MARKO ŠPOLJAREC, Ph.D.[1]
E-mail: marko.spoljarec@internationalvalueservices.com
ROBERT MANGER, Ph.D.[2]
(Corresponding author)
E-mail: manger@math.hr
[1] Intesa Sanpaolo International Value Services
Radnička cesta 44, 10000 Zagreb, Croatia
[2] Department of Mathematics
Faculty of Science, University of Zagreb
Bijenička cesta 30, 10000 Zagreb, Croatia

Science in Traffic and Transport
Original Scientific Paper
Submitted: 8 Mar. 2020
Accepted: 25 Sep. 2020

SOLVING ROBUST VARIANTS OF INTEGER FLOW PROBLEMS WITH UNCERTAIN ARC CAPACITIES

ABSTRACT

This paper deals with robust optimization and network flows. Several robust variants of integer flow problems are considered. They assume uncertainty of network arc capacities as well as of arc unit costs (where applicable). Uncertainty is expressed by discrete scenarios. Since the considered variants of the maximum flow problem are easy to solve, the paper is mostly concerned with NP-hard variants of the minimum-cost flow problem, thus proposing an approximate algorithm for their solution. The accuracy of the proposed algorithm is verified by experiments.

KEYWORDS

network flow; integer flow; robust optimization; algorithm.

1. INTRODUCTION

Network flows [1-3] are an important modelling paradigm used in optimization. Models based on networks and flows are encountered in various areas of operation research, e.g. resource assignment, transportation, traffic regulation, and others. Two most common types of network flow problems are the *maximum flow problem* and the *minimum-cost flow problem.*

An instance of a network flow problem is specified by exact values of its parameters, such as arc capacities or arc unit costs. However, in real-life situations, those values are often hard to specify since they may depend on some unforeseen circumstances or may be too volatile for accurate measurement. Then, we experience *uncertainty* in the problem formulation. Ignoring uncertainty is not recommended since it can easily lead to low-quality or even infeasible solutions.

A state-of-the-art method for dealing with the mentioned uncertainty is called *robust optimization*. The method has been established by a series of papers and books [4-7]. More recent surveys and some general results can be found in [8-11]. Among the above references, the most important for our purposes is book [7] which provides a framework for robust *discrete* optimization.

According to the approach from [7], uncertainty in problem parameters should be captured by a discrete (finite and explicitly given) set of *scenarios.* Each scenario specifies a consistent combination of parameter values. The only solutions that are taken into account are those that are feasible for all scenarios. The behaviour of any considered solution under any scenario is evaluated according to some criteria. Then the so-called *robustly optimal* solution is chosen as the one whose worst behaviour, measured over all scenarios, is the best possible one.

The framework from [7] obviously allows many options. Indeed, the behaviour of a solution under a scenario can be measured by different criteria. Also, the set of problem parameters that are considered as uncertain can be more or less extensive. Thus, for the same conventional (non-robust) optimization problem one can construct several robust problem variants, which may be more or less difficult to solve.

It is well known that the conventional flow problems can be solved in polynomial time. This is true even if one insists on only integer solutions. On the other hand, robust variants of the same problems can easily become NP-hard. For that reason, designing efficient algorithms for robust variants turns out to be an important area of research.

There is a fair number of papers on robust network flows found in literature, e.g. [5, 12-21]. However, such works are hard to compare since they use different definitions and concepts. Most of them do not fit into the framework from [7] since they assume infinite, but rather regular sets of scenarios defined, e.g. by intervals for parameter values. The authors of the available papers have been mainly concerned with complexity issues or with custom-designed solutions for special problem cases. There are not too many publications which could be regarded as useful to decision-makers in real-life situations.

An interesting contribution to the considered topic has been given by the recently published paper [22]. It differs from the above cited works by trying to be more practical. Indeed, it studies relatively general and practically relevant robust problem variants and solves them by applying general-purpose algorithmic paradigms. More precisely, in [22] the authors have considered two robust variants of the standard minimum-cost integer flow problem. The approach from [7] has been used. However, uncertainty in problem formulation has been limited only to arc unit costs. Still, the authors have shown that even such restricted problem variants are already NP-hard. Also, they have demonstrated that the considered variants, although NP-hard, can be solved with reasonable accuracy by heuristics based on local search or evolutionary computing [23, 24]. Those heuristics include several interesting and original operators for flow initialization, neighbourhood generation, crossover or mutation.

The aim of this paper is to build upon the ideas and results from [22] in order to produce even more applicable solutions to robust integer flow problems. On a more abstract level, the aim is to give additional contribution to the scarce literature on useful robust optimization algorithms. In fact, this paper can be regarded as an extension and improvement of [22]. Extensions go in three directions. First, two types of network flow problems are considered instead of one, i.e. the maximum integer flow problem in addition to the minimum-cost integer flow problem. Next, uncertainty in problem formulation now includes arc capacities along with arc unit costs. Finally, as an alternative to the heuristics, a more accurate approximate algorithm based on relaxation and rounding is now introduced and tested on large problem instances.

The rest of this paper is organized as follows. Section 2 specifies three robust variants of the maximum integer flow problem (with uncertain arc capacities) and shows that all of them can be reduced to the conventional (non-robust) variant. Consequently, the considered robust maximum flow problem variants are skipped from the subsequent sections since they can be solved by well-known polynomial-time algorithms. Section 3 specifies three robust variants of the minimum-cost integer flow problem (with uncertain arc capacities and arc unit costs). It is shown that at least two of those variants are NP-hard. Section 4 presents our new approximate algorithm for solving robust variants of the minimum-cost integer flow problem, which is based on relaxation and rounding. The same section also explains how the new algorithm can be implemented by combining general-purpose routines and more specific flow procedures. The next Section 5 presents robust minimum-cost integer flow problem instances that are used in experiments. The instances are too large to be solved exactly, but still small enough to be solved in relaxed form. The same section also describes experiments where the chosen problem instances have been solved repeatedly by the algorithm from Section 4. Experimental results are presented, which allow estimation of algorithm accuracy depending on the robust variant involved. The last Section 6 gives conclusions.

2. VARIANTS OF THE MAXIMUM FLOW PROBLEM

First, the *conventional* (non-robust) variant of the maximum integer flow problem is described. Let $G=(V, A)$ be *a network* (directed graph), where $V=\{v_1,v_2,\ldots,v_n\}$ is a set of n elements called *vertices* and $A\subset V\times V$ is a set of ordered pairs of vertices called *arcs*. Each arc (v_i, v_j) is assigned its *capacity* u_{ij}. We consider *feasible flows* that transfer an amount of flow F from *source* v_1 to *sink* v_n. A feasible flow is a non-negative function defined on arcs, which obeys the *flow conservation rule* in each vertex, as well as the *capacity constraint* along each arc. We denote the *arc flow value* assigned to an arc (v_i, v_j) by x_{ij}. The objective is to construct a feasible flow with *maximal* value F.

In this work it is assumed that all capacities u_{ij} are non-negative integers. Moreover, we restrict to flows that consist of integer arc values x_{ij}, thus producing integral values F. In this way we indeed

consider the maximum *integer* flow problem. The restriction to integers is appropriate in many applications where discrete quantities are handled.

