The Shortest Path Problem on Networks with Fuzzy Parameters

Fábio Hernandes, Maria Teresa Lamata, José Luis Verdegay, and Akebo Yamakami

Fuzzy Sets and Systems, 2007

Presented by Drew Buck

2/14/2013

Overview

- Motivation
- Definitions
  - Graphs
  - Fuzzy numbers
  - Fuzzy ranking methods
- Fuzzy shortest path problem
  - Bellman-Ford algorithm
  - Fuzzy shortest path algorithm
- Examples
- Conclusions

Motivation

Shortest path problems occur in many applications including:

- Transportation
- Routing
- Communications
- Supply chain management
- Models involving agents

Graph Definitions

A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E.

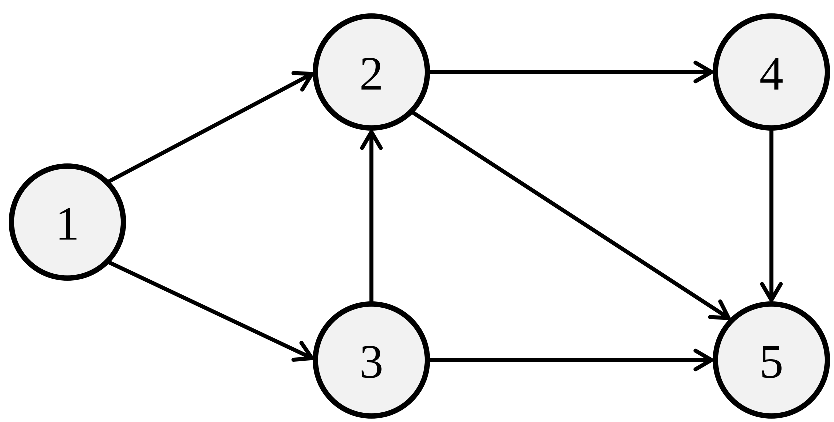
In a directed graph, each edge is an ordered pair (i, j) representing an arc connecting nodes i and j.

A path p_{ij} is a sequence $p_{ij} = \{i, (i, i_1), i_1, \ldots, i_k, (i_k, j), j\}$ of alternating nodes and arcs that connect two nodes i and j.

Graph Definitions

For the shortest path problem, we define a source node s with index 1, and choose t as the destination node.

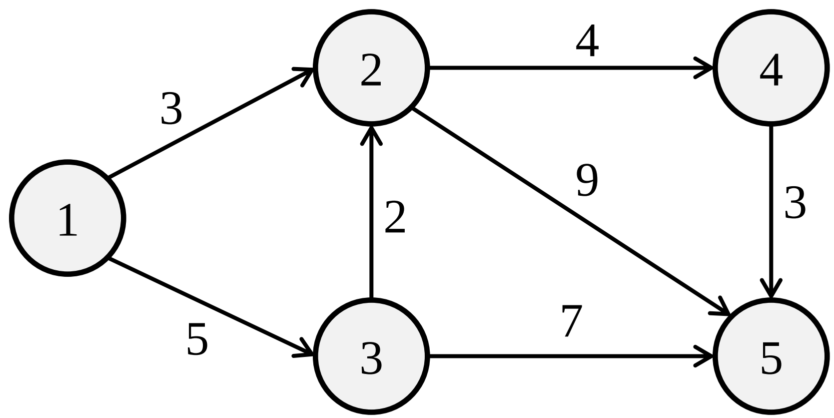
We assume that there exists at least one path p_{si} in $G = (V, E)$ for each node $i \in V - \{s\}$.



Graph Definitions

Each arc (i, j) is assigned a value representing the cost, time, distance, etc. required to traverse from i to j.

Formally, the SPP seeks to find the path corresponding to the minimum cost of travel between the source and destination nodes.



Graph Definitions

Traditionally, the arc weights are defined as real numbers, giving rise to an optimally shortest path.

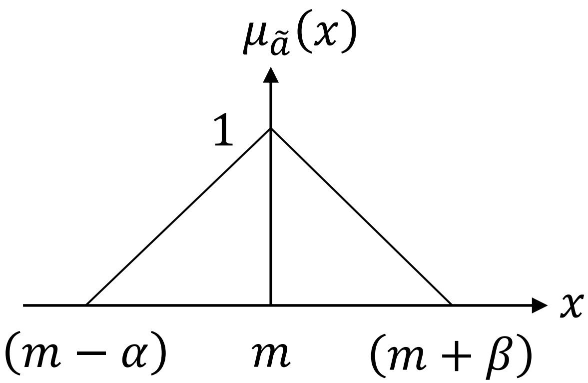
However, many practical applications may find fuzzy numbers to be more appropriate for defining arc weights.

This requires the choice of a ranking function to determine the smaller of two fuzzy numbers.

Fuzzy Numbers

Let us define a triangular fuzzy number $\tilde{a} = (m, \alpha, \beta)$ with the membership function $\mu_{\tilde{a}}(x)$, defined as

