OPTIMAL PATH PROBLEM WITH POSSIBILISTIC WEIGHTS

Jan CAHA[1], Jiří DVORSKÝ[2]

[1,2]Department of Geoinformatics, Faculty of Science, Palacký University in Olomouc,

17. listopadu 50, 771 46, Olomouc, Czech Republic

[1]jan.caha@upol.cz, [2]jiri.dvorsky@upol.cz

Abstract

The selection of optimal path is one of the classic problems in graph theory. Its utilization have various practical uses ranging from the transportation, civil engineering and other applications. Rarely those applications take into account the uncertainty of the weights of the graph. However this uncertainty can have high impact on the results. Several studies offer solution by implementing the fuzzy arithmetic for calculation of the optimal path but even in those cases neither of those studies proposed complete solution to the problem of ranking of the fuzzy numbers. In the study the ranking system based on the Theory of Possibility is used. The biggest advantage of this approach is that it very well addresses the indistinguishability of fuzzy numbers. Lengths of the paths are compared based on the possibility and the necessity of being smaller than the alternative. The algorithm offers the user more information than only the optimal path, instead the list of possible solutions is calculated and the alternatives can be ranked using the possibility and the necessity to identify the possibly best variant.

Keywords: fuzzy numbers, Dijkstra algorithm, optimal path, uncertainty

INTRODUCTION

The selection of optimal or least-cost path through space is one of the common issues in the GIS. The optimal path may be chosen either in a network or on a surface. In both cases the algorithms used for selecting optimal path are based on graph theory, so there is actually little difference between the raster and the vector datasets. The range of possible utilization is very wide, from route planning to many civil engineering applications especially in construction of various networks [14].

Like any other type of data even data for selecting optimal path are affected by uncertainty. The main uncertainty affecting selection of optimal path is the uncertainty of weights or in other words cost for travelling from one node to another. These weights of edges can represent many real world phenomena, for example geographical distance of nodes, time necessary to cover the distance or amount of fuel needed for this particular distance. While distance can be measured quite exactly it is not particularly suitable as a measure for finding optimal path [9], mainly because distance alone does not tell anything about fitness of the solution. On the other hand, the time and/or amount of fuel necessary for travel are good indicators for optimal path selection. Yet none of those two can be expressed exactly for real world problems. Both of them are highly dependant on many other variables and thus unfit to be expressed as a crisp number [11]. It is much better to express them as a vague and ill-know values, using fuzzy set theory as fuzzy numbers [2].

Modifications of the Dijkstra and other algorithms for selection of the optimal path were studied in several studies [2,5,7,8,9,11,12,13,14]. All of those papers aims at calculating the optimal or the shortest path in a network when the uncertainty of the arc weights is presented in the graph, however each of these studies utilize different ways to obtain the results. The process has two main challenges that have to be addressed in order to produce the algorithm. These challenges are addition of the fuzzy numbers and their ranking. The addition of fuzzy numbers is usually described for the triangular and the trapezoidal fuzzy numbers [2,7,11], however these are not all the possible shapes of the fuzzy numbers and other variants can be also used. The addition is presented for mentioned shapes mainly because it can be easily implemented and calculated. But there are more general solutions that work for variety of other fuzzy number shapes [6]. Some authors [2] even use methods that provides a crisp value as the result of addition of fuzzy numbers. While this may be easier for further ranking of the results it leads to the loss of information about vagueness and/or imprecision of the fuzzy number.

The second challenge that need to be solved is ranking of fuzzy numbers. As noted by Dubois and Prade [4] there is no natural total-ordering structure for a set of fuzzy numbers and many of the approaches to the problem are either counter-intuitive and/or consider only one point of view on the matter. Some studies propose algorithms where any of indices for comparison of fuzzy numbers can be used [7]. While some use specific indice or even distance of fuzzy numbers for their ranking [9,13]. While all mentioned approaches have their possible use, none of them really address the problem of indistinguishability and overlap of fuzzy numbers, which certainly should be solved. The solution can be found in use of Possibility theory and indices proposed by Dubois and Prade [4].