The above maximum integer flow problem can be formally defined as the following *integer linear programming* (ILP) problem [3]:

MIF...

$$\text{maximize } z = F$$

subject to:

$$\sum_{\substack{v_j \in V \\ (v_i,v_j)\in A}} x_{ij} - \sum_{\substack{v_j \in V \\ (v_j,v_i)\in A}} x_{ji} = \begin{cases} F \text{ if } v_i = v_1 \\ -F \text{ if } v_i = v_n \\ 0 \text{ if } v_i \notin \{v_1, v_n\} \end{cases}, \text{ for all } v_i \in V \quad (1)$$

$$0 \le x_{ij} \le u_{ij}, \text{ for all } (v_i, v_j) \in A$$

$$x_{ij} \text{ integer, for all } (v_i, v_j) \in A$$

The described maximum integer flow problem has obvious applications in the area of traffic and transportation. For instance, the whole network can be interpreted as a transportation system, where vertices represent locations (junctions) and arcs represent roads. Certain commodity must be continuously shipped from one location (source) to another (sink). Arc capacities determine how many units of commodity can travel through the corresponding roads in parallel. A network flow specifies one possible way of shipping, i.e. it defines how many commodity units should be sent through particular roads. The restriction to integer flows makes sense when our commodity consists of indivisible (packaged) units that cannot be split into parts. The maximum flow determines the most-intensive way of shipping the commodity from the source to the sink.

Next, we will specify three *robust* variants of the maximum integer flow problem. According to our adopted framework from [7], uncertainty in input data should be expressed through a discrete set of *scenarios S*. A particular scenario $s \in S$ comprises a specific set of arc capacities u_{ij}^s. It is assumed that the network structure is the same for all scenarios. Again, according to [7], we restrict to flows that are feasible simultaneously for all scenarios, i.e. to flows that satisfy

$$0 \le x_{ij} \le u_{ij}^s, \text{ for all } (v_i, v_j) \in A \text{ and all } s \in S \quad (2)$$

or equivalently

$$0 \le x_{ij} \le \min_{s \in S} u_{ij}^s, \text{ for all } (v_i, v_j) \in A. \quad (3)$$

The first robust variant is called *absolute robust* [7] or *max-min* [8] variant. There, the behaviour of a feasible flow under a scenario is measured as the actual flow value *F*. For each feasible flow, its worst behaviour (i.e. minimal flow value) over all scenarios is recorded. As the robust solution, the flow is chosen whose worst behaviour is the best (i.e. maximal) among all feasible flows. More precisely, the absolute robust variant is defined as the following optimization problem:

RMIF-A...

$$\text{maximize } z = \min_{s \in S} \{F\}$$

subject to:

$$\sum_{\substack{v_j \in V \\ (v_i,v_j)\in A}} x_{ij} - \sum_{\substack{v_j \in V \\ (v_j,v_i)\in A}} x_{ji} = \begin{cases} F \text{ if } v_i = v_1 \\ -F \text{ if } v_i = v_n \\ 0 \text{ if } v_i \notin \{v_1, v_n\} \end{cases}, \text{ for all } v_i \in V \quad (4)$$

$$0 \le x_{ij} \le u_{ij}^s, \text{ for all } (v_i, v_j) \in A \text{ and all } s \in S$$

$$x_{ij} \text{ integer, for all } (v_i, v_j) \in A$$

The above robust objective function follows the general pattern from [7]. However, in our case the conventional objective function happens to achieve the same value *F* under any scenario $s \in S$. Thus, the above operator $\min_{s \in S}$ is in fact obsolete.

The second robust variant is called *robust deviation* [7] or *min-max regret* [8] variant. There, the behaviour of a feasible flow under scenario *s* is measured as deviation of the actual flow value *F* from the optimal flow value z^s for that scenario. Again, the flow is chosen whose worst behaviour over all scenarios is the best among all feasible flows. Thus, the robust deviation variant is formally defined as the following optimization problem:

RMIF-D...

$$\text{minimize } z = \max_{s \in S} \{z^s - F\}$$

$$\text{subject to the same constraints as for RMIF - A} \quad (5)$$

The operator $\max_{s \in S}$ in the above robust objective function is not obsolete. Namely, value z^s can be different for each $s \in S$.

The third robust variant is called *relative robust deviation* [7] or *min-max relative regret* [8] variant. There, the behaviour of a feasible flow under scenario *s* is measured as relative deviation of the actual flow value *F* from the optimal flow value z^s for that scenario. Once more, the flow is chosen whose worst behaviour over the whole set of scenarios is the best among all feasible flows. Or more formally, the relative robust deviation variant is defined in the following way:

RMIF-R...

$$\text{minimize } z = \max_{s \in S} \left\{ \frac{z^s - F}{z^s} \right\}$$

$$\text{subject to the same constraints as for RMIF - A} \quad (6)$$

It is assumed here that z^s is non-zero for each $s \in S$. Again, the operator $\max_{s \in S}$ is not obsolete since z^s can differ from one scenario to another.

Let us now discuss how the assumed uncertainty in arc capacities can be interpreted within the previously described shipping application. Uncertain capacity of an arc means that the throughput of the corresponding road can vary according to circumstances. For instance, the throughput can be reduced in case of snow or flooding. So, one scenario could model normal weather circumstances, another scenario a snowy winter, and yet another a rainy season, etc. An additional reason for uncertainty in arc capacities can be utilization of the same roads by some other traffic that is not a part of our shipping. Then one scenario could model empty roads, another scenario heavily congested roads, yet another scenario a situation where some roads are partially loaded with other traffic and some are empty, etc.

The above three robust variants RMIF-A, RMIF-D and RMIF-R have been constructed according to the general guidelines from [7], which are applicable to any type of conventional optimization problem. However, it is easy to see that in our case (the maximum flow problem) all three variants are in fact trivial, i.e. they are reduced to the conventional variant. To see this, let us consider an auxiliary (conventional) maximum flow problem defined as:

MIF*...

$$\begin{aligned} &\text{maximize } z = F \\ &\text{subject to the constraints from } \textit{Formula 1} \text{ where the } 2^{\text{nd}} \\ &\text{constraint is replaced with:} \\ &0 \le x_{ij} \le u^*_{ij}, \text{ for all } (v_i, v_j) \in A \end{aligned} \tag{7}$$

Here the notation $u^*_{ij} := \min_{s \in S} u^s_{ij}$ is used. Then the set of feasible flows for MIF* obviously coincides with the set of feasible flows considered in RMIF-A, RMIF-D and RMIF-R, respectively. Moreover, the following can be observed.

