Capacity Inverse Minimum Cost Flow Problem

Çiğdem Güler[†*] and Horst W. Hamacher[†]
Department of Mathematics, University of Kaiserslautern
67653 Kaiserslautern, Germany

November 26, 2007

Abstract

Given a directed graph $G = (N, A)$ with arc capacities u_{ij} and a minimum cost flow problem defined on G, the *capacity inverse minimum cost flow problem* is to find a new capacity vector $\hat{u}$ for the arc set A such that a given feasible flow $\hat{x}$ is optimal with respect to the modified capacities. Among all capacity vectors $\hat{u}$ satisfying this condition, we would like to find one with minimum $\|\hat{u} - u\|$ value.

We consider two distance measures for $\|\hat{u} - u\|$, rectilinear (L_1) and Chebyshev (L_∞) distances. By reduction from the feedback arc set problem we show that the capacity inverse minimum cost flow problem is $\mathcal{NP}$-hard in the rectilinear case. On the other hand, it is polynomially solvable by a greedy algorithm for the Chebyshev norm. In the latter case we propose a heuristic for the bicriteria problem, where we minimize among all optimal solutions the number of affected arcs. We also present computational results for this heuristic.

Keywords: inverse problems, network flows, minimum cost flows

1 Introduction

In the past few decades, optimization problems with estimated problem parameters have drawn considerable attention from researchers. For this kind of problems one often knows a priori an optimal solution based on observations or experiments, but is interested in finding a set of parameters, such that the known solution is optimum (i) and the deviation from the initial estimates is minimized (ii). The problem of recalculating the parameters satisfying (i) and (ii) is known as *inverse optimization problem*. In their paper, Ahuja and Orlin [2] mention that the major application area for inverse optimization is geophysical sciences and it were, indeed, geophysicists to first study such problems. At the begining of 90's, a well-known study by Burton and Toint [6, 7] attracted the interest of mathematicians to this topic. In their papers, the authors study the inverse shortest path problems used to predict the movements of earthquakes.

Heuberger [13] published a first thorough survey on inverse problems. He describes several classes of inverse problems in detail and reviews solution methods proposed in the literature.

In our paper, we consider a class of inverse network flow problems which has so far not been treated in the literature, but seems to have some potential in applications. It is defined on a directed

*Author of correspondence: gueler@mathematik.uni-kl.de

†The research has been partially supported by the Deutsche Forschungsgemeinschaft (DFG), Grant HA 1737/7 "Algorithmik großer und komplexer Netzwerke" and by the Rheinland-Pfalz cluster of excellence "Dependable adaptive systems and mathematical modeling".

graph $G = (N, A)$ with node set N and arc set A. The arcs in the graph have flow capacities $u_{ij} \geq 0$ and costs $c_{ij} \geq 0$, the nodes have supply or demands of value $b(i)$. The well-known [1] LP formulation of the minimum cost flow problem is

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1a}$$

subject to

$$\sum_{j\in N^+(i)} x_{ij} - \sum_{j\in N^-(i)} x_{ji} = b(i) \qquad \forall i \in N \tag{1b}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in A \tag{1c}$$

Here, $N^+(i)$ and $N^-(i)$ is the set of nodes adjacent from and to node i, respectively.

Minimum cost flow problems have already been considered in the context of inverse optimization by several authors. In the *(cost) inverse minimum cost flow problem*, the initial cost vector c is replaced by $\hat{c}$ such that a given feasible network flow $\hat{x}$ is optimal with respect to cost $\hat{c}$. Zhang and Liu [20] use a linear programming approach to analyze the inverse minimum cost flow problem under unit weight L_1 norm. Ahuja and Orlin [2] show that the inverse problem of a linear program is also a linear program and analyze the min cost flow problem as special case. If the change of cost is measured by the L_1 norm the inverse problem reduces to a unit-capacity minimum cost flow problem. For the L_∞ norm the inverse problem turns out to be solvable as a minimum mean cycle problem. In the case of weighted objective functions a minimum cost-to-time ratio cycle problem has to be solved.

