RAIRO-Oper. Res. 52 (2018) 1107–1121
https://doi.org/10.1051/ro/2017088

RAIRO Operations Research
www.rairo-ro.org

REVERSE MAXIMUM FLOW PROBLEM UNDER THE WEIGHTED CHEBYSHEV DISTANCE

Javad Tayyebi[1,2], Abumoslem Mohammadi[2]
and Seyyed Mohammad Reza Kazemi[1,*]

Abstract. Given a network $G(V, A, \mathbf{u})$ with two specific nodes, a source node s and a sink node t, the reverse maximum flow problem is to increase the capacity of some arcs (i, j) as little as possible under bound constraints on the modifications so that the maximum flow value from s to t in the modified network is lower bounded by a prescribed value v^0. In this paper, we study the reverse maximum flow problem when the capacity modifications are measured by the weighted Chebyshev distance. We present an efficient algorithm to solve the problem in two phases. The first phase applies the binary search technique to find an interval containing the optimal value. The second phase uses the discrete type Newton method to obtain exactly the optimal value. Finally, some computational experiments are conducted to observe the performance of the proposed algorithm.

Mathematics Subject Classification. 90C35, 90B10, 05C21

Received 9 April 2017. Accepted 30 November 2017.

1. Introduction

For an optimization problem, the corresponding inverse problem is to modify some parameters of the problem as less cost as possible in such a way that some desired goals are achieved. Two general classes of inverse problems are interested in the literature (see [11, 16] for a survey):

- Given an optimization problem as well as its a feasible solution $\mathbf{x}^0$, modify some parameters of the problem as little as possible so that $\mathbf{x}^0$ becomes an optimal solution with respect to new parameters.
- Given a maximization problem, adjust some parameters of the problem minimally so that the optimal objective value of the modified problem is lower bounded by a prescribed value v^0.

The first class is called "the inverse optimization problem" in the literature [2, 3]. To make distinction from the first, the other is referred to as "the reverse optimization problem" [24, 25, 27].

In the inverse and reverse optimization problems, suppose that we wish to modify the vector $\mathbf{u} = (u_i)_{i\in I}$ into $\hat{\mathbf{u}} = (\hat{u}_i)_{i\in I}$. The objective function is to minimize the distance between the initial vector $\mathbf{u}$ and the modified vector $\hat{\mathbf{u}}$. For instance, each one of the following distance functions can be applied as the objective function:

Keywords and phrases: Maximum flow problem, reverse problem, Chebyshev distance, network design, Newton method, binary search.

[1] Department of Industrial Engineering, Birjand University of Technology, Birjand, Iran.
[2] Department of Mathematics, Faculty of sciences, Imam Ali University, Tehran, Iran.

* Corresponding author: kazemi@birjandut.ac.ir

Article published by EDP Sciences

- Euclidean distance (l_2): $(\sum_{i\in I} w_i(\hat{u}_i - u_i)^2)^{\frac{1}{2}}$;
- Manhattan distance (l_1): $\sum_{i\in I} w_i|\hat{u}_i - u_i|$;
- Chebyshev distance (l_∞): $\max_{i\in I} w_i|\hat{u}_i - u_i|$;
- Sum-type Hamming distance (H_1): $\sum_{i\in I} w_i H(u_i, \hat{u}_i)$;
- Bottleneck-type Hamming distance (H_∞): $\max_{i\in I} w_i H(u_i, \hat{u}_i)$;

in which $w_i > 0$ is a weight (or a penalty) associated with u_i and $H(u_i, \hat{u}_i)$ is the Hamming distance between u_i and $\hat{u}_i$, *i.e.*, $H(u_i, \hat{u}_i) = 0$ if $\hat{u}_i = u_i$ and $H(u_i, \hat{u}_i) = 1$ otherwise.

Burton and Toint [5, 6] were the first to introduce the reverse shortest path problem under the Euclidean distance. Since then, different inverse and reverse optimization problems have been studied by many authors under the Manhattan, Chebyshev and Hamming distances. For example, the reverse shortest path problem under l_1, l_2 and H_1 is investigated in [26, 27]. It is proved that the problem is strongly NP-hard. For the special case that the network is a tree, efficient algorithms is designed to solve the problem [21]. The inverse shortest path problem is also considered by several authors [3, 7, 8, 14]. It is shown that the inverse shortest path problem under the distances l_1, l_∞ and H_∞ can be solved in polynomial time while the problem under the sum-type Hamming distance is NP-hard.

The inverse maximum flow problem is to modify arc capacities minimally so that a given flow becomes a maximum flow with respect to the modified capacities. This problem under the distances l_1 and H_1 is investigated by several authors [10, 18, 19, 23]. The general result is that the problem can be reduced to a minimum cut problem in an auxiliary network and consequently, it can be solved in strongly polynomial time. The inverse maximum flow problem under l_∞ is studied by Deaconu [9] and an optimal solution of the problem is computed by finding a min-max cut on an auxiliary network.

The reverse maximum flow problem is to look for a new capacity vector such that the maximum flow value with respect to new capacities is lower bounded by a prescribed value v^0. The goal of the problem is to minimize the distance between the initial and the new capacity vectors. A problem, called the budget-constrained flow improvement problem, has a structure similar to the reverse maximum flow problem. This problem is to increase the capacities within a budget constraint such that the flow value from the source to the sink is maximized. Notice that by applying the binary search, we can obtain an optimal solution to either of the problems by solving the other. However, both the problems arise from some practical applications in designing transport networks.

The budget-constrained flow improvement problem is studied in [17] whenever the budget constraint is with respect to the distances l_1 and H_1. It is shown that the problem under l_1 is solvable in polynomial time while it under H_1 is strongly NP-hard.

1.1. Our contribution

In this paper, the reverse maximum flow problem is considered under the weighted Chebyshev distance. The problem under l_2 and H_∞ is not studied in this paper because the first case yields a quadratic programming problem while the other can be solved by a formal search procedure [12, 22].

Our contribution is to develop an efficient algorithm for solving the reverse maximum flow problem under the weighted Chebyshev distance. The algorithm contains two phases. The first phase uses the binary search technique to find an interval containing the optimal value. The second phase reduces the dual of the problem to a maximum ratio cut problem and then, applies the discrete type Newton method to obtain the optimal value. Finally, an optimal solution is constructed from the optimal value. In a series of computational experiments, the performance of the algorithm is illustrated.

The rest of the paper is organized as follows: Section 2 provides some primary notions about the maximum flow problem. In Section 3, the reverse maximum flow problem is formulated and some primary results are presented. Section 4 considers the reverse maximum flow problem under the Chebyshev distance and presents an efficient algorithm for solving it. In Section 5, some computational experiments are conducted to observe the performance of the algorithm. Finally, some concluding remarks are given in Section 6.

