The rough interval shortest path problem

Ali Moghanni[(1)], Marta Pascoal[(1,2)*]

[(1)] University of Coimbra, CMUC, Department of Mathematics, 3001-501 Coimbra, Portugal
E-mail: {amoghanni,marta}@mat.uc.pt

[(2)] University of Coimbra, CMUC, Department of Mathematics, Institute for Systems Engineering and Computers – Coimbra, Portugal

2 May 2019

Abstract: The shortest path problem is one of the most popular network optimization problems and it is of great importance in areas such as transportation, network design or telecommunications. This model deals with determining a minimum weighted path between a pair of nodes of a given network. The deterministic version of the problem can be solved easily, in polynomial time, but sometimes uncertainty or vagueness is encountered. In this work we consider the rough interval shortest path problem, where each arc's wight is represented by a lower approximation interval and an upper approximation interval, which surely contain the real weight value and that may possibly contain the real weight value, respectively. A labeling algorithm is developed to find efficient solutions of the problem.

Keywords: rough sets, shortest path, labeling, efficient solutions

1 Introduction

The shortest path problem is a classical network optimization model with a wide range of applications in areas such as transportation, network design, telecommunications, etc. This model is used whenever we want to minimize a linear function of a path between a certain pair of nodes of a given network, such as the distance, the time or the cost [1].

Traditionally a weight is associated with each directed arcs of the network and the goal of the problem is to find the shortest path, with respect to the sum of the weights, from an origin to a destination. On the other hand, the weights may fluctuate depending on the real problem conditions, like traffic, payload and so on, and this brings the concepts of vagueness and uncertainty. The rough set theory, proposed by Pawlak in 1982 [6, 7] can be seen as a mathematical approach to vagueness. Although the rough set theory overlaps with other theories, this approach seems to be of fundamental importance in research areas such as machine learning, intelligent systems, inductive reasoning, pattern recognition, knowledge discovery, decision analysis, and expert systems [3, 8]. In such cases, fuzzy numbers have often been used as the arc weights. However, it is not always easy for a decision maker to determine the membership functions, and it may be more intuitive to express them as rough intervals than as fuzzy numbers [9].

The focus of this work is a shortest path problem between a pair of nodes in a network, the arcs of which are associated with rough intervals. Our purpose is to introduce an algorithm for finding the set of efficient paths for this problem. The method results from the combination of

*Corresponding author

the labeling algorithm proposed by Martins [4], modified in order to handle rough intervals, also considering interval arithmetic operation used by Okada and Gen [5].

This paper is structured as follows. In Section 2 notation and some preliminary definitions are presented. In Section 3, the rough interval shortest path problem (RISPP) is formulated, under the assumption that each arc parameter is defined by a lower approximation and an upper approximation intervals. In Section 4, a labeling algorithm is introduced to find efficient paths with respect to the criteria defined before. Section 5 presents computational results, which are followed by concluding remarks.

2 Preliminary Concepts

In this section, we introduce several concepts related with roughness and interval arithmetic used along this paper.

2.1 Rough set theory

The rough set theory intends to jointly manage vagueness and uncertainty. A basic assumption in this theory is that any concept can be defined through the collection of all the objects that exhibit the properties associated with it. Therefore, a vague concept is assumed to be decomposed in two well-delimited concepts that can be handled independently. Namely, for a vague concept R, it can be formulated:

1. a lower approximation $\underline{R}$ containing the elements that are surely included in R, and
2. an upper approximation $\overline{R}$ containing those elements that can possibly be included in R, that is, which cannot, beyond doubt, be excluded from R.

Rough intervals are a particular case of rough sets, used to modeling continuous variables. Such an interval is characterized by two parts: a lower approximation interval and an upper approximation interval, which satisfy the rough set upper approximation and lower approximation conditions above. Given a variable x and upper and lower approximation intervals, $\underline{X}$ and $\overline{X}$, these notions can be represented by the following implications:

1. $x \in \underline{X} \Rightarrow x \in X$,
2. $x \notin \overline{X} \Rightarrow x \notin X$.

Then, $\underline{X} \subseteq X \subseteq \overline{X}$. If $\overline{X} - \underline{X} \neq \emptyset$, the interval is said to be rough and is denoted by $(\underline{X}, \overline{X})$. It follows from these conditions that a rough interval can be written as an ordered pair $([a_1, a_2], [b_1, b_2])$ such that

$$b_1 \leq a_1 < a_2 \leq b_2.$$

Examples of pairs of intervals that define (the top ones), and do not define (at the bottom), rough intervals are depicted in Figure 1.

