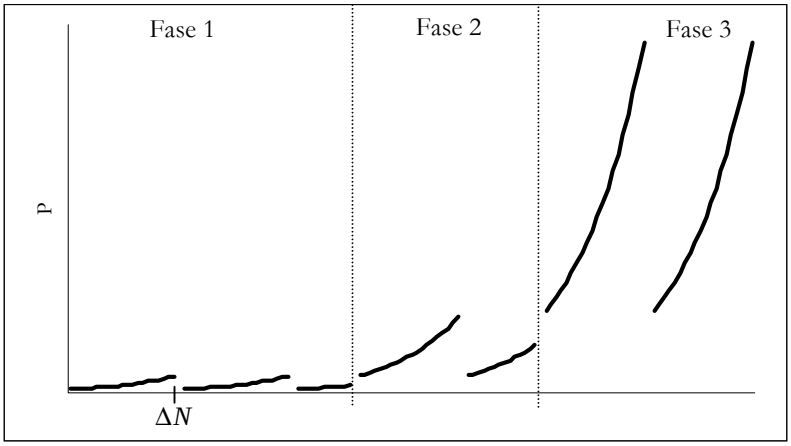


Fig. VII.12 – Um exemplo da evolução de P na versão 3 de 0-1TS.

A versão 3 foi aquela que adoptámos e que designámos no texto por *0-1TS*. Os parâmetros de entrada são ΔN e L.

Capítulo VIII

Conclusões

Neste capítulo revemos as principais contribuições deste trabalho, fazemos uma síntese das conclusões mais significativas e apontamos algumas vias de investigação futura nesta área.

O objectivo central do trabalho foi o de contribuir com novas metodologias para o apoio à decisão em problemas de programação linear inteira multiobjectivo (PLIMO) e problemas de programação linear inteira-mista multiobjectivo (PLIMMO). No cumprimento desse objectivo, desenvolvemos métodos interactivos de pontos de referência particularmente vocacionados para *pesquisas direccionais* de soluções eficientes (não dominadas). Desenvolvemos ainda uma outra abordagem interactiva para problemas de PLIMO com variáveis binárias que utiliza meta-heurísticas (*simulated annealing* e *tabu search*) para o cálculo de soluções aproximadas das soluções não dominadas.

Os métodos desenvolvidos procuram satisfazer os seguintes requisitos, que julgamos importantes para apoiar eficazmente o AD na análise de problemas de PLIMO/PLIMMO:

– possuir um protocolo simples de interacção com o AD, não exigindo demasiada informação sobre as preferências do AD em cada interacção.
– Assumir uma 'comunicação aberta' com o AD em que o processo de decisão não é orientado por qualquer função utilidade implícita. O processo de decisão termina quando o AD considerar que encontrou uma solução de compromisso satisfatória.
– Reduzir o esforço computacional envolvido no cálculo de soluções eficientes relativamente à abordagem tradicional que resolve separadamente cada programa escalarizante.

Vários autores têm admitido a importância de que o esforço computacional não deve ser excessivo para não comprometer o processo interactivo, o que é particularmente relevante em problemas de PLIMO/PLIMMO. Algumas propostas têm sido feitas nesse sentido, mas seguindo sempre uma de duas linhas de orientação. A primeira consiste em diminuir o número de programas escalarizantes que se resolvem por cada interacção, o que pode implicar a

apresentação de menos informação ao AD em cada interacção. A segunda consiste em utilizar técnicas heurísticas nas fases de cálculo ou calcular soluções contínuas que são eficientes para a relaxação linear do problema, o que implica a apresentação de informação mais "pobre" ao AD. Não desprezamos nenhuma destas vias, e chegámos mesmo a experimentar a segunda na abordagem que propusemos baseada em meta-heurísticas. O cálculo de soluções eficientes aproximadas através de técnicas heurísticas talvez seja o único processo viável para determinados problemas de grandes dimensões. Julgamos, contudo, que as vias possíveis para reduzir o esforço computacional não se esgotam nestas duas. Foi nessa perspectiva que decidimos investigar formas de aproveitamento do cálculo de uma solução eficiente para obter outras soluções eficientes, usando técnicas exactas. Este estudo conduziu ao desenvolvimento de formas de resolução "encadeada" dos sucessivos programas escalarizantes de uma pesquisa direccional.

Começámos por utilizar planos de corte, porque se afiguravam interessantes do ponto de vista teórico para o cumprimento dessa tarefa. Em face da instabilidade numérica decorrente do uso de planos de corte em programação inteira monocritério, e que se repercutia na abordagem multicritério, decidimos conduzir a investigação seguinte com base na técnica de *branch-and-bound*. O resultado foi manifestamente vantajoso.

Assim, propusemos dois métodos interactivos de pontos de referência, em que o primeiro apenas se aplica a problemas de PLIMO e usa técnicas de planos de corte, e o segundo trata problemas de PLIMO e de PLIMMO e usa *branch-and-bound*. Este métodos oferecem ao AD opções semelhantes de pesquisa de soluções eficientes, diferindo apenas nas técnicas usadas para o fazer. O segundo método revelou-se muito superior ao primeiro, visto que é mais robusto e permite tratar uma gama mais alargada de problemas.