2. Preliminaries

In this section, we provide some notions and notations used throughout this paper.

Suppose that $G(V, A, \mathbf{u})$ is a directed network where $V = \{1, 2, \ldots, n\}$ is the node set and A is the set of m arcs. A nonnegative capacity u_{ij} is associated with each arc (i, j). Without loss of generality, we assume that G does not contain both arcs (i, j) and (j, i) for every $i, j \in V$. If not, we split (i, j) into two arcs (i, k) and (k, j) by introducing a dummy node k.

In the maximum flow problem, we wish to send as much flow as possible between two specific nodes, a source node s and a sink node t, without exceeding the capacity of any arc. This problem is formulated as follows:

$$\max\ v \tag{2.1a}$$

$$\text{s.t.} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} v & i = s, \\ 0 & i \in V\backslash\{s, t\}, \\ -v & i = t, \end{cases} \tag{2.1b}$$

$$0 \le x_{ij} \le u_{ij} \quad \forall (i, j) \in A, \tag{2.1c}$$

The problem (2.1) has been studied extensivelyin both theoretic and algorithmic aspects and has wide range of applications [1]. A vector $\mathbf{x}$ satisfying constraints (2.1b) and (2.1c) for some $v \ge 0$ is refer to as a feasible flow and v is called its value.

For a feasible flow $\mathbf{x}^0$ to the problem (2.1), its residual network $G_{\mathbf{x}^0}(V, A', \mathbf{u}')$ can be constructed by the following algorithm in which $u'(i, j)$ is called the residual capacity of arc (i, j) [1].

Algorithm 2.1. Constructing the residual network.

1: The node set is still V.
2: **for** $(i, j) \in A$
3: **if** $x^0_{ij} < u_{ij}$ **then**
4: Add (i, j) to A' with $u'_{ij} = u_{ij} - x^0_{ij}$.
5: **end if**
6: **if** $x^0_{ij} > 0$ **then**
7: Add (j, i) to A' with $u'_{ji} = x^0_{ij}$.
8: **end if**
9: **end for**

We denote the arc sets created by Steps 2 and 3 by A'_1 and A'_2, respectively. Obviously, $A'_1 \subseteq A$ and $A' = A'_1 \cup A'_2$.

Let us now recall some definitions. An st-path is a directed path from s to t. An st-cut C is a minimal set of arcs so that if they are removed, then there is not any st-path in G. Thus the removal of arcs belonging to an st-cut will separate some $S, \bar{S} \subseteq V$ into exactly two connected components so that $s \in S$ and $t \in \bar{S}$. We refer to an arc (i, j) with $i \in S$ and $j \in \bar{S}$ as a forward arc of C, and refer to an arc (i, j) with $i \in \bar{S}$ and $j \in S$ as a backward arc. Suppose $(S, \bar{S})$ and $(\bar{S}, S)$ denote respectively the set of forward and backward arcs in an st-cut C. The capacity of an st-cut is the sum of capacities of its forward arcs. An st-cut with minimum capacity is called a minimum st-cut. It is customary to denote an st-cut by $C = [S, \bar{S}]$ and its capacity by $\mathbf{u}(C) = \mathbf{u}[S, \bar{S}]$. The following theorem establishes the relationship between the maximum flow problem and the minimum cut problem [15].

Theorem 2.2 (Max-flow Min-cut Theorem)**.** *The value of a maximum flow from s to t in a capacitated network $G(V, A, \mathbf{u})$ equals the capacity of its minimum st-cut.*

3. Problem statement

In this section, we introduce and formulate the reverse maximum flow problem and then, establish some primary results on the problem. Given an instance of the problem (2.1), the reverse maximum flow problem is to look for a new capacity vector $\hat{\mathbf{u}}$ so that the following conditions are satisfied:

- The maximum flow value in the network $G(V, A, \hat{\mathbf{u}})$ is lower bounded by a prescribed value $v^0 \geq 0$.
- $u_{ij} \leq \hat{u}_{ij} \leq u_{ij} + p_{ij}$ for every $(i, j) \in A$ where $p_{ij} \geq 0$ is a given bound for increasing u_{ij}.
- The distance between $\mathbf{u}$ and $\hat{\mathbf{u}}$ is minimized.

Thus the reverse maximum flow problem is formulated as follows:

$$\min z = d(\mathbf{u}, \hat{\mathbf{u}}) \tag{3.1a}$$

$$\text{s.t.} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} \begin{cases} \geq v^0 & i = s, \\ = 0 & i \neq s, t, \\ \leq -v^0 & i = t, \end{cases} \quad \forall i \in V, \tag{3.1b}$$

$$0 \leq x_{ij} \leq \hat{u}_{ij} \quad \forall (i, j) \in A, \tag{3.1c}$$

$$u_{ij} \leq \hat{u}_{ij} \leq u_{ij} + p_{ij} \quad \forall (i, j) \in A, \tag{3.1d}$$

in which $d(.,.)$ is a function to measure the distance between $\mathbf{u}$ and $\hat{\mathbf{u}}$. Throughout this paper, we suppose that two following assumptions hold.

Assumption 3.1. The optimal value of the problem (3.1) is greater than zero.

Assumption 3.2. All data of the problem (3.1) are integral.

Let $\mathbf{x}^*$ be a maximum flow in the network $G(V, A, \mathbf{u})$ with the maximum flow value v^*. Obviously, if $v^0 \leq v^*$, then $(\hat{\mathbf{u}}, \mathbf{x}) = (\mathbf{u}, \mathbf{x}^*)$ is an optimal solution to the problem (3.1) whose objective value is equal to zero. Assumption 3.1 guarantees that the initial capacity vector $\mathbf{u}$ is not an optimal solution to the problem (3.1).

Let $\mathbf{x}^0$ be a feasible flow in G. A significant point about the reverse maximum flow problem is that one can simply introduce the problem on the residual network $G_{\mathbf{x}^0}(V, A', \mathbf{u}')$ instead of $G(V, A, \mathbf{u})$ in the following manner.

$$\min z = d(\mathbf{u}', \hat{\mathbf{u}}') \tag{3.2a}$$

$$\text{s.t.} \sum_{(i,j)\in A\cup A'_2} x_{ij} - \sum_{(j,i)\in A\cup A'_2} x_{ji} \begin{cases} \geq v^0 - v^* & i = s, \\ = 0 & i \neq s, t, \\ \leq -v^0 + v^* & i = t, \end{cases} \quad \forall i \in V, \tag{3.2b}$$

$$0 \leq x_{ij} \leq \hat{u}'_{ij} \quad \forall (i, j) \in A \cup A'_2, \tag{3.2c}$$

$$u'_{ij} \leq \hat{u}'_{ij} \leq u'_{ij} + p_{ij} \quad \forall (i, j) \in A. \tag{3.2d}$$

$$\hat{u}'_{ij} = x^*_{ji} \quad \forall (i, j) \in A'_2. \tag{3.2e}$$

In the problem (3.2), notice that arc $(i, j) \in A$ with zero residual capacity is added to the residual network whenever its residual capacity changes to a positive value. For this reason, we have to modify the residual capacity of arcs belonging to A and not only that of arcs in A'_1. On the other hand, we don't modify the residual capacity of arcs belonging to A'_2 because their residual capacity are not in term of the capacity vector $\mathbf{u}$. The following proposition is an immediate result on the relationship between the problems (3.1) and (3.2).