The main advantage of the proposed algorithm is in generalization of fuzzy numbers addition. There is no assumption about shape of fuzzy numbers, instead the methods that use piecewise linear fuzzy number are used. Such fuzzy numbers can have any shape. The framework of Possibility theory is used for ranking of fuzzy numbers for calculation of possibility and necessity of equality and/or exceedance of fuzzy numbers.

The structure of paper is following: Section 2 offers basic preliminaries and notions of graph theory, for further elaborations about this topic please see [1]. Section 3 briefly summarizes informations about Fuzzy numbers, addition of Fuzzy numbers and ranking of Fuzzy numbers in framework of possibility theory. The proposed Algorithm is shown in section 4 and a case study is presented in section 5. The discussion and conclusions are presented in section 6.

GRAPH THEORY PRELIMINARIES

Prior to explanation of the algorithm some basic definitions need to be set up, the full reference can be found in Bondy and Murty [1]. Through the work we consider directed weighted network $G(N, A)$ that consist of set of nodes $N=\{1,\dots,n\}$ and a set of directed arcs $A=\{1,\dots,m\}$. Each of these arcs is defined by an ordered pair of nodes (i, j) that $i, j\in N$ and never $i\neq j$. Since the arcs are directed then $A(i, j)\neq A(j, i)$. Each of these arcs has a weight, that specifies cost for passing from start to final node. Usually this weight w_k is specified as a crisp positive number and it is used to identify the optimal path from starting node to the destination.

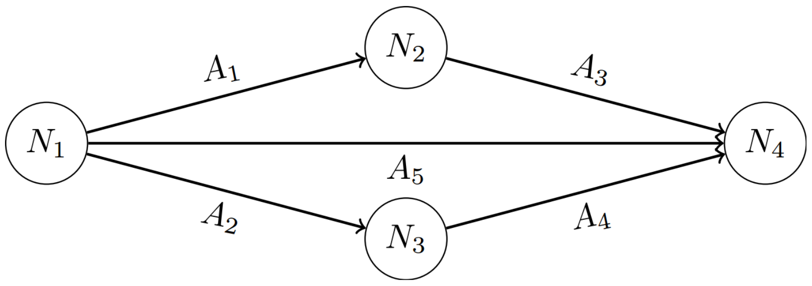

Fig. 1. Simple graph with nodes N_i and directed arcs $A_{j.}$

The selection of optimal path is a process of selecting path from node N_i to node N_j, where the sum of the weights is minimal. There are many algorithms that solve the problem proposed by Bellman – Ford, Dijkstra, Dreyfus and others [8]. Between these algorithms the Dijkstra algorithm [3] is undoubtedly one of the most commonly used, not only in practical applications, but also in scientific studies [2].

FUZZY NUMBERS AND POSSIBILITY THEORY

In case when there is need to model uncertainty that originates in indistinguishability, vagues etc. it is not suitable to use statistical approaches and alternative approaches is necessary [6]. Alternative framework for all essential operations can be found in Fuzzy set theory and Possibility theory.

Fuzzy Numbers

Fuzzy numbers are special cases of convex, normal fuzzy sets defined on $\mathbb{R}$ with at least piecewise continuous membership function, that represent vague, imprecise or ill-known value [6]. There are several types of fuzzy numbers, commonly used are triangular and trapezoidal ones, however other shapes are possible as well [6]. Triangular and trapezoidal fuzzy numbers are often used because calculations with them and their comparison can be done relatively easily, but it is much better if calculations and comparisons can be done for any shape of fuzzy numbers.

The most general type of fuzzy numbers that can be utilized for calculations are so called piecewise linear fuzzy numbers, these fuzzy numbers are defined as a set of α-cuts [6]. They can approximate any given shape and in their most simple representation are equal to triangular or trapezoidal fuzzy numbers.

If there is need to combine fuzzy numbers with classic crisp values then crisp numbers are treated as special case of fuzzy number, where all α-cuts are the same degenerative interval [6].