No seguimento dos métodos anteriores, desenvolvemos um sistema computacional de apoio à decisão que integra os referidos métodos, bem como outras ferramentas auxiliares de pesquisa. Estas ferramentas incluem meios complementares para o cálculo de soluções eficientes (tais como, a optimização de somas pesadas das funções objectivo adequada para uma pesquisa estratégica inicial ou a introdução de limitações adicionais nos valores das funções objectivo). Incluem também diferentes formas de apresentação de resultados, sendo uma delas a representação gráfica de regiões de indiferença associadas aos pontos de referência que conduzem às soluções não dominadas calculadas (esta representação está limitada a problemas de PLIMO com duas ou três funções objectivo). Esta representação gráfica vem no seguimento do estudo teórico que efectuámos sobre pontos de referência que conduzem à mesma solução não dominada.

Não tendo sido nossa intenção descrever detalhadamente o funcionamento do sistema computacional, apresentámos nesta dissertação apenas algumas das suas características aquando da análise de um problema de localização-transporte.

Como já referimos atrás, propusemos também uma abordagem interactiva com meta-heurísticas para o tratamento de problemas de PLIMO com variáveis binárias. Esta abordagem foi integrada no sistema computacional anterior.

Em síntese, as principais contribuições deste trabalho são as seguintes:

– Concepção de um método interactivo de pontos de referência para problemas de PLIMO recorrendo a técnicas de planos de corte.

– Concepção um método interactivo de pontos de referência para problemas de PLIMO e PLIMMO recorrendo a técnicas de *branch-and-bound*.

  A contribuição mais significativa destes métodos de pontos de referência está na forma como são conduzidas as pesquisas direccionais. O AD indica a função objectivo que pretende melhorar em cada instante, ficando a cargo do método o ajuste automático do ponto de referência. Após esse ajuste, o método determina a solução eficiente seguinte da pesquisa direccional aproveitando, para o efeito, informação do cálculo da solução anterior.

– Experimentação de uma abordagem combinada dos dois métodos anteriores.

– Estudo teórico de regiões de indiferença associadas aos pontos de referência em problemas de PLIMO, e sua representação gráfica.

– Análise de um problema multiobjectivo de localização-transporte.

– Desenvolvimento de uma abordagem interactiva para problemas de PLIMO 0-1 que recorre a meta-heurísticas para o cálculo aproximado de soluções não dominadas.

– Implementação computacional de um sistema de apoio à decisão que integra os desenvolvimentos anteriores.

O sistema computacional permitiu-nos testar e aferir os anteriores desenvolvimentos. Das experiências que realizámos, para além da respectiva análise teórica, consideramos que o método baseado em *branch-and-bound* é a contribuição mais relevante deste trabalho.

Podemos resumir as principais conclusões deste trabalho nos seguintes pontos:

– O ajuste automático do ponto de referência durante as pesquisas direccionais revelou-se bastante útil na medida em que garante o cálculo de soluções não dominadas próximas,

mas distintas, ao longo de uma dada direcção. As pesquisas direccionais são particularmente interessantes para uma fase avançada do processo de decisão em que o AD já possui um conhecimento global do problema.

– O primeiro método de pontos de referência usa planos de corte como única técnica de resolução dos programas escalarizantes inteiros. Não nos propusemos fazer investigação na área dos planos de corte, mas apenas utilizar alguns dos planos de corte mais conhecidos. A instabilidade numérica das técnicas utilizadas torna o método pouco robusto e limita naturalmente a sua aplicabilidade. A resolução "encadeada" dos sucessivos programas escalarizantes não parece, porém, aumentar as dificuldades numéricas.

– O método baseado em *branch-and-bound* mostrou-se muito mais robusto do que o anterior. As experiências computacionais mostraram que as pesquisas direccionais são eficazes. A estratégia consiste em aproveitar a informação da anterior árvore de *branch-and-bound* para ajustar o ponto de referência (por análise de sensibilidade) e resolver o programa escalarizante seguinte. Na maior parte dos problemas testados, o tempo de cálculo de uma solução foi reduzido relativamente ao tempo que seria necessário para resolver, desde o início, o correspondente programa escalarizante.

– Nas experiências computacionais efectuadas, o uso da abordagem combinada (entre o método baseado em *branch-and-bound* e o de planos de corte) revelou-se, em geral, pior do que o uso isolado do método baseado em *branch-and-bound*. Apenas numa classe dos problemas testados a situação se inverteu. Esta é uma classe de problemas em que um dos tipos de planos de corte introduzidos permite, de facto, acelerar a resolução dos programas escalarizantes. Este caso reforça a ideia de que a incorporação de outras desigualdades válidas fortes permitiria melhorar o desempenho das abordagens propostas, quer a que usa apenas planos de corte, quer a combinada.

– As pesquisas direccionais permitem alcançar qualquer solução não dominada do problema multiobjectivo. O AD pode percorrer diferentes partes do conjunto das soluções não dominadas através da mudança de direcção, podendo também alterar o ponto de referência de partida para recomeçar a pesquisa de outro local. As direcções possíveis são k (tantas quantas as funções objectivo).

  Em problemas bi-objectivo, este tipo de pesquisa traduz-se num "varrimento" de soluções no sentido do melhoramento de uma das funções objectivo. De uma solução para a seguinte, o AD sabe que a função objectivo escolhida irá melhorar piorando a outra. Pode inclusive calcular todas as soluções eficientes – basta começar num extremo (óptimo de uma das funções) e terminar no outro. Mas, em problemas com mais do que

duas funções objectivo, o deslocamento é feito segundo uma direcção que pode não ser a única que melhora a função objectivo escolhida pelo AD. O AD pode sentir a necessidade de ter também algum controlo na variação das outras funções, o que não lhe é permitido com as pesquisas direccionais simples. Nesta perspectiva, propusemos a introdução de limitações adicionais nos valores das funções objectivo, um tipo de informação de preferências que o AD está possivelmente mais habilitado a fornecer. Julgamos, porém, que devem ser investigadas outras formas auxiliares de condução da pesquisa.

