The Quest for Optimal Paths in Fuzzy Weighted Graphs

Peter De Kesel, Chris Cornelis, Etienne E. Kerre

Department of Applied Mathematics and Computer Science, Ghent University
Fuzziness and Uncertainty Modelling Research Unit
Krijgslaan 281 (S9), B-9000 Ghent, Belgium
E-mail: `Peter.De.Kesel@ibsbe.be`, `{chris.cornelis,etienne.kerre}@rug.ac.be`
WWW homepage:`http://fuzzy.rug.ac.be`

Abstract

Algorithms for finding multicriteria shortest paths in graphs have been studied and applied in a multitude of optimization applications during the past 50 years. In a lot of cases, the system to be modelled is inherently imprecise; fuzzy graphs, along with generalizations of algorithms for finding optimal paths within them, have been proposed to meet this characteristic. The correctness of the algorithms depends on the ranking method used to rank paths. We introduce and prove conditions that the ranking method should satisfy, and present an efficient method for finding those optimal paths.

1 Introduction and Preliminaries

In many transportation, routing, communications and other applications, graphs emerge naturally as a mathematical model of the problem to be solved. Due to their transparent semantics in terms of set and relation theory, they have proven rewarding candidates for generalization to a fuzzy framework, and many variations on the theme of a fuzzy graph exist; see e.g. Rosenfeld [5]. In this paper, we consider *fuzzy weighted graphs*, i.e. vertices (or nodes) and edges (or arrows) remain crisp, but the edge weights will be fuzzy numbers, as in [4]. Let us recall some necessary definitions.

A fuzzy number is defined as a fuzzy set in the real line $\mathbb{R}$, i.e. an $\mathbb{R} \to [0,1]$ mapping A such that A is upper semicontinuous, convex and has bounded support. If A and B are fuzzy numbers, their sum $A \oplus B$ and product $A \odot B$ are defined by Zadeh's extension principle. For $z \in \mathbb{R}$,

$$(A \oplus B)(z) \quad = \quad \sup_{x \in \mathbb{R}} \min(A(x), B(z-x)), (A \odot B)(z) \quad = \quad \sup_{x \in \mathbb{R} \setminus \{0\}} \min\left(A(x), B\left(\tfrac{z}{x}\right)\right)$$

A fuzzy weighted graph $\tilde{G} = (V, E, \tilde{c})$ consists of a set V of vertices or nodes v_i and a binary relation E of edges $e_k = (v_i, v_j) \in V \times V$; we denote $head(e_k) = v_i$ and $tail(e_k) = v_j$. with each edge (v_i, v_j), a weight or cost $\tilde{c}(v_i, v_j) = (\tilde{c}(v_i, v_j)^1, ..., \tilde{c}(v_i, v_j)^r)$, a vector of fuzzy numbers with $r \geq 1$, is associated. For simplicity, in this paper we will focus on the case $r = 1$.

An s,t*–path* p in a fuzzy weighted graph $\tilde{G} = (V, E, \tilde{c})$ is an n–tuple $p = (e_1, e_2, ..., e_n)$ with $e_i \in E$ and $tail(e_i) = head(e_{i+1})$, for $i = 1, ..., n-1$ and $head(e_1) = s, tail(e_n) = t$. $P(s,t)$ is the set of all s,t–paths. An s,t–path p is called a cycle if $s = t$. A special path denoted $\emptyset$ is called the empty path; it does not contain any edges. Suppose we have two paths in a graph $\tilde{G}$: $p = (v_1, v_2, ..., v_{n_p})$ and $q = (v_{n_p}, v_{n_p+1}, ..., v_n)$, we say $p \Diamond q = (v_1, v_2, ..., v_n)$ is the concatenation of the two paths in $\tilde{G}$.