Zhang and Liu [21] propose an optimization model for general inverse problems and show that most of the combinatorial problems can be fit into this model as special cases. They also suggest a Newton-type algorithm for their model under L_∞ norm.

To the best of our knowledge, until now only changes of the cost function have been considered for min cost flows. Capacity modifications were examined for minimum cut problems by Ahuja and Orlin [3]. They use combinatorial arguments to prove that the inverse minimum cut problem under L_1 norm can be efficiently solved using maximum flow computations in the graph. For the Chebyshev norm, the inverse problem requires solving a polynomial sequence of minimum cut problems. In a recent paper, Zhang and Liu [19] analyze the inverse maximum flow problem under the weighted Hamming distance and propose strongly polynomial algorithms.

Our goal is to close the gap between the capacity perturbing inverse problems and the cost perturbing ones by analyzing the inverse minimum cost flow problem, in which only the arc capacities are changed. We call this problem *capacity inverse minimum cost flow problem* in order to distinguish it from the inverse minimum cost flow problem where the cost vector is changed. In general, we use denotations IMCF_u and IMCF_c for the capacity and cost inverse case, respectively, but mostly drop the capacity index subsequently, since we only deal with this class of problems.

The practical motivation to work on IMCF_u arises from radiation therapy planning where we would like to find a therapy plan that minimizes the underdosage of cancerous tissue and the overdosage of healthy organs [12]. In order to model the radiation problem as a network flow problem, we interpret the voxels as arcs of some graph. The flows on these arcs represent the doses deposited in the voxels where lower and upper bounds on the dosage is modeled as lower and upper capacities. Then, minimizing the overdosage and the underdosage on the voxels is nothing but minimizing the change in the arc capacities. Due to the large number of voxels, the resulting network model of the radiation problem is very large scale and complex.

The rest of this paper is organized as follows: Section 2 describes the capacity inverse minimum cost flow problem in detail and gives a combinatorial formulation of the problem. Section 3 analyzes the problem and its complexity under the rectilinear norm L_1. In Section 4 we present a polynomial algorithm to solve the inverse problem under L_∞ norm and propose a heuristic for the bicriteria problem, where we minimize among all optimal solutions the number of affected arcs. In Section 5 we discuss the results of computational experiments for the proposed heuristic and conclude the paper in the last section with a summary of our results and a discussion of current and future work in this area.

2 Problem Definition

In this section, our aim is to propose a mathematical formulation for the capacity inverse minimum cost flow problem. For a given flow network and a feasible integer flow $\hat{x}$, the capacity inverse problem can be described as follows:

$$\begin{aligned} \min \quad & \|(u - \hat{u})\| && \text{(2a)} \\ \text{subject to} \quad & \\ & \hat{x}_{ij} \leq \hat{u}_{ij} \quad \forall (i,j) \in A && \text{(2b)} \\ & \hat{x} \text{ optimal min cost flow} && \text{(2c)} \\ & \text{with respect to capacity } \hat{u} \end{aligned}$$

Note that $\hat{x}$ and u are part of the data while $\hat{u}$ is the vector of variables to be determined.

In order to model this inverse problem mathematically, it is first necessary to replace the verbal formulation (2c) by a formal one. To achieve this, we use the following well-known optimality condition of minimum cost flow problems [1].

Negative Cycle Property: The flow $\hat{x}$ is an optimal flow of the minimum cost flow problem if and only if the corresponding residual graph $G(\hat{x}, u)$ does not contain any negative (cost) cycles.

As a consequence of the Negative Cycle Property, we know that the residual graph $G(\hat{x}, u)$ corresponding to the given feasible solution $\hat{x}$ for arc capacities u contains negative cost cycles unless $\hat{x}$ is already optimal. Hence, another interpretation of the capacity inverse problem would be to destroy the negative cycles in the residual graph $G(\hat{x}, u)$ by perturbing the arc capacities in the original graph $G = (N, A)$.