1. Rough intervals

2. Pair of intervals

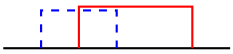


Figure 1: Rough intervals

1. $A \leq_{LR} B$

2. Uncomparable A and B

Figure 2: Comparison between intervals A and B

2.2 Interval arithmetic

In the following some notions on interval arithmetic and interval order relations, used later on to evaluate and to compare paths in a network, are introduced.

Real number intervals can be seen as ordered pairs of real numbers, and thus their arithmetic operations can be defined similarly [2].

Definition 2.1 *Let $A = [a_1, a_2]$ and $B = [b_1, b_2]$ be two intervals of real numbers. Then, the sum $A + B$ is an interval defined as*

$$A + B = [a_1 + b_1, a_2 + b_2].$$

Example 2.1 *According to this definition, the sum of intervals $[1, 3]$ and $[2, 4]$ is the interval $[1, 3] + [2, 4] = [3, 7]$.*

For comparing two intervals, an order relation based on the comparison between the left and right limits, is introduced [5].

Definition 2.2 *For any intervals $A = [a_1, a_2]$ and $B = [b_1, b_2]$, the relation $\leq_{LR}$ is defined as*

$$A \leq_{LR} B \quad \text{if and only if} \quad a_1 \leq b_1 \text{ and } a_2 \leq b_2.$$

It can be shown that $\leq_{LR}$ is reflexive, anti-symmetric and transitive, and therefore it is an order relation. However it is not a total order relation, as there may be intervals than cannot be compared to one another. This is illustrated in Figure 2.

3 The Rough Interval Shortest Path Problem

Let $G = (N, A)$ be a directed network with a set of nodes $N = \{1, \ldots, n\}$ and a set of arcs $A \subseteq N \times N$. Each arc $(i, j) \in A$ is associated with a rough interval that can represent the distance between the nodes i and j, or the time for traversing the arc (i, j). This rough interval is denoted by $C_{ij} = (\underline{C}_{ij}, \overline{C}_{ij})$.

A path in G from node $i_0 \in N$ to node $i_l \in N$ is a sequence

$$\langle (i_0, i_1), (i_1, i_2), \ldots, (i_{l-1}, i_l) \rangle$$

of arcs in A. The RISPP with source node $s \in N$ and target node $t \in N$ can be formulated as follows

$$\begin{array}{ll} \text{“minimize”} & \displaystyle\sum_{(i,j)\in A} C_{ij} x_{ij} \\ \text{subject to} & \displaystyle\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in N - \{s, t\} \\ -1 & \text{if } i = t \end{cases} \\ & x_{ij} \in \{0, 1\}, (i, j) \in A \end{array} \tag{1}$$

where $x \in \{0, 1\}^{|A|}$ is the array with unitary values for the components associated with the arcs in the solution. The objective function is the summation of rough intervals, C_{ij}, which is defined as an extension of Definition 2.1.

Definition 3.1 *Let $(\underline{A}, \overline{A})$ and $(\underline{B}, \overline{B})$ be two rough intervals. Then their sum is defined as follows:*

$$(\underline{A}, \overline{A}) + (\underline{B}, \overline{B}) = (\underline{A} + \underline{B}, \overline{A} + \overline{B}).$$

It can be shown that the result of the operation introduced in Definition 3.1, $(\underline{A}, \overline{A}) + (\underline{B}, \overline{B})$, is still a rough interval.

Lemma 3.1 *If $(\underline{A}, \overline{A})$ and $(\underline{B}, \overline{B})$ are two given rough intervals, then*

$$(\underline{A} + \underline{B}, \overline{A} + \overline{B})$$

is also a rough interval.

Additionally in the RISPP different paths need to be compared. In order to do that, the order relation $\leq_{LR}$, given in Definition 2.2, is now adapted for rough intervals as a dominance relation.

