International Journal of Control and Automation
Vol.6, No.5 (2013), pp.171-180
http://dx.doi.org/10.14257/ijca.2013.6.5.16

The Minmax Regret Shortest Path Problem with Interval Arc Lengths

Jun-Gyu Kang

Department of Industrial & Management Engineering,
Sungkyul University, Korea

jun-gyu.kang@sungkyul.ac.kr

Abstract

This paper considers the shortest path problem on directed acyclic graphs, where the uncertainty of input data (lengths of arcs) is modeled in the form of intervals. In order to handle the interval data the minmax criterion to the regret values is applied, where the original objective function with interval coefficients is transformed into that of finding the least maximum worst-case regret, which is called the minmax regret criterion. A heuristic algorithm exploiting properties to reduce the computational times and to improve the solution quality is developed and its performance is shown with computational experiments.

Keywords: *Shortest Path, Interval Data, Minmax regret*

1. Introduction

The classical shortest path problem on directed acyclic graphs (DAGs) is a very well-known problem with lots of applications, *i.e.*, route planning for transportation networks or for telecommunication networks, and community clustering for E-commerce [1]. Its deterministic version can be solved efficiently, for example, by using Dijkstra's algorithm or using Bellman-Ford's algorithm. For large scale problems, meta-heuristics, *i.e.*, the ant-colony optimization [2] or the genetic algorithm [3], can be applied. But in practical applications, the exact values of input data like lengths of arcs or duration of activities are often unknown in advance, due to the uncertain travel times on transportation networks and the uncertain transmission delays on telecommunication or on electric power networks [4]. In such cases, the most common approach to determine the optimal path is that the uncertainty is characterized by random distributions and the expected value of a utility function is maximized [5]. Nevertheless, the stochastic approach has limits to its application since it is often hard to determine proper random distributions due to the variable nature of the world or to a lack of information about a considered system.

With the least assumption about ambiguous variables, the simplest form of the uncertainty representation to adopt is that the value of arc lengths lies within a given range, independently from the values taken by the other parameters [6]. Here, the possible realization of the arc length corresponds to the interval ranges defined by known lower and upper bounds, but the actual realization and the probability distribution function are unknown. For the shortest path problem with interval data, the *minmax regret* criterion can be applied to find the relatively robust path [7]. Here, the regret denotes the deviation of length between a given path and the optimal path under a certain realization of arc lengths. Note that it is also called the *relative robustness* criterion or the *robust deviation* criterion.

ISSN: 2005-4297 IJCA

The *Minmax Regret Shortest Path Problem with interval arc lengths* (MRSPP hereafter) is identifying the one; of which maximum regret over all possible realization of arc lengths is minimized, and a mixed integer programming formulation of the problem is presented in [8]. It is hard to obtain the exact solution, since the MRSPP is *NP*-hard even if the network is directed, acyclic, has a layered structure, and all bounds of intervals of uncertainty belong to {0, 1} [9, 10]. As a practical approach for large scale problems, an enumerative greedy algorithm for the minmax regret longest path problem, where the *k* non-dominated paths are generated by a branch-and-bound algorithm, was developed in [11]. Here, the non-dominated path indicates the path that is the shortest in at least one realization of arc lengths. Note that the longest path problem on DAGs can be transformed to the shortest path problem on DAGs, without loss of generality. Similar work was done by [12], which enumerates the *k* shortest paths with improvement rules. In both heuristic algorithms, generating the paths until the near-optimal solution comes, takes extensive computational time, as checking the non-dominancy is *NP*-complete and generating the *k* non-dominated paths is *NP*-hard [13]. As a constructive method, heuristic algorithms to estimate the maximum regret of the path from the source node to each intermediate node were developed in [14, 15].

This paper expands results in [14] by refining some properties, which reduces computational times and improves solution quality. It works for general directed graphs, although the performance of the suggested algorithm is illustrated with directed acyclic, layered graphs.

This paper is organized as follows: In Section 2, we describe the problem as that of finding the shortest path in a directed acyclic graph with interval arc lengths. In Section 3, some properties and an algorithm by exploiting the properties are suggested. In Section 4, the computational experiments on directed acyclic, layered graphs with small widths are presented and the results compared with those of the algorithm in [15] are followed. Finally, conclusions can be found in Section 5.