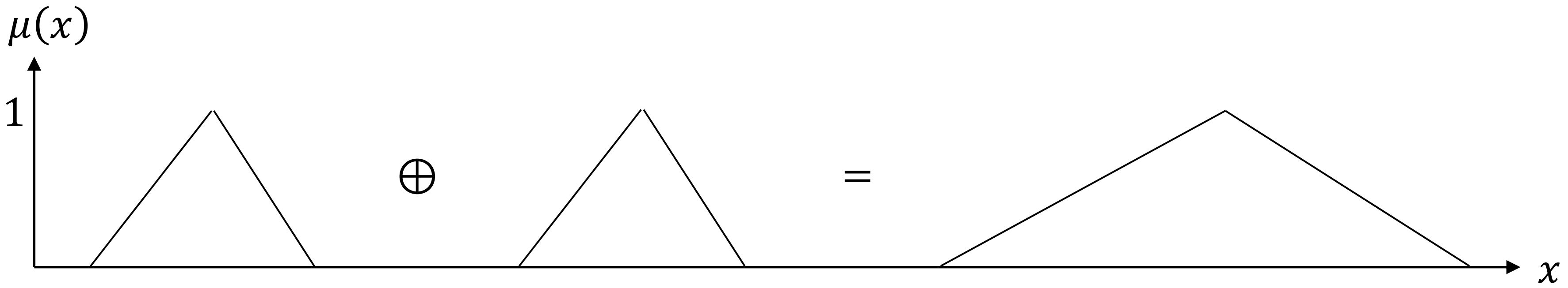
$$\mu_{\tilde{a}}(x) = \begin{cases} 0 & \text{if } x \leqslant m - \alpha, \\ \dfrac{x - (m - \alpha)}{\alpha} & \text{if } m - \alpha < x < m, \\ 1 & \text{if } x = m, \\ \dfrac{(m + \beta) - x}{\beta} & \text{if } m < x < m + \beta, \\ 0 & \text{if } x \geqslant m + \beta \end{cases}$$



Fuzzy Addition

The sum of two fuzzy numbers $\tilde{a} = (m_1, \alpha_1, \beta_1)$ and $\tilde{b} = (m_2, \alpha_2, \beta_2)$ is given as

$$\begin{aligned} \tilde{a} \oplus \tilde{b} &= (m_1, \alpha_1, \beta_1) \oplus (m_2, \alpha_2, \beta_2) \\ &= (m_1 + m_2, \alpha_1 + \alpha_2, \beta_1 + \beta_2) \end{aligned}$$



Fuzzy Ranking

Consider a simple network with edge lengths defined by fuzzy numbers.

The total length of each path is a new fuzzy number.

How can we decide which path is to be preferred?

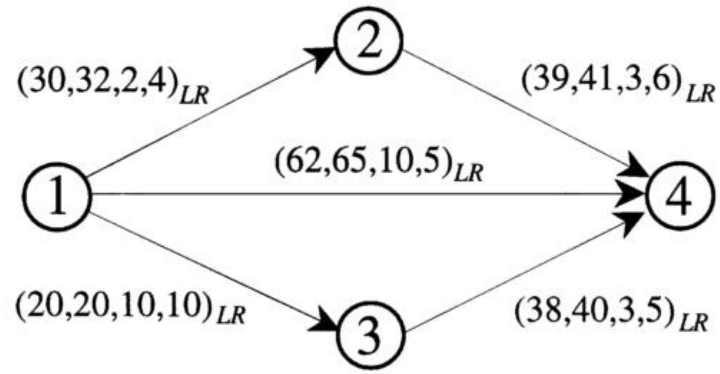


Fig. 1. Small example network.

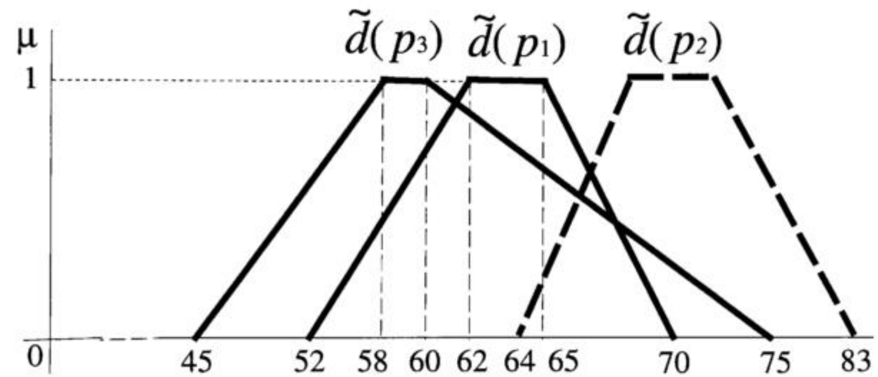


Fig. 2. Each membership function of the path distance.

From “A shortest path problem on a network with fuzzy arc lengths” by S. Okada and T. Soper

Fuzzy Ranking

We say that $\tilde{a}$ is preferred to $\tilde{b}$ $\left(\tilde{a} \prec \tilde{b}\right)$ iff $\tilde{a} < \tilde{b}$.

The following six ranking functions are considered:

- Yager's index
- Liou and Wang index
- García and Lamata index
- Okada and Soper relation
- Nayeem and Pal acceptability index
- Dubois and Prade's possibility index

Yager's Index

Compare the centroids of the two fuzzy numbers.

$$f(\tilde{a}) = \frac{\int x\tilde{a}_x \,\mathrm{d}x}{\int \tilde{a}_x \,\mathrm{d}x}$$

$$\tilde{a} < \tilde{b} \text{ iff } f(\tilde{a}) < f\left(\tilde{b}\right)$$

Liou and Wang Index

Weight the left and right centroids.

$$LW^{\lambda}(\tilde{a}) = \lambda S_D(\tilde{a}) + (1-\lambda) S_I(\tilde{a})$$

where

$$S_D(\tilde{a}) = m + \int_m^{m+\beta} f_{\tilde{a}}^R(x)\,\mathrm{d}x = \int_0^1 f_{\tilde{a}}^{R^{-1}}(y)\,\mathrm{d}y$$

$$S_I(\tilde{a}) = (m-\alpha) + \int_{m-\alpha}^{m} f_{\tilde{a}}^L(x)\,\mathrm{d}x = \int_0^1 f_{\tilde{a}}^{L^{-1}}(y)\,\mathrm{d}y$$

$\lambda \in [0,1]$ represents the decision-maker's optimism/pessimism

$$\tilde{a} < \tilde{b} \text{ iff } LW^{\lambda}(\tilde{a}) < LW^{\lambda}(\tilde{b})$$

García and Lamata Index

Add a modality index $\delta \in [0,1]$ to the previous index, indicating a weighted preference for the modal value of the fuzzy number.

$$I(\tilde{a}) = (1 - \delta)[\lambda S_D(\tilde{a}) + (1 - \lambda) S_I(\tilde{a})] + \delta m$$

$$\tilde{a} < \tilde{b} \text{ iff } I(\tilde{a}) < I(\tilde{b})$$

Okada and Soper Relation

Let $\tilde{a} = (m_1, \alpha_1, \beta_1)$ and $\tilde{b} = (m_2, \alpha_2, \beta_2)$ be two triangular fuzzy numbers and $\varepsilon \in [0,1]$ be an optimism factor.