If we investigate the effects of changing the capacities of arcs in the initial graph G onto the residual graph $G(\hat{x}, u)$, we observe that there are 3 alterations that can occur in the residual graph:

1. *A new arc can be added to $G(\hat{x}, u)$:* If we increase the capacity of an arc from $u_{ij} = \hat{x}_{ij}$ to $\hat{u}_{ij} > u_{ij}$ then we create a new arc from node i to node j in $G(\hat{x}, u)$ with capacity $(\hat{u}_{ij} - \hat{x}_{ij})$.

2. *An existing arc can be deleted from $G(\hat{x}, u)$:* This takes place by decreasing the capacity $u_{ij} > \hat{x}_{ij}$ to $\hat{u}_{ij} = \hat{x}_{ij}$.

3. *The residual capacity of an already existing arc can be changed without deleting*

Proposition 1. *A negative cycle in the residual graph $G(\hat{x}, u)$ can be destroyed if and only if an existing arc is deleted from $G(\hat{x}, u)$, i.e. the capacity of an arc (i, j) in the original graph G is set to its flow value $\hat{x}_{ij}$.*

Proof: In order to eliminate the negative cycles in $G(\hat{x}, u)$, we have to either change the costs in G or break these cycles. Since we are not allowed to modify the cost function, the only possibility remaining is to break the cycles, which has to be achieved through capacity modifications. As mentioned above, changing the capacities of the arcs in the original graph can lead to 3 alterations in the residual graph. Obviously adding a new arc to $G(\hat{x}, u)$ or changing the capacity of an existing arc (i, j) to $\hat{u}_{ij} \neq 0$ does not break any cycles. Hence, the only way is to delete an arc from $G(\hat{x}, u)$, which is attained by setting the capacity of the arc to $\hat{x}_{ij}$.

□

Using Proposition 1, IMCF under L_1 norm (or L_∞ norm) can be reformulated as choosing a set of arcs A_D to be deleted from the residual graph $G(\hat{x}, u)$ such that $G(\hat{x}, u)$ is free of negative cycles. The objective is to find $A_D \subseteq A(\hat{x}, u)$ such that $\sum_{(i,j)\in A_D} (u_{ij} - \hat{x}_{ij})$ (or $\max_{(i,j)\in A_D} (u_{ij} - \hat{x}_{ij})$) is minimized.

While studying IMCF, the first question that arises is the feasibility of the problem. We can formulate the feasibility condition for the capacity inverse problem as follows:

Lemma 2. *There does not exist a feasible solution to the capacity inverse minimum cost flow problem if and only if there exists a cycle C in G with positive flows on the cycle arcs and a positive sum of arc costs, i.e. $\hat{x}_{ij} > 0 \quad \forall (i, j) \in C$ and $\sum_{(i,j)\in C} c_{ij} > 0$.*

Proof: "⇐" The reverse cycle $\bar{C}$ of C is contained in the residual graph $G(\hat{x}, u)$ and is a negative cycle with capacities equal to the flow values $\hat{x}_{ij}$. By Proposition 1, any feasible solution $\hat{u}$ of the inverse problem has to break the negative cycles in $G(\hat{x}, u)$. However, independent of the choice of $\hat{u}$ the reverse cycle $\bar{C}$ remains in the residual graph since $\hat{u}_{ij} \geq \hat{x}_{ij}$.
"⇒" If there does not exist a feasible solution for the inverse problem, then for all capacities $\hat{u}$ (such that $\hat{x}$ is feasible) the residual graph $G(\hat{x}, \hat{u})$ of $\hat{x}$ with respect to arc capacities $\hat{u}$ contains a negative cycle $\bar{C}$. That means $\bar{C}$ does not contain any forward arc $(i, j) \in A$ (since otherwise $\hat{u}_{ij} = \hat{x}_{ij}$ would destroy the cycle). Consequently, the reverse cycle C has $\sum_{(i,j)\in C} c_{ij} > 0$ and $\hat{x}_{ij} > 0$ holds for all arcs in C.