2. Problem Description and Preliminaries.

Let $\mathbf{G} = (\mathbf{N}, \mathbf{A})$ denote a directed graph, where **N** is the set of nodes and **A** is the set of arcs. An interval $[l_{ij}, u_{ij}]$ associated with each arc $(i, j) \in \mathbf{A}$ represents the range of possible length for each arc, and is independent from the value of other arcs. Then, the shortest path problem can be formulated as a traditional linear programming model as follows:

$$(\mathrm{P})\ \text{Min} \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = \begin{cases} 1 & \text{if } i = o, \\ -1 & \text{if } i = t, \\ 0 & \textit{otherwise}, \end{cases} \tag{2}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in \mathbf{A}, \tag{3}$$

where $l_{ij} \le c_{ij} \le u_{ij}$ for all $(i, j) \in \mathbf{A}$, and *o* and *t* denote the origin node and the terminal node, respectively. x_{ij} is a binary decision variable, where $x_{ij} = 1$ if arc (i, j) belongs to the shortest path, $x_{ij} = 0$ otherwise. In case where all of c_{ij} are deterministic, the

problem can be solved by using the well-known Dijkstra's algorithm and other efficient algorithms. However, in this paper, since c_{ij} is assumed to be given in the form of interval $[l_{ij}, u_{ij}]$, no path can always be the shortest path over all realization of arc lengths and thus, the minmax regret criterion is applied as a surrogate measure. The MRSPP can be formally described as follows.

A particular realization of arc's length, such that $c_{ij}^{\mathbf{s}} \in [l_{ij}, u_{ij}]$ is fixed $\forall (i, j) \in \mathbf{A}$, is called a scenario **s**. Thus, every scenario represents a specific realization of arc's lengths, which possibly occurs with a positive, but unknown probability. Suppose a given arbitrary scenario **s**, path **x** and path **y** from *o* to *t*, the regret of path **x** against path **y** is defined by the difference between the lengths of paths as follows:

$$R(\mathbf{x}, \mathbf{y}, \mathbf{s}) = |\, \mathbf{c}^{\mathbf{s}}\mathbf{x} - \mathbf{c}^{\mathbf{s}}\mathbf{y} \,| \tag{4}$$

Let Φ be the set of all possible paths from *o* to *t* and Γ be the set of all scenarios. Then the maximal regret of a given path **x** can be defined as follows:

$$R_{\max}(\mathbf{x}) = \max_{\mathbf{s} \in \Gamma, \mathbf{y} \in \Phi} \mathbf{c}^{\mathbf{s}}\mathbf{x} - \mathbf{c}^{\mathbf{s}}\mathbf{y} \tag{5}$$

In equation (5), the scenario **s** and the path **y** that maximize the right hand side is called the *worst-case scenario* and the *worst-case alternative* for **x**, respectively. Equation (5) can be solved straightforwardly by setting arc's length, which is called *c-consistency* by [16], as follows:

$$c_{ij} = \begin{cases} u_{ij} & \text{if } arc(i, j) \in \mathbf{x} \\ l_{ij} & otherwise \end{cases} \tag{6}$$

For a given path **x**, the arc's length can be fixed as equation (6) to maximize equation (5) and the scenario is called the worst-case scenario for **x**. Equation (6) signifies that it is sufficient to deliberate the extreme point of $c_{ij} \in \{l_{ij}, u_{ij}\}$ while computing the maximum regret, and thus the number of possible scenarios is $|\mathbf{S}| = 2^m$ ($|A| = m$). Under the worst case scenario **s** is induced by the given path **x**, the shortest path maximizes the right-hand side of equation (5) and thus it is called the worst-case alternative for **x** or the worst-case alternative induced by the scenario **s** [12]. So, using equation (6), equation (4) and (5) can be simply written as follows:

$$R_{\max}(\mathbf{x}) = R(\mathbf{x}, \mathbf{y}) = \mathbf{c}^{u}\mathbf{x} - \mathbf{c}^{l}\mathbf{y} \tag{7}$$

where, $\mathbf{c}^{u}$ and $\mathbf{c}^{l}$ denote the cost vector induced by c-consistency and **y** denotes the worst-case alternative of **x**, in that, the shortest path derived from $\mathbf{c}^{u}$ and $\mathbf{c}^{l}$.