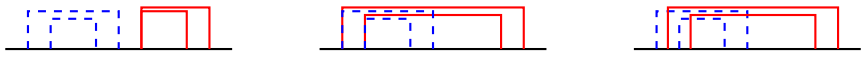
Definition 3.2 *Let $(\underline{A}, \overline{A})$ and $(\underline{B}, \overline{B})$ be two rough intervals. Then, $(\underline{A}, \overline{A})$ dominates $(\underline{B}, \overline{B})$ if and only if*

$$(\underline{A}, \overline{A}) \leq_{LR} (\underline{B}, \overline{B}) \text{ and } (\underline{A}, \overline{A}) \neq (\underline{B}, \overline{B}).$$

Figure 3 illustrates the dominance relation between several pairs of rough intervals. It also shows that, as expected, not every two rough intervals can be compared, that is, there may be rough intervals $(\underline{A}, \overline{A})$ and $(\underline{B}, \overline{B})$ such that neither the first dominates the latter, nor the opposite holds

Since not all pairs of rough intervals can be compared, in general there is not a unique solution for the RISPP, but there rather is a set of paths that are not dominated by any other. These solutions are called efficient paths. The purpose of the method presented in the next section is to compute the set of all the efficient paths from s to t in G.

1. $(\underline{A}, \overline{A})$ dominates $(\underline{B}, \overline{B})$



2. Uncomparable $(\underline{A}, \overline{A})$ and $(\underline{B}, \overline{B})$

Figure 3: Comparison between rough intervals $(\underline{A}, \overline{A})$ and $(\underline{B}, \overline{B})$

4 Algorithmic Approach

In this section an algorithm is introduced for finding the set of efficient paths between two nodes with respect to given rough intervals. The algorithm is based on the labeling algorithm proposed by Martins for the multicriteria shortest path problem [4].

As seen before each path from node s to node $i \in N$, p_{si}, has a well defined rough interval $C(p_{si}) = (\underline{C}(p_{si}), \overline{C}(p_{si}))$. Because there may exist more than one path from s to i and because not all of them can be compared, in order to find the efficient rough interval shortest paths from s to t, it is necessary to store information about all the partial paths that are not dominated. Thus each node $i \in N$ is associated with a set of labels, each one representing a path starting in s. Let $i \in N$ be a node of G and let p_{si} represent a path from s to i in G. The l-th label associated with i is the tuple $l_i = [C(p_{si}), j, k]$, where:

- $C(p_{si})$ represents the rough interval that gives the lower and the upper approximations for the cost of path p_{si}, and
- j_k is the k-th label of node $j \in N$, of the path from s to node j that precedes node i in path p_{si}.

Two labels representing paths from s to the same node can be compared by looking at the rough interval costs of each one and applying Definition 3.2. The new definition is presented below.

Definition 4.1 *For some node $i \in N$, we say that the label $[(\underline{C}(p_{si}), \overline{C}(p_{si})), -, -]$ dominates the label $[(\underline{C}(p'_{si}), \overline{C}(p'_{si})), -, -]$ if and only if*

$$(\underline{C}(p_{si}), \overline{C}(p_{si})) \leq_{LR} (\underline{C}(p'_{si}), \overline{C}(p'_{si})) \text{ and } (\underline{C}(p_{si}), \overline{C}(p_{si})) \neq (\underline{C}(p'_{si}), \overline{C}(p'_{si}))$$

The method stores node labels as temporary as soon as they are set. These labels may dominate others, which are then deleted. They may also be deleted later themselves, due to the appearance of newer labels that dominate it.

A temporary label becomes permanent (or definite) if it is chosen to be scanned in a suitable order and, thus, to extend the partial paths. This goal can be accomplished by selecting the temporary labels by lexicographic order. The lexicographic order for rough intervals results from the notion of lexicographic order for 4-tuples.

Definition 4.2 *Let $A = ([a_1, a_2], [a_3, a_4])$ and $B = ([b_1, b_2], [b_3, b_4])$ be two rough intervals, such that $a_3 \leq a_1 < a_2 \leq a_4$ and $b_3 \leq b_1 < b_2 \leq b_4$. We say that A is lexicographically smaller than or equal to B, denoted by $A \leq_{Lex} B$, if and only if*

$$A = B$$

or if there exists $i \in \{1, 2, 3, 4\}$ such that

$$a_i < b_i \text{ and } a_j = b_j, \text{ for all } j < i.$$

It is remarked that the lexicographic order is a total order relation, which ensures that a minimum label can be found at any iteration of the algorithm.

Definition 4.3 *Given the node $i \in N$, the label $[(\underline{C}(p_{si}), \overline{C}(p_{si})), -, -]$ is lexicographically smaller than the label $[(\underline{C}(p'_{si}), \overline{C}(p'_{si})), -, -]$ if and only if*

$$(\underline{C}(p_{si}), \overline{C}(p_{si})) \leq_{Lex} (\underline{C}(p'_{si}), \overline{C}(p'_{si})).$$

From a permanent label of some node $i \in N$, a temporary label is assigned to every node $j \in N$, for any arc $(i, j) \in A$, provided that it is dominated by another one. The labeling method for finding the set of efficient rough interval shortest paths is outlined in Algorithm 1.