□

3 Complexity of IMCF Under Rectilinear Norm (L_1)

In this section, we study the complexity of the capacity inverse minimum cost flow problem under unit weight L_1 norm and illustrate that the problem is $\mathcal{NP}$-complete. For this purpose, we show that the *feedback arc set problem*, which is known to be $\mathcal{NP}$-complete [10, 15], is reducible to the rectilinear IMCF (R-IMCF).

A *feedback set* in a directed graph is a set of arcs that includes at least one arc of every directed cycle. The *feedback arc set problem (FAS)* seeks a feedback set of minimum size, so that the removal of all arcs in this set makes the resulting graph acyclic. In the weighted version of FAS, there exists a nonnegative weight function associated with the arcs of the digraph. Then, the objective is to find a minimum weight set of arcs A^0 such that the directed graph $G^0 = (N, A \backslash A^0)$ is acyclic.

First, we start with the simpler case and show the reducibility of FAS (cardinality) to R-IMCF on a graph with unit capacity arcs (R-1-IMCF). Since any feasible integer flow $\hat{x}$ is a 0-1 flow, capacities of arcs with positive flow cannot be changed, i.e. arcs in $A_R := \{(j, i) \in A(\hat{x}, u) : x_{ij} = 1\}$ cannot be deleted from the residual graph $G(\hat{x}, u)$. The decision problems of the two problems are therefore due to Proposition 1 as follows:

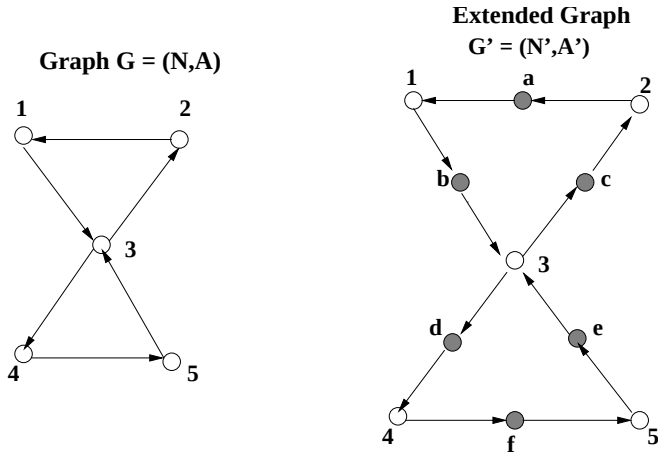

Figure 1: After the 1^{st} and 2^{nd} steps of transformation are applied

- *FAS:* Given a directed graph $G = (N, A)$, does there exist $\tilde{A} \subseteq A$ such that $|\tilde{A}| \leq k$ and $G \backslash \tilde{A}$ is acyclic?
- *R-1-IMCF:* Given a directed graph $G' = (N', A')$ with unit arc capacities, costs associated with arcs and a subset of arcs $A_R \subsetneq A'$, does there exist $\tilde{A} \subseteq A' \backslash A_R$ such that $|\tilde{A}| \leq k$ and $G' \backslash \tilde{A}$ defines a graph that does not contain any negative cost cycles?

Transformation Algorithm: Given a directed graph $G = (N, A)$ where $|N| = n$, $|A| = m$ and capacities $u_{ij} = 1 \quad \forall (i, j) \in A$, we modify graph G to obtain $G' = (N', A')$ such that G' is the residual graph for a feasible solution of a minimum cost flow problem and both graphs have the same cycles. For this purpose, we apply the following steps:

1. Split each arc of G into 2 arcs using a dummy node k and preserving the direction of the original arc, i.e. replace any arc (i, j) with two arcs (i, k) and (k, j).
2. For each arc pair $[(i, k), (k, j)]$ with dummy node k, set the cost of the arc (k, j) to $-\epsilon$ where $\epsilon > 0$. All the remaining arcs of G' will have costs of 0 (see figure 1).
3. Add a source node s and a sink node t.
4. For each arc pair $[(i, k), (k, j)]$ where $(i, j) \in A$, add the arc from the sink node to the dummy node k of the arc pair, and the arc from the end node j to the source node. If there exist parallel arcs between node j and source node s, we add new dummy nodes d_i and replace (j, s) with the two arcs (j, d_i) and (d_i, s) (see figure 2).
5. For all arcs the capacities are equal to 1.