Then the MRSPP can be re-defined as follows:

$$\min_{\mathbf{x} \in \Phi} R_{\max}(\mathbf{x}) = \min_{\mathbf{x} \in \Phi} \max_{\substack{\mathbf{s} \in \Gamma \\ \mathbf{y} \in \Phi}} (\mathbf{c}^{\mathbf{s}}\mathbf{x} - \mathbf{c}^{\mathbf{s}}\mathbf{y}) \tag{8}$$

Applying equation (6), a mixed integer programming formulation is presented in [8] as follows:

$$(\text{MRSPP}) \ \text{Min} \quad \sum_{(i, j) \in A} u_{ij} x_{ij} - y_t \tag{9}$$

$$\text{subject to} \quad y_j \le y_i + l_{ij} + (u_{ij} - l_{ij})x_{ij} \quad \forall (i, j) \in A, \tag{10}$$

$$y_j \ge 0 \quad \forall i \in N, \tag{11}$$

(2) and (3)

where $x_{ij} = 1$, if arc(i, j) belongs to the minmax regret shortest path; $x_{ij} = 0$ otherwise, y_j holds the cost of the shortest path from *o* to node *i* in the scenario induced by the path defined by **x** variables. The significant inequalities of the formulation are those in (10), which determine the length of each arc $c_{ij} \in \{l_{ij}, u_{ij}\}$ to keep *c-consistency* induced by **x** variables. The remaining constraints are basically the same as those of the classical shortest path problem described in equation (2) and (3).

3. The Algorithm

In this section, we present properties, which reduce computational times and improve the quality of the solution, develop rules based on the properties and build an efficient heuristic algorithm exploiting the rules.

3.1 Main Ideas

First, we suggest a property to reduce the computation time to obtain the worst-case alternative path for a given solution path. From the minmax regret and the *c-consistency* defined earlier, it is easy to find the worst-case alternative for a given solution as follows:

Property 1 [16]**.** The worst-case alternative for a given solution **x** can be determined as the shortest path from the origin *o* to the terminal *t* in the scenario where each arc on **x** has cost $c_{ij} = u_{ij}$ and the rest has cost $c_{ij} = l_{ij}$.

Property 1 is already described in Section 2, through equation (5), (6) and (7). From Property 1, the following rule can be derived directly as follows:

Rule 1: Initiation. For a null path from *o* to *o*, the scenario *r* which maximizes the regret is that $c_{ij} = l_{ij}$ for all arcs and the worst case alternative is the shortest path where $c_{ij} = l_{ij}$ for all arcs.

From Rule 1, the algorithm initiates a null path with the corresponding worst-case alternative and the maximum regret. Then, if a partial path from *o* to $i \in \mathbf{N}$ is determined, the worst-case alternative and the maximum regret can be found easily by extending Property 1. Given a partial path **x** from *o* to $i \in \mathbf{N}$, the worst case scenario **s** is that $c_{ij} = u_{ij}$ for each arc (i, j) on **x** and $c_{ij} = l_{ij}$ for the rest, and the worst case alternative is the shortest path from *o* to *t* under the scenario **s**.

However, to avoid solving the shortest path problem at each node, the following proposition can be used by expanding the partial path to an immediate successor arc. In the proposition given below, $\mathbf{x}^i$ denotes the minmax regret partial path to *i*, defined as the partial path from *o* to $i \in \mathbf{N}$, which has the least maximum regret among all feasible partial paths from *o* to *i* over all possible scenarios. And, $\mathbf{y}^i$ denotes the worst-case

alternative path from *o* to *t* for $\mathbf{x}^i$. Suppose that $\mathbf{x}^i$ and $\mathbf{y}^i$ are given for node $i \in \mathbf{N}$ ($i \neq t$), let node $j \in \mathbf{N}$ be an immediate successor of node *i* and $\mathbf{x}^i_{+ij}$ be $\mathbf{x}^i \cup \text{arc}(i, j)$.

Proposition 1. Given a minmax regret partial path $\mathbf{x}^i$ and the corresponding worst-case alternative $\mathbf{y}^i$, if $\text{arc}(i, j) \notin \mathbf{y}^i$ then the path $\mathbf{x}^i_{+ij}$ is a minmax regret partial path from *o* to $\text{arc}(i, j)$ and $\mathbf{y}^i$ is the corresponding worst-case alternative.