For α-cuts in $[\varepsilon, 1]$, $\tilde{a}$ dominates $\tilde{b}$ with a degree ε $\left(\tilde{a} \prec_\varepsilon \tilde{b}\right)$ iff

$$\begin{gathered} m_1 \le m_2, \\ (m_1 - \alpha_1)_\varepsilon \le (m_2 - \alpha_2)_\varepsilon, \\ (m_1 + \beta_1)_\varepsilon \le (m_2 + \beta_2)_\varepsilon, \\ \tilde{a} \neq \tilde{b} \end{gathered}$$

$$\tilde{a} < \tilde{b} \text{ iff } \tilde{a} \prec_\varepsilon \tilde{b}$$

Nayeem and Pal Acceptability Index

For two triangular fuzzy numbers $\tilde{a} = (m_1, \alpha_1, \beta_1)$ and $\tilde{b} = (m_2, \alpha_2, \beta_2)$,

$$A(\tilde{a} \prec \tilde{b}) = \frac{m_2 - m_1}{\beta_1 + \alpha_2}$$

$$\tilde{a} < \tilde{b} \text{ iff } A(\tilde{a} \prec \tilde{b}) > A(\tilde{b} \prec \tilde{a})$$

Dubois and Prade's Possibility Index

For two fuzzy numbers $\tilde{a}$ and $\tilde{b}$,

$$\text{Poss}\left(\tilde{a} < \tilde{b}\right) = \sup_{x_i \le x_j} \min\left(\tilde{a}(x_i), \tilde{b}\left(x_j\right)\right)$$

$$\tilde{a} < \tilde{b} \text{ iff } \text{Poss}\left(\tilde{a} < \tilde{b}\right) > \text{Poss}\left(\tilde{b} < \tilde{a}\right)$$

Fuzzy Ranking

These six ranking indexes can be classified into two groups:

- Indexes that map fuzzy numbers into crisp numbers
  - Yager’s index
  - Liou and Wang index
  - García and Lamata index
- Indexes that compare the ordering of two fuzzy numbers
  - Okada and Soper relation
  - Nayeem and Pal acceptability index
  - Dubois and Prade’s possibility index

Fuzzy Shortest Path Problem

Given the set of directed edges E, where each arc $(i, j) \in E$ is assigned a fuzzy number $\tilde{c}_{ij}$, FSPP is formally defined as a linear programming problem:

$$\min \quad \tilde{f}(x) = \sum_{(i,j)\in E} \tilde{c}_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{if } i \neq 1, t\,(i = 1, \ldots, r), \\ -1 & \text{if } i = t, \end{cases}$$

$$x_{ij} = 0 \text{ or } 1 \quad \text{for } (i, j) \in E,$$

where r is the number of nodes, t is the destination node and $\sum$ refers to the addition of fuzzy numbers.

Fuzzy Shortest Path Problem

Because of the various ranking methods for fuzzy numbers, we cannot solve the linear program directly.

This has lead to the development of several specialized algorithms.

- Dubois and Prade’s extension of the Floyd-Warshall and Bellman-Ford algorithms
- Klein’s dynamic programming method
- Lin and Chern searched for arcs that increase total cost when removed from the path
- Okada *et al.* extended Dijkstra’s algorithm to find a Pareto optimal path
- Blue *et al.* used a modified *k*-shortest path algorithm proposed by Eppstein
- Okada considered the possibility of an arc being on the shortest path
- Nayeem and Pal used an algorithm based on their acceptability index

Fuzzy Shortest Path Problem

These methods often present peculiarities and/or problems:

- They find costs without an existing path.
- They do not provide decision-makers with any guidelines for choosing the best path.
- They can only be applied to graphs with fuzzy non-negative parameters.

Bellman-Ford Algorithm (Crisp)

The Bellman-Ford algorithm finds the shortest paths in a graph G, given a source vertex s and a set of weights w. For each vertex v, $d[v]$ stores an upper bound on the distance from s to v and $\pi[v]$ stores the best path predecessor of v.

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i ← 1 to |V[G]| − 1
    do for each edge (u, v) ∈ E[G]
        do RELAX(u, v, w)
for each edge (u, v) ∈ E[G]
    do if d[v] > d[u] + w(u, v)
        then return FALSE
return TRUE
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ V[G]
    do d[v] ← ∞
       π[v] ← NIL
d[s] ← 0
```