- RMIF-A is (due to obsoleteness of $\min_{s \in S}$) *identical* to MIF*, thus their optimal flows must be the same, and their objective functions achieve the same value.
- For the objective function of RMIF-D the following holds:

$$\max_{s \in S} \{ z^s - F \} = \max_{s \in S} z^s - F \tag{8}$$

  The above expression will be minimal if F is maximal. Thus, within RMIF-D the optimal objective function value is

$$z = \max_{s \in S} z^s - z_* \tag{9}$$

  and the optimal flow for RMIF-D must be the same as for MIF*. Here z_* denotes the optimal objective function value for MIF*.
- For the objective function of RMIF-R the following holds:

$$\max_{s \in S} \left\{ \frac{z^s - F}{z^s} \right\} = 1 - \min_{s \in S} \frac{F}{z^s} = 1 - \frac{F}{\max_{s \in S} z^s} \tag{10}$$

  This expression will be minimal if F is maximal. Thus, within RMIF-R the optimal objective function value is

$$z = 1 - \frac{z_*}{\max_{s \in S} z^s} \tag{11}$$

  and the optimal flow is again the same as for MIF*.

Putting it all together, it turns out that an instance of RMIF-A or RMIF-D or RMIF-R can be solved by solving the corresponding instance of MIF*. In order to compute the exact value of the robust objective function in RMIF-D or RMIF-R it is also necessary to find the values z^s for all $s \in S$, which means solving additional $|S|$ instances of MIF. Thus, on the one hand, one robust instance is always reduced to one or more conventional instances. On the other hand, it is well known [2] that the conventional instances can be solved by polynomial algorithms, even if only integer flows are sought. Consequently, there is no need to design new algorithms for solving robust maximum integer flow problem variants, and they will be skipped from the rest of this paper.

3. VARIANTS OF THE MINIMUM-COST FLOW PROBLEM

Analogously as in the previous section, this section starts with the conventional (non-robust) variant of the minimum-cost integer flow problem. As before, a network $G=(V,A)$ is considered, where $V=\{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $A \subset V \times V$ is a set of arcs. However, each arc (v_i, v_j) is now characterized not only by its capacity u_{ij} but also by its *unit cost* (cost per unit of flow) c_{ij}. Feasible flows are sought that transfer an amount of flow F from source v_1 to sink v_n. A feasible flow is again a non-negative function defined on arcs obeying the flow conservation rule in vertices and the capacity constraints along arcs. The flow value assigned to an arc (v_i, v_j) is denoted by x_{ij} as before. But now additionally the flow cost is computed as the sum of products $c_{ij} x_{ij}$ over all arcs (v_i, v_j). The flow value

F is not maximized; instead, it is given in advance. The objective now is to find a feasible flow with the required value F having *minimal cost*.

Analogously as in the previous section, it is assumed that all input data u_{ij}, c_{ij}, F are non-negative integers, and that the flow itself consists of integer arc values x_{ij}. Thus, the minimum-cost *integer* flow problem is considered according to the following ILP definition:

MCIF...

$$\text{maximize } z = \sum_{(v_i,v_j)\in A} c_{ij}x_{ij}$$

subject to:

$$\sum_{\substack{v_j\in V\\(v_i,v_j)\in A}} x_{ij} - \sum_{\substack{v_j\in V\\(v_j,v_i)\in A}} x_{ji} = \begin{cases} F \text{ if } v_i = v_1 \\ -F \text{ if } v_i = v_n \\ 0 \text{ if } v_i \notin \{v_1, v_n\} \end{cases}, \text{for all } v_i \in V \quad (12)$$

$$0 \le x_{ij} \le u_{ij}, \text{ for all } (v_i,v_j) \in A$$

$$x_{ij} \text{ integer, for all } (v_i,v_j) \in A$$

Similarly as the maximum integer flow problem, the described minimum-cost integer flow problem can also be applied in the area of traffic and transportation. For instance, we can again interpret the network and its flows as a model for shipping, as in Section 2. Then the unit cost of an arc has an obvious meaning: it is the cost of transporting one unit of commodity along the corresponding road, measured in terms of fuel, work effort, etc. The objective is now to ship the commodity from the source to the sink, so that a desired intensity of shipping is achieved, but at minimum cost.

In real-world applications, it is very common that the considered commodity is produced at many locations and also consumed at many locations. Then the most appropriate model is a network with multiple sources and sinks, as specified in [1, 2, 25]. However, it is easy to see that such a network can easily be reduced to our single-source-single-sink form. Reduction is done by introducing an artificial source, an artificial sink, and some extra arcs with appropriate capacities. There are also some real-world applications that require solving the minimum-cost maximum flow problem [26]. But such seemingly more complicated task can be modelled as a combination of our two problems considered in the previous and in the present section, respectively.

In the next three paragraphs three robust variants of the minimum-cost integer flow problem are introduced. Similarly to the previous section and according to [7], uncertainty in input data is expressed through a discrete set of scenarios S. But now, each scenario $s\in S$ specifies also the arc unit costs c_{ij}^s together with arc capacities u_{ij}^s. The network structure and the required flow value F are assumed to be the same for all scenarios. Again, according to [7], the restriction is done to flows with value F that are feasible for all scenarios simultaneously, i.e. to flows that among other things satisfy

$$0 \le x_{ij} \le \min_{s\in S} u_{ij}^s, \text{ for all } (v_i,v_j) \in A \quad (13)$$

As in the previous section, the first robust variant is the *absolute robust* [7] or *min-max* [8] variant, where the behaviour of a feasible flow under a scenario is measured as the actual flow cost. As the robust solution, the flow is chosen whose worst (i.e. maximal) cost, measured over all scenarios, is the best possible (i.e. minimal). Here follows the strict definition:

GRMCIF-A...

$$\text{minimize } z = \max_{s\in S}\left\{\sum_{(v_i,v_j)\in A} c_{ij}^s x_{ij}\right\}$$

subject to:

$$\sum_{\substack{v_j\in V\\(v_i,v_j)\in A}} x_{ij} - \sum_{\substack{v_j\in V\\(v_j,v_i)\in A}} x_{ji} = \begin{cases} F \text{ if } v_i = v_1 \\ -F \text{ if } v_i = v_n \\ 0 \text{ if } v_i \notin \{v_1, v_n\} \end{cases}, \text{for all } v_i \in V \quad (14)$$

$$0 \le x_{ij} \le \min_{s\in S} u_{ij}^s, \text{ for all } (v_i,v_j) \in A$$

$$x_{ij} \text{ integer, for all } (v_i,v_j) \in A$$

The second robust variant is the *robust deviation* [7] or *min-max regret* [8] variant. The behaviour of a feasible flow under scenario s is assessed as deviation of its actual cost from the optimal cost z^s for that scenario. As a robust solution, the flow is chosen whose worst (i.e. largest) deviation, measured over all scenarios, is the best possible (i.e. the smallest). Or more precisely:

GRMCIF-D...

$$\text{minimize } z = \max_{s\in S}\left\{\sum_{(v_i,v_j)\in A} c_{ij}^s x_{ij} - z^s\right\} \quad (15)$$

subject to the same constraints as for GRMCIF - A

The third robust variant is the *relative robust deviation* [7] or *min-max relative regret* [8] variant. The behaviour of a feasible flow under scenario s is measured as relative deviation of its actual cost from the optimal cost z^s for that scenario. It is assumed again that z^s is non-zero for each $s\in S$. As the robust solution, the flow is chosen whose worst (i.e. largest) relative deviation, measured over all scenarios, is the best possible (i.e. the smallest). The precise definition looks as follows:

GRMCIF-R...

$$\text{minimize } z = \max_{s\in S}\left\{\frac{\sum_{(v_i,v_j)\in A} c_{ij}^s x_{ij} - z^s}{z^s}\right\} \quad (16)$$

subject to the same constraints as for GRMCIF - A

Let us now see how the above assumed uncertainty of input data can be interpreted within the previously described shipping application. It has already been explained why arc capacities can be uncertain. But similar explanations are also valid for arc unit costs, i.e. they can also vary according to circumstances. For instance, costs may rise during the winter since driving through snow will require more fuel. Also, the price of fuel will rise in case of an economic crisis. Thus, a scenario could correspond to a particular weather condition, or a particular economic situation, or a combination of both.