Fuzzy Arithmetic

In order to perform basic arithmetic operations with fuzzy numbers there is need for apparatus that allows and specifies such operations. The most general form of such rule is specified by so called extension principle [15], however this particular definition is complicated in terms of implementation, so alternative approaches that utilize decomposition theorem and interval arithmetic are used [6]. The decomposition theorem states that every fuzzy number (or generally any fuzzy set) $\tilde{A}$ can be described by associated sequence of α-cuts. An α-cut is an interval where all the objects have membership at least equal α. Formally it can is written as: $cut_\alpha(\tilde{A})=\tilde{A}_\alpha=\{x\in X|\mu_{\tilde{A}}(x)\geq\alpha\}$ [6]. Such α-cut of a fuzzy number is always closed interval $A_\alpha=[\underline{a_\alpha},\overline{a_\alpha}]$. The only necessary arithmetic operation for determination of shortest path is addition, using decomposition theorem and interval arithmetic the addition of fuzzy number $\tilde{A},\tilde{B}$ is [10]:

$$\tilde{A}_\alpha+\tilde{B}_\alpha=[\underline{a_\alpha}+\underline{b_\alpha},\overline{a_\alpha}+\overline{b_\alpha}] \quad (1)$$

for each $\alpha\in[0,1]$. Using this approach the addition of any two fuzzy numbers is possible.

Possibility theory

To allow decision making based on fuzzy numbers there is a need for a system that will allow ranking of fuzzy numbers. There are several such systems however most of them consider only one point of view on the problem [4]. The complete set of ranking indices in the framework of possibility theory was proposed in [4]. This ranking system uses possibility and necessity measures to determine relation of two fuzzy numbers.

Utilization of possibility theory allows also semantically describe fuzzy numbers as possibility distributions [16]. This semantic than help us explaining what such fuzzy numbers mean. The values with membership value 1 are believed to be absolutely possible or unsurprising, thus they should cove the most likely result. With decreasing degree of membership the possibility of obtaining given result decreases and the surprise rises. When membership value reaches 0 then such result is impossible (or almost impossible at some cases) and the surprise that such result would present is maximal. Such semantics helps with practical explanation what the results truly mean.

To asses position of fuzzy number $\tilde{X}$ to the fuzzy number $\tilde{Y}$ four indices are needed [4]. Two of them define possibility and necessity that $\tilde{X}$ is at least equal or greater than $\tilde{Y}$:

$$\Pi_{\tilde{X}}([\tilde{Y},\infty)) = \sup_x \min(\mu_{\tilde{X}}(x), \sup_{y\leq x} \mu_{\tilde{Y}}(y)) \quad (2)$$

$$\mathcal{N}_{\tilde{X}}([\tilde{Y},\infty)) = \inf_x \max(1-\mu_{\tilde{X}}(x), \sup_{y\leq x} \mu_{\tilde{Y}}(y)) \quad (3)$$

The other two determine if $\tilde{X}$ is strictly greater than $\tilde{Y}$:

$$\Pi_{\tilde{X}}(]\tilde{Y},\infty)) = \sup_x \min(\mu_{\tilde{X}}(x), \inf_{y\geq x} 1-\mu_{\tilde{Y}}(y)) \quad (4)$$

$$\mathcal{N}_{\tilde{X}}(]\tilde{Y},\infty)) = \inf_x \max(1-\mu_{\tilde{X}}(x), \inf_{y\geq x} 1-\mu_{\tilde{Y}}(y)) \quad (5)$$

Together these indices allow comparison of any two fuzzy numbers, based on pairwise comparison any set of fuzzy numbers can be sorted.

For both set of indices there are four situations of the combinations of possibility and necessity that can be outcome of the calculation. In this paragraph both relations – at least equal or greater, and strictly greater – are referred as relation, because the descriptions are valid for both pairs of indices. The first situation is when $\Pi_{\tilde{X}}([\tilde{Y},\infty))=N_{\tilde{X}}([\tilde{Y},\infty))=0,$ which means that $\tilde{X}$ is definitely does not fulfil the given relation to $\tilde{Y}.$ Then there is opposite situation $\Pi_{\tilde{X}}([\tilde{Y},\infty))=N_{\tilde{X}}([\tilde{Y},\infty))=1$ in which $\tilde{X}$ completely satisfy the relation. The other two relations contains some uncertainty, because they indicate certain results but they can not provide them absolutely. The first of those is situation when $\Pi_{\tilde{X}}([\tilde{Y},\infty))>0$ and $N_{\tilde{X}}([\tilde{Y},\infty))=0.$ This means that there is possibility that $\tilde{X}$ might satisfy the relation, but it is not necessary. Obviously that means that the indicators are not strong. The last possible combination of values is $\Pi_{\tilde{X}}([\tilde{Y},\infty))=1$ and $N_{\tilde{X}}([\tilde{Y},\infty))>0.$ In such case again it the relation is not satisfied absolutely but the indicators are much stronger than in previous case.