```
RELAX(u, v, w)
if d[v] > d[u] + w(u, v)
    then d[v] ← d[u] + w(u, v)
         π[v] ← u
```

From "Introduction to Algorithms" by T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein

Bellman-Ford Algorithm (Crisp)

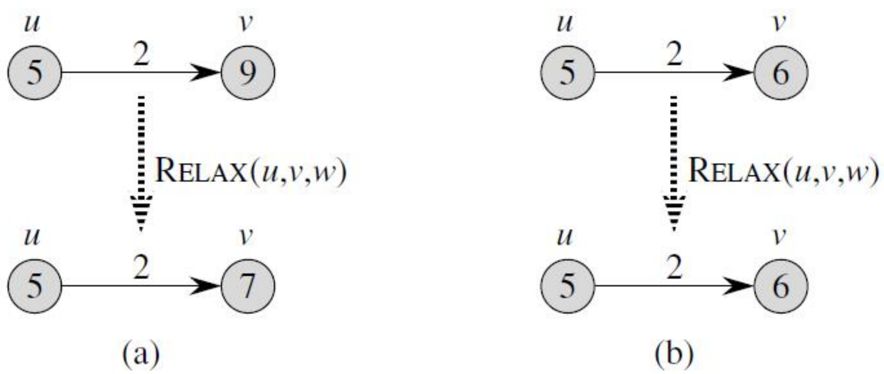


Figure 24.3 Relaxation of an edge (u, v) with weight $w(u, v) = 2$. The shortest-path estimate of each vertex is shown within the vertex. **(a)** Because $d[v] > d[u] + w(u, v)$ prior to relaxation, the value of $d[v]$ decreases. **(b)** Here, $d[v] \le d[u] + w(u, v)$ before the relaxation step, and so $d[v]$ is unchanged by relaxation.

From "Introduction to Algorithms" by T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein

Bellman-Ford Algorithm (Crisp)

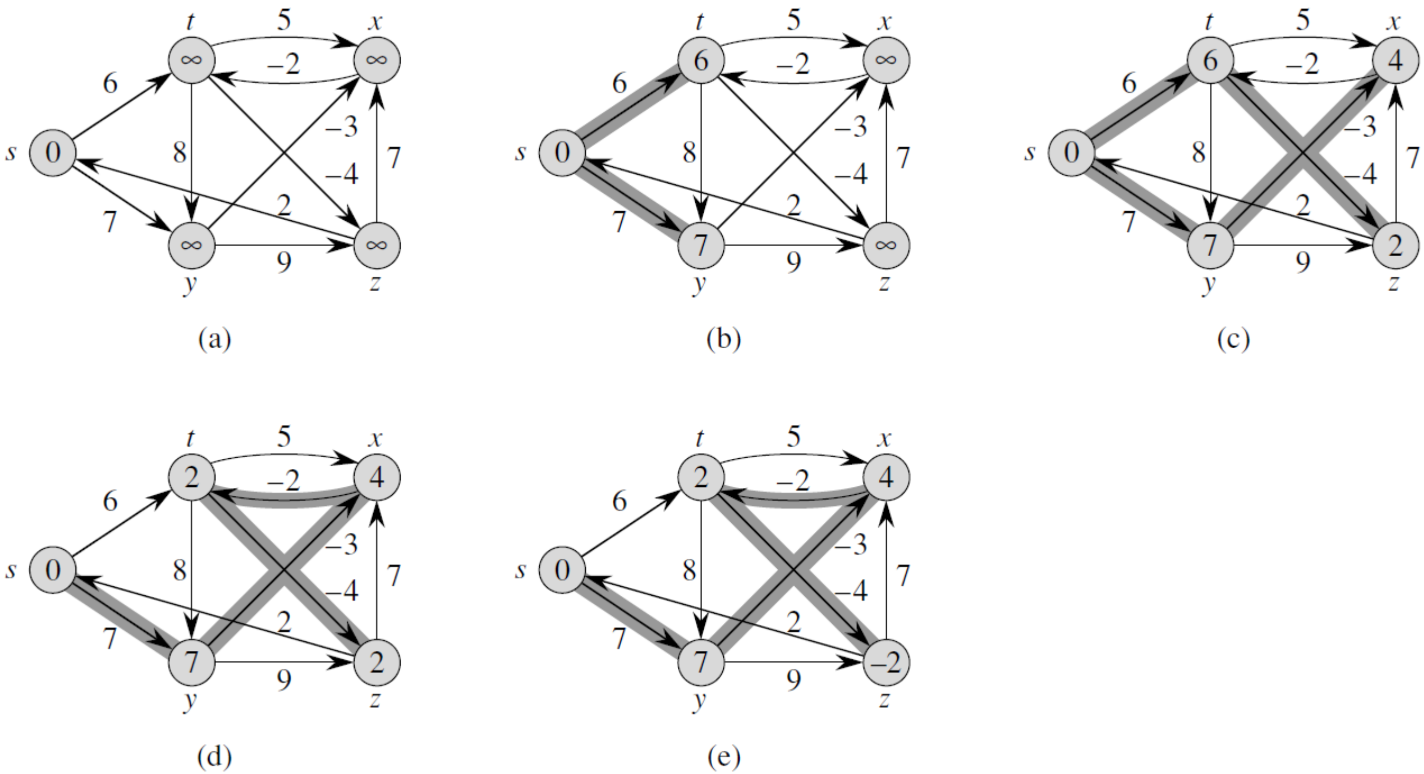


Figure 24.4 The execution of the Bellman-Ford algorithm. The source is vertex s. The d values are shown within the vertices, and shaded edges indicate predecessor values: if edge (u, v) is shaded, then $\pi[v] = u$. In this particular example, each pass relaxes the edges in the order $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$. **(a)** The situation just before the first pass over the edges. **(b)–(e)** The situation after each successive pass over the edges. The d and π values in part (e) are the final values. The Bellman-Ford algorithm returns TRUE in this example.

From "Introduction to Algorithms" by T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein

Fuzzy Shortest Path Algorithm

Notation:

r	number of nodes;
it	iteration counter;
E	set of edges;
$M = \sum_{i=1}^{E} \lvert (m+\beta)^i \rvert$	a large number substituting ∞;
$\tilde{c}_{ji}$	cost of arc (j, i);
$p_{(1,t)}$	path between nodes 1 and t;
Γ_i^{-1}	set of predecessor nodes of i;
$\tilde{d}_k^{it}(p_{(1,t)})$	distance along path $p_{(1,t)}$ of the k^{th} label in the iteration it;