Proof. Suppose that $\text{arc}(i, j)$ is not included in any worst-case alternative for path $\mathbf{x}^i$. Then, by the definition of the minmax regret shortest path derived from equation (8), we have,

$$\min_{\mathbf{x} \in \Lambda \cup \text{arc}(i,j)} R_{\max}(\mathbf{x}) = \min_{\mathbf{x} \in \Lambda} \left(u_{ij} - R_{\max}(\mathbf{x}) \right) = u_{ij} - \min_{\mathbf{x} \in \Lambda} R_{\max}(\mathbf{x}), \tag{12}$$

where, Λ denotes the set of all feasible paths from *o* to node *i*. From the assumption of Proposition 1, we have,

$$\min_{\mathbf{x} \in \Lambda} R_{\max}(\mathbf{x}) = R(\mathbf{x}^i, \mathbf{y}^i), \tag{13}$$

Using equation (7), the maximum regret of the partial path $\mathbf{x}^i_{+ij}$ can be calculated as follows:

$$R_{\max}(\mathbf{x}^i_{+ij}) = R(\mathbf{x}^i_{+ij}, \mathbf{y}^i) = u_{ij} - R(\mathbf{x}^i, \mathbf{y}^i), \tag{14}$$

From (12), (13) and (14),

$$\min_{\mathbf{x} \in \Lambda \cup \text{arc}(i,j)} R_{\max}(\mathbf{x}) = u_{ij} - R(\mathbf{x}^i, \mathbf{y}^i) = R_{\max}(\mathbf{x}^i_{+ij}), \tag{15}$$

This completes the proof. □

Proposition 1 allows $\text{arc}(i, j)$ to take over the minmax regret partial path and the worst-case alternative from its predecessor, if the condition of Proposition 1 is satisfied. Therefore, we can avoid searching all feasible paths to find the minmax regret partial path from *o* to $i \in \mathbf{N}$, $i \neq t$. The path expanding rule, derived from Proposition 1, can be summarized as follows:

Rule 2: Expanding a Path. Given a minmax regret partial path $\mathbf{x}^i$, it can be easily extended to $\mathbf{x}^i_{+ij}$ with a supplemented $\text{arc}(i, j)$, if $\text{arc}(i, j)$ is not on $\mathbf{y}^i$.

If Proposition 1 is not satisfied, in that $\text{arc}(i, j)$ is on $\mathbf{y}^i$, the worst-case alternative, $\mathbf{y}^i_{+ij}$, for $\mathbf{x}^i_{+ij}$ can be recomputed by Property 1. But, it is not guaranteed that $\mathbf{x}^i_{+ij}$ is a minmax regret partial path and has the least maximum regret among all candidate paths, of which worst-case alternative is $\mathbf{y}^i_{+ij}$. The following simple rule can be used to find a better candidate path for a given alternative $\mathbf{y}^i_{+ij}$.

Rule 3: Generating a Candidate Path for a Given Alternative Path. Given candidate $\mathbf{x}^i_{+ij}$ and alternative $\mathbf{y}^i_{+ij}$, let G' be a sub graph of G, where arcs on $\mathbf{y}^i_{+ij}$ but not on $\mathbf{x}^i_{+ij}$ are uninvolved. Then, a new candidate path, $\mathbf{x}^{i-}_{+ij}$, is the shortest path from o to arc(i, j), where $c_{ij} = u_{ij}$ for all arcs on G'.

The Rule 3 can be derived from the following Theorem 1.

Lemma 1. $\mathbf{y}^i_{+ij}$ is the worst-case alternative for $\mathbf{x}^{i-}_{+ij}$ on G.