THE ALGORITHM

The Dijkstra algorithm was proposed in 1959 [3] and it purpose is to identify shortest path between given source node of the graph and all the other nodes. With small modification it can be also used to identify shortest path from starting node to the destination node [2]. It is this variant of the algorithm which will be shown. The algorithm is relatively simple and makes use of only two operations - addition and comparison of crisp numbers. The algorithm works in several steps [2]:

1. assign all nodes distance value: zero for the starting node and infinity for all others
2. set all nodes as unvisited, set starting node as current
3. calculate cumulative distances from current node to all its neighbours, if the distance is smaller then previously recorded distance then overwrite the distance and write the previous node
4. set all neighbours of the current node as visited
5. move to next unvisited node
6. if all the nodes were visited stop the algorithm.

The distance from one node to another is equal to the weight of the arc between those nodes in the necessary direction. The distance of node N_i from starting node is equal to the sum of distances between those two nodes. For each node we store the distance from the starting node and also the previous node on the path from starting node. That way the shortest path from any node to the starting node can be identified easily.

From the definition of the algorithm it is obvious that there is only one solution to the problem of finding optimal path between two nodes if such path exists. However identification of such ideal path is only possible in the environment without uncertainty. As soon as uncertainty is introduced there maybe be several solutions that can be hard or impossible to order. For the modified version of Dijkstra algorithm there is no assumption of one ideal path, in fact there is assumption that several such paths exist and that there are differences that allow their basic ordering.

There are several modifications to the algorithm. First is that all the weights in graph are expected to be fuzzy numbers. The calculation on distances from starting node to neighboring nodes is done according to decomposition theorem and Eq. (1). The second change of the algorithm is that for each node the list of distances is stored (Code Sample 1).

Code sample 1. Dijkstra algorithm (modified from: [2])

```
function Dijkstra(Graph, source):
  for each vertex v in Graph:            // Initializations
    v.distanceList.add(infinity);        // Each vertex stores list of distances
  end for

  source.distanceList.add(0);            // Distance from source to source
  Q = the set of all nodes in Graph;

  while Q is not empty:                  // The main loop

    u = vertex in Q with smallest distance to last visited node or source;
    remove u from Q;

    if u.distanceList.contains(infinity):
      break;                                       // all remaining vertices are
    end if                                         // inaccessible from source

    for each neighbor v of u:   // where v has not yet been removed from Q
                                // the distance is calculated for fuzzy numbers
      newDistanceList = calculateDistance(u.distanceList, dist_between(u, v));
      v.distanceList = compareFuzzyValues(v.distanceList, newDistanceList);

      select new v in Q;      // Reorder v in the Queue

    end for
  end while

  return distancesLists of all nodes;
end function
```

When calculating distance of new node there is necessity to take into account all the possible paths to this node and add the new distance to all of them (Code Sample 2).

Code sample 2. Calculation of distances for node

```
//each node may contain more than one solution, the function calculates those
function calculateDistance(distanceList, distance)
  for each distance i in distanceList:
    distanceList[i] = distanceList[i] + distance; // according to Eq.(1)
  end for
  return distanceList;
end function
```

The next step in the algorithm is comparison of the calculated distance with the distances already known for the node. Possibility and necessity is calculated according to Eqs. (2,3). There are 3 possible outcomes of the comparison the new values to the already known distances. First the necessity result may be equal to 1. Which means that it is definitely bigger, in such case the value is of no interest because the alternative is for sure shorter. If the values of necessity and possibility are both equal to zero than the value is definitely smaller and it should replace the originally recorded values. If $\Pi_{\tilde{X}}([\tilde{Y},\infty))>0$ and $N_{\tilde{X}}([\tilde{Y},\infty))<1$ that is and indicator, that there is overlap between two values and there is no clear preference which is one is actually smaller. In such case the new value is added to the list (Code Sample 3).