Algorithm 1: Labeling algorithm for the rough interval shortest path problem

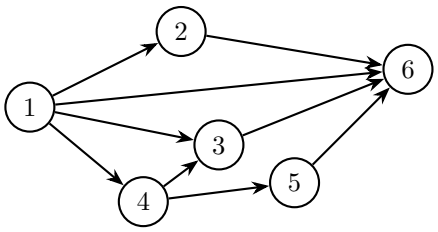
Input: A graph G with one rough variable associated with each arc
Output: Rough interval shortest paths

1 Assign the temporary label $[([0, 0], [0, 0]), -, -]_1$ to node s
2 **while** *the set of temporary labels is not empty* **do**
3 Find the lexicographically smallest temporary label. Let it be the l-th label of node i
4 Set this label as permanent
5 **for** $j \in N$ *such that* $(i, j) \in A$ **do**
6 $(\underline{C}(p_{sj}), \overline{C}(p_{sj})) \leftarrow (\underline{C}(p_{si}) + \underline{C}_{ij}, \overline{C}(p_{si}) + \overline{C}_{ij})$
7 Let $[(\underline{C}(p_{sj}), \overline{C}(p_{sj})), i, l]_\epsilon$ be a new temporary label of node j
8 Delete the temporary labels of node j that correspond to paths from s to j dominated by the new path
9 Find the efficient paths from s to t by using the two pointers of each label

To illustrate the Algorithm 1 we apply it to the network depicted in Figure 4. The origin and the destination nodes are $s = 1$ and $t = 6$, respectively. The arc rough intervals are shown in the table on the right-hand side of Figure 4.

The method starts by assigning the label $l_1 = [([0, 0], [0, 0]), -, -]_1$ to node $s = 1$. This label is chosen at the first iteration of the **while** loop at line 2, as a consequence it is set as a permanent label. When scanning node $s = 1$, the nodes 2, 3, 4 and 6 are labeled with

- $l_2 = [([16, 29], [10, 30]), 1, 1]_1$,
- $l_3 = [([22, 48], [8, 69]), 1, 1]_1$,
- $l_4 = [([13, 23], [6, 36]), 1, 1]_1$,



(i,j)	$(\underline{C}_{ij}, \overline{C}_{ij})$
$(1,2)$	$([16,29],[10,30])$
$(1,3)$	$([22,48],[8,69])$
$(1,4)$	$([13,23],[6,36])$
$(1,6)$	$([33,60],[20,81])$
$(2,6)$	$([15,30],[8,50])$
$(3,6)$	$([18,25],[4,47])$
$(4,3)$	$([7,17],[1,20])$
$(4,5)$	$([12,27],[9,30])$
$(5,6)$	$([5,8],[2,13])$

Figure 4: Network $G = (N, A)$ and rough interval arc weights

Table 1: Labels of the efficient paths from $s = 1$ to any node in the network in Figure 2

Node	Labels
1	$(([0,0],[0,0]),-,-)_1$
2	$(([16,29],[10,30]),1,1)_1$
3	$(([20,40],[7,56]),4,1)_2$
4	$(([13,23],[6,36]),1,1)_1$
5	$(([25,50],[15,66]),4,1)_1$
6	$(([30,58],[17,79]),5,1)_4$
	$(([38,65],[11,103]),3,2)_3$

- $l_6 = [([33,60],[20,81]),1,1]_1$,

respectively. In the next iteration of the loop, the label l_4 is the lexicographically smallest temporary label. When scanning it the node 5 is also labeled, with $l_5 = [([25,50],[15,66]),4,1]_1$ and a new label of node 3 is created, $l_3 = [([20,40],[7,56]),4,1]_2$ that dominates the former, which is thus deleted. Then l_3 is the lexicographically smallest temporary label, thus the node 6 is labeled with $l_6 = [([32,53],[12,84]),3,1]_1$. In the next step of the algorithm, the l_2 is the lexicographically smallest temporary label, thus node 6 is labeled again, now with $l_6 = [([31,59],[18,80]),2,1]_2$, which dominates the former label.