To make the notion of "shortest path" precise, we need an *objective function* $\tilde{f}$ that associates an objective value (a fuzzy number) with each path $p \in P(s,t)$; it is assumed that $\tilde{f}(\emptyset) = 0$ and $\tilde{f}(p) = \tilde{c}(e_i)$ if the s,t–path p consists of only one edge e_i. The objective function determines the *cost criterion.* For instance, if the objective function is such that $\tilde{f}(p\Diamond q) = \tilde{f}(p) \oplus \tilde{f}(q)$ for any two paths p and q in $P(s,t)$, we say that we work with the additive cost criterion. In case of the multiplicative criterion, we have: $\tilde{f}(p\Diamond q) = \tilde{f}(p) \odot \tilde{f}(q)$. Other criteria are not considered in this paper.

For our purposes, paths will be compared based on their objective values; therefore, we need a method $\prec_M$ to rank fuzzy numbers. Then, $\tilde{f}(p) \prec_M \tilde{f}(q)$ means that path p is "shorter", or more optimal than q. More concisely, we write $p \prec_{D_M} q$. Note that such a ranking method will not be total, i.e.$\neg(\tilde{f}(p) \prec_M \tilde{f}(q)$ or $\tilde{f}(q) \prec_M \tilde{f}(p))$ may occur for distinct p and q; in this case p and q are called *non–dominated alternatives*; as in [6], we write $\tilde{f}(p) \sim_M \tilde{f}(p)$. A *fuzzy shortest–path algorithm* finds those s,t–paths which are not dominated by other ones; so in general the solution will not be a single path, but rather a set of non–dominated alternatives. This situation occurs also in crisp multi–criteria shortest path problems. Many ranking methods are available, in section 3.1 we demonstrate why only a few of them are suitable.

2 Label Correcting Algorithm for Fuzzy Weighted Graphs

This section describes the generic label correcting algorithm (adapted from the work of Martins [3] and listed in table 1) for finding all non–dominated s,v_i–paths in a connected weighted fuzzy graph $\tilde{G} = (V,E,\tilde{c})$, for all nodes v_i in $\tilde{G}$, where s denotes a fixed starting vertex of $\tilde{G}$ for all paths. We also require that $0 \prec_M \tilde{f}(c)$ applies for every cycle c of $\tilde{G}$, so the algorithm will finish; all non–dominated paths will be finite and have a finite objective value.

The algorithm uses the following data structures: Q is a first–in–first–out (FIFO) queue, and the linear list $L(v_j)$ contains, at any given moment during algorithm execution, the labels of the s,v_j–paths that have been found so far and that are not dominated by other examined paths. Corresponding to the k–th non–dominated path that is found, we store the k–th label in $L(v_j)$ as follows:

$$L(v_j,k) = [\tilde{f}(p^k_{s,v_j}),(v_i,k_i)]^k_{v_j},$$

in which v_i is the vertex preceeding v_j in the path p^k_{s,v_j}, and k_i indicates that for the k_i–th s,v_i–path $p^{k_i}_{s,v_i}$ in $L(v_i)$ applies: $\tilde{f}(p^k_{s,v_j}) = \tilde{f}(p^{k_i}_{s,v_i}) \oplus \vec{c}_{i,j}$. When the algorithm terminates, all non–dominated paths can be reconstructed by walking reversily through the pointers in all intermediate $L(v_i)$ until the starting vertex s is reached.

The order in which the algorithm walks through the graph is not fixed, but since objective values of s,v_j–paths are calculated as the sum (or product in case of a multiplicative criterium) of the objective value of an s,v_i–path and $\tilde{c}(v_i,v_j)$, and since $\oplus$ and $\otimes$ are commutative and associative, the used order for walking through the graph does not impact the algorithm's correctness (it may impact the speed of finding the result). Furthermore, it is not necessary to consider all s,v_i–paths, and select the non–dominated ones afterwards. The search space can be reduced due to the strong optimality principle: every subpath of an optimal path is also optimal. In case a journey from Amsterdam to Paris through Brussels seemed better than through Rome, it is useless to consider a stop in Rome before getting to Paris if you are looking for an optimal Amsterdam–Paris–Madrid path. This pruning in the search space is implemented on lines 12 and 15. The ranking technique used in these lines must fullfil certain conditions in order to be sure that we get a correct result, i.e. the result contains all non-dominated paths. Those conditions are discussed in section 3.1.