The described algorithm provides list of distances for each node. This list can be ordered according to pairwise comparisons of fuzzy numbers in the list to produce best possible outcome. Also all several outcomes can be presented with ranking according to Eqs. (2,3).

Code sample 3. Comparison of fuzzy distances stored in two lists

```
//comparison of existing list of distances store for node with list of new nodes
function compareFuzzyValues(distanceList, newDistanceList):

  addToList = false;

  if distanceList is empty:
    distanceList = newDistanceList;

  else:
    for each newDistance in newDistanceList:
      for each distance in distanceList:
        possibility = possibility(newDistance >= distance); // using Eq.(2)
        necessity = necessity(newDistance >= distance);     // using Eq.(3)

        if necessity = 1:
          break;
        end if

        if possibility = 0:
          remove distance from distanceList;
          addToList = true;
        end if

        if possibility > 0 and necessity < 1:
          addToList = true;
        end if
      end for

        if addToList = true:
          add newDistance to distanceList;
        end if
    end for
  end if

  return distanceList;
end function
```

The algorithm can be implemented according to the Code samples 1., 2. and 3. The addition of fuzzy numbers as well as their ranking is computationally much more complicated operation than same operations with crisp numbers. Also the algorithm does not search only for one solution but rather for a set solutions which is also more computationally expensive than the classic Dijkstra Algorithm. Based on these facts it can be reasoned that the algorithm will be both computationally and time demanding when compared to classic Dijkstra algorithm. But these are properties of the algorithm that on the other hand allows calculation with the uncertainty.

CASE STUDY

The case study shows rather simple example of a graph with weights represented as fuzzy numbers or possibility distributions. This case study may describe real word example of travelling in the city where weights show time necessary to reach the node. Obviously time cannot be expressed exactly because it may depend on external conditions that are unknown at the time, when the model was created. Such conditions could be weather, time when the the travel should be made, amount of traffic etc. Because none of those conditions can be know in advance, it is reasonable to model them as possibility distributions.

The example used in case study is obvious. A simple graph containing 4 nodes and 5 directed arcs (Fig. 2). **Node *a* is a starting point of path and destination node is *d*.** From the visualization of the graph it is visible that 3 path can be identified: a→b→d, a→c→d and finally direct path from a→d. Since all the weights of the graph are defined as triplets in form $[\underline{\tilde{A}_0}, \underline{\tilde{A}_1} = \overline{\tilde{A}_1}, \overline{\tilde{A}_0}]$ they represents triangular fuzzy numbers.

Known property of fuzzy numbers is that if they are aggregated the result is again triangular number [6]. So the Eqs (1) is applied only for α values of 0 and 1. Obtained results are summarized in Table 1.

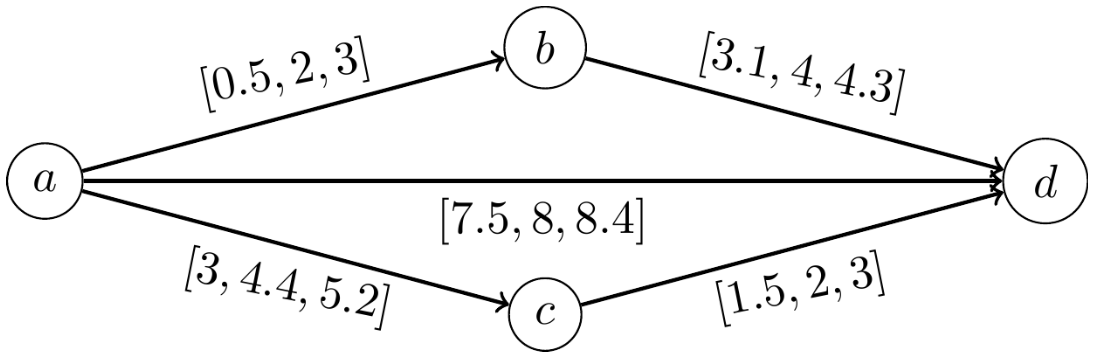

Fig. 2. Simple graph with fuzzy weights.