Proof. Let $\boldsymbol{s}^1$ be the scenario, where $c_{ij} = u_{ij}$ for all arcs on $\mathbf{x}^i_{+ij}$ and $c_{ij} = l_{ij}$ for the rest on G, and $\boldsymbol{s}^2$ be the scenario, where $c_{ij} = u_{ij}$ for all arcs on G' and $c_{ij} = l_{ij}$ for the rest on G. The cost of each arc under $\boldsymbol{s}^2$ is greater than or equal to that under $\boldsymbol{s}^1$, since more arcs are set to u_{ij} under $\boldsymbol{s}^2$ and the rest are the same. Thus, the shortest path under $\boldsymbol{s}^1$ is shorter than or equal to the shortest path under $\boldsymbol{s}^2$. The cost of arc on $\mathbf{y}^i_{+ij}$ is the same under $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$, since the arcs on $\mathbf{y}^i_{+ij}$ are pegged for G' or the costs are already set to u_{ij}. The shortest path on G under $\boldsymbol{s}^2$ is the worst-case alternative for each path on G or G' under $\boldsymbol{s}^2$. By definition, $\mathbf{y}^i_{+ij}$ is the shortest path on G under $\boldsymbol{s}^1$. Therefore, $\mathbf{y}^i_{+ij}$ is the shortest path on G under $\boldsymbol{s}^2$. Since $\mathbf{x}^{i-}_{+ij}$ is a path on G' under $\boldsymbol{s}^2$, $\mathbf{y}^i_{+ij}$ is the worst-case alternative for $\mathbf{x}^{i-}_{+ij}$. The result follows. □

From Lemma 1, the following Theorem 1 can be derived.

Theorem 1. The maximum regret of $\mathbf{x}^{i-}_{+ij}$ is less or equal to the maximum regret of $\mathbf{x}^i_{+ij}$.

Proof. From Lemma 1, it is obvious that $\mathbf{y}^i_{+ij}$ is the worst-case alternative for $\mathbf{x}^{i-}_{+ij}$ and $\mathbf{x}^i_{+ij}$. By definition of G', $\mathbf{x}^{i-}_{+ij} \subset G'$ and $\mathbf{x}^i_{+ij} \subset G'$. By definition of $\mathbf{x}^{i-}_{+ij}$ in Rule 4, $\mathbf{x}^{i-}_{+ij}$ is the shortest path under $\boldsymbol{s}^2$, which embraces the scenario induced by $\mathbf{x}^i_{+ij}$. By definition of the maximum regret described in equation (7), $\mathbf{c}^l\mathbf{y}$ is constant since $\mathbf{y}^i_{+ij}$ is given. $\mathbf{c}^u\mathbf{x}^{i-}_{+ij}$ is less than or equal to $\mathbf{c}^u\mathbf{x}^i_{+ij}$. This completes the proof. □

Note that, it is normally not guaranteed that there always exists a minmax regret shortest path which comprises an arbitrary arc(i, j), since some arcs, which are called *non-weak* arcs, are never in an optimal solution of the MRSPP [8]. In this paper, it is supposed that the given graph is exclusive of non-weak arcs.

3.2 Pseudo-code

A heuristic algorithm exploiting the properties been described in Section 3.1 is proposed with the pseudo-code briefed in Figure 1. The algorithm starts at the origin node by initializing the maximum regret, r_j, for each node j at line 0-1 in Figure 1. And then the null path and the corresponding worst-case alternative are determined by Rule 1 at line 2. Then, an iterative procedure is performed to find the minmax regret partial path by examining each incoming arc(i, j) for each node j at line 3-15. At each step, if arc$(i, j) \notin \mathbf{y}^i$, at line 5, Rule 2 is applied to get the worst-case alternative and the corresponding maximum regret, straightforwardly, at line 6-7. Else, Rule 3 can be applied to find a better candidate path from o to arc(i, j) at line 9-10. When all incoming arcs to node j are probed, the least maximum deviated path among them is taken as the minmax regret partial path for node j at line 12-13. This iterative procedure is repeated onto node t.