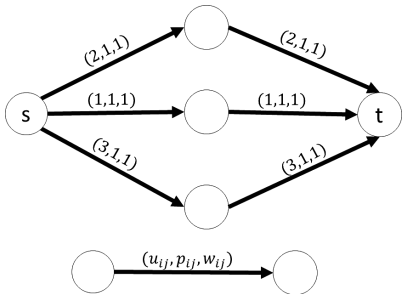


FIGURE 1. An instance of the problem (4.1) whose optimal solution is not integer. By assuming $v^0 = 7$, the problem has the optimal solution $\hat{u}_{ij} = u_{ij} + \frac{1}{3}$ for every arc (i, j) and consequently, its optimal value is $\frac{1}{3}$.

Proposition 3.3. *The problem (3.2) has an optimal solution $(\mathbf{x}^*, (\hat{\mathbf{u}}')^*)$ if and only if the problem (3.1) has an optimal solution $(\mathbf{x}^{**}, \hat{\mathbf{u}}^*)$ in which $\hat{u}^*_{ij} = (\hat{u}')^*_{ij} + x^0_{ij}$ and $x^{**}_{ij} = x^*_{ij} - x^*_{ji} + x^0_{ij}$ for each $(i, j) \in A$. Furthermore, the optimal objective values of both the problems are the same.*

Proof. The proof is straightforward. □

Lemma 3.4. *There exists an optimal solution of the problem (3.1) to satisfy the constraints (3.1b) in the equality form.*

Proof. Suppose that $(\mathbf{x}^*, \hat{\mathbf{u}}^*)$ is an optimal solution to the problem (3.1) and $\hat{v}^*$ is the value of the flow $\mathbf{x}^*$ in the network $G(V, A, \hat{\mathbf{u}}^*)$. It is easy to verify that $(\frac{v^0}{\hat{v}^*}\mathbf{x}^*, \hat{\mathbf{u}}^*)$ is also a feasible solution and its objective value is equal to that of $(\mathbf{x}^*, \hat{\mathbf{u}}^*)$. Moreover, the solution $(\frac{v^0}{\hat{v}^*}\mathbf{x}, \hat{\mathbf{u}})$ satisfies the constraints (3.1b) in the equality form. This completes the proof. □

4. Reverse problem under the weighted Chebyshev distance

In this section, we consider the reverse maximum flow problem in the case that the distance function $d(.,.)$ is the weighted Chebyshev distance and we propose an efficient algorithm to solve the problem.

The reverse maximum flow problem under the weighted Chebyshev distance is formulated as follows:

$$\min z = \max_{(i,j)\in A} w_{ij}(\hat{u}_{ij} - u_{ij}) \tag{4.1a}$$

$$\text{s.t.} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} \begin{cases} \geq v^0 & i = s, \\ = 0 & i \neq s, t, \\ \leq -v^0 & i = t, \end{cases} \quad \forall i \in V, \tag{4.1b}$$

$$0 \leq x_{ij} \leq \hat{u}_{ij} \quad \forall (i, j) \in A, \tag{4.1c}$$

$$u_{ij} \leq \hat{u}_{ij} \leq u_{ij} + p_{ij} \quad \forall (i, j) \in A, \tag{4.1d}$$

in which w_{ij} is a nonnegative weight associated with each arc (i, j) and the other parameters are defined as in the problem (3.1).

One can simply show that the reverse maximum flow problems under l_1 and H_1 have at least an integer optimal solution whenever Assumption 3.2 is satisfied (see the similar results to the budget-constrained flow

improvement problem in [17]). Unfortunately, this result is not valid in the case of the Chebyshev distance (see Fig. 1 for an example). However, we show that the problem can be solved in strongly polynomial time.

Define $\hat{p}_{ij} = \hat{u}_{ij} - u_{ij}$ for each $(i,j) \in A$ and $\hat{p}_{\max} = \max_{(i,j)\in A} w_{ij}\hat{p}_{ij}$. By Lemma 3.4, the problem (4.1) is reduced to the following linear programming problem.

$$\min \quad z = \hat{p}_{\max} \tag{4.2a}$$

$$\text{s.t.} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} v^0 & i = s, \\ 0 & i \neq s,t, \\ -v^0 & i = t, \end{cases} \quad \forall i \in V, \tag{4.2b}$$

$$0 \leq x_{ij} \leq u_{ij} + \hat{p}_{ij} \quad \forall (i,j) \in A, \tag{4.2c}$$

$$w_{ij}\hat{p}_{ij} \leq \hat{p}_{\max} \quad \forall (i,j) \in A, \tag{4.2d}$$

$$0 \leq \hat{p}_{ij} \leq p_{ij} \quad \forall (i,j) \in A. \tag{4.2e}$$

Lemma 4.1. *If $(\mathbf{x}, \hat{\mathbf{p}})$ is a feasible solution to the problem (4.2), then $(\frac{v^0}{v^*}\mathbf{x}^*, \hat{\mathbf{p}})$ is also feasible in which $\mathbf{x}^*$ is a maximum flow with the value v^* in the network $G(V, A, \mathbf{u} + \hat{\mathbf{p}})$. Furthermore, the values of both the solutions are the same.*

Proof. The proof follows from Lemma 3.4. □

Based on Lemma 4.1, it is sufficient to calculate the vector $\hat{\mathbf{p}}$ to determine a feasible solution to the problem (4.2) because $\mathbf{x}$ can be computed directly from $\hat{\mathbf{p}}$. We henceforth focus only on finding $\hat{\mathbf{p}}$. A simple property of the problem (4.2) is that if $\hat{\mathbf{p}}$ is a feasible solution, then any vector $\hat{\mathbf{p}}'$ with $\hat{\mathbf{p}} \leq \hat{\mathbf{p}}' \leq \mathbf{p}$ is also feasible. Using this property, the following result is immediate.