– Na análise do problema de PLIMMO de localização-transporte verificámos algumas das vantagens, já referidas atrás, das pesquisas direccionais. Do ponto de vista cognitivo, podemos salientar a vantagem da condução automática da pesquisa (através do ajuste do ponto de referência), que percorre troços contínuos de soluções não dominadas e salta automaticamente as descontinuidades. Do ponto de vista técnico, podemos referir a rapidez do cálculo das sucessivas soluções não dominadas. A combinação com limitações auxiliares nos valores das funções objectivo mostrou-se, várias vezes, indispensável. Contudo, sentimos igualmente a necessidade de proceder a uma pesquisa estratégica inicial que fornecesse um conhecimento de carácter mais global sobre o problema.

– A representação gráfica de sub-regiões de indiferença associadas aos pontos de referência de problemas de PLIMO é, do nosso ponto de vista, uma ferramenta útil para auxiliar o AD na pesquisa de novas soluções não dominadas. Neste gráfico podem visualizar-se as regiões onde se localizam os pontos de referência que conduzem à mesma solução não dominada – e, por isso, a escolha desses pontos levaria a cálculos desnecessários – fornecendo uma indicação da distribuição espacial dessas soluções. Trata-se, porém, de uma informação parcial porque apenas definimos sub-regiões de indiferença para cada solução. Seria claramente vantajoso se conseguíssemos definir a região de indiferença completa correspondente a cada nova solução não dominada que se calculasse.

– A utilização de meta-heurísticas em abordagens multiobjectivo permite lidar com problemas em que os métodos interactivos “exactos” não se mostram adequados (pelo tempo ou quantidade de recursos que exigiriam). A contrapartida é a obtenção de soluções que podem não ser eficientes. Na abordagem que desenvolvemos preocupámo-nos com a qualidade das soluções geradas, pelo que nos concentrámos essencialmente no desempenho dos algoritmos de *simulated annealing* e *tabu search* (monocritério) que resolvem os programas escalarizantes.

  A maior parte das versões de meta-heurísticas que se encontram na literatura da área é dedicada a classes de problemas particulares. Estamos certos que o desempenho dessas

versões nos problemas a que se destinam deverá ser superior ao de um algoritmo de carácter genérico. No entanto, pretendemos construir algoritmos genéricos que pudessem ser usados numa gama alargada de problemas. Testámo-los em problemas monocritério de mochila (*knapsack*) com múltiplas restrições – problemas muito utilizados como padrão (*benchmark*) na literatura da área – e observámos que as soluções obtidas foram, em geral, tão boas ou melhores do que as obtidas por outros algoritmos genéricos de meta-heurísticas. Porém, as experiências foram limitadas, mesmo nos problemas monocritério. E, no que diz respeito aos problemas multiobjectivo, a aferição da qualidade das soluções potencialmente eficientes torna-se ainda mais difícil. Pelos exemplos que testámos, julgamos que é possível obter um compromisso aceitável entre a qualidade das soluções e o tempo computacional mas, como em qualquer outra abordagem heurística, só experiências mais exaustivas poderiam confirmar ou desmentir esta ideia.

As conclusões do trabalho suscitam-nos algumas vias de investigação futura.

Vimos as potencialidades e limitações das *pesquisas direccionais* no método baseado em *branch-and-bound*. A pesquisa de soluções eficientes é eficaz (em diferentes aspectos) mas de carácter fundamentalmente "local", e são limitadas as opções oferecidas para as direcções de deslocamento. Ao permitir que o AD introduza restrições adicionais nos valores das funções objectivo, o método torna-se mais flexível concedendo ao AD maior controlo na pesquisa de soluções eficientes. Esta opção foi incluída no método, mas não explorámos processos que aproveitem os cálculos anteriores para obter a primeira solução eficiente da nova pesquisa. Na versão actual do método, destrói-se a árvore anterior de *branch-and-bound* e recomeça-se uma nova árvore para o programa escalarizante modificado. A extensão do processo de actualização da árvore de *branch-and-bound* para esta situação é um trabalho que julgamos merecer um empenho futuro.

No contexto deste método abrem-se ainda outras perspectivas de desenvolvimento futuro. Uma delas consiste em investigar processos de análise de sensibilidade e aproveitamento da árvore de *branch-and-bound* para novas direcções de pesquisa de soluções eficientes. Uma outra via de trabalho consiste na exploração do método para casos particulares de problemas multiobjectivo com relevância do ponto de vista prático.

Estudámos o comportamento dos pontos de referência quando usados na métrica de Tchebycheff para gerar soluções eficientes, o que conduziu à definição de sub-regiões de indiferença das soluções eficientes associadas aos pontos de referência. Seria interessante continuar este estudo com vista a aumentar o conhecimento teórico nessa matéria e, em particular, poder definir regiões de indiferença mais alargadas.