Algorithm RDSP(G = (N, A), $c_{ij} \in [l_{ij}, u_{ij}]$) for each arc $(i, j) \in A$, *source*)
0 **for each** node $j \in$ N // *Initializations*
1 r_j:=infinity // *all maximum regret are set to infinity*
2 (Apply **Rule 1**):
$\mathbf{x}^0$=NULL, $\mathbf{y}^o$:= the shortest path on G under the scenario l, r_0=the length of $\mathbf{y}^o$
3 **for each** node $j \in$ N
4 **for each** $i \in$ N such that arc$(i, j) \in$ A
5 **if** arc$(i, j) \notin \mathbf{y}^i$ **then** (apply **Rule 2**)
6 $\mathbf{x}^i_{+ij} := \mathbf{x}^i \cup$ arc(i, j)
7 $\mathbf{y}^i_{+ij} := \mathbf{y}^i$
8 **else** (apply **Rule 3**)
9 Find $\mathbf{y}^i_{+ij}$ and $\mathbf{x}^{i-}_{+ij}$
10 $\mathbf{x}^i_{+ij} := \mathbf{x}^{i-}_{+ij}$
11 $r_{ij} := \mathbf{c}^u \mathbf{x}^i_{+ij} - \mathbf{c}^l \mathbf{y}^i_{+ij}$
12 **if** $r_{ij} < r_j$ **then**
13 $r_j := r_{ij}$, $\mathbf{x}^j := \mathbf{x}^i_{+ij}$, $\mathbf{y}^j := \mathbf{y}^{ij}$
14 **end for**
15 **end for**

Figure 1. Pseudo-code of the Proposed Algorithm

4. Computational Experiments

This section presents some computational results to evaluate the performance of the proposed algorithm RDSP, described in the previous section. To demonstrate the performance and the efficiency, MILP model in [8] was adopted to find the optimal solution and the algorithm in [15] was selected to show the accuracy of the proposed algorithm.

4.1 Network Model

The algorithm has been tested on graphs suggested in [8], called Karaşan's graphs. As stated in [8], these graphs are suitable to simulate telecommunication networks. Karaşan's graphs are *directed acyclic* and *layered* graphs with a small *width*. Note that a directed acyclic graph is a directed graph with no directed cycles, and a layered graph

is a connected graph where layers partition the nodes. Each edge, which has a nonnegative integral weight, connects only nodes in successive layers. The width is the greatest number of vertices in any layer. Particularly, in Karaşan's graphs, the width of each layer is the same except for the first and the last layer, which are the origin and the terminal, respectively, and each node is connected to every node in the successive layer.

As stated in [17], each interval cost $[l_{ij}, u_{ij}]$ is obtained by generating a random number $c_{ij} \in [1, c]$ and by randomly selecting l_{ij} in $[(1-d)c_{ij}, (1+d)c_{ij}]$ and u_{ij} in $[l_{ij}, (1+d)c_{ij}]$, where $0 < d < 1$. Here, parameter c and d are used to control the degree of uncertainty. The bigger c spreads the center of interval for each arc, the bigger d widens the interval.

4.2 Experimental Result

The algorithm is coded in C^{++} using the Standard Template Library. All the tests were done on a personal computer with Intel Core2Duo @ 2.33GHz CPU / 1GB main memory. To obtain the optimal solution based on Karaşan's MILP model, Cbc (Coin-or branch and cut, open-source MILP solver [18]) ver. 1.1.2 was applied. Although it is said that Cbc is slower than commercial MILP solvers, *i.e.*, 5 times slower than ILOG CPLEX 12.0 [19], the result of Cbc is only used for calculating the relative errors of the proposed algorithm from the optimal solution, not for comparing the computational time.

Tests were done with combinations of 4 parameters, *i.e.*, the number of layers$\in$\{50, 100\}, width$\in$\{2,4\}, $c\in$\{10, 20\}, and $d\in$\{0.3, 0.9\}. For each combination of parameters, the test was repeated 100 times with randomly generated instances.

The computational results are summarized in Table 1. In the first part of the table shows the computational time of Cbc and the proposed algorithm. Note that 'mean' indicates the arithmetic mean, STD indicates the standard deviation. The computational time of the heuristic algorithm in [15] is almost 0 and so not presented. Although the computational time for the exact solution increases about 50 times as the width of networks increases from 2 to 4, the computational time of the proposed algorithm increase only twice as shown in the second column of Table 1. The second part shows the relative errors between the algorithm results and the exact solution. On average, the relative errors of the proposed algorithm are much less than that of the fast heuristic algorithm in [15]. Especially, it shows that the performance of the algorithm is robust against any combination of the parameters, since the relative errors are stable around 1%. And the third part 'Correct Ratio (%)' shows the number of problems, that the algorithms optimally solved, out of 100 instances. It shows that the proposed algorithm can find optimal solutions with relatively high percentages. The computational results show that the proposed algorithm outperforms other heuristic algorithms developed previously, to author's knowledge, with the most accurate solutions.