$d_k^{it}(p_{(1,t)})$	the ranking index applied to $\tilde{d}_k^{it}(p_{(1,t)})$;

Fuzzy Shortest Path Algorithm

Step 0: [Initialization]

(1) $\tilde{d}_1^0(p_{(1,1)}) = (0,0,0)$

(2) $\tilde{d}_1^0(p_{(1,j)}) = (M+2, 1, 1), \quad j = 2,3, \dots, r$

(3) $it \leftarrow 1$

Step 1: [Determination of distance paths, dominance check, and negative circuit]

(1) $\tilde{d}_1^{it}(p_{(1,1)}) = (0,0,0)$

(2) [Determination of fuzzy path distances: The distance between nodes 1 and i is the fuzzy addition of the distance of the path with the predecessor node in the previous iteration $\tilde{d}_l^{it-1}(p_{(1,j)})$ and the cost of arc (j,i)]

- $\forall j \in \Gamma_i^{-1}, \; i = 2, 3, \dots, r,$ do:
  $\tilde{d}_k^{it}(p_{(1,i)}) = \tilde{d}_l^{it-1}(p_{(1,j)}) \oplus \tilde{c}_{ji}$

Fuzzy Shortest Path Algorithm

(3) [Dominance check: For each node $i \in N$, the dominance is checked for all the labels of the node i, being compared one to one]

- If $d_k^{it}\left(p_{(1,i)}\right) > d_l^{it}\left(p_{(1,i)}\right) \Longrightarrow$ delete the label k
- If $d_l^{it}\left(p_{(1,i)}\right) < d_k^{it}\left(p_{(1,i)}\right) \Longrightarrow$ delete the label l

(4) [Verification of a negative circuit: The verification of the existence of a negative circuit is performed by means of the applied index on the distance of the path $p_{(1,j)}$. If the results are negative, the algorithm will be in an infinite loop]

- If there is at least one node i such that $d_k^{it}\left(p_{(1,i)}\right) < 0$ then
  - Go to step 4 [negative circuit]
  - Otherwise go to step 2 (next slide)

Fuzzy Shortest Path Algorithm

Step 2: [Stop criterion] For all nodes and all labels do:

(1) If $\left(\tilde{d}_k^{it}\left(p_{(1,i)}\right) = \tilde{d}_k^{it-1}\left(p_{(1,i)}\right)\right)$ or $(it = r)$ do:

- If $it = r$ and $\left(\tilde{d}_k^{it}\left(p_{(1,i)}\right) \neq \tilde{d}_k^{it-1}\left(p_{(1,i)}\right)\right)$ then