Uma outra via de investigação consiste na aplicação das metodologias propostas no estudo de casos. Um estudo preliminar mostrou que o método interactivo de pontos de referência (baseado em *branch-and-bound*) poderia ser útil na análise de um problema de selecção de estratégias de controlo remoto de cargas numa rede de distribuição de energia eléctrica (brevemente introduzido no capítulo V). Esta é uma aplicação que perspectivamos, sendo para tal indispensável a intervenção de peritos da área a que o problema diz respeito.

A abordagem interactiva com meta-heurísticas destina-se actualmente a problemas multiobjectivo com variáveis binárias. A extensão, em primeiro lugar, ao caso 0-1 misto e, em segundo lugar, aos casos inteiro (puro) e inteiro-misto constituem direcções de trabalho a explorar. Além disso, a abordagem deveria ser testada com mais problemas e problemas de maiores dimensões, o que permitiria uma aferição mais acurada, aspecto importante para a sua utilização futura no estudo de casos.

Por fim, e no âmbito do desenvolvimento de *software*, seria interessante investigar a possibilidade de ligar o sistema computacional (que integra as metodologias propostas) a códigos comerciais de resolução de problemas de programação inteira e inteira-mista. Actualmente o sistema não recorre a qualquer *software* comercial. Por um lado, pode ser facilmente distribuído e utilizado porque não necessita de recursos especiais (apenas um PC com o Windows 95 ou 98). Por outro lado, o seu desempenho poderia ser melhorado pela utilização de sub-rotinas mais robustas existentes hoje em dia em *software* comercial.

Referências

Aarts, E. e J. Korst (1989). *Simulated Annealing and Boltzmann machines*. Wiley, Chichester.

Aboudi, R. e K. Jörnsten (1994). Tabu search for general zero-one integer programs using the pivot and complement heuristic. *ORSA Journal on Computing*, vol. 6 (1), 82-93.

Abramson, D. e M. Randall (1999). A simulated annealing code for general linear programs. *Annals of Operations Research*, vol. 86, 3-21.

Abramson, D., H. Dang e M. Krishnamoorthy (1996). A comparison of two methods for solving 0-1 integer programs using a general purpose simulated annealing algorithm. *Annals of Operations Research*, vol. 63, 129-150.

Acevedo, J. e E. N. Pistikopoulos (1999). An algorithm for multiparametric mixed-integer linear programming problems. *Operations Research Letters*, vol. 24, 139-148.

Aksoy, Y. (1990). An interactive branch-and-bound algorithm for bicriterion nonconvex/mixed integer programming. *Naval Research Logistics*, vol. 37, 403-417.

Alves, M. J. e J. Clímaco (1997). Um procedimento interactivo de pontos de referência para programação linear inteira multiobjectivo – Abordagem com recurso a planos de corte. *Relatório de Investigação, Série Métodos Científicos de Gestão*, nº 3 de 1997, Faculdade de Economia da Universidade de Coimbra.

Alves, M. J. e J. Clímaco (1998[A]). Multiobjective mixed-integer programming. Artigo convidado e aceite em 1998 para integrar a *Encyclopedia of Optimization,* P. M. Pardalos e C. A. Floudas (Eds.), Kluwer Academic Publishers.

Alves, M. J. e J. Clímaco (1998[B]). Multiobjective mixed-integer programming: an interactive approach. *Relatório de Investigação, Série Métodos Científicos de Gestão*, nº 2 de 1998, Faculdade de Economia da Universidade de Coimbra.

Alves, M. J. e J. Clímaco (1999A). Using cutting planes in an interactive reference point approach for multiobjective integer linear programming problems. *European Journal of Operational Research*, vol. 117 (3), 565-577.

Alves, M. J. e J. Clímaco (1999B). On indifference sets of reference points in multiobjective integer linear programming. Submetido para publicação em 1999.

Alves, M. J. e J. Clímaco (1999C). Aplicação de uma abordagem interactiva baseada em pontos de referência a um problema multiobjectivo de localização-transporte. *Relatório de Investigação, Série Métodos Científicos de Gestão*, nº 3 de 1999, Faculdade de Economia da Universidade de Coimbra.

Alves, M. J. e J. Clímaco (2000A). An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research*, vol. 124 (3), 478-494.

Alves, M. J. e J. Clímaco (2000B). An interactive method for 0-1 multiobjective problems using simulated annealing and tabu search. *Journal of Heuristics*, vol. 6 (3), 385-403.

Bäck, T., M. Schütz e S. Khuri (1996). A comparative study of a penalty function, a repair heuristic, and a stochastic operators with the set-covering problem. In J. M. Alliot, E. Lutton, E. Ronald, M. Schoenhauer, D. Snyers (Eds), *Artificial Evolution*, 3-20, Springer-Verlag, Berlin.

Bailey, M. G. e B. E. Gillett (1980). Parametric integer programming analysis: a contraction approach. *Journal of the Operational Research Society*, vol. 31, 257-262.

Balas, E. (1965). An additive algorithm for solving linear problems with zero-one variables. *Operations Research*, vol. 15, 517-546.

Balas, E. (1971). Intersection cuts – a new type of cutting planes for integer programming. *Operations Research*, vol. 19, 19-39.

Balas, E. e C. H. Martin (1980). Pivot and Complement – a heuristic for 0-1 programming. *Management Science*, vol. 26, 86-96.

Balas, E., S. Ceria e G. Cornuéjols (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, vol. 58, 295-324.