Theorem 4.2. *If the problem (4.2) has a feasible solution $\hat{\mathbf{p}}$ whose objective value is less than or equal to z, then $\hat{\mathbf{p}}^z$ defined as*

$$\hat{p}^z_{ij} = \begin{cases} \min\{\frac{z}{w_{ij}}, p_{ij}\} & w_{ij} > 0, \\ p_{ij}, & w_{ij} = 0, \end{cases} \quad \forall (i,j) \in A, \tag{4.3}$$

is also feasible and its objective value is less than or equal to z.

Proof. The result follows from the fact that $\hat{p}_{ij} \leq \hat{p}^z_{ij} \leq p_{ij}$ for every $(i,j) \in A$. □

Suppose that we have sorted the values of $\{w_{ij}p_{ij} : (i,j) \in A\} \cup \{0\}$ in nondecreasing order and let

$$0 = z_0 \leq z_1 \leq z_2 \leq z_3 \leq \ldots \leq z_m = \max_{(i,j)\in A} w_{ij}p_{ij}$$

denote the sorted list of these values. Based on Theorem 4.2, the following results are immediate.

Corollary 4.3. *The problem (4.2) is feasible if and only if $\hat{\mathbf{p}}^{z_m}$ defined by (4.3) for $z = z_m$ is a feasible solution of the problem.*

Proof. The proof of the sufficiency is trivial. On the proof of necessity, since any feasible solution has an objective value less than or equal to z_m, Theorem 4.2 implies the result. □

Corollary 4.4. *If the problem (4.2) is feasible, then its optimal objective value belongs to $(0, \min\{Wv^0, z_m\}]$ where $W = \max_{(i,j)\in A} w_{ij}$.*

Proof. Assumption 3.1 states that the optimal objective value of the problem (4.2) is greater than zero. Consider $\hat{\mathbf{p}}'$ defined by $\hat{p}'_{ij} = v^0$ for every $(i,j) \in A$. Obviously, $\hat{\mathbf{p}}'$ satisfies the constraints (4.2b)–(4.2c) but is not necessarily feasible because it does not satisfy the bound constraints (4.2e). The result is immediate by noting that the objective values of $\hat{\mathbf{p}}^{z_m}$ and $\hat{\mathbf{p}}'$ are respectively z_m and Wv^0. □

Corollary 4.5. *If the optimal objective value of the problem (4.2) is z^*, then $\hat{\mathbf{p}}^{z^*}$ defined by (4.3) for $z = z^*$ is an optimal solution to this problem.*

Proof. The result is a direct conclusion from Theorem 4.2. □

Corollaries 4.4 and 4.5 imply that the problem (4.2) is reduced to finding a minimum value $z^* \in (0, \min\{Wv^0, z_m\}]$ such that $\hat{\mathbf{p}}^{z^*}$ defined by (4.3) is a feasible solution to the problem (4.2). Our proposed algorithm for finding z^* contains two phases. In the first phase, the algorithm looks for an interval $(z_{k-1}, z_k]$, $k = 1, 2, \ldots, m$, such that the optimal objective value z^* belongs to it. In the second phase, the algorithm calculates exactly the value z^* by solving a minimum ratio cut problem.

In the first phase, the algorithm searches an index $k \in \{1, 2, \ldots, m\}$ such that $\hat{\mathbf{p}}^{z_k}$ is a feasible solution to the problem (4.2) while $\hat{\mathbf{p}}^{z_{k-1}}$ is not feasible where $\hat{\mathbf{p}}^{z_{k-1}}$ and $\hat{\mathbf{p}}^{z_k}$ are defined by (4.3) for $z = z_{k-1}$ and $z = z_k$, respectively. The fact that $\hat{\mathbf{p}}^{z_k}$ is feasible implies that $z^* \le z_k$ and based on Theorem 4.2, the infeasibility of $\hat{\mathbf{p}}^{z_{k-1}}$ implies that $z^* > z_{k-1}$. Then, $z_{k-1} < z^* \le z_k$. In order to check feasibility of $\hat{\mathbf{p}}^z$ for $z = z_{k-1}, z_k$, it suffices to solve the maximum flow problem in $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^z)$. If the maximum flow value is greater than or equal to v^0, then $\hat{\mathbf{p}}^z$ is feasible and otherwise, it is not feasible (see Lemma 4.1). Therefore, the algorithm looks for one index $k \in \{1, 2, \ldots, m\}$ such that the maximum flow value in $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^{z_k})$ is greater than or equal to v^0 and the maximum flow value in $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^{z_{k-1}})$ is less than v^0. The algorithm uses the binary search technique to find such the index k.

Hereafter, we describe the second phase. We recall that in the first phase, the algorithm has found one index $k^* \in \{1, 2, \ldots, m\}$ such that the optimal objective value z^* belongs to $(z_{k^*-1}, z_{k^*}]$. Thus we can restrict ourselves to feasible solutions with objective value $z \in (z_{k^*-1}, z_{k^*}]$ to find the optimal objective value. For any value $z \in (z_{k^*-1}, z_{k^*}]$, the vector $\hat{\mathbf{p}}^z$ defined by (4.3) is obtained directly from

$$\hat{p}^z_{ij} = \begin{cases} \frac{z}{w_{ij}} & (i,j) \in A_{k^*}, \\ p_{ij} & (i,j) \in A \backslash A_{k^*}, \end{cases} \tag{4.4}$$

where $A_{k^*} = \{(i,j) \in A : w_{ij} p_{ij} \ge z_{k^*}\}$.
Notice that the optimal objective value z^* of the problem (4.2) is the minimum value of $z \in (z_{k^*-1}, z_{k^*}]$ so that the network $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^z)$ can send v^0 units of flow from s to t. Consequently, z^* is the optimal value of the following problem.

$$\min z \tag{4.5a}$$

$$\text{s.t.} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} v^0 & i = s, \\ 0 & i \ne s, t, \\ -v^0 & i = t, \end{cases} \quad \forall i \in V, \tag{4.5b}$$

$$x_{ij} - \frac{1}{w_{ij}} z \le u_{ij} \quad \forall (i,j) \in A_{k^*}, \tag{4.5c}$$

$$x_{ij} \le u_{ij} + p_{ij} \quad \forall (i,j) \in A \backslash A_{k^*}, \tag{4.5d}$$

$$z \ge 0, x_{ij} \ge 0 \quad \forall (i,j) \in A. \tag{4.5e}$$

The problem (4.5) is to find the minimum value of z so that the network $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^z)$ can send v^0 units of flow from s to t. The first phase guarantees that the optimal value z^* belongs to $(z_{k^*-1}, z_{k^*}]$. Hence, one can

apply the binary search technique to find the optimal value z^*. Here, we will use an other procedure which has less complexity rather than the binary search technique.