  - Go to step 4 [negative circuit]
  - Otherwise go to step 3

(2) Otherwise $it = it + 1$, go to step 1.

Step 3: [Shortest paths composition: Find the shortest paths from 1 to i $(i = 2, 3, \ldots, r)$. It is sufficient to store in block 1.2 the predecessor nodes of i that are used to rebuild the shortest paths]

Step 4: [Termination: Finish the execution of the algorithm]

Complexity Analysis

The algorithm takes at most $r - 1$ iterations to converge.

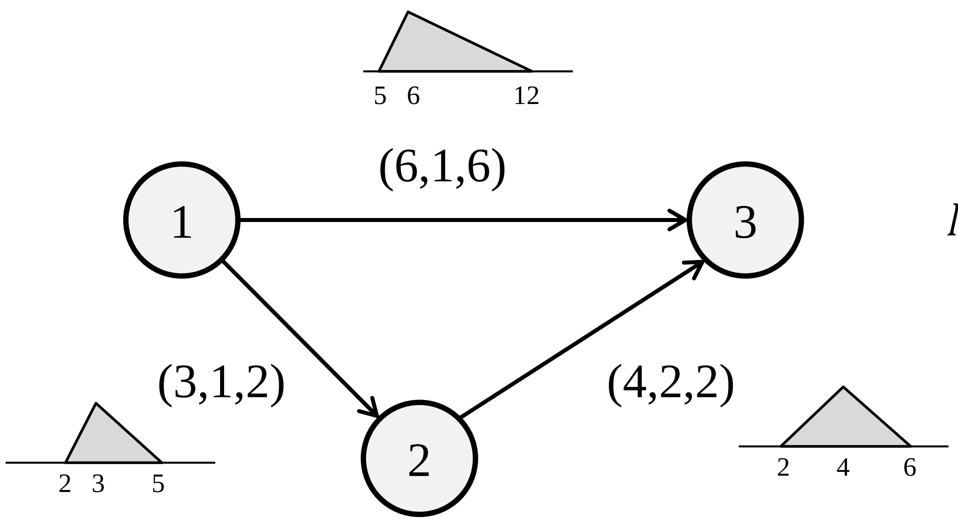
In step 1 of each iteration a maximum of $rV_{\max}$ additions are computed where $V_{\max}$ is the maximum number of labels that can be assigned to a node.

In step 2 of each iteration, a maximum of $rV_{\max}^2$ dominance comparisons are required.

This gives an overall complexity of

$$O\left((r-1)\left(rV_{\max}^2\right)\right) = O\left(r^2V_{\max}^2 + rV_{\max}^2\right) = O\left(r^2V_{\max}^2\right)$$

Example



$$M = \sum_{i=1}^{E} \left|(m+\beta)^i\right| = 5 + 6 + 12 = 23$$

Initialization

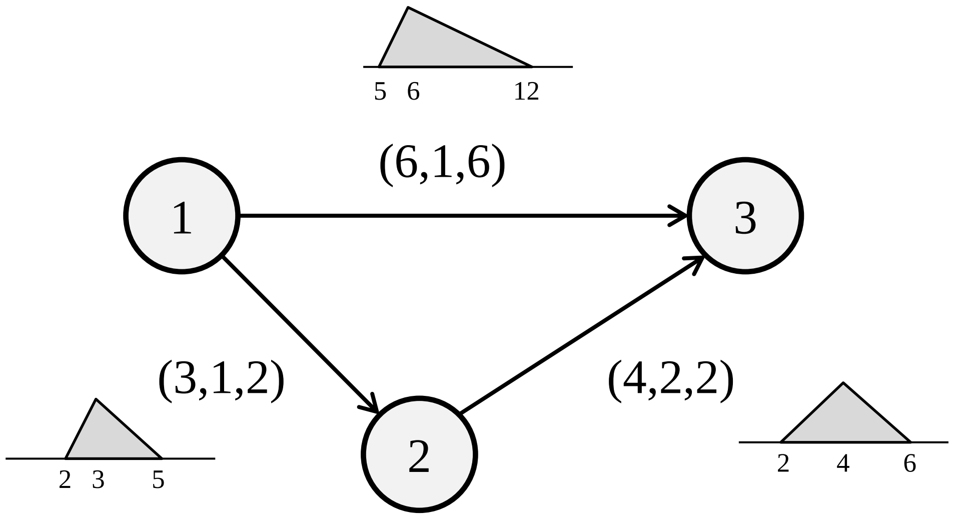
$it = 0$:

$\tilde{d}_1^0\left(p_{(1,1)}\right) = (0,0,0)$

$\tilde{d}_1^0\left(p_{(1,2)}\right) = (25,1,1)$

$\tilde{d}_1^0\left(p_{(1,3)}\right) = (25,1,1)$

Example



Determination of fuzzy path distances

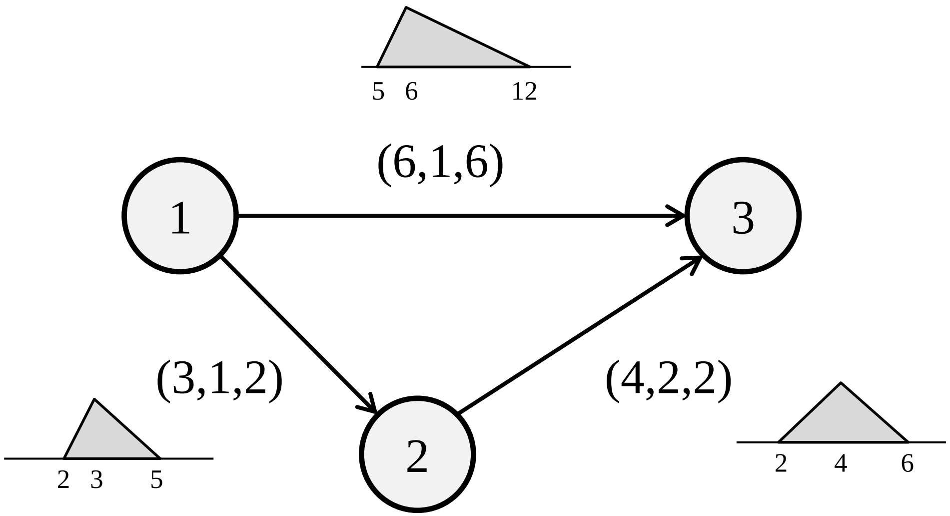
$it = 1$:

$\tilde{d}_1^1\left(p_{(1,1)}\right) = (0,0,0)$

$\tilde{d}_1^1\left(p_{(1,2)}\right) = (25,1,1)$ $\quad \tilde{d}_2^1\left(p_{(1,2)}\right) = (3,1,2)^1$

$\tilde{d}_1^1\left(p_{(1,3)}\right) = (25,1,1)$ $\quad \tilde{d}_2^1\left(p_{(1,3)}\right) = (6,1,6)^1$ $\quad \tilde{d}_3^1\left(p_{(1,3)}\right) = (29,3,3)^2$

Example



Dominance check

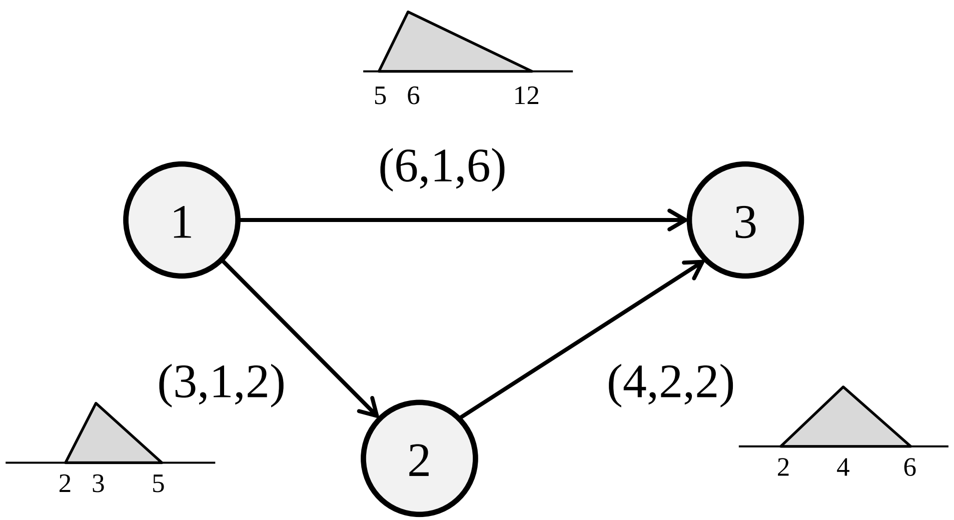
$it = 1$:

$\tilde{d}_1^1\left(p_{(1,1)}\right) = (0,0,0)$

$\tilde{d}_1^1\left(p_{(1,2)}\right) = (3,1,2)^1$

$\tilde{d}_1^1\left(p_{(1,3)}\right) = (6,1,6)^1$