Balas, E., S. Ceria e G. Cornuéjols (1996A). Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, vol. 42 (9), 1229-1246.

Balas, E., S. Ceria, G. Cornuéjols e N. Natraj (1996[B]). Gomory cuts revisited. *Operations Research Letters*, vol. 19, 1-9.

Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of Operational Research Society*, vol. 41 (11), 1069-1072. Problemas disponíveis na WWW a partir de http://mscmga.ms.ic.ac.uk/info.html.

Benayoun, R., J. Montgolfier, J. Tergny e O. Larichev (1971). Linear programming with multiple objective functions: Step method (STEM). *Mathematical Programming*, vol. 1, 366-375.

Bitran, G. R. (1977). Linear multiple objective programs with zero-one variables. *Mathematical Programming*, vol. 13, 121-139.

Bitran, G. R. (1979). Theory and algorithms for linear multiple objective programs with zero-one variables. *Mathematical Programming* vol. 17 (3), 362-389.

Bouyssou, D. (1993). Décision multicritère ou aid multicritère? *Newsletter of the European Working Group "Multicriteria Aid for Decisions"*, series 2, nº 2.

Bowman Jr., V. J. (1976). On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thiriez, S. Zionts (Eds), Multiple Criteria Decision Making, *Lecture Notes in Economics and Mathematical Systems*, vol. 130, 76-86, Springer-Verlag, Berlin.

Boyd, E. A. (1994). Fenchel cutting planes for integer programs. *Operations Research*, vol. 42 (1), 53-64.

Brearly, A. L., G. Mitra e H. P. Williams (1975). Analysis of mathematical programming problems prior applying the simplex algorithm. *Mathematical Programming*, vol. 8, 54-83.

Ceria, S., C. Cordier, H. Marchand e L. A. Wolsey (1998). Cutting planes for integer programs with general integer variables. *Mathematical Programming*, vol. 81, 201-214.

Chalmet, L. G., L. Lemonidis e D. J. Elzinga (1986). An algorithm for the bi-criterion integer programming problem. *European Journal of Operational Research*, vol. 25, 292-300.

Clímaco, J., C. Ferreira e M. E. Captivo (1997). Multicriteria integer programming: an overview of the different algorithmic approaches. In João Clímaco (Ed.), *Multicriteria Analysis*, 248-258, Springer-Verlag, Berlin.

Cohon, J. L., R. L. Church e D. P. Sheer (1979). Generating multiobjective tradeoffs: an algorithm for bicriterion problems. *Water Resources Research*, vol. 15, 1001-1010.

Connolly, D. (1992). General purpose simulated annealing. *Journal of the Operational Research Society*, vol. 43 (5), 495-505.

Coutinho-Rodrigues, J., J. Clímaco, J. Current e S. Ratick (1997). A guided tour to an interactive location-routing software package using an HAZMAT case. *Working Paper*.

Coutinho-Rodrigues, J., J. Current, J. Clímaco e S. Ratick (1995). An interactive spatial decision support system for multiobjective HAZMAT location-routing problems. *WPS 95-39*, Max M. Fisher College of Business, The Ohio State University, 1995. Uma outra versão foi publicada em *Transportation Research Board*, Record 1602, 101-109, 1997.

Crema, A. (1997). A contraction algorithm for the multiparametric integer linear programming problem. *European Journal of Operational Research*, vol. 101, 130-139.

Crema, A. (1998). A procedure to verify the completeness of the right-hand-side parametric analysis for a mixed integer linear programming problem. *European Journal of Operational Research*, vol. 108, 684-695.

Crema, A. (1999). An algorithm to perform a complete right-hand-side parametrical analysis for a 0-1 integer linear programming problem. *European Journal of Operational Research*, vol. 114, 569-579.

Crowder, H., E. L. Johnson e M. Padberg (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, vol. 31 (5), 803-834.

Current, J. e S. Ratick (1995). A model to assess risk, equity and efficiency in facility location and transportation of hazardous materials. *Location Science*, vol. 3 (3), 187-201.

Czyżak, P. e A. Jaszkiewicz (1996). A multiobjective metaheuristic approach to the location of petrol stations by the capital budgeting model**.** *Control and Cybernetics*, vol. 25 (1), 177-187 **.**

Czyżak, P. e A. Jaszkiewicz (1998). Pareto Simulated Annealing – a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, vol. 7, 34-47.

Deckro, R. F. e E. P. Winkofsky (1983). Solving zero-one multiple objective programs through implicit enumeration. *European Journal of Operational Research*, vol. 12, 362-374.

Dowsland, K. A. (1993). Simulated Annealing. In C. Reeves (Ed.), *Modern Heuristics for Combinatorial Problems*, 20-69, Blackwell Scientific Publications, Oxford.

Drexl, A. (1988). A Simulated Annealing approach to the multiconstraint zero-one knapsack problem. *Computing*, vol. 40, 1-8.

Durso, A. (1992). An interactive combined branch-and-bound/Tchebycheff algorithm for multiple criteria optimization. In A. Goicoechea, L. Duckstein, S. Zionts (Eds), *Multiple Criteria Decision Making: Theory and Applications in Business, Industry and Government*, 107-122, Springer-Verlag, New York.

Faigle, U. e W. Kern (1992). Some convergence results for probabilistic tabu search**.** *ORSA Journal on Computing*, vol. 4 (1), 32-37.