Finally, we focus on a few important lines in the algorithm:

- If a new s,t–path p_{new} is found, it will not be kept in list $L(t)$ if there is already an s,t–path p_{old} in list $L(t)$ which dominates p_{new} (i.e. if $\tilde{f}(p_{old}) \prec_M \tilde{f}(p_{new})$) [line 12].

- If there is no s,t–path in $L(t)$ that dominates p_{new} in step I, p_{new} is added to the list [line 14] in order to be able to extend this path on the quest for optimal paths to vertices that can be reached from t. In step I, all earlier found paths $p_{old,l}$ in $L(t)$ that were dominated by p_{new} (i.e. if $\tilde{f}(p_{new}) \prec_M \tilde{f}(p_{old,l})$) are removed in line 15.
- In line 16, the paths starting from s to all children of v_j in $\tilde{G}$ have to be reconsidered. Suppose that, in this step, for a child k of v_j in $\tilde{G}$ we have an s,k–path via parent x that is not dominated by a path via v_j. Now that we have found a new optimal path towards v_j, it is possible that this new path dominates the s,k–path via x. For this reason, this algorithm is called *label correcting*: if we are only interested in an optimal path from s to one particular vertex v, we cannot be sure that a constructed s,v–pad will not be dominated by an other alternative until the entire algorithm has finished.

```
1   'initialize:
2   $\forall v \in V : L(v,1) \leftarrow [\infty,(-,-)]^1_v$;
3   $Q \leftarrow \emptyset$;
4   $L(s,1) \leftarrow [\tilde{0},(-,-)]^1_s$; add($Q, L(s,1)$);
5   'best-first search:
6   while ($Q \neq \emptyset$) do {
7     $x \leftarrow$ takeNext($Q$);
8     '$x$ contains $L(v_i,k) = [\tilde{f}(p^k_{s,v_i}),(v_p,k_p)]^k_{v_i}$;
9     $\forall v_j \in A(v_i)$ do :{
10      $p^l_{s,v_j} \leftarrow p^k_{s,v_i} \diamondsuit (v_i,v_j)$; (*)
11      $\tilde{f}(p^l_{s,v_j}) \leftarrow \tilde{f}(p^k_{s,v_i}) \oplus \tilde{c}_{v_i,v_j}$;
12      if $p^l_{s,v_j}$ is not dominated in $L(v_j)$ {
13        $L(v_j,l) \leftarrow [\tilde{f}(p^l_{s,v_j}),(v_i,k)]^l_{v_j}$ ;
14        add $L(v_j,l)$ to list $L(v_j)$;
15        make $L(v_j)$ consistent;
16        add($Q, L(v_j,l)$);
17      }
18    }
19  }
20  stop
(*) with $l$ an empty position.  If line 15 caused a free position, that one may be reused,
    otherwise a new position is added to the list.
```

Table 1: Label correcting algorithm

3 Suitable Ranking Techniques for Dominance Checking

3.1 Necessary and Sufficient Conditions for Correctness of the Labelling Algorithms

Given a fuzzy weighted graph $\tilde{G} = (V, E, \tilde{c})$; the labelling algorithm from the previous section searches all optimal paths using either the additive or the multiplicative cost criterion. The ranking method $\prec_M$ will have to satisfy certain properties in order to come up with the correct solution. We set the scene by proving a number of necessary conditions on $\prec_M$.

Theorem 3.1 (Necessary conditions) *In order to obtain a result that contains all non–dominated paths, the following constraints for $\prec_M$ are needed*[1]*:*
(C.1) $\prec_M$ *is antireflexive* $\Leftrightarrow (\forall A \in \mathcal{A})(\neg(A \prec_M A))$

[1](C.3) is necessary when using the additive cost criterion, (C.3') when using the multiplicative cost criterion.