In the previously mentioned paper [22] the authors have studied two similar but restricted robust variants of the minimum-cost integer flow problem, denoted as RMCIF-A and RMCIF-D, where scenarios can influence only the arc unit costs but not arc capacities. The above definitions GRMCIF-A, GRMCIF-D and GRMCIF-R are apparently more general, and for that reason their acronyms start with "G". An important question is whether the G-variants from this paper are really different and more general than their counterparts from [22].

It is easy to see that variant GRMCIF-A is in fact equivalent to the corresponding variant RMCIF-A from [22]. Indeed, for each arc (v_i,v_j) we can compute its minimal capacity $u_{ij}^* = \min_{s \in S} u_{ij}^s$. In each scenario we can replace any capacity with the corresponding minimal capacity. Then we will obviously obtain the same solution as with the original capacities. Thus, an instance of GRMCIF-A can be reduced to a simpler instance of RMCIF-A, where only arc unit costs change through scenarios while capacities are fixed.

On the other hand, it is also easy to notice that variant GRMCIF-D is not equivalent to the corresponding variant RMCIF-D from [22]. Namely, although all feasible flows considered in GRMCIF-D conform to the minimal arc capacities u_{ij}^*, the optimal cost z^s for a particular scenario s is still computed with the original capacities u_{ij}^s. Since u_{ij}^s can be larger than u_{ij}^*, cost z^s computed with u_{ij}^s can be better (smaller) than if it were computed with u_{ij}^*. Thus, the objective function in GRMCIF-D is different than in RMCIF-D, and the corresponding robust solution can also be different. In our experiments we have encountered concrete instances where such difference in solutions really occurs.

Next, it can also be observed that variant GRMCIF-R from this paper is not equivalent to a corresponding variant where arc capacities are fixed through scenarios. The reasons are the same as for GRMCIF-D. The difference in solutions is again easily spotted in concrete instances.

Now follows a simple example that illustrates the introduced problem variants. We consider the network with fourteen vertices that is shown twice in *Figure 1*. The source is vertex v_1 and the sink is vertex v_{14}. The desired flow value F is equal to 2. There are two scenarios for arc unit costs and capacities, specified by the left and right part of *Figure 1*, respectively. Within each part, unit costs are shown as left-hand arc labels, and capacities as right-hand labels. We would like to solve the conventional variant MCIF for each scenario. Also, we would like to solve all three robust variants, i.e. GRMCIF-A, GRMCIF-D and GRMCIF-R, respectively.

As can be seen from *Figure 1*, our network consists of four separate paths connecting the source to the sink. In the case of Scenario 1 each of those paths has capacity 1. So, under Scenario 1 any feasible flow with value F=2 must be a combination of two so-called unit flows, each of them sending one unit of flow through a different path. In the case of Scenario 2 any path has capacity 2. Thus Scenario 2 allows four additional feasible flows, each sending two units of flow through only one path. In robust problem variants we are restricted to solutions that are simultaneously feasible for both scenarios, thus again to combinations of two unit flows going through distinct paths.

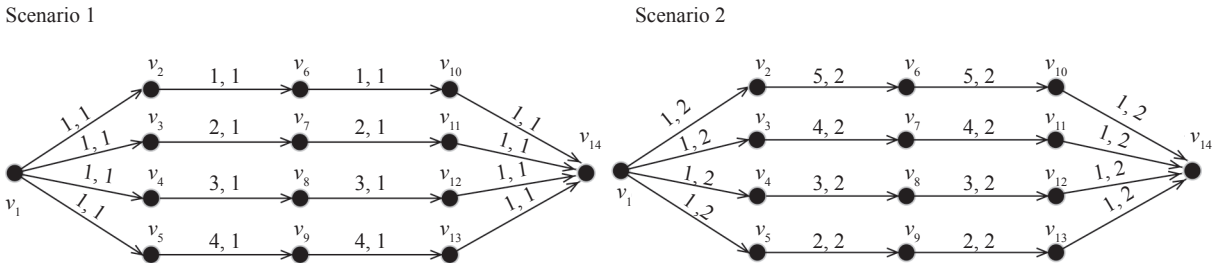

Figure 1 – A sample problem instance with two scenarios

Let us denote the uppermost path in our network with 1, the next uppermost path with 2, …, the lowermost with 4. By considering all possibilities, we can easily check that the optimal solution under Scenario 1 is the combination of unit flows going through paths 1 and 2 – the respective cost is 10. Similarly, the optimal solution under Scenario 2 is the flow sending two units of flow through path 4 with cost 12. On the other hand, the combination of unit flows through paths 2 and 4 gives the optimal solution according to the absolute robustness criterion (GRMCIF-A) and its cost is 16. Next, the optimum in the sense of robust deviation (GRMCIF-D) can be achieved in three ways as a combination of two unit flows: either through paths 1 and 4, or through paths 2 and 3, or through paths 2 and 4 – the optimal deviation of cost is 6. Finally, the optimum in the sense of relative robust deviation (GRMCIF-R) is obtained by combining the unit flows either through paths 1 and 4, or through paths 2 and 3 – the optimal relative deviation of cost is 1/2.

Our example clearly demonstrates that a robust solution can differ from any conventional solution corresponding to a particular scenario. Also, it is shown that the considered three criteria of robustness can produce different results. Finally, the comparison with a similar example from [22] confirms that variant GRMCIF-D from this paper is indeed not equivalent to the restricted variant RMCIF-D from [22].

Let us finally give some remarks about the computational complexity of the considered robust variants of the minimum-cost integer flow problem. In [22] it has been proven that the restricted RMCIF-A and RMCIF-D are both NP-hard [3]. Since GRMCIF-A and GRMCIF-D are either equivalent or more general than their counterparts from [22], they must surely be NP-hard as well. An open question is whether GRMCIF-R is also NP-hard. Our impression is that GRMCIF-R cannot be any easier to solve than GRMCIF-D. However, at this moment we cannot provide a rigorous proof for such a claim.

4. AN APPROXIMATE ALGORITHM

In the previously mentioned paper [22] the authors have developed 13 heuristics for solving robust variants of the minimum-cost integer flow problem, which are based either on local search or on evolutionary computing. Those heuristics have originally been designed for the restricted problem variants RMCIF-A and RMCIF-D, but with minor modifications they can also be applied to GRMCIF-A, GRMCIF-D and GRMCIF-R considered in this paper. Rather than recycling the old solutions, we now describe a different approximate algorithm, which is based on relaxation and rounding. It will turn out that our new algorithm is more advantageous than the heuristics from [22] since it assures better accuracy.