For computing z^*, we will solve the dual of the problem (4.5) because the objective values of both the problems are the same (see the fundamental theorem of duality in [4]). We associate the dual variable π_i with the ith constraint in (4.5b). Weassociate the dual variable α_{ij}, $(i,j) \in A_{k^*}$, with each constraint (4.5c) and also the dual variable α_{ij}, $(i,j) \in A \backslash A_{k^*}$, with each constraint (4.5d). Using these notations, the dual of the problem (4.5) is

$$\begin{aligned} \max\ w = & \sum_{(i,j)\in A_{k^*}} u_{ij}\alpha_{ij} + \sum_{(i,j)\in A\backslash A_{k^*}} (u_{ij}+p_{ij})\alpha_{ij} + v^0\pi_s - v^0\pi_t, \\ \text{s.t. } & \pi_i - \pi_j + \alpha_{ij} \le 0 \quad \forall (i,j) \in A, \\ & \sum_{(i,j)\in A_{k^*}} -\frac{1}{w_{ij}}\alpha_{ij} \le 1, \\ & \alpha_{ij} \le 0, \quad \forall (i,j) \in A. \end{aligned} \tag{4.6}$$

This problem is a minimum cut problem with one additional constraint as well as two additional terms in the objective function. In the following, we will show that the problem is reduced to a maximum ratio cut problem and then, present an optimal solution to it.

Without loss of generality, we can assume that $\pi_t = 0$ because if (π, α) is a feasible solution to the problem (4.6), then for any real number d, $(\pi + d, \alpha)$ is also feasible with the same objective value. By introducing nonnegative variable $\hat{\alpha}_{ij} = -\alpha_{ij}$ for every $(i,j) \in A$, the problem (4.6) is converted into the following problem.

$$\max\ w = -\sum_{(i,j)\in A_{k^*}} u_{ij}\hat{\alpha}_{ij} - \sum_{(i,j)\in A\backslash A_{k^*}} (u_{ij}+p_{ij})\hat{\alpha}_{ij} + v^0\pi_s \tag{4.7a}$$

$$\text{s.t. } \pi_i - \pi_j \le \hat{\alpha}_{ij} \quad \forall (i,j) \in A, \tag{4.7b}$$

$$\sum_{(i,j)\in A_{k^*}} \frac{1}{w_{ij}}\hat{\alpha}_{ij} \le 1, \tag{4.7c}$$

$$\pi_t = 0,\ \hat{\alpha}_{ij} \ge 0, \quad \forall (i,j) \in A. \tag{4.7d}$$

In the following, we show that the problem (4.7) is equivalent to a minimum ratio cut problem.

Lemma 4.6. *For each feasible solution $(\hat{\alpha}^0, \pi^0)$ to the problem (4.7), the set $\{(i,j) \in A : \hat{\alpha}^0_{ij} > 0\}$ contains an st-cut.*

Proof. By contradiction, assume that the set $\{(i,j) \in A : \hat{\alpha}^0_{ij} > 0\}$ does not contain any st-cut. Then, there exists a path P from s to t so that $\hat{\alpha}^0_{ij} = 0$ for each $(i,j) \in P$. This together with the constraints (4.7b) imply that $\pi_s \le 0$. Consequently, $(\hat{\alpha}, \pi) = (0,0)$ is an optimal solution to the problem (4.7) whose optimal value equals to zero. This contradicts Assumption 3.1. □

By Lemma 4.6, there exists at least one optimal solution $(\hat{\alpha}^*, \pi^*)$ to the problem (4.7) so that $C^* = \{(i,j) \in A : \hat{\alpha}^*_{ij} > 0\}$ is an st-cut. For this reason, we henceforth restrict ourself to solutions $(\hat{\alpha}, \pi)$ with the property that

- $\hat{\alpha}_{ij} > 0$ for every $(i,j) \in C$,
- $\hat{\alpha}_{ij} = 0$ for every $(i,j) \in A \backslash C$,

for some $st-$cut C.

Consider the network $G'(V, A, \mathbf{u}')$ where $\mathbf{u}'$ is defined as follows:

$$u'_{ij} = \begin{cases} u_{ij} & (i,j) \in A_{k^*}, \\ u_{ij} + p_{ij} & (i,j) \in A \backslash A_{k^*}. \end{cases} \tag{4.8}$$

Suppose that $(\hat{\alpha}^0, \pi^0)$ is a feasible solution to the problem (4.7) and $C = [S, \bar{S}] = \{(i,j) : \hat{\alpha}^0_{ij} > 0\}$ is the corresponding st-cut. For each $(i,j) \in C$, there exists at least one st-path P containing (i,j). By summing the constraints (4.7b) corresponding to arcs of P, we have $\pi_s = \pi_s - \pi_t \leq u'_{ij}$. Thus,

$$\pi_s \leq \min\{u'_{ij} : (i,j) \in C\}. \tag{4.9}$$

On the other hand, by substituting $\hat{\alpha}_{ij} = 0$ into the constraints (4.7b), we have

$$\pi_i \leq \pi_j, \quad \forall (i,j) \in A \backslash C. \tag{4.10}$$

Using inequalities (4.9) and (4.10), one can simply show that there is at least an optimal solution to the problem (4.7) so that the variables $\hat{\alpha}_{ij}$, $(i,j) \in C$, and π_i, $i \in S$, have the same value of π_s while the variables $\hat{\alpha}_{ij}$, $(i,j) \in A \backslash C$, and π_i, $i \in \bar{S}$, are zero. Let $\pi_s = \gamma$. Based on the above argument, we define a solution $(\hat{\alpha}, \pi)$ as follows:

$$\hat{\alpha}_{ij} = \begin{cases} \gamma & (i,j) \in C, \\ 0 & (i,j) \notin C, \end{cases} \quad \forall (i,j) \in C, \tag{4.11}$$

$$\pi_i = \begin{cases} \gamma & i \in S, \\ 0 & i \in \bar{S}, \end{cases} \quad \forall i \in V, \tag{4.12}$$

for some st-cut $C^* = [S, \bar{S}]$. Obviously, a solution defined by (4.11) and (4.12) satisfies the constraints (4.7b). Substituting the solution in the problem (4.7), we have

$$\begin{aligned} &\max w = \gamma \left(v^0 - \sum_{(i,j) \in C} u'_{ij} \right), \\ &\gamma \left(\sum_{(i,j) \in A_{k^*} \cap C} \frac{1}{w_{ij}} \right) \leq 1, \\ &\gamma \geq 0, \end{aligned} \tag{4.13}$$

in which an st-cut C and a value γ are to be determined. Based on Assumption 3.1, $\gamma > 0$. On the other hand, any st-cut C with $\mathbf{u}'(C) < v^0$ contains at least one element of A_{k^*} because if not, the problem (4.13) has an infinite optimal value and consequently, the problem (4.5) is infeasible which leads to a contradiction. Then, we can set $\gamma = (\sum_{(i,j) \in A_{k^*} \cap C} \frac{1}{w_{ij}})^{-1}$ and reduce the problem (4.13) to the following problem.

$$\max_C w = \frac{v^0 - \sum_{(i,j) \in C} u'_{ij}}{\sum_{(i,j) \in A_{k^*} \cap C} \frac{1}{w_{ij}}}. \tag{4.14}$$

This problem is a special case of linear fractional combinatorial (LFC) problems which is called the maximum ratio cut problem. Radzik [20] proposed a type Newton method to solve any LFC problem in strongly polynomial

Algorithm 4.7. Algorithm to solve the problem (4.1).