Table 1. Resulting path and their lengths of the case study

Path	Triangular number
a→b→d	[3.6,6,7.3]
a→c→d	[4.5,6.4,8.2]
a→d	[7.5,8,8.4]

Results can be best asses when they are visualized (Fig. 3). Now the ranking needs to established. Since the interest is in finding values that are smaller or equal to the given value the Eqs. (2,3) will be used to calculate possibility and necessity.

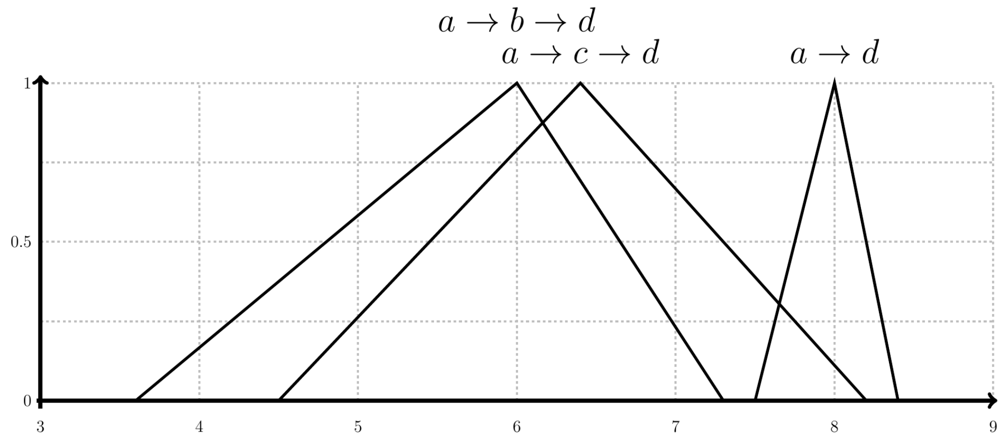

Fig. 3. Results of case study compared in a graph.

The comparison of the obtained results is summarized in Tab. 2. From that can be reasoned that solution a→d is the worst as it has possibility and necessity of at least equality to both other solution equal to 1. Also a→c→d ≥ a→d has both possibility and necessity equal to 0 which excludes this solution from set of possible shortest paths. From comparison of solutions a→b→d and a→c→d it is clear that there is no strict ordering of these solutions, because there is quite a big overlap. But even such solutions can be ordered, there is higher both possibility and necessity of a→c→d ≥ a→b→d than a→b→d ≥ a→c→d. Than can be interpreted as a→b→d being more suitable solution. However the fact that neither of indices is equal to 1 means that the solutions can not be distinguished very well, in fact they are rather similar (Fig. 3). According to those facts the results of the algorithm can be shortly summarized by defining solutions a→b→d and a→c→d as acceptable solutions, while a→d is unacceptable solution (Tab. 3).

Table 2. Comparison of the resulting paths

Comparison	Possibility	Necessity
a→b→d ≥ a→c→d	0.875	0.349
a→c→d ≥ a→d	0	0
a→c→d ≥ a→b→d	1	0.651
a→c→d ≥ a→d	0.304	0
a→d ≥ a→b→d	1	1
a→d ≥ a→c→d	1	1

Such cases are classic situations when presenting both solution and their ranking would be useful for decision maker. Mainly because the solutions are quite similar and there is possibility that choosing any of those might lead to optimal decision. Unfortunately because of the presented uncertainty the solutions can not be ranked directly.

Table 3. Results of the algorithm

Path	Status
a→b→d	Accepted
a→c→d	Accepted
a→d	Not accepted

DISCUSSION AND CONCLUSION

The presented modification of the Dijkstra algorithm aims to provide better support of decision making in situations where uncertainty of the data exists. It should be helpful mainly by providing all solutions that can-not be distinguished, or in other words that have rather high similarity. This is archived by utilization of Fuzzy set theory and Possibility theory to manage the uncertainty and the vagueness through the calculation. The results obtained from the algorithm provide the user not only with one optimal path but also with other options that are quite similar under the given amount of uncertainty.

The use of fuzzy numbers as weights in the graph allows better modelling of the real world situations where the time to travel from one point to another can not be specified exactly, or other similar cases. Specifying the time as a crisp number can be to much idealization and simplification of the problem, because the algorithms for finding optimal path then produce way to idealized solutions that do not take into account either uncertainty or the amount of dissimilarity of the solutions.