Example



Determination of fuzzy path distances

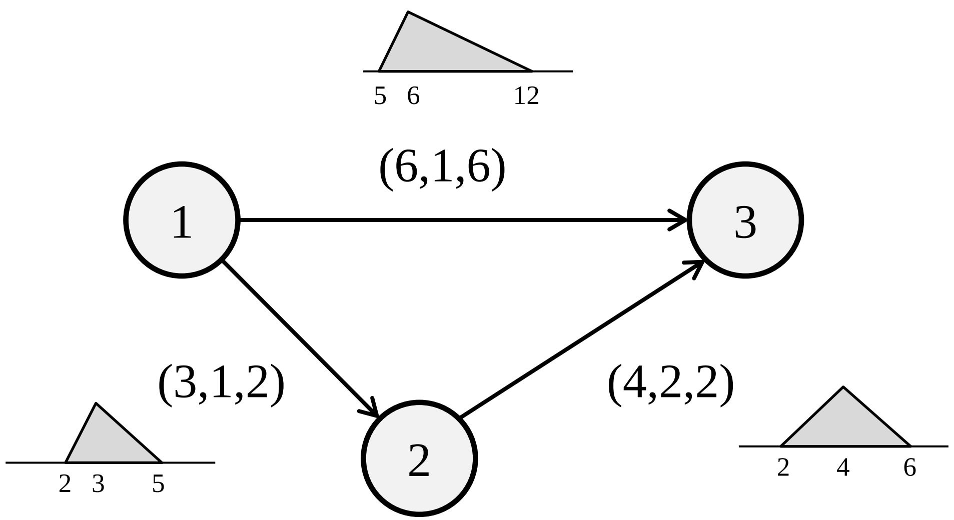
$it = 2$:

$\tilde{d}_1^2\left(p_{(1,1)}\right) = (0,0,0)$

$\tilde{d}_1^2\left(p_{(1,2)}\right) = (3,1,2)^1$

$\tilde{d}_1^2\left(p_{(1,3)}\right) = (6,1,6)^1$ $\qquad$ $\tilde{d}_2^2\left(p_{(1,3)}\right) = (7,3,4)^2$

Example



Dominance check

$it = 2$:

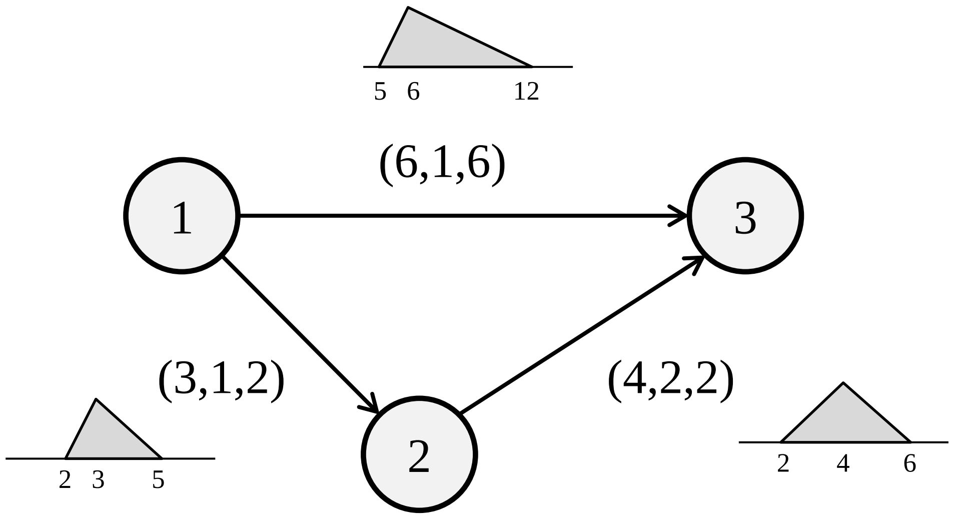
$\tilde{d}_1^2\left(p_{(1,1)}\right) = (0,0,0)$

$\tilde{d}_1^2\left(p_{(1,2)}\right) = (3,1,2)^1$

$\tilde{d}_1^2\left(p_{(1,3)}\right) = (6,1,6)^1$ $\tilde{d}_2^2\left(p_{(1,3)}\right) = (7,3,4)^2$

Depends on the choice of ranking function

Example



No change in 3rd iteration; algorithm terminates

$it = 3$:

$\tilde{d}_1^2(p_{(1,1)}) = (0,0,0)$

$\tilde{d}_1^2(p_{(1,2)}) = (3,1,2)^1$

$\tilde{d}_1^2(p_{(1,3)}) = (6,1,6)^1$ $\tilde{d}_2^2(p_{(1,3)}) = (7,3,4)^2$

Depends on the choice of ranking function

Example

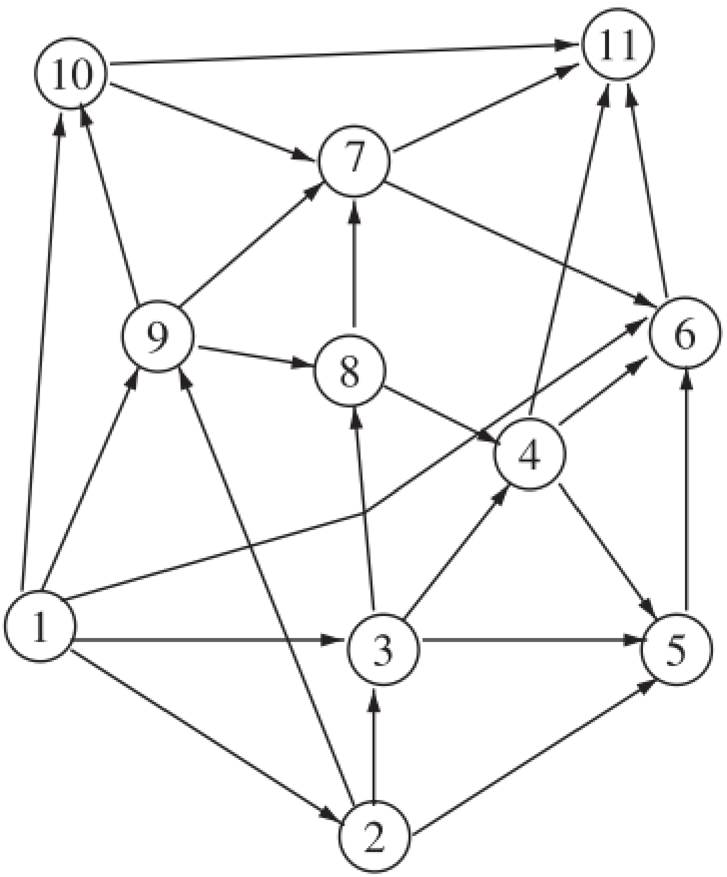


Fig. 1. Example network.

Table 1
Edge information—Example 1

Source node	Destination node	Arc cost
1	2	(820 20 20)
1	3	(361 11 9)
1	6	(677 27 6)
1	9	(300 10 50)
1	10	(450 30 20)
2	3	(186 6 7)
2	5	(510 15 15)
2	9	(930 30 30)
3	4	(667 17 216)
3	5	(748 18 22)
3	8	(443 18 22)
4	5	(199 9 11)
4	6	(340 30 20)
4	11	(740 30 30)
5	6	(660 50 30)
6	11	(242 12 18)
7	6	(410 20 30)
7	11	(472 22 18)
8	4	(730 20 5)
8	7	(242 12 13)
9	8	(137 7 8)
9	7	(130 10 20)
9	10	(242 12 18)
10	7	(342 12 8)
10	11	(1310 60 130)

Example

Table 2
Results of Example 1

Destination node	Shortest path	Cost path	Order relation
2	$1 \to 2$	(820 20 20)	All
3	$1 \to 3$	(361 11 9)	All
4	$1 \to 3 \to 4$	(1028 28 225)	All
4	$1 \to 9 \to 8 \to 4$	(1167 37 63)	Okada and Soper ($\varepsilon = 0$)
5	$1 \to 3 \to 5$	(1109 29 31)	All
6	$1 \to 6$	(677 27 6)	All
7	$1 \to 9 \to 7$	(430 20 70)	All
8	$1 \to 9 \to 8$	(437 17 58)	All
9	$1 \to 9$	(300 10 50)	All