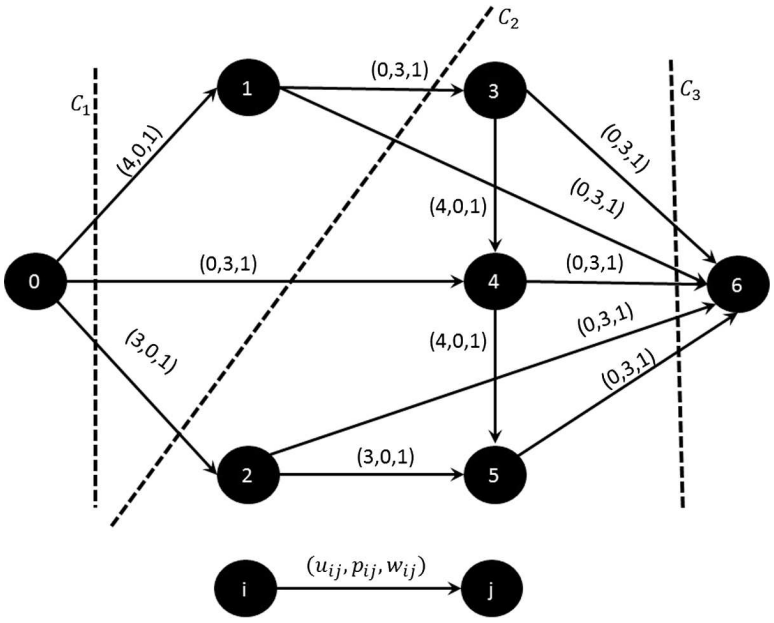
Input: A network $G(V, A, \mathbf{u}, \mathbf{p}, \mathbf{w})$ with two specific nodes s and t and a prescribed value v^0.
Output: A new capacity vector $\hat{\mathbf{u}}$.
Phase I:
Sort the values of $\{w_{ij}p_{ij} : (i,j) \in A\} \cup \{0\}$ in nondecreasing order and let $z_0 = 0 \leq z_1 \ldots \leq z_m$ be the sorted list. Obtain the maximum flow value v in $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^{z_m})$.
if $v < v^0$ **then**
  Stop because the problem (4.1) is infeasible.
end if
Obtain the maximum flow value v in $G(V, A, \mathbf{u})$.
if $v \geq v^0$ **then**
  Stop because the problem (4.1) has the trivial optimal solution $\hat{\mathbf{u}} = \mathbf{u}$.
end if
Set $k_L = 0$ and $k_U = m$.
while $k_U - k_L > 1$
  Set $k = [\frac{k_L + k_U}{2}]$.
  Obtain the maximum flow value v in $G(V, A, \mathbf{u} + \hat{\mathbf{p}}^{z_k})$.
  if $v < v^0$ **then**
    $k_L = k$.
  else
    $k_U = k$.
  end if
end while
Phase II:
Set $\epsilon = 10^{-10}$.
Set $w = z_{k_L}$.
Obtain $\mathbf{u}'$ using (4.8).
while True **do**
  Obtain $\mathbf{u}^{(w)}$ using (4.15).
  Find a minimum cut C in $G(V, A, \mathbf{u}^{(w)})$ with capacity v.
  if $|v - v^0| < \epsilon$ **then**
    Stop because the optimal solution $\hat{\mathbf{u}} = \mathbf{u}^{(w)}$ is found.
  else
    Set $w = \frac{v^0 - \sum_{(i,j)\in C} u'_{ij}}{\sum_{(i,j)\in C \cap A_{k^*}} \frac{1}{w_{ij}}}$.
  end if
end while

time. Therefore, we apply this method in the second phase to solve the problem in strongly polynomial time. Based on the result obtained from the first phase, we know that the optimal value w^* of the problem (4.14) is in $(z_{k^*-1}, z_{k^*}]$. In the second phase, the algorithm generates an increasing sequence $\{w_l\}$ converging to w^* by starting the initial point $w_0 = z_{k^*-1}$. Suppose that the algorithm has calculated the lth term of the sequence, that is, w_l. The algorithm finds the minimum cut C^* in the network $G(V, A, \mathbf{u}^{(w_l)})$ where the capacity vector $\mathbf{u}^{(w_l)}$ is defined as follows:

$$u_{ij}^{(w_l)} = \begin{cases} u'_{ij} + \frac{w_l}{w_{ij}} & (i,j) \in A_{k^*}, \\ u'_{ij} & (i,j) \in A \backslash A_{k^*}, \end{cases} \qquad \forall (i,j) \in A. \tag{4.15}$$

Algorithm 4.8. An implementation of Algorithm 4.7 based on the binary search in the second phase.

Input: A network $G(V, A, \mathbf{u}, \mathbf{p}, \mathbf{w})$ with two specific nodes s and t and a prescribed value v^0.
Output: A new capacity vector $\hat{\mathbf{u}}$.
Phase I:
This fuzz is the same first fuzz of Algorithm 4.7.
Phase II:
Set $\epsilon = 10^{-10}$.
Set $w_L = z_{k_L}, w_U = z_{k_U}$.
while True **do**
  Set $w = \frac{w_L + w_U}{2}$.
  Obtain $\mathbf{u}^{(w)}$ using (4.15).
  Obtain the maximum flow value v in $G(V, A, \mathbf{u}^{(w)})$.
  if $|v - v^0| < \epsilon$ **then**
    Stop because the optimal solution $\hat{\mathbf{u}} = \mathbf{u}^{(w)}$ is found.
  else if $v < v^0$ **then**
    Set $w_L = w$.
  else
    Set $w_U = w$.
  end if
end while



Iter.	Value of w	Minimum cut	Maximum flow value
1	0.00	C_3	0
2	2.00	C_2	9
3	2.33	C_1	9.33
4	3.00	C_1	10

FIGURE 2. An instance of the problem (4.1) with $v^0 = 10$ as well as the results obtained from the phase II of Algorithm 4.7 for solving the problem.