Ferreira, C., B. S. Santos, M. E. Captivo, J. Clímaco e C. C. Silva (1996). Multiobjective location of unwelcome or central facilities involving environmental aspects: a prototype of a decision support system. *Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 36 (2-3), 159-172.

Ferreira, Carlos M. S. (1997). *Problemas de localização e distribuição multicritério – aproximações e estudo de alguns casos com implicações ambientais.* Dissertação de doutoramento, Universidade de Aveiro.

Feyerabend, P. (1975). *Against the Method*. New Left Books, London.

Gabbani, D. e M. Magazine (1986). An interactive heuristic approach for multi-objective integer-programming problems. *Journal of the Operational Research Society*, vol. 37 (3), 285-291.

Gal, T. e K. Wolf (1986). Stability in vector maximization – a survey. *European Journal of Operational Research*, vol. 25, 169-182.

Geoffrion, A. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, vol. 22, 618-630.

Geoffrion, A. M. e R. Nauss (1977). Parametric and postoptimality analysis in integer linear programming. *Management Science*, vol. 23 (5), 453-466.

Glover, F. (1986). Future paths for integer programming and links to Artificial Intelligence**.** *Computers and Operations Research*, vol. 13 (5), 533-549.

Glover, F. (1989). Tabu search - Part I. *ORSA Journal on Computing*, vol. 4 (3), 190-206.

Glover, F. (1990A). Tabu search - Part II. *ORSA Journal on Computing*, vol. 2 (1), 4-32.

Glover, F. (1990B). Tabu search: a tutorial. *Interfaces*, vol. 20 (4), 74-94.

Glover, F. e M. Laguna (1993). Tabu Search. In C. Reeves (Ed.), *Modern Heuristics for Combinatorial Problems*, 70-150, Blackwell Scientific Publications, Oxford.

Goemans, M. X. (1989). Valid inequalities and separation for mixed 0-1 constraints with variable upper bounds. *Operations Research Letters*, vol. 8, 315-322.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Mass.

Gomory, R. E. (1963). An algorithm for integer solutions to linear programs. In R. L. Graves, P. Wolfe (Eds), *Recent Advances in Mathematical Programming*, 269-302, McGraw-Hill, New York.

Gonzalez, J. J., G. R. Reeves e L. S. Franz (1985). An interactive procedure for solving multiple objective integer linear programming problems. In Y. Y. Haimes, V. Chankong (Eds.), Decision Making with Multiple Objectives, *Lecture Notes in Economics and Mathematical Systems*, vol. 242, 250-260, Springer-Verlag, Berlin.

Grossman, T. e A. Wool (1997). Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, vol. 101 (1), 81-92.

Gu, Z., G. L. Nemhauser e M. W. P. Savelsbergh (1998). Lifted cover inequalities for 0-1 integer programs: computation. *INFORMS Journal on Computing*, vol. 10 (4), 427-437.

Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research*, vol. 13 (2), 311-329.

Hanne, T. (1999). On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, vol. 117, 553-564.

Hansen, M. P. (1997). Tabu Search for multiobjective optimization: MOTS. Artigo apresentado na "13th Conference on MCDM", University of Cape Town, South Africa, Janeiro 6-10/1997.

Isermann, H. (1974). Proper efficiency and the linear vector maximum problem. *Operations Research*, vol. 22, 189-191.

Jenkins, L. (1982). Parametric mixed integer programming: an application to solid waste management. *Management Science*, vol. 28 (11), 1270-1284.

Jenkins, L. (1987). Parametric-objective integer programming using knapsack facets and Gomory cutting planes. *European Journal of Operational Research*, vol. 31, 102-109.

Johnson, E. L., M. M Kostreva e U. H. Suhl (1985). Solving 0-1 integer programming problems from large scale planning models. *Operations Research*, vol. 33 (4), 803-819.

Jorge, H., C. H. Antunes e A. G. Martins (2000). A multiple objective decision support model for the selection of remote load control strategies. *IEEE Transactions on Power Systems* , vol. 5 (2), 865-872.

Kaliszewski, I. (1994). *Quantitative Pareto Analysis by Cone Separation Technique*. Kluwer Academic Publishers, Boston.

Karaivanova, J. N., S. C. Narula e V. Vassilev (1993). An interactive procedure for multiple objective integer linear programming problems. *European Journal of Operational Research*, vol. 68, 344-351.

Karaivanova, J., P. Korhonen, S. Narula, J. Wallenius e V. Vassilev (1995). A reference direction approach to multiple objective integer linear programming. *European Journal of Operational Research*, vol. 81, 176-187.

Karwan, M. H., S. Zionts, B. Villarreal e R. Ramesh (1985). An improved interactive multicriteria integer programming algorithm. In Y. Y. Haimes, V. Chankong (Eds), Decision Making with Multiple Objectives, *Lecture Notes in Economics and Mathematical Systems*, vol. 242, 261-271, Springer-Verlag, Berlin.

Khuri, S., T. Bäck e J. Heitkötter (1994). The zero-one multiple knapsack problem and genetic algorithms. In E. Deaton, D. Oppenheim, J. Urban, H. Berghel (Eds), *Proceedings of the 1994 ACM Symposium on Applied Computing*, 188-193, ACM-Press, New York.

Kirkpatrick, S., C. D. Gellat e M. P. Vecchi (1983). Optimization by simulated annealing. *Science*, vol. 220, 671-680.

Kiziltan, G. e E. Yucaoglu (1983). An algorithm for multiobjective zero-one linear programming. *Management Science*, vol. 29 (12), 1444-1453.