First, let us note that GRMCIF-A, GRMCIF-D and GRMCIF-R are in fact ILP problems, in spite of their seemingly more complex min-max objective functions. Namely, all three problems can easily be transformed into an equivalent form which is obviously ILP. Transformation is done by introducing an additional variable, as it has been shown in [7]. Here follow the corresponding "linearized" problem versions denoted with GRMCIF-A', GRMCIF-D' and GRMCIF-R', respectively.

GRMCIF-A'...

$$\begin{aligned}
&\text{minimize } y \\
&\text{subject to:} \\
&\sum_{(v_i,v_j)\in A} c_{ij}^s x_{ij} \le y, \text{ for all } s \in S \\
&\sum_{\substack{v_j\in V \\ (v_i,v_j)\in A}} x_{ij} - \sum_{\substack{v_j\in V \\ (v_j,v_i)\in A}} x_{ji} = \begin{cases} F \text{ if } v_i = v_1 \\ -F \text{ if } v_i = v_n \\ 0 \text{ if } v_i \notin \{v_1, v_n\} \end{cases}, \text{ for all } v_i \in V \qquad (17) \\
&0 \le x_{ij} \le \min_{s\in S} u_{ij}^s, \text{ for all } (v_i,v_j) \in A \\
&x_{ij} \text{ integer, for all } (v_i,v_j) \in A
\end{aligned}$$

GRMCIF-D'...

$$\begin{aligned}
&\text{minimize } y \\
&\text{subject to the constraints from } \textit{Formula 17} \text{ where the 1}^{\text{st}} \\
&\text{of them is changed to:} \qquad (18) \\
&\sum_{(v_i,v_j)\in A} c_{ij}^s x_{ij} \le y + z^s, \text{ for all } s \in S
\end{aligned}$$

GRMCIF-R',

$$\begin{aligned}
&\text{minimize } y \\
&\text{subject to the constraints from } \textit{Formula 17} \text{ where the 1}^{\text{st}} \\
&\text{of them is changed to:} \qquad (19) \\
&\sum_{(v_i,v_j)\in A} c_{ij}^s x_{ij} \le (y+1) z^s, \text{ for all } s \in S
\end{aligned}$$

Next, let us note that the above linearized problem versions can be further simplified by relaxation, i.e. by skipping integrality constraints for arc values x_{ij}. Of course, the relaxed problem versions are not equivalent to the original versions since their solutions can be non-integral. Still, relaxation is useful because it provides lower bounds for integer solutions. Also, relaxed problem versions, being ordinary linear programming (LP) tasks, are much easier to solve.

Now we can explain in more detail how our new approximate algorithm works. For a given instance of GRMCIF-A (or GRMCIF-D or GRMCIF-R), the algorithm finds a solution in two steps.

1) The corresponding instance of the relaxed GRMCIF-A' (or GRMCIF-D' or GRMCIF-R') is solved as an ordinary LP instance. The obtained solution is a flow with the required value F, but possibly non-integer arc values x_{ij}.
2) The flow obtained in Step 1 is transformed into an integer flow with the same value F and arc values x_{ij} rounded to integers. That integer flow is regarded as an approximate solution of the original problem instance.

Step 1 in our algorithm can be implemented by any general-purpose optimization package able to solve LP instances, e.g. by Simplex method [3]. In our experiments we have used IBM ILOG CPLEX Optimization Studio [27], which is a state-of-the-art package able to solve very large LP instances.

To realize Step 2 of our algorithm, we have used a basic procedure called *flow rounding*, which in turn uses another basic procedure called *flow augmentation*. Both procedures have been developed in [22], but here we are using them in a novel way. A detailed description of flow rounding, as well as of flow augmentation, can be found in [22]. In our experiments, we have implemented the two procedures by our own C# program [28].

An illustration of flow rounding is given in *Figure 2*. The left part of the figure specifies a non-integer flow in a network. Thereby arc flow values x_{ij} are presented as left-hand arc labels, and arc capacities as right-hand labels. The overall flow value F equals 10. The right part of the figure shows the rounded version of the same flow obtained by applying the rounding procedure. We see that the flow value F is the same as before, i.e. 10, and that each x_{ij} has been rounded to a nearby integer (either smaller or larger).

Now follows a more detailed analysis of the example given in *Figure 2*. The rounding procedure is described step-by-step.

- First, a pseudo-flow is constructed, where all initial (non-integer) arc flow values x_{ij} from the left part of *Figure 2* are rounded to their nearest integers $\lfloor x_{ij} + 0.5 \rfloor$. It is obvious that the obtained pseudo-flow must obey the capacity constraints along arcs, thanks to the fact that all arc capacities are integral. Namely, if a certain non-integer value is below a certain integer limit, then the rounded version of the same value cannot exceed the same limit. On the other hand, the pseudo-flow may violate the flow conservation rule in vertices. Indeed, in our particular case the conservation rule is infringed in vertices v_2 and v_6 (namely, v_2 receives 9 units of flow but sends only 3+5=8 units, while v_6 receives 3+3=6 units but sends 7 units).
- In order to correct these violations, the rounding procedure next decomposes the pseudo-flow into unit flows (i.e. integer flows that transfer one unit of flow from the source to the sink). More precisely, the procedure extracts from the pseudo-flow as many unit flows as possible. The extracted unit flows are summed up in order to produce the first version of the sought rounded flow. It is obvious that the obtained sum must be a feasible integer flow: namely, it satisfies the flow conservation rule (since it combines unit flows) and it also obeys the capacity constraints (since its arc values cannot exceed the corresponding values in the initial pseudo-flow).
- The only obstacle with the described sum of unit flows is that its value F can be smaller than desired. This is exactly what happens in our case, where only 9 unit flows can be extracted from the initial pseudo-flow. Thus, the first version of the rounded flow has value F equal to 9 instead of 10. Thereby the arcs (v_2,v_5) and (v_5,v_6) are loaded below their capacity, i.e. with only 3 units of flow.
- The final part of flow rounding consists of invoking the auxiliary flow augmentation procedure. It finds possibilities to send additional units of flow from the source to the sink, maybe through paths

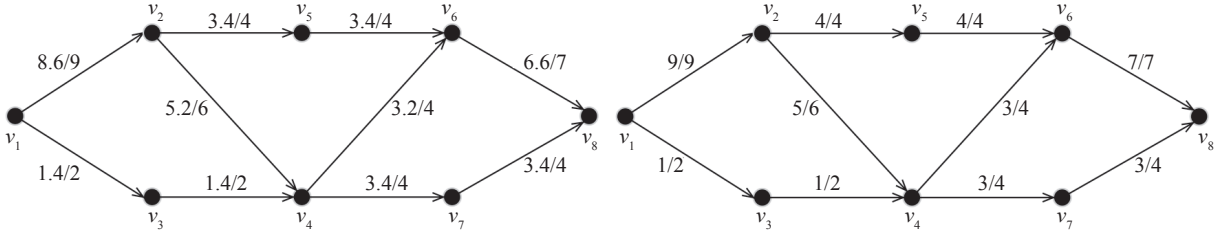

Figure 2 – An example of flow rounding

that have not been used by the pseudo-flow. In our case, the flow augmentation sends an extra flow unit through the uppermost path in the network. In this way, the final integer flow is produced with value F equal to 10, as shown in the right part of *Figure 2*.