By construction of the graph G' the following lemmata hold.

Lemma 3. *For any directed cycle of graph G, there exists a negative cost cycle in the transformed graph G' and vice versa.*

Lemma 4. *The directed graph G' that is constructed from the digraph G by applying the transformation algorithm defines a residual graph for a feasible solution of the minimum cost flow problem in a unit capacity graph.*

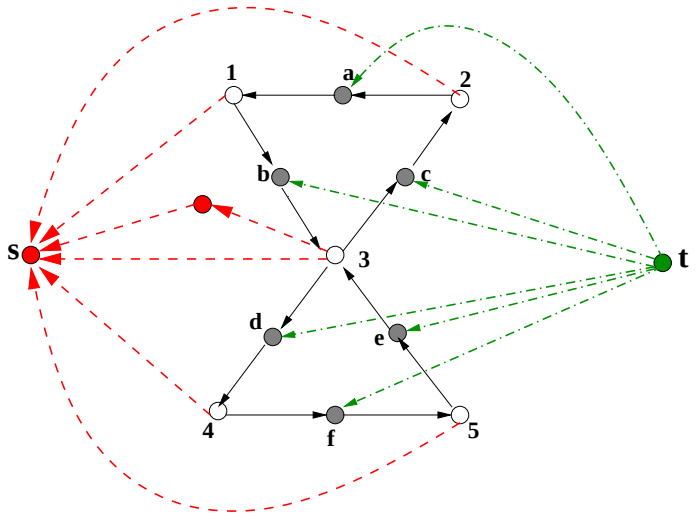

Figure 2: After the 3^{rd} and 4^{th} steps of transformation are applied

Proof: Reverse the direction of the arcs from the sink to the dummy nodes, from nodes j to the source node, and of the outgoing arcs from the dummies with negative costs (see figure 3). This part of the network allows a flow of value m=$|A|$ thus establishing a feasible solution to the minimum cost flow problem with a total cost of $m\epsilon$.

□

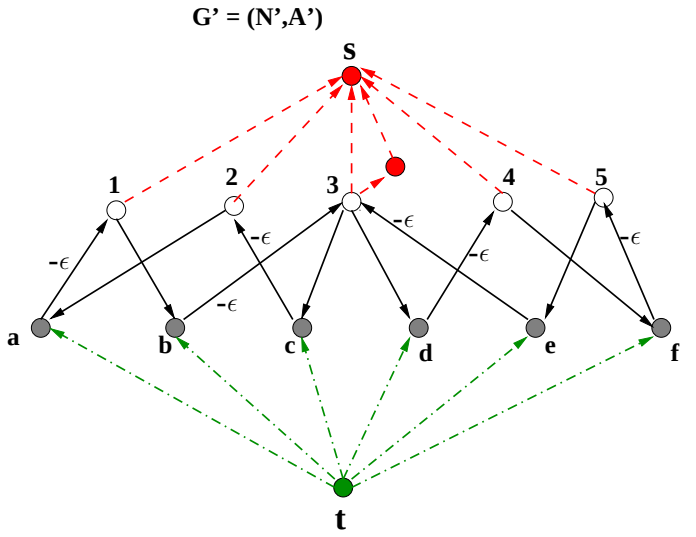

Figure 3: The transformed graph has now 2 negative cost cycles that correspond to the directed cycles of the original graph

As a result of the given two lemmata (3 and 4), we can conclude that the transformation algorithm generates an instance of R-IMCF on a unit capacity graph from any given cardinality FAS. Moreover, the transformation can be done in polynomial time since the number of required

changes at each step of the algorithm is bounded by $\mathcal{O}(\mathrm{m})$. Thus, we are now ready to show the $\mathcal{NP}$-completeness of the inverse problem.