TABLE 1. Average performance statistics of Algorithms 4.7 and 4.8 for the binomial graphs with $p = \frac{1}{2}$.

n	Algorithm 4.7						Algorithm 4.8					
	m	Phase I		Phase II		Total time	m	Phase I		Phase II		Total time
		Time	Num. of iter.	Time	Num. of iter.			Time	Num. of iter.	Time	Num. of iter.	
50	618.10	0.10	9.40	0.04	2.00	0.13	625.60	0.10	9.30	0.39	35.60	0.49
100	2480.90	0.53	11.40	0.19	2.00	0.72	2480.50	0.47	11.40	1.45	35.50	1.92
250	15526.40	4.44	13.80	1.27	2.00	5.71	15556.00	3.99	14.00	10.21	35.80	14.20
500	62375.40	18.63	16.00	4.99	2.00	23.62	62365.60	18.32	16.00	40.55	34.80	58.87

TABLE 2. Average performance statistics of Algorithms 4.7 and 4.8 for the binomial graphs with $n = 100$.

p	Algorithm 4.7						Algorithm 4.8					
	m	Phase I		Phase II		Total time	m	Phase I		Phase II		Total time
		Time	Num. of iter.	Time	Num. of iter.			Time	Num. of iter.	Time	Num. of iter.	
0.10	489.10	0.07	9.10	0.03	2.00	0.10	487.80	0.07	9.00	0.32	43.00	0.39
0.20	994.10	0.18	10.00	0.07	2.00	0.24	987.50	0.17	9.80	0.67	41.70	0.84
0.30	1486.80	0.30	10.80	0.11	2.00	0.40	1501.70	0.28	10.90	1.10	41.40	1.38
0.40	1972.00	0.41	10.90	0.15	2.00	0.56	1957.70	0.36	10.90	1.40	41.30	1.76
0.50	2457.90	0.53	11.20	0.18	2.00	0.71	2461.70	0.50	11.20	1.93	40.80	2.43
0.60	2961.90	0.61	11.90	0.21	2.00	0.82	2971.70	0.61	11.70	2.13	40.70	2.74
0.70	3464.30	0.65	11.80	0.22	2.00	0.87	3450.60	0.67	11.80	2.45	40.40	3.12
0.80	3956.60	0.76	11.90	0.26	2.00	1.01	3958.50	0.77	12.00	2.61	40.00	3.38
0.90	4460.90	0.87	12.20	0.29	2.00	1.16	4464.20	0.86	12.10	2.88	38.80	3.74
1.00	4950.00	0.94	12.30	0.31	2.00	1.26	4950.00	0.99	12.10	3.26	38.40	4.25

In this state, two distinct cases may occur:

Case I ($\mathbf{u^{(w_1)}(C^*) = v^0}$): In this case, we have

$$w_l = \frac{v^0 - \sum_{(i,j)\in C^*} u'_{ij}}{\sum_{(i,j)\in A_{k^*}\cap C^*} \frac{1}{w_{ij}}} \geq \frac{v^0 - \sum_{(i,j)\in C} u'_{ij}}{\sum_{(i,j)\in A_{k^*}\cap C} \frac{1}{w_{ij}}}$$

for each st-cut C. Then, w_l is the optimal value of the problem (4.14).

Case II ($\mathbf{u^{(w_1)}(C^*) < v^0}$): In this case, $\frac{v^0-\sum_{(i,j)\in C^*} u'_{ij}}{\sum_{(i,j)\in A_{k^*}\cap C^*} \frac{1}{w_{ij}}} > w_l$. Consequently, w_l is a lower bound on the optimal value. Then the algorithm sets $w_{l+1} = \frac{v^0-\sum_{(i,j)\in C^*} u'_{ij}}{\sum_{(i,j)\in A_{k^*}\cap C^*} \frac{1}{w_{ij}}}$. Notice that $w_l < w_{l+1} \leq w^*$.

It is notable that the case that $\mathbf{u}^{(w_l)}(C^*) > v^0$ never occur because the inequality $\mathbf{u}^{(w_0)}(C^*) < v^0$ is guaranteed from the first phase and if $\mathbf{u}^{(w_l)}(C^*) > v^0$ for $l \geq 1$, then $\mathbf{u}^{(w_l)}(C) > v^0$ for every st-cut C and consequently, $w_l > \frac{v^0-\sum_{(i,j)\in C} u'_{ij}}{\sum_{(i,j)\in A_{k^*}\cap C} \frac{1}{w_{ij}}}$ for every st-cut C which contradicts the fact that w_l is constructed in the preceding iteration by $w_l = \frac{v^0-\sum_{(i,j)\in C^0} u'_{ij}}{\sum_{(i,j)\in A_{k^*}\cap C^0} \frac{1}{w_{ij}}}$ for some st-cut C^0.

Our proposed algorithm is formally stated in Algorithm 4.7. Radzik [20] proved that the discrete type Newton method solves any LFC problem in strongly polynomial time. Thus, the following result is immediate.

Theorem 4.9. *Algorithm 4.7 solves the problem (4.1) in strongly polynomial time.*

5. Experimental results

We recall that for solving the problem (4.1), Algorithm 4.7 performs two phases. The first phase finds an interval $(z_k, z_{k+1}]$ containing the optimal value. The second phase uses the Newton method to solve the dual of the problem (4.5) for obtaining the optimal value of the problem (4.1). A straight idea is to apply the binary search technique in the second phase for solving the problem (4.5) (see Algorithm 4.8). In this section, we have conducted a computational study to observe the performance of these two algorithms.

The following computational tools were used to develop algorithms: Python 2.7.5, Matplotlib 1.3.1 and NetworkX 1.8.1. All computational experiments were conducted on a 32-bit Windows 10 with Processor Intel(R) Core(TM) i5 $-$ 3210M CPU @2.50GHz and 4 GB of RAM.