Note that in our example most of the original arc flow values x_{ij} have been rounded to their nearest integers $\lfloor x_{ij} + 0.5 \rfloor$, only the flow values for arcs (v_2,v_5) and (v_5,v_6) have eventually been rounded to their next larger integers $\lceil x_{ij} \rceil$.

Finally, let us note that in the case of (relative) robust deviation, prior to running our algorithm, it is also necessary to compute the optimal (conventional) cost z^s for each scenario s. Namely, those values must be specified within problem constraints. Computing of z^s can be accomplished by any algorithm for solving the conventional minimum-cost flow problem. In our experiments, we have employed the Malholtra-Kumar-Maheshvari algorithm for finding flows with given values, combined with the Floyd-Warshall algorithm for finding negative-length cycles in the so-called displacement networks [2, 25].

5. EXPERIMENTS

In our experiments, we have used a set of 30 carefully constructed GRMCIF problem instances, denoted with I-61, I-62, …, I-90. They are in fact slight modifications of similar RMCIF instances from [22]. The instances from both sets have been chosen to be large enough, so that they cannot be solved to optimality by a general-purpose optimization package such as IBM ILOG CPLEX [27]. Indeed, any attempt to find their exact (i.e. optimal and integral) solution by CPLEX results in an out-of-memory error. On the other hand, the chosen instances are still small enough to be solvable in a relaxed form. It means that CPLEX can find their optimal non-integer solution after the integrality constraints to arc flow values have been abandoned. It also means that our approximate algorithm from Section 4 can find its nearly optimal solution by rounding the result from CPLEX.

Note that the relaxed solution of a problem instance is never worse than the exact solution. Consequently, the robust cost of the relaxed solution can serve as a lower bound for robust cost of the exact solution. Note also that the approximate solution obtained by our algorithm is never better than the exact solution. Consequently, robust cost of the approximate solution can serve as an upper bound for robust cost of the exact solution. Putting it all together, although we do not know the exact solutions of our problem instances, we can still obtain fairly tight interval estimates of their robust costs by using the above described lower and upper bounds.

All our problem instances involve layered networks like those occurring in real-life applications of network flows. The structure of a layered network is illustrated in Figure 4 in [22]. Thus, all vertices except the source and the sink are grouped into layers. Arcs can connect only vertices between adjacent layers. The number of vertices within a layer is called *layer width*, and it can be fixed (the same for each layer) or varying.

Essential properties of our problem instances are summarized in *Table 1*. The values shown in the table have been chosen by hand, and the remaining details not visible from the table (such as configuration of arcs and concrete values of u_{ij}^s or c_{ij}^s) have been generated randomly. The complete specification of each problem instance can be found in our repository at the address http://hrzz-rodiopt.math.pmf.unizg.hr.

Regarding our problem instances, we admit that it would be much better if we used some standard benchmark data instead of modifications of data from [22]. However, to the best of our knowledge, we are not aware of any suitable benchmark collection. There are some well-known repositories of minimum-cost flow problem instances, e.g. [29, 30], but they cover only the conventional (non-robust) problem variant. Expanding a conventional benchmark instance with more scenarios would not make sense since it would produce a new instance that cannot be compared with the original one.

In our experiments, we have employed our approximate algorithm from Section 4 to solve each of the 30 problem instances, i.e. I-61, I-62, …, and I-90, according to each of the three problem variants, i.e. GRMCIF-A, GRMCIF-D and GRMCIF-R, respectively. The whole computation has been done on a standard notebook computer with a 2.60 GHz Intel Core i5-6440HQ processor and 4 GBytes of memory. The obtained results are presented in *Table 2*.

As we can see, for any problem instance and any problem variant *Table 2* reports two robust costs, the first one is obtained after the first step of the algorithm (relaxation) and the second is obtained after the second step (rounding). Also, there is relative difference of the second cost vs. the first cost, expressed as a percentage. As already mentioned

Table 1 – Properties of the chosen GRMCIF problem instances

Instance identifier	# of vertices $\|V\|$	# of arcs $\|A\|$	# of scenarios $\|S\|$	Flow value F	# of layers	Layer width	Range for u_{ij}^s, c_{ij}^s
I-61	122	696	60	215	20	6 (fixed)	0-99
I-62	122	472	60	183	30	4 (fixed)	0-99
I-63	122	357	60	96	40	3 (fixed)	0-99
I-64	122	240	60	68	60	2 (fixed)	0-99
I-65	122	2,818	60	35	7	48,40,18,5,4,3,2	0-99
I-66	102	485	60	179	20	5 (fixed)	0-99
I-67	102	200	60	60	50	2 (fixed)	0-99
I-68	102	2,041	60	115	4	61,28,9,2	0-99
I-69	162	1,232	30	329	20	8 (fixed)	0-99
I-70	162	632	30	85	40	4 (fixed)	0-99
I-71	162	320	30	17	80	2 (fixed)	0-99
I-72	162	4,466	30	321	4	68,34,52,6	0-99
I-73	152	735	30	224	30	5 (fixed)	0-99
I-74	152	447	30	147	50	3 (fixed)	0-99
I-75	152	4,404	30	161	4	100,40,7,3	0-99
I-76	222	2,321	15	457	20	11 (fixed)	0-99
I-77	222	440	15	26	110	2 (fixed)	0-99
I-78	222	7,174	15	186	5	138,40,34,4,4	0-99
I-79	212	1,435	15	145	30	7 (fixed)	0-99
I-80	212	627	15	95	70	3 (fixed)	0-99
I-81	212	5,543	15	274	7	104,32,32,30,3,4,5	0-99
I-82	252	1,235	10	15	50	5 (fixed)	0-9
I-83	252	14,138	10	53	5	87,120,29,3,11	0-9
I-84	242	2,760	10	56	20	12 (fixed)	0-9
I-85	242	1,872	10	41	30	8 (fixed)	0-9
I-86	242	1,416	10	35	40	6 (fixed)	0-9
I-87	242	952	10	26	60	4 (fixed)	0-9
I-88	242	717	10	19	80	3 (fixed)	0-9
I-89	242	480	10	6	120	2 (fixed)	0-9
I-90	242	6,680	10	20	5	168,32,26,10,4	0-9

earlier, the two reported values provide an interval estimate for the unknown exact-solution robust cost. Therefore, the computed percentage can be interpreted as an upper bound for the relative error of our approximate solution vs. the exact solution.