10	$1 \to 10$	(450 30 20)	All
11	$1 \to 9 \to 7 \to 11$	(902 42 88)	Yager; Liou and Wang ($\lambda = 1$); García and Lamata ($\lambda = 1, \delta = 0$); Okada and Soper ($\varepsilon = 0$ and 0.5)
11	$1 \to 6 \to 11$	(919 39 24)	Liou and Wang ($\lambda = 0$ and 0.5); García and Lamata (except $\lambda = 1, \delta = 0$); Okada and Soper ($\varepsilon = 0$ and 0.5); Nayeem and Pal and Dubois and Prade

Example

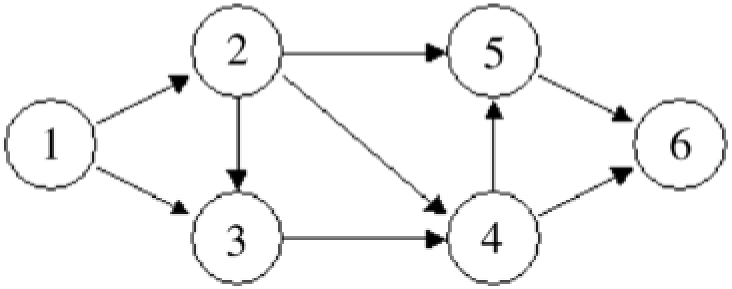


Fig. 2. Acyclic network.

Table 3
Cost of acyclic network—Example 2

Source node	Destination node	Arc cost
1	2	(2 1 1)
1	3	(7 2 2)
2	3	(4 3 5)
2	4	(11 1 1)
2	5	(6 1 1)
3	4	(9 1 1)
4	5	(−8 1 1)
4	6	(13 2 1)
5	6	(9 1 1)

Example

Table 4
Results of acyclic network—Example 2

Destination node	Shortest path	Cost path	Order relation
2	$1 \to 2$	(2 1 1)	All
3	$1 \to 3$	(7 2 2)	Okada and Soper ($\varepsilon = 0$ and 0.5); Liou and Wang ($\lambda = 1$); García and Lamata ($\lambda = 1$, $\delta = 0$ and $\lambda = 1$, $\delta = 0.5$)
3	$1 \to 2 \to 3$	(6 4 6)	All, except Liou and Wang ($\lambda = 1$); García and Lamata ($\lambda = 1$, $\delta = 0$)
4	$1 \to 2 \to 4$	(13 2 2)	All
4	$1 \to 2 \to 3 \to 4$	(15 5 7)	Okada and Soper ($\varepsilon = 0$)
5	$1 \to 2 \to 4 \to 5$	(5 3 3)	All
5	$1 \to 2 \to 3 \to 4 \to 5$	(7 6 8)	Okada and Soper ($\varepsilon = 0$)
6	$1 \to 2 \to 4 \to 5 \to 6$	(14 4 4)	All
6	$1 \to 2 \to 3 \to 4 \to 5 \to 6$	(16 7 9)	Okada and Soper ($\varepsilon = 0$)

Conclusions

The fuzzy shortest path problem has a wide range of applications.

Depending on the fuzzy ranking, the presented algorithm can return a single path or a set of non-dominated paths.

The algorithm can work with fuzzy numbers of any type, so long as an appropriate ordering is defined.

By extending the Bellman-Ford algorithm, this method can handle graphs with negative arc weights and can detect negative cycles.