For computational experiments, we have used a special class of undirected random graphs, called binomial graphs, which are first introduced by Paul Erdõs and Alfréd Rényi [13] in 1959. The binomial graphs are generated by determining two parameters n and p which n is the number of nodes and any edge (i, j) is included in the graph with the probability $p \in [0, 1]$. The parameter p can be thought of as a weighting function; as p increases from 0 to 1, the graph becomes more dense. In particular, the case $p = 1$ corresponds to a complete graph. In experiments, to construct a directed graph $G(V, A)$, an undirected binomial graph is first generated and then, any edge (i, j) with $i < j$ is directed from i to j. In generated instances, we have supposed that the source is node 1 and the sink is node n. We have used random data generated using a uniform random distribution as follows:

$$\begin{aligned}
&u_{ij} \sim U(0, n) \quad \forall (i, j) \in A,\\
&p_{ij} \sim U(0, n) \quad \forall (i, j) \in A,\\
&w_{ij} \sim U(0, n) \quad \forall (i, j) \in A,\\
&v^0 \sim U(v^*, v^{**}),
\end{aligned}$$

in which v^* and v^{**} are respectively the maximum flow value in $G(V, A, \mathbf{u})$ and $G(V, A, \mathbf{u} + \mathbf{p})$. The condition $v^* \leq v^0 \leq v^{**}$ guarantees that the problem (4.1) is feasible and has not the trivial optimal solution $\hat{\mathbf{u}} = \mathbf{u}$.

We have tested the algorithms on five classes of networks which differ from the number of nodes, varying from 50 to 500 (see Tab. 1) and also, differ from the probability value p, varying from 0.1 to 1.0 (see Tab. 2). There are 10 random instances generated for each class of graphs. Tables 1 and 2 present average performance statistics of the algorithms. The experimental results illustrate that the performance of Algorithm 4.7 is significantly better than Algorithm 4.8.

An important observation of experiments is that the number of iterations of Algorithm 4.7 is even equal to 2. In spite of this observation, unfortunately, this is not a general result. Figure 2 provides us with a counterexample to this observation.

6. Conclusion

In this paper, we studied the reverse maximum flow problem under the weighted Chebyshev distance and proposed an efficient algorithm to solve the problem in two phases. The first phase uses the binary search technique to find an interval containing the optimal value and the second phase applies the discrete type Newton method to obtain the optimal value. As a result from Radzik's paper [20], our proposed algorithm solves the problem in strongly polynomial time. Furthermore, the experimental results show that the performance of the algorithm is better than one which uses the binary search in the second phase.

As future works, we propose studying the problem under other distances and its extension on hypergraphs.

Acknowledgements. The first author would like to thank Imam Ali University for granting this research.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows. Prentice-Hall, Englewood Cliffs (1993).
[2] K. Ahuja and J.B. Orlin, Inverse optimization. *Oper. Res.* **49** (2001) 771–783.
[3] K. Ahuja and J.B. Orlin, Combinatorial algorithms for inverse network flow problems. *Networks* **40** (2002) 181–187.
[4] M.S. Bazaraa, J. Jarvis and H.D. Sherali, Linear Programming and Network Flows. John Wiley & Sons (2011).
[5] D. Burton and Ph.L. Toint, On an instance of the inverse shortest paths problem. *Math. Program.* **53** (1992) 45–61.
[6] D. Burton and Ph.L. Toint, On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Math. Program.* **63** (1994) 1–22.
[7] M. Call, Inverse Shortest Path Routing Problems in the Design of IP Networks. Department of Mathematics Linkping University, Sweden (2010).
[8] T. Cui and D.S. Hochbaum, Complexity of some inverse shortest path lengths problems. *Networks* **56** (2010) 20–29.
[9] A. Deaconu, The inverse maximum flow problem considering l_∞ norm. *RAIRO: OR* **42** (2008) 401–414.
[10] A. Deaconu, The inverse maximum flow problem with lower and upper bounds for the flow. *Yugosl. J. Oper. Res.* **18** (2008) 13–22.
[11] M. Demange and J. Monnot, An introduction to inverse combinatorial problems, in Paradigms of Combinatorial Optimization (Problems and New Approaches), edited by V.Th. Paschos. Wiley, London, Hoboken (2010).
[12] C.W. Duin and A. Volgenant, Some inverse optimization problems under the Hamming distance. *Eur. J. Oper. Res.* **170** (2006) 887–899.
[13] P. Erds and A. Rnyi, On random graphs I. *Publ. Math.* **6** (1959) 290–297.
[14] S.P. Fekete, W.H. Hochstattler, S. Kromberg and C. Moll, The complexity of an inverse shortest paths problem, in Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future, edited by R.L. Graham, J. Kratochvil, J. Nesetrol and F.S. Roberts. American Mathematical Society DIMATIA-DIMACS Conference, Stirin Castle, Czech Republic (1997) 49–113.
[15] L.R. Ford and D.R. Fulkerson, Maximal flow through a network. *Can. J. Math.* **8** (1956) 399–404.
[16] C. Heuberger, Inverse optimization: a survey on problems, methods, and results. *J. Comb. Optim.* **8** (2004) 329–336
[17] S.O. Krumke, H. Noltemeier, S. Schwarz, H.-C. Wirth and R. Ravi, *Operations Research Proceedings 1998 – Selected Papers of the International Conference on Operations Research, Zurich, August 31–September 3, 1998* (1998) 158–167.
[18] L.C. Liu and J.Z. Zhang, Inverse maximum flow problems under the weighted Hamming distance. *J. Comb. Optim.* **12** (2006) 395–408.
[19] L. Liu, Inverse Maximum Flow Problems under the Combining Norms, edited by M. Fellows, X. Tan, and B. Zhu. In: FAW-AAIM, Vol. 7924 of *Lecture Notes in Computer Science* (2013) 221–230.
[20] T. Radzik, Parametric flows, weighted means of cuts, and fractional combinatorial optimization, in Complexity in Numerical Optimization, edited by P.M. Pardalos. World Scientific Publishing Co. (1993) 351–386.

[21] J. Tayyebi and M. Aman, Efficient algorithms for the reverse shortest path problem on trees under the Hamming distance. *Yugosl. J. Oper. Res.* **27** (2017) 46–60.
[22] J. Tayyebi and M. Aman, On inverse linear programming problems under the bottleneck-type weighted Hamming distance. *Discret. Appl. Math.* **240** (2018) 92–10.
[23] C. Yang, J. Zhang and Z. MA, Inverse maximum flow and minimum cut problems. *Optimization* **40** (1997) 147–170.
[24] J. Zhang, Z. Liu and Z. Ma, Some reverse location problem. *Eur. J. Oper. Res.* **124** (2000) 77–88.
[25] J. Zhang, X. Yang and M. Cai, Reverse Center Location Problem. Vol. 1741 of *Lecture Notes in Computer Science* (1999) 279–294.
[26] B.W. Zhang, J.Z. Zhang and L.Q. Qi, The shortest path improvement problems under Hamming distance. *J. Comb. Optim.* **12** (2006) 351–361.
[27] J.Z. Zhang and Y.X. Lin, Computation of the reverse shortest path problem. *J. Glob. Optim.* **25** (2003) 243–261.