In its last row, *Table 2* also comprises average bounds for relative errors. Averaging is done over the whole set of problem instances and for each problem variant separately. The presented averages indicate that our approximate algorithm is very accurate. Actual approximation errors depend on the problem variant, and they are better for GRMCIF-A than for GRMCIF-D or GRMCIF-R. We can expect that a solution according to the absolute robustness criterion will be accurate up to 1%. For the (relative) deviation robustness criteria errors could be slightly larger but still below 2%.

The experimental results from this paper can easily be compared with similar results from [22]. In [22] there is a set of 30 RMCIF instances, denoted with I-31, I-32, …, I-60, which are solved according to the RMCIF-A and RMCIF-D problem variants by 13 heuristics. In fact, I-31, I-32, …, and I-60 are almost identical to I-61, I-62, …, and I-90, respectively. The

Table 2 – Robust solution costs obtained with relaxation and rounding

	GRMCIF-A			GRMCIF-D			GRMCIF-R		
Instance identifier	Relaxed solution	Rounded relaxed solution	Relat. error ≤ [%]	Relaxed solution	Rounded relaxed solution	Relat. error ≤ [%]	Relaxed solution	Rounded relaxed solution	Relat. error ≤ [%]
I-61	211,168	212,427	0.60	143,261	143,834	0.40	2.287610	2.300072	0.54
I-62	269,898	271,864	0.73	153,656	154,750	0.71	1.405138	1.412041	0.49
I-63	192,327	193,476	0.60	100,652	101,434	0.78	1.137112	1.154330	1.51
I-64	211,414	212,474	0.50	79,579	80,333	0.95	0.644371	0.650414	0.94
I-65	15,440	15,987	3.54	7,832	8,735	11.53	1.133585	1.205848	6.37
I-66	179,401	180,226	0.46	116,453	117,553	0.94	1.974020	1.989308	0.77
I-67	155,463	156,182	0.46	57,379	57,927	0.96	0.651568	0.658212	1.02
I-68	29,104	29,468	1.25	19,656	20,721	5.42	3.177696	3.196130	0.58
I-69	303,919	304,706	0.26	216,885	217,642	0.35	2.542881	2.548876	0.24
I-70	157,961	158,742	0.49	98,859	99,555	0.70	1.764815	1.780503	0.89
I-71	66,113	66,219	0.16	26,357	27,317	3.64	0.683370	0.709696	3.85
I-72	68,149	68,448	0.44	51,538	51,935	0.77	3.671287	3.689214	0.49
I-73	315,410	316,464	0.33	193,483	194,473	0.51	1.641403	1.651402	0.61
I-74	356,849	357,682	0.23	160,471	162,029	0.97	0.844588	0.848251	0.43
I-75	36,179	36,439	0.72	25,069	25,347	1.11	2.969617	3.002195	1.10
I-76	382,518	383,353	0.22	284,219	285,310	0.38	2.942799	2.948734	0.20
I-77	131,974	133,031	0.80	47,090	48,026	1.99	0.564684	0.574444	1.73
I-78	44,809	44,982	0.39	31,452	31,688	0.75	2.610898	2.637670	1.03
I-79	179,904	180,155	0.14	131,487	133,271	1.36	2.750578	2.752951	0.09
I-80	305,734	306,843	0.36	147,924	148,735	0.55	0.955830	0.961266	0.57
I-81	89,713	89,978	0.30	58,645	59,303	1.12	2.011219	2.020388	0.46
I-82	3,126	3,144	0.58	1,706	1,779	4.28	1.204502	1.256445	4.31
I-83	1,240	1,272	2.58	658	688	4.56	1.145100	1.158756	1.19
I-84	4,495	4,501	0.13	2,800	2,885	3.04	1.652181	1.683111	1.87
I-85	5,086	5,199	2.22	2,916	2,926	0.34	1.347415	1.401887	4.04
I-86	6,024	6,098	1.23	3,205	3,210	0.16	1.143578	1.171286	2.42
I-87	7,047	7,057	0.14	2,941	2,949	0.27	0.717120	0.719810	0.38
I-88	7,121	7,133	0.17	2,490	2,502	0.48	0.540152	0.560176	3.71
I-89	3,349	3,358	0.27	1,000	1,008	0.80	0.427695	0.431049	0.78
I-90	472	500	5.93	252	266	5.56	1.179771	1.267045	7.40
Avg.			0.87			1.85			1.67

only difference is that their arc capacities are fixed through scenarios. Thereby the fixed capacity of each arc (v_i,v_j) in each RMCIF instance is equal to u^*_{ij} in the corresponding GRMCIF instance. As explained earlier, in such circumstances the RMCIF-A variant is equivalent to the GRMCIF-A variant, so that a direct comparison of solutions (instance by instance) is possible. On the other hand, the RMCIF-D and GRMCIF-D variants are not quite equivalent, but their solutions are still comparable on the level of average relative errors. Namely, RMCIF-D instances can be considered as simplified GRMCIF-D instances, whose solving should not be harder than for general GRMCIF-D instances.

The performance of 13 considered heuristics over the considered RMCIF instances is presented by Table 9 in [22]. We can see that the best results are produced by a hybrid of evolutionary computing

and local search denoted as EC-9. Thereby the obtained average error bounds for the RMCIF-A and RMCIF-D variants are 6.30% and 12.40%, respectively. These values are worse than the corresponding (analogously computed) values from *Table 2* in this paper, where instead of 6.30% and 12.40% we have 0.87%, and 1.85%, respectively. Thus, we can observe that the approximate algorithm from this paper assures much better accuracy than the best heuristic from [22].

6. CONCLUSION

In this paper several robust variants of the maximum flow problem and the minimum-cost flow problem have been considered. Thereby, only integer flows have been taken into account. A common property of all the considered robust problem variants is that they assume uncertainty of network arc capacities. In the case of the minimum-cost flow problem, arc unit costs are uncertain as well. All uncertainties are expressed by discrete (finite and explicitly given) sets of scenarios.

In the paper it has been shown that the considered robust variants of the maximum flow problem are easy to solve, i.e. they can be reduced to the non-robust variant of the same problem. On the other hand, the considered robust minimum-cost flow problem variants turn out to be NP-hard. In order to solve such NP-hard tasks, the paper has proposed an approximate algorithm, which is based on relaxation and rounding. Experiments have been presented where the proposed algorithm was implemented and tested on appropriate problem instances.

According to the presented experiments, our new approximate algorithm for solving robust variants of the minimum-cost integer flow problem seems to be very accurate. Depending on the involved robustness criterion, relative errors of the obtained approximate solutions vs. the corresponding exact solutions range from 1% to 2%. Such accuracy cannot be achieved by other relevant algorithms found in the literature whose relative errors are several times larger.

The just mentioned new algorithm can surely be regarded as the most important original contribution of the paper. However, there are also some other interesting contributions that should be noted. For instance, the analysis of robust maximum flow problem variants is very useful, since it provides a way for solving such variants by conventional (already existing) algorithms.

The present implementation of our new algorithm relies on a general-purpose linear programming package. This fact can be interpreted as a potential drawback. It implies namely, that the implemented algorithm is applicable only to moderately sized problem instances whose relaxed versions are still solvable by the used software. It is true that state-of-the-art general-purpose packages installed on state-of-the-art computers can accommodate considerably large problem instances. Still, extremely large instances could exceed the available resources.