Table 1. Summary of computational results

layers	width	c	d	Computation Time (sec.) MIP mean	MIP STD	Proposed algorithm mean	Proposed algorithm STD	Relative Error (%) Heuristic in [15] mean	Heuristic in [15] STD	Proposed algorithm mean	Proposed algorithm STD	Correct Ratio (%) Heuristic in [15]	Proposed algorithm
50	2	10	0.3	0.11	0.156	0.03	0.021	20.8	0.179	0.88	0.020	8	71
			0.9	0.44	0.496	0.03	0.018	14.3	0.115	1.06	0.016	4	46
		20	0.3	0.09	0.048	0.03	0.033	19.7	0.179	1.02	0.029	14	77
			0.9	0.40	0.488	0.03	0.024	15.0	0.105	1.12	0.022	3	59
	4	10	0.3	3.82	2.511	0.06	0.026	10.8	0.066	1.17	0.021	3	54
			0.9	8.68	6.899	0.06	0.018	7.7	0.042	0.84	0.013	2	40
		20	0.3	3.08	1.797	0.06	0.023	10.6	0.088	0.98	0.019	5	56
			0.9	6.79	3.878	0.06	0.017	9.0	0.053	0.99	0.015	1	49
100	2	10	0.3	0.30	0.173	0.24	0.018	17.7	0.108	0.81	0.017	0	61
			0.9	1.86	2.137	0.18	0.012	16.1	0.079	0.99	0.011	0	28
		20	0.3	0.37	0.278	0.24	0.022	22.2	0.142	1.06	0.019	0	52
			0.9	1.33	0.802	0.18	0.014	17.3	0.073	1.03	0.011	0	29
	4	10	0.3	17.21	9.978	0.38	0.014	10.1	0.049	1.08	0.013	1	24
			0.9	58.67	51.408	0.39	0.010	9.2	0.043	1.04	0.009	0	10
		20	0.3	14.93	10.401	0.39	0.016	9.8	0.070	1.18	0.014	1	28
			0.9	58.13	64.654	0.40	0.010	7.3	0.040	0.79	0.008	1	21

5. Conclusions

This paper considered a generalization of the shortest path problem on DAGs, in which the lengths of arcs are given in the form of intervals. To treat the ambiguity of the arc lengths, we applied the minmax regret criterion and transformed the shortest path problem with interval data into the minmax regret shortest path problem. The solution algorithm includes 3 major rules based on properties proved in this paper, which ease the complexity to find the minmax regret partial path and the corresponding worst-case alternative. The computational experiments were carried out, and the results show that the proposed algorithm provides optimal solutions with relatively high percentage or near optimal solutions with about 1% of relative errors from optimal solutions, within a reasonable amount of computation time.

Acknowledgements

This research was supported by the research fund of Sungkyul University under grant No. 2011-1142-1.

References

[1] S. Zhang, J. Chen, H. Zhong, Z. Fang and J. Shi, "Trust Network and Trust Community Clustering based on Shortest Path Analysis for E-commerce", International Journal of u- and e-Service, Science and Technology, vol. 5, no. 2, **(2012)**, pp. 31-420, http://www.sersc.org/journals/IJUNESST/vol5_no2/3.pdf.

[2] H. Kim, "Ant-Q agent System Based Path Optimization Service for a Multi-Objective Mobile Robot and Real World", International Journal of u- and e- Service, Science and Technology, vol. 5, no. 2, **(2012)**, pp. 77-86, http://www.sersc.org/journals/IJUNESST/vol5_no2/6.pdf.

[3] C. Lin, J. Yu, J. Liu, W. Lai and C. Ho, "Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems", International Journal of Hybrid Information Technology, vol. 2, no. 1, **(2009)**, pp. 21-30, http://www.sersc.org/journals/IJHIT/vol2_no1_2009/3.pdf.