The proposed algorithm proposes solution to both challenges, that were mentioned previously. It allows identification of optimal path in uncertain environment. This uncertain or vague environment is however better model of reality than exact environment, where all the values are expected to be known precisely. The second issue is addressed by providing not only one solution but a list of solutions. This provides more alternatives that can be ranked using possibility and necessity measures.

The variant of Dijkstra algorithm is in GIS also used for selecting least cost paths on surfaces [14]. The proposed algorithm can be used for calculating optimal path on surfaces that contain uncertainty, especially on so called fuzzy surfaces. Further studies of the topic could be focus on this issue - selection of optimal paths on fuzzy surfaces.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support by the Operational Program Education for Competitiveness - European Social Fund (projects CZ.1.07/2.3.00/20.0170 and CZ.1.07/2.2.00/28.0078 of the Ministry of Education, Youth and Sports of the Czech Republic).

REFERENCES

[1] BONDY, J. A. and MURTY, U. S. R. (2008) Graph Theory. Springer.

[2] DENG, Y., CHEN, Y., ZHANG, Y. and MAHADEVAN, S., (2012) Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing*. Vol. 12, no. 3, pp. 1231–1237.

[3] DIJKSTRA, E. W. (1959) A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*. Vol. 1, pp. 269–271.

[4] DUBOIS, D. and PRADE, H. (1983) Ranking Fuzzy Numbers in the Setting of Possibility Theory. *Information Sciences*. Vol. 30, no. 3, pp. 183–224.

[5] GHATEE, M. and HASHEMI, S. M. (2009) Application of fuzzy minimum cost flow problems to network design under uncertainty. *Fuzzy Sets and Systems*. 2009. Vol. 160, no. 22, pp. 3263–3289.

[6] HANSS, M. (2005) Applied fuzzy arithmetic : an introduction with engineering applications. Berlin ; New York: Springer-Verlag.

[7] HERNANDES, F., LAMATA, M. T., VERDEGAY, J. L. and YAMAKAMI, A. (2007) The shortest path problem on networks with fuzzy parameters. *Fuzzy Sets and Systems*. Vol. 158, no. 14, pp. 1561–1570.

[8] JI, X., IWAMURA, K. and SHAO, Z. (2007) New models for shortest path problem with fuzzy arc lengths. *Applied Mathematical Modelling*. Vol. 31, no. 2, pp. 259–269.

[9] MAHDAVI, I., NOURIFAR, R., HEIDARZADE, A. and AMIRI, N. M. (2009) A dynamic programming approach for finding shortest chains in a fuzzy network. *Applied Soft Computing*. Vol. 9, no. 2, pp. 503–511.

[10] MOORE, R. E., KEARFOTT, R. B. and CLOUD, M. J. (2009) Introduction to interval analysis. Philadelphia, PA: Society for Industrial and Applied Mathematics.

[11] OKADA, S. and SOPER, T. (2000) A shortest path problem on a network with fuzzy arc lengths. *Fuzzy Sets and Systems*. Vol. 109, no. 1, pp. 129–140.

[12] OKADA, S. (2004) Fuzzy shortest path problems incorporating interactivity among paths. *Fuzzy Sets and Systems*. Vol. 142, no. 3, pp. 335–357.

[13] TAJDIN, A., MAHDAVI, I., MAHDAVI-AMIRI, N. and SADEGHPOUR-GILDEH, B. (2010) Computing a fuzzy shortest path in a network with mixed fuzzy arc lengths using α-cuts. *Computers & Mathematics with Applications*. Vol. 60, no. 4, pp. 989–1002.

[14] YU, C., LEE, J. and MUNRO-STASIUK, M. J. (2003) Extensions to least-cost path algorithms for roadway planning. *International Journal of Geographical Information Science*. Vol. 17, no. 4, pp. 361–376.

[15] ZADEH, L. A., (1965) Fuzzy Sets. *Information and Control*. Vol. 8, no. 3, pp. 338–353.

[16] ZADEH, L. A. (1978) Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*. Vol. 1, pp. 3–28.