In our future research we plan to explore limitations of our implemented algorithm regarding problem instance size. Also, we will try to develop a dedicated software solution that would overcome possible limitations of the presently used general-purpose package. An additional plan is to test our algorithm and software on a real-world optimization problem belonging to the area of traffic and transportation. It could be a shipping application, where road capacities and costs of transport are uncertain. Such uncertainty can be expressed through scenarios that correspond to weather conditions and/or economic circumstances.

ACKNOWLEDGEMENT

This work has been fully supported by the Croatian Science Foundation under the project IP-2018-01-5591.

Dr. sc. **MARKO ŠPOLJAREC**[1]
E-mail: marko.spoljarec@internationalvalueservices.com
Dr. sc. **ROBERT MANGER**[2]
E-mail: manger@math.hr
[1] Intesa Sanpaolo International Value Services
Radnička cesta 44, 10000 Zagreb, Hrvatska
[2] Matematički odsjek
Prirodoslovno-matematički fakultet
Sveučilište u Zagrebu
Bijenička cesta 30, 10000 Zagreb, Hrvatska

RJEŠAVANJE ROBUSNIH VARIJANTI PROBLEMA CJELOBROJNOG TOKA S NESIGURNIM KAPACITETIMA LUKOVA

SAŽETAK

Ovaj rad bavi se robusnom optimizacijom i tokovima u mrežama. Promatra se nekoliko robusnih varijanti za probleme cjelobrojnog toka. U tim varijantama pretpostavlja se da postoji nesigurnost u pogledu kapaciteta lukova u mreži te također nesigurnost jediničnih cijena za lukove (tamo gdje one postoje). Nesigurnost

se izražava preko diskretnih scenarija. S obzirom da se promatrane varijante problema maksimalnog toka mogu lagano riješiti, rad se uglavnom koncentrira na NP-teške varijante problema toka s minimalnom cijenom te predlaže algoritam za njihovo približno rješavanje. Točnost predloženog algoritma provjerava se pomoću eksperimenata.

KLJUČNE RIJEČI

tok u mreži; cjelobrojni tok; robusna optimizacija; algoritam.

REFERENCES

[1] Bazaraa MS, Jarvis JJ, Sherali HD. *Linear Programming and Network Flows.* Fourth Edition. Hoboken NJ: Wiley; 2010.

[2] Jungnickel D. *Graphs, Networks and Algorithms.* Fourth Edition. Springer, Berlin; 2013.

[3] Papadimitriou CH, Steiglitz K. *Combinatorial Optimization – Algorithms and Complexity.* Mineola NY: Dover Publications; 1998.

[4] Ben-Tal A, El Ghaoui L, Nemirovski A. *Robust Optimization*. Princeton NJ: Princeton University Press; 2009.

[5] Bertsimas D, Sim M. Robust discrete optimization and network flows. *Mathematical Programming.* 2003;98: 49-71.

[6] Bertsimas D, Sim M. The price of robustness. *Operations Research*. 2004;52: 35-53.

[7] Kouvelis P, Yu G. *Robust Discrete Optimization and Its Applications*. Berlin: Springer; 1997.

[8] Aissi H, Bazgan C, Vanderpooten D. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*. 2009;197: 427-438.

[9] Aissi H, Bazgan C, Vanderpooten D. General approximation schemes for min-max (regret) versions of some (pseudo-)polynomial problems. *Discrete Optimizaton*. 2010;7: 136-148.

[10] Bertsimas D, Brown DB, Caramanis C. Theory and applications of robust optimization. *SIAM Review*. 2011;53: 464-501.

[11] Kasperski A, Zielinski P. Robust discrete optimization under discrete and interval uncertainty: A survey. In: Doumpos M, Zopounidis C, Grigoroudis E. (eds.) *Robustness Analysis in Decision Aiding, Optimization, and Analytics.* Cham CH: Springer; 2016. p. 113-143.

[12] Aissi H, Vanderpooten D. Robust capacity expansion of a network under demand uncertainty: A bi-objective approach. *Networks*. 2016;68: 185-199.

[13] Atamtürk A, Zhang M. Two-stage robust network flow and design under demand uncertainty. *Operations Research*. 2007;55: 662-673.

[14] Bertsimas D, Nasrabadi E, Stiller S. Robust and adaptive network flows. *Operations Research.* 2013;61: 1218-1242.

[15] Boginski VL, Commander CW, Turko T. Polynomial-time identification of robust network flows under uncertain arc failures. *Optimization Letters*. 2009;3: 461-473.

[16] Minoux M. On robust maximum flow with polyhedral uncertainty sets. *Optimization Letters*. 2009;3: 367-376.

[17] Minoux M. Robust network optimization under polyhedral demand uncertainty is NP-hard. *Discrete Applied Mathematics*. 2010;158: 597-603.

[18] Ordoñez F, Zhao J. Robust capacity expansion of network flows. *Networks.* 2007;50: 136-145.

[19] Poss M. A comparison of routing sets for robust network design. *Optimization Letters*. 2014;8: 1619-1635.

[20] Righetto GM, Morabito R, Alem D. A robust optimization approach for cash flow management in stationery companies. *Computers and Industrial Engineering*. 2016;99: 137-152.

[21] Rui M, Jinfu Z. Robust discrete optimization for the minimum cost flow problem. In: Wang C, Ye Z. (eds.) *Proceedings of the International Workshop on Intelligent Systems and Applications – ISA 2009. Wuhan, China, 23-24 May 2009.* Piscataway NJ: IEEE; 2009. p. 1-4.

[22] Špoljarec M, Manger R. Heuristic solutions to robust variants of the minimum-cost integer flow problem. *Journal of Heuristics*. 2020;26: 531-559.

[23] Eiben AE, Smith JE. *Introduction to Evolutionary Computing*. Second edition. Berlin: Springer; 2015.

[24] Talbi EG. *Metaheuristics – From Design to Implementation.* Hoboken NJ: Wiley; 2009.

[25] Korte B, Vygen J. *Combinatorial Optimization – Theory and Algorithms*. Fifth Edition. Berlin: Springer; 2012.

[26] Carre B. *Graphs and Networks.* Oxford UK: Oxford University Press; 1979.

[27] IBM Corporation. *IBM ILOG CPLEX Optimization Studio, CPLEX User's Manual, Version 12, Release 8*. IBM Knowledge Center; 2017. Available from: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0 [Accessed 17 Dec 2018].

[28] Troelsen A, Japikse P. *C# 6.0 and the .NET 4.6 Framework.* 7th Edition. New York NY: Apress; 2016.

[29] Beasley JE. *OR-Library*. Brunel University London; 2018. Available from: http://people.brunel.ac.uk/~mastjjb/jeb/info.html [Accessed 17 Dec 2018].

[30] DIMACS – Center for Discrete Mathematics and Theoretical Computer Science. DIMACS *Implementation Challenges*. Piscataway NJ: Rutgers University; 2017. Available from: http://archive.dimacs.rutgers.edu/Challenges [Accessed 17 Dec 2018].