The procedure continues until there no temporary labels left to scan. At the end two efficient paths are obtained from $s = 1$ to $t = 6$:

- $p_1 = \langle(1,4),(4,5),(5,6)\rangle$,
- $p_2 = \langle(1,4),(4,3),(3,6)\rangle$.

The final results for each node, that is, the labels of efficient paths from s to any node in the network are reported in Table 1.

5 Computational experiments

This section is dedicated to assessing the empirical performance of Algorithm 1 for two sets of instances of the RISPPs. The algorithm was implemented in Matlab (R2018a). The tests ran on a 64-bit PC with an Intel Core i7-7500U at 3.5GHz with 12GB of RAM.

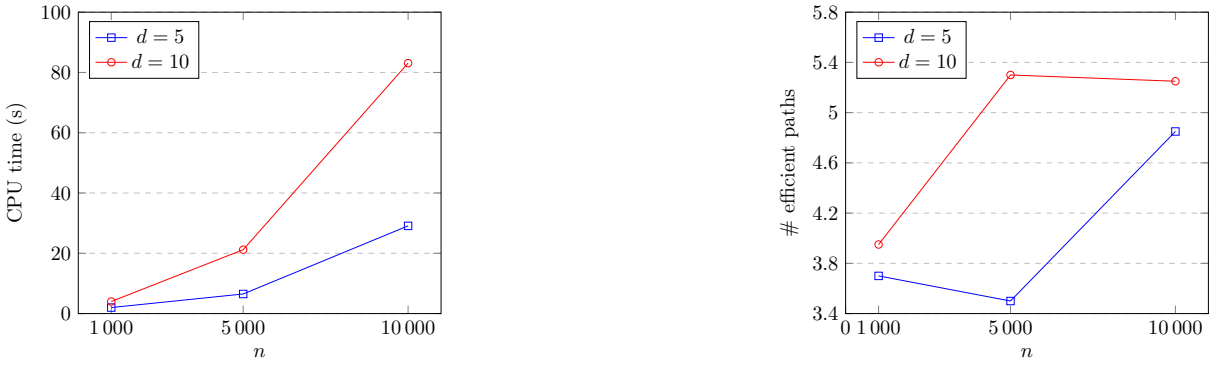


Figure 5: Mean results for Algorithm 1 in random networks

A first set of tests was performed on randomly generated networks with $n = 1\,000, 5\,000, 10\,000$ nodes, $m = dn$ arcs, for average degrees $d = 5, 10$. The arc rough intervals are defined by 4 integer numbers, generated uniformly in $[1, 1\,000]$. These instances are denoted by $R_{n,d}$.

The second set of instances are grid networks, denoted by G, where nodes are arranged in a rectangular, or square, grid with given height and width. Any pair of adjacent nodes is connected by an arc in both directions, and each arc is associated with a rough interval generated as in the random networks case. The characteristic of this set of networks are summarized in Table 3.

In order to analyze the performance of Algorithm 1, the code ran over 30 different instances for each dimension, and the mean CPU times and number of efficient paths were calculated.

Table 2: Mean results for Algorithm 1 in random networks

$R_{n,d}$	CPU time (s)	# efficient paths
$R_{1\,000,5}$	0.925	3.6
$R_{1\,000,10}$	1.685	4.0
$R_{5\,000,5}$	5.860	3.5
$R_{5\,000,10}$	19.853	5.3
$R_{10\,000,5}$	29.257	4.9
$R_{10\,000,10}$	82.408	5.3

For the two cases, the mean values increase rapidly with the size of the network, given by n and d.

Table 2 and Figure 5 show the mean CPU times and the mean number of efficient paths obtained by Algorithm 1 for the first set of experiments. Those results show that the CPU times increase rapidly, up to 83 seconds, with the increase of n and d. The plots also show a fairly similar growth of the number of efficient paths with the number of nodes of the network.

The mean CPU times and number of efficient paths of Algorithm 1 when applied to the grid networks are reported in Table 3 and in Figure 6. It is remarked that the problem is much more difficult for square grids than for rectangular grids with similar number of nodes, which is a consequence of the higher number of efficient paths on this type of grids. All instances were solved in less than 90 seconds, but the RISPP was also more difficult to solve for grid instances than for random instances. The mean number of computed efficient paths ranged from 23.8 to 1345.9.