(C.2) $\prec_M$ *is transitive* $\Leftrightarrow (\forall(A,B,C)\in\mathcal{A}^3)(A \prec_M B$ *and* $B \prec_M C \Rightarrow A \prec_M C)$
(C.3) $\prec_M$ *is compatible with* $\oplus \Leftrightarrow (\forall(A,B,C)\in\mathcal{A}^3)(A \prec_M B \Rightarrow A\oplus C \prec_M B \oplus C)$
(C.3') $\prec_M$ *is compatible with* $\odot \Leftrightarrow (\forall(A,B,C)\in\mathcal{A}^3)(A \prec_M B \Rightarrow A\odot C \prec_M B \odot C)$

Proof: (C.1)–(C.3) can be proved by contraposition, here we only prove (C.3). Suppose that in step I, the s,t–path p_I is constructed, and in a later step J the s,t–path p_J is constructed. In a still later step K, non–dominated paths to u, a child of t are searched.
In this situation, the algorithm may yield an incorrect result if $\prec_M$ is not compatible with $\oplus$: if $\tilde{f}(p_I) \prec_M \tilde{f}(p_J)$, then p_J will not be added to the list $L(t)$. In step K, $p_I\diamondsuit(t,u)$ will be considered and $p_J\diamondsuit(t,u)$ will be omitted, while it is possible that:

$$\neg\left((\tilde{f}(p_I)\oplus\tilde{c}(t,u)) \prec_M (\tilde{f}(p_J)\oplus\tilde{c}(t,u))\right)$$

and hence,

$$p_I\diamondsuit(t,u) \not\prec_{D_M} p_J\diamondsuit(t,u)$$

so the algorithm fails by ignoring the path $p_J\diamondsuit(t,u)$. □

An antireflexive and transitive binary relation is called a *strict* order relation, and all fuzzy ranking methods mentioned in [6] meet these two constraints. (C.3), sufficient to entail the strong optimality principle, and hence correctness[2], is more restrictive, as will be seen in subsection 3.2.

Theorem 3.2 (Correctness) *The search algorithm to find the additive (multiplicative) optimal paths in a fuzzy weighted graph is correct iff the used* strict *order* $\prec_M$ *is compatible with* $\oplus$ *(*$\odot$ *if there are no negative costs*[3]*.)*

3.2 Suitable Ranking Methods

In [6], Wang and Kerre gave a comprehensive overview of fuzzy ranking methods and their properties. For LR flat fuzzy numbers[4], it turns out that the ranking based on the popular center of gravity defuzzification (see also section 4) cannot be used because neither (C.3) nor (C.3') hold for it. (C.3) holds for a number of methods (Campos and Muñoz, Yager, Fortemps and Roubens, Yuan, Saade and Schwarzlander, Kolodziejczyk) which surprisingly all result in the same final ranking. (C.3') applies for Dubois and Prade's PSD–method if continuous LR flat fuzzy numbers are used with strict monotonous membership functions on both sides of their kernel.

3.3 An Optimization : the Label Setting Algorithm

The efficiency of the label correcting algorithm in table 1 can be considerably increased if $\tilde{G}$ does not have negative weights. The idea is as follows: we replace the list Q by a *priority queue* $PQ_{\preceq_N}$ (using a total order relation $\preceq_N$), such that on line 7, the label returned by $takeNext(PQ_{\preceq_N})$ will contain an objective value that will never be dominated later on, hence a reference to an optimal

[2]This claim is a straightforward extension of its analogon in classical graph theory. It is however not as trivially satisfied as its crisp counterpart, as our discussion reveals.

[3]A fuzzy number A is called non–negative if $Supp(A) \subseteq [0,+\infty[$; note that since $\ln A \oplus \ln B = \ln(A\odot B)$, searching for multiplicative optimal paths in $\tilde{G} = (V,E,\tilde{c})$ is equivalent to searching additive optimal paths in $\tilde{G}' = (V,E,\ln\circ\tilde{c})$.