Klein, D. e E. Hannan (1982). An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research*, vol. 9, 378-385.

Klein, D. e S. Holm (1979). Integer programming post-optimal analysis with cutting planes. *Management Science*, vol. 25 (1), 64-72.

Korhonen, P. e J. Wallenius (1988). A Pareto Race. *Naval Research Logistics*, vol. 35, 615-623.

Korhonen, P., S. Salo e R, Steuer (1997). A heuristic for estimating nadir criterion values in multiple objective linear programming, *Operations Research*, vol. 45 (5), 751-757.

Lewandowski, A. e A. Wierzbicki (1988). Aspiration based decision analysis and support. Part I: Theoretical and methodological backgrounds. *Working paper WP-88-03*, IIASA, Laxenburg, Austria.

Lewandowski, A., T. Kreglewski, T. Rogowski e A. Wierzbicki (1989). Decision support of DIDAS family (Dynamic Interactive Decision Analysis & Support). In A. Lewandowski, A. P. Wierzbicki (Eds), Aspiration Based Decision Support Systems – Theory, Software and Applications, *Lecture Notes in Economics and Mathematical Systems*, vol. 331, 21-47, Springer-Verlag, Berlin.

L'Hoir, H. e J. Teghem (1995). Portfolio selection by MOLP using an interactive branch and bound. *Foundations of Computing and Decision Sciences*, vol. 20 (3), Institute of Computing Science, Technical University of Poznan, Poland.

Løkketangen, A. e F. Glover (1998). Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, vol. 106, 624-658.

Løkketangen, A., K. Jörnsten e S. Storøy (1994). Tabu Search within a pivot and complement framework. *International Transactions in Operations Research*, vol. 1 (3) 305-316.

Marcotte, O. e R. M. Soland (1980). *Branch and bound algorithm for multiple criteria optimization*. University of Cornell and George Washington.

Marcotte, O. e R. M. Soland (1986). An interactive branch-and-bound algorithm for multiple criteria optimization. *Management Science*, vol. 32 (1), 61-75.

Mavrotas, G. e D. Diakoulaki (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, vol. 107, 530-541.

Metev, B. e I. Yordanova (1993). Use of reference points for MOLP problem analysis. *European Journal of Operational Research*, vol. 68, 374-378.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller e E. Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics.*, vol 21, 1087-1092.

Murty, K. G. (1983). *Linear Programming*. Wiley, New York.

Nakayama, H. (1997). Multi-objective mathematical programming. In João Clímaco (Ed), *Multicriteria Analysis*, 126-129, Springer-Verlag, Berlin.

Narula, S. C. e V. Vassilev (1994). An interactive algorithm for solving multiple objective integer linear programming problems. *European Journal of Operational Research*, vol. 79, 443-450.

Nauss, R. (1979). *Parametric Integer Programming*. University of Missouri Press, Columbia.

Nemhauser, G. L. e L. A. Wolsey (1988). *Integer and Combinatorial Optimization*. Wiley, New York.

Ogryczak, W., K. Studzinski e K. Zorychta (1989). A solver for the multi-objective transshipment problem with facility location. . *European Journal of Operational Research*, vol. 43, 53-64.

Ogryczak, W., K. Studzinski e K. Zorychta (1992). DINAS: A computer-assisted analysis system for multiobjective transshipment problems with facility location. *Computers and Operations Research*, vol. 19 (7), 637-647.

Ohtake, Y. e N. Nishida (1985). A branch-and-bound algorithm for 0-1 parametric mixed integer programming. *Operations Research Letters*, vol. 4 (1), 41-45.

Park, M.-W. e Y.-D. Kim (1998). A systematic procedure for setting parameters in simulated annealing. *Computers and Operations Research*, vol. 25 (3), 207-217.

Pasternak, H. e U. Passy (1973). Bicriterion mathematical programs with Boolean variables. In J. L. Cochrane, M. Zeleny (Eds), *Multiple Criteria Decision Making*, 327-348, University of South Carolina Press.

Petersen, C. C. (1967). Computational experience with variants of the Balas algorithm applied to the selection of R&D projects. *Management Science*, vol. 13 (9), 736-750.

Ramesh, R., M. H. Karwan e S. Zionts (1990). An interactive method for bicriteria integer programming. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20 (2), 395-403.

Ramesh, R., S. Zionts e M. H. Karwan (1986). A class of practical interactive branch and bound algorithms for multicriteria integer programming. *European Journal of Operational Research*, vol. 26, 161-172.

Rasmussen, L. M. (1986). Zero-one programming with multiple criteria. *European Journal of Operational Research*, vol. 26, 83-95.

Roodman, G. M. (1972). Postoptimality analysis in zero-one programming by implicit enumeration. *Naval Research Logistics Quarterly*, vol. 19, 435-447.

Rountree, S. L. K. e B. E. Gillett (1982). Parametric integer linear programming: a synthesis of branch and bound with cutting planes. *European Journal of Operational Research*, vol. 10, 183-189.

Roy, B. (1987). Meaning and validity of interactive procedures as tools for decision making, *European Journal of Operational Research*, vol. 31, 297-303.

Schrage, L. e L. Wolsey (1985). Sensitivity analysis for branch and bound integer programming. *Operations Research*, vol. 33, 1008-1023.