[4] K. Ramakrishnan and G. Ray, "Robust H∞ Controller Synthesis for Linear Uncertain Systems with Interval Time-delay: A Less Conservative Result", International Journal of Control and Automation, vol. 4, no. 4, **(2011)**, pp. 39-54, http://www.sersc.org/journals/IJCA/vol4_no4/3.pdf.
[5] D. Rasteiro and A. Anjo, "Optimal paths in probabilistic networks", Journal of Mathematical Sciences, vol. 120, no. 1, **(2004)**, pp. 974-987, http://dx.doi.org/10.1023/B:JOTH.0000013560.69722.c1.
[6] A. Kasperski, "Discrete Optimization with Interval Data - Minmax Regret and Fuzzy Approach", Springer, **(2008)**, http://www.springer.com/engineering/computational+intelligence+and+complexity/book/978-3-540-78483-8.
[7] P. Kouvelis and G. Yu, "Robust Discrete Optimization and Its Application", Kluwer Academic Publishers, Boston, **(1997)**, http://dx.doi.org/10.1007/978-1-4757-2620-6.
[8] O. E. Karaşan, M. Ç. Pinar and H. Yaman, "The Robust Shortest Path Problem with Interval Data", **(2001)**, Unpublished Material, http://www.ie.bilkent.edu.tr/~mustafap/pubs/.
[9] I. Averbakh and V. Lebedev, "Interval data minmax regret network optimization problems", Discrete Applied Mathematics, vol. 138, no. 3, **(2004)**, pp. 289-301, http://dx.doi.org/10.1016/S0166-218X(03)00462-1.
[10] P. Zieliński, "The computational complexity of the relative robust shortest path problem with interval data", European Journal of Operational Research, vol. 158, no. 3, **(2004)**, pp. 570-576, http://dx.doi.org/10.1016/S0377-2217(03)00373-4.
[11] J. Kang, D. Lee and P. Xirouchakis, "Disassembly Sequencing with Imprecise Data: a Case Study", Int'l Journal of Industrial Engineering - Theory, Applications and Practice, vol. 10, no. 4, **(2003)**, pp. 407-412.
[12] R. Montemanni and L. M. Gambardella, "An Exact Algorithm for the Robust Shortest Path Problem with Interval Data", Computers and Operations Research, vol. 31, no. 10, **(2004)**, pp. 1667-1680, http://dx.doi.org/10.1016/S0305-0548(03)00114-X.
[13] S. Chanas and P. Zieliński, "The computational complexity of the criticality problems in a network with interval activity times", European Journal of Operational Research, vol. 136, no. 3, **(2002)**, pp. 541-550, http://dx.doi.org/10.1016/S0377-2217(01)00048-0.
[14] J. Kang, "Robust Disassembly Sequencing with Interval Data", Proc. of the Global Conf. on Sustainable Product Development and Life Cycle Eng'g, Busan, Korea, **(2008)** September 20 - October 1, pp. 109-113.
[15] E. Conde, "A minmax regret approach to the critical path method with task interval times", European Journal of Operational Research, vol. 197, no. 1, **(2009)**, pp. 235-242, http://dx.doi.org/10.1016/j.ejor.2008.06.022.
[16] M. Inuiguchi and M. Sakawa, "Minimax Regret Solution To Linear-Programming Problems With An Interval Objective Function", European Journal of Operational Research, vol. 86, no. 3, **(1995)**, pp. 526-536, http://dx.doi.org/10.1016/0377-2217(94)00092-Q.
[17] R. Montemanni, L. M. Gambardella and A. V. Donati, "A branch and bound algorithm for the robust shortest path problem with interval data", Operations Research Letters, vol. 32, no. 3, **(2004)**, pp. 225-232, http://dx.doi.org/10.1016/j.ejor.2003.10.008.
[18] Cbc Home Page, https://projects.coin-or.org/Cbc.
[19] Mixed Integer Linear Programming Benchmark, http://plato.asu.edu/ftp/milpc.html.

Authors

Jun-Gyu KANG

He is currently an Assistant Professor in Department of Industrial and Management Engineering at Sungkyul University, Republic of Korea. He received a B.S. degree in Industrial Engineering from Busan National University in 1998 and the M.S. in Industrial Engineering from Korea Advanced Institute of Science and Technology in 2000. He did his PhD in Mechanical Engineering at the Swiss Federal Institute of Technology – Lausanne (École Polytechnique Fédéral de Lausanne) in 2005. From 2006 to 2007, he worked at Grenoble Institute of Technology (Institut Polytechnique de Grenoble, France) as a research fellow. His areas of research include Robust Optimization, Product Lifecycle Management, Product Design and Development for End-of-Life, and System Modeling and Simulation. E-mail: jun-gyu.kang@sungkyul.ac.kr.