[4]If L and R are two decreasing $[0,+\infty[\rightarrow[0,+\infty[$ mappings satisfying $L(0)=R(0)=1$ and $L(1)=R(1)=0$ and the real numbers m_l, m_r, α, β satisfy $m_l \le m_r$ and $\alpha,\beta \ge 0$, then the LR flat fuzzy number $(m_l,m_r,\alpha,\beta)_{LR}$ is defined by

$$(m_l,m_r,\alpha,\beta)_{LR}: \quad \mathbb{R} \rightarrow [0,1]$$
$$x \mapsto \begin{cases} L\left(\frac{m_l-x}{\alpha}\right) & x \le m_l, \alpha > 0 \\ 1 & x \in [m_l,m_r] \\ R\left(\frac{x-m_r}{\beta}\right) & x \ge m_r, \beta > 0 \\ 0 & \text{otherwise} \end{cases}$$

path. We can therefore partition the labels of a given node v_i into two classes: $L(v_i)$ contains temporary labels, while $\overline{L}(v_i)$ contains permanent labels that correspond to optimal paths. $PQ_{\preceq_N}$ contains only temporary labels; any label that is selected from $PQ_{\preceq_N}$ becomes permanent. To check whether a new path is dominated, both $L(v_i)$ and $\overline{L}(v_i)$ have to be examined, but only $L(v_i)$ has to be made consistent, warranting another significant gain in time. This is called the *label setting algorithm*, an extension of the well–known Dijkstra algorithm.

Okada and Soper [4] applied this optimization in a network using LR flat fuzzy numbers as weights. They casually remarked that "the order [used by $PQ_{\preceq_N}$] does not play so important a role", but this is not always true. In fact, the optimization is only possible if the following link exists between the ranking method $\preceq_N$ used in the priority queue and the strict order relation $\prec_M$ used for dominance–checking (see also [3]):

$$(Q = \min(\mathcal{A}) \text{ in } (\mathcal{A}, \preceq_N)) \Rightarrow (Q = \min(\mathcal{A}) \text{ in } (\mathcal{A}, \prec_M))$$

for arbitrary fuzzy numbers A and B belonging to a class $\mathcal{A}$, with $Q = \min(\mathcal{A})$ in $(X, \preceq_N) \Leftrightarrow (Q \in \mathcal{A})$ and $\neg(\exists P \in \mathcal{A})(P \neq Q$ and $P \preceq_N Q)$ and $(Q = \min(\mathcal{A})$ in $(X, \prec_M) \Leftrightarrow (Q \in \mathcal{A})$ and $\neg(\exists P \in \mathcal{A})(P \prec_M Q)$. A sufficient condition for this link is: $(A \prec_M B) \Rightarrow (A \preceq_N B)$

4 A Fast Search Method

Defuzzification is a synthesis process that reduces the information contained by a fuzzy number to a crisp (real) number. Based on the total order available in $\mathbb{R}$, we can straightforwardly rank fuzzy numbers after defuzzification[5]. For fuzzy numbers A and B and a defuzzification operator DF we define:

$$A =_{DF} B \Leftrightarrow DF(A) = DF(B), A \prec_{DF} B \Leftrightarrow DF(A) < DF(B), A \preceq_{DF} B \Leftrightarrow DF(A) \leq DF(B)$$

We can apply defuzzification at different moments during the searching process for optimal paths in $\tilde{G} = (V, E, \tilde{c})$:

- **Plan A**: defuzzification of costs (before line 1): the fuzzy weighted graph $\tilde{G} = (V, E, \tilde{c})$ becomes a crisp graph $G = (V, E, c)$ (with $c = DF(\tilde{c})$) in which all optimal paths can be obtained by using real operators $(+,*)$ to calculate the objective values, $<_{\mathbb{R}}$ for dominance–checking and $\leq_{\mathbb{R}}$ for ranking the temporary labels in the priority queue

- **Plan B**: fuzzy costs and using defuzzification for dominance-checking (in line 12 and 15): The fuzzy objective value is calculated with the extended operators $\oplus$ and $\odot$). The dominance–relation is determined by $\prec_{DF}$, label ranking in the priority queue by $\leq_{L,DF}$ (such that the condition of sect. 3.3 is fulfilled).

Plan A is the fastest: all imprecision in the fuzzy graph disappears before the (now crisp) search algorithm starts. This strategy has been successfully applied for dynamic routing in wide heterogeneous telecommunication networks where particular levels of Quality of Service (QoS) had to be guaranteed; see e.g. Arnold et al. [1] where a "fuzzy link evaluator" uses a fuzzy rule base to derive one single crisp cost that is derived from lots of criteria). The advantages of plan A are otherwise obvious:

- Specialized implementations of the search algorithm for crisp graphs can be used for fuzzy weighted graphs without any modification

- In order to guarantee the correctness of the labelling algorithms, there are no constraints on the defuzzification method used.