Table 3: Mean results for Algorithm 1 in grid networks

Name	Size	n	m	CPU time (s)	# efficient paths
G_1	2×50	100	296	0.647	23.8
G_2	50×2	100	296	0.808	43.1
G_3	10×10	100	360	3.408	504.1
G_4	2×72	144	428	1.703	61.2
G_5	72×2	144	428	1.520	65.9
G_6	3×48	144	474	2.706	84.9
G_7	48×3	144	474	2.370	123.3
G_8	4×36	144	496	3.325	250.3
G_9	36×4	144	496	2.936	257.3
G_{10}	12×12	144	528	37.605	1345.9
G_{11}	3×75	225	744	29.738	268.9
G_{12}	75×3	225	744	38.367	232.0
G_{13}	5×45	225	800	57.414	970.1
G_{14}	45×5	225	800	84.661	858.1

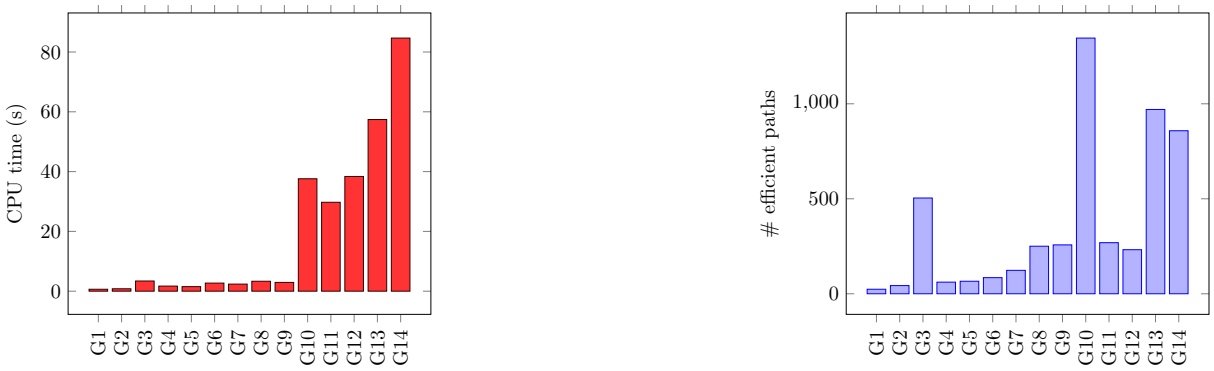


Figure 6: Mean results for Algorithm 1 in grid networks

6 Conclusions

This paper addressed the rough interval shortest path problem. It was assumed that the arc weights are defined as pairs of intervals, a lower approximation interval, which surely contains the arc weight, and am upper approximation interval, which may contain the arc weight. A labeling method was presented which is able to find the efficient paths from a single origin to all other nodes with respect to the rough interval arc weights. The proposed algorithm was tested regarding the CPU time and the number of produced solutions over random and grid networks. The obtained results are promising as the algorithm was always able to obtain a set of efficient rough interval shortest paths within less than 2 minutes, in average. Future lines of research may include studying a bicriteria version of the problem with rough interval costs and capacities.

Acknowledgment This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) under project grants UID/MAT/00324/2013 and UID/MULTI/00308/2013.

The work was also partially financially supported by project P2020 SAICTPAC/0011/2015, co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Union's European Regional Development Fund, and FCT, and by FEDER Funds and National Funds under project CENTRO-01-0145-FEDER-029312.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] G. Alefeld and J. Herzberger. *Introduction to interval computations*. Computer Science and Applied Mathematics. Elsevier Science; Academic Press, 1 edition, 1983.

[3] P. Mahajan, R. Kandwal, and R. Vijay. Rough set approach in machine learning: A review. *International Journal of Computer Applications*, 56:1–13, oct 2012.

[4] E. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236 – 245, 1984.

[5] S. Okada and M. Gen. Order relation between intervals and its application to shortest path problem. *Computers & Industrial Engineering*, 25, 1993.

[6] Z. Pawlak. Rough sets. *International Journal of Computer & Information Sciences*, 11:341–356, 1982.

[7] Z. Pawlak. *Imprecise Categories, Approximations and Rough Sets*, pages 9–32. Springer Netherlands, Dordrecht, 1991.

[8] M. Rebolledo. Rough intervals—enhancing intervals for qualitative modeling of technical systems. *Artificial Intelligence*, 170:667–685, 2006.

[9] R. Slowinski and D. Vanderpooten. A generalized definition of rough approximations based on similarity. *IEEE Transactions on Knowledge and Data Engineering*, 12:331–336, Mar 2000.