Senyu, S. e Y. Toyoda. (1967). An approach to linear programming with 0-1 variables. *Management Science*, vol. 15B, 196-207.

Serafini, P. (1994). Simulated Annealing for multiobjective optimization problems. In G. H. Tzeng, H. F. Wang, U. P. Wen, P. L. Yu (Eds), *Multiple Criteria Decision Making*, 283-292, Springer-Verlag, New York.

Shin, W. S. e D. B. Allen (1994). An interactive paired comparison method for bicriterion integer programming. *Naval Research Logistics*, vol. 41, 423-434.

Simon, H. A. (1957). *Models of Man*. Macmillan, New York.

Soland, R. M. (1979). Multicriteria Optimization: A general characterization of efficient solutions. *Decision Sciences*, vol. 10, 26-38.

Solanki, R. (1991). Generating the noninferior set in mixed integer biobjective linear programs: an application to a location problem. *Computers and Operations Research*, vol. 18 (1), 1-15.

Steuer, R. (1986). *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley, New York.

Steuer, R. E. e E.-U. Choo (1983). An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, vol. 26, 326-344.

Steuer, R. E., J. Silverman e A. W. Whisman (1993). A combined Tchebycheff/aspiration criterion vector interactive multiobjective programming procedure. *Management Science*, vol. 39 (10), 1255-1260.

Teghem Jr., J. e P. L. Kunsch (1986[A]). A survey of techniques for finding efficient solutions to multi-objective integer linear programming. *Asia-Pacific Journal of Operational Research*, vol. 3, 95-108.

Teghem, J. e P. L. Kunsch (1986[B]). Interactive methods for multi-objective integer linear programming. In G. Fandel, M. Grauer, A. Kurzhanski, A. P. Wierzbicki (Eds), Large-Scale Modelling and Interactive Decision Analysis, *Lecture Notes in Economics and Mathematical Systems*, vol. 273, 75-87, Springer-Verlag, Berlin.

Ulungu, E. L. e J. Teghem (1994). Multi-objective combinatorial optimization problems: a survey. *Multi-Criteria Decision Analysis*, vol. 3, 83-104.

Ulungu, E. L., J. Teghem e Ch. Ost (1998). Efficiency of interactive multi-objective simulated annealing through a case study. *Journal of the Operational Research Society*, vol. 49, 1044-1050.

Ulungu, E. L., J. Teghem e Ph. Fortemps (1995). Heuristics for multi-objective combinatorial optimization problem by simulated annealing. In J. Gu, G. Chen, Q. Wei, S. Wang (Eds), *MCDM: Theory and Applications 1995,* Sci-Tech, Windsor, 228-238 .

Van Roy, T. J. e L. A. Wolsey (1987). Solving mixed integer programming problems using automatic reformulation. *Operations Research*, vol 35 (1), 45-57.

Vanderpooten, D. (1992). Three basic conceptions underlying multiple criteria interactive procedures. In A. Goicoechea, L. Duckstein, S. Zionts (Eds), *Multiple Criteria Decision Making: Theory and Applications in Business, Industry and Government*, 441-448, Springer-Verlag, New York.

Vassilev, V. e S. C. Narula (1993). A reference direction algorithm for solving multiple objective integer linear programming problems. *Journal of the Operational Research Society*, vol. 44 (12), 1201-1209.

Villareal, B. e M. H. Karwan (1981). Multicriteria integer programming: A (hybrid) dynamic programming recursive approach. *Mathematical Programming*, vol. 21, 204-223.

Villarreal, B., M. H. Karwan e S. Zionts (1980). An interactive branch and bound procedure for multicriterion integer linear programming. In G. Fandel, T. Gal (Eds), Multiple Criteria Decision Making Theory and Application, *Lecture Notes in Economics and Mathematical Systems*, vol. 177, 448-467, Springer-Verlag, Berlin.

Walker, J. (1978). An interactive method as an aid in solving bicriterion mathematical programming problems. *Journal of the Operational Research Society*, vol. 29 (9), 915-922.

Wang, H.-F. e J.-S. Horng (1996). Structural approach to parametric analysis of an IP – on the case of the right-hand side. *European Journal of Operational Research*, vol. 92, 148-156.

Weingarter, H. M. e D. N. Ness. (1967). Methods for the solution of the multi-dimensional 0/1 knapsack problem. *Operations Research*, vol. 15, 83-103.

White, D. J. (1984). A branch and bound method for multi-objective boolean problems. *European Journal of Operational Research*, vol. 15, 126-130.

White, D. J. (1985). A multiple objective interactive Lagrangean relaxation approach. *European Journal of Operational Research*, vol. 19, 82-90.

White, D. J. (1990). A bibliography on the applications of mathematical programming multiple-objective methods. *Journal of the Operational Research Society*, vol. 41 (8), 669-691.

Wierzbicki, A. P. (1986). On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR Spectrum* 8, 73-87.

Wierzbicki, A. P. (1998). Reference point methods in vector optimization and decision support. *Interim Report IR-98-017*, IIASA, Laxenburg, Austria.

Wolsey, L. A. (1981). Integer programming duality: price functions and sensitivity analysis. *Mathematical Programming*, vol. 20, 173-195.

Wolsey, L. A. (1998). *Integer Programming*, Wiley, New York.

Zionts, S. e J. Wallenius (1983). An interactive multiple linear programming method for a class of underlying nonlinear utility functions. *Management Science*, vol. 29 (5), 519-529.