[5]Generally, using this kind of methods there will be only one non–dominated path; this could be seen as a drawback because part of the imprecision in the problem is sacrifised, and as a result some alternative paths which a decision maker might also consider meaningful are left out.

- The uncertainty about the length of the shortest path (i.e. the left and right spreads of the fuzzy number) does not increase when this path is extended.

The following theorem shows when the result of plan B can be obtained by the faster plan A.

Theorem 4.1 *Searching for the optimal paths using the additive or multiplicative cost criterion, plan A and plan B are equivalent if the defuzzification operator satisfies:*
(C.A) $DF(A \oplus B) = DF(A) + DF(B), \forall (A, B) \in \mathcal{A}^2$.
(C.M) $DF(A \odot B) = DF(A).DF(B), \forall (A, B) \in \mathcal{A}^2$.

Proof: Bearing in mind theorem 3.2, it can easily be proved that $\prec_{DF}$ meets (C.1–C.3) if DF meets (C.A) and we use the additive cost criterion. We prove e.g. (C.3). Because $DF(\mathcal{A}) \subseteq \mathbb{R}$ applies for the set of fuzzy objective values $\mathcal{A}$ we can prove for arbitrary A, B and C from $\mathcal{A}$:

$$\begin{aligned} A \prec_{DF} B \quad &\Leftrightarrow DF(A) < DF(B) \Leftrightarrow DF(A) + DF(C) < DF(B) + DF(C) \\ &\Leftrightarrow DF(A \oplus C) < DF(B \oplus C) \Leftrightarrow A \oplus C \prec_{DF} B \oplus C \end{aligned}$$

Analogously, from (C.M) follow (C.1–C.3) using the multiplicative cost criterion because the fuzzy algorithm can only find multiplicative optimal paths in graphs with no negative costs. □

As mentioned before, all ranking methods examined in [6] for which (C.3) applies result in the same final ordening. One of them is the very fast method of Campos and Muñoz which uses the following defuzzification operator: $DF_{CM^\lambda}(A) = \int_0^1 (\lambda a_\alpha^- + (1-\lambda) a_\alpha^+) \mathrm{d}\alpha$. Based on the linearity of the integral–operator, it is easy to prove that (C.A) applies for DF_{CM^λ} and therefore we can conclude that using $<_{DF_{CM^\lambda}}$ insures the correctness of the algorithm and that we can use plan A.

5 Conclusion

We spelled out the necessary and sufficient conditions for correctness of the labelling algorithms for fuzzy weighted graphs; unfortunately, not all commonly used fuzzy ranking techniques satisfy them, so an algorithm developer should take care to check them. Moreover, it was shown that if a defuzzification method is used in the ranking of fuzzy numbers, an optimization by defuzzifying all fuzzy costs before algorithm execution is possible under certain conditions on the defuzzification operator. It is also shown that center of gravity defuzzification does not meet these conditions.

References

[1] W. Arnold, H. Hellendoorn, R. Seising, C. Thomas, A. Weitzel, Fuzzy routing, Fuzzy sets and systems, **85**, 1997, 131–151.

[2] P. De Kesel, Shortest paths in fuzzy graph theory, M. Sc. Thesis (in Dutch), Ghent University, 2002.

[3] E. Q. V. Martins, J. L. E. Santos, The labelling algorithm for the multiobjective shortest path problem, http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/mospp.ps.gz, 1999.

[4] S. Okada, T. Soper, A shortest path problem on a network with fuzzy arc lengths, Fuzzy sets and systems, **109**, 2000, 129–140.

[5] A. Rosenfeld, Fuzzy graphs, Fuzzy sets and their applications to cognitive and decision processes (L. A. Zadeh, K. S. Fu, K. Tanaka, M. Shimura, eds.), Academic Press, New York, 1975, 77–95.

[6] X. Wang, E. E. Kerre, Reasonable properties for the ordering of fuzzy quantities (I - II), Fuzzy sets and systems, **118**, 2001, 375–405.