Theorem 5. *The rectilinear capacity inverse minimum cost flow problem with unit weights is $\mathcal{NP}$-complete on a unit capacity graph.*

Proof: Obviously, the inverse problem on graph $G' = (N', A')$ is in $\mathcal{NP}$: Given a certificate $\tilde{A} \subseteq A' \backslash A_R$, it is possible to check in polynomial time whether $G' \backslash \tilde{A}$ is free of negative cycles by using any negative cycle detecting algorithm [1].

The transformation algorithm converts any FAS into an instance of R-IMCF in polynomial time. Hence, it is only left to prove

$$\begin{array}{rcl} \exists \tilde{A} \subseteq A: \quad |\tilde{A}| \leq k \quad \text{and} \quad G \backslash \tilde{A} \quad \text{is acyclic} \\ \iff \\ \exists \hat{A} \subseteq A' \backslash A_R : |\hat{A}| \leq k \quad \text{and} \quad G' \backslash \hat{A} \quad \text{is free of negative cost cycles} \end{array}$$

" $\Rightarrow$ " Suppose (i, j) is an arc in $\tilde{A}$. Then, by the construction of G' there exists an arc pair $[(i, d), (d, j)]$ that is the splitted version of this arc. By Lemma 4 one of these arcs, (d, j) with a cost of $-\epsilon$ is an element of A_R, but the other one, (i, d) can be deleted since it is not contained in A_R. So, we set $\hat{A} = \{(i, d) : (i, j) \in \tilde{A} \text{ and } (d, j) \in A_R\}$. Clearly, $|\hat{A}| = |\tilde{A}| \leq k$ and since each directed cycle of graph G is by Lemma 3 associated with a negative cycle in G', $G' \backslash \hat{A}$ is free of negative cycles.
" $\Leftarrow$ " Analogously we set $\tilde{A} = \{(i, j) : (i, d) \in \hat{A} \text{ and } (d, j) \in A_R\}$.

□

We can generalize this result to the rectilinear capacity inverse minimum cost flow problems for graphs with arc capacities $u_{ij} \geq 0$. The following corollary states this generalization.

Corollary 6. *The rectilinear capacity inverse minimum cost flow problem with unit weights is $\mathcal{NP}$-complete on graphs with arc capacities $u_{ij} \geq 0$.*

Proof: By using the same transformation algorithm as in the preceding case of unit capacities, we can reduce the **weighted** feedback arc set problem to R-IMCF. In this case, the weights of arcs in FAS define the capacities of the corresponding arc pairs in R-IMCF. Additionally, the capacities of the arcs $(i, s) : \ i \in N$ and $(t, j) : \ j \in N' \backslash N$ are set to be equal to the capacities of the arcs $(j, i) \in A'$. Thus, in the associated minimum cost flow problem, the flow to be sent from source node to sink node is $\sum_{(i,j) \in A} w_{ij}$, i.e. the total weight of the arcs in FAS.

□

With this corollary and the fundamental assumption $\mathcal{P} \neq \mathcal{NP}$, we can assume that R-IMCF cannot be solved in polynomial time. Hence, it is essential to find *good* approximations of the optimum solution, which gives rise to the question how good the problem R-IMCF can be approximated.

$\mathcal{APX}$ is the class of all $\mathcal{NP}$ optimization problems such that, for some $r \geq 1$, there exists a polynomial-time r-approximate algorithm for a problem P. Problem P is said to be **$\mathcal{APX}$-hard** if there exists a reduction from all problems $Q \in \mathcal{APX}$ to the problem P. If also $P \in \mathcal{APX}$, then P is called **$\mathcal{APX}$-complete**. Kann [14] reports that the minimum weighted feedback arc set problem is $\mathcal{APX}$-hard for directed graphs and no constant approximation algorithm is known for it [4].

In order to show the $\mathcal{APX}$-hardness of a problem, we need an **approximation preserving reduction** [4]. Let P_1 and P_2 be two optimization problems in $\mathcal{NP}$. P_1 is said to be *AP-reducible* to P_2 if two functions f and g and a positive constant $\alpha \geq 1$ exist satisfying the following conditions.

1. For any instance $x \in I_{P_1}$ and for any rational $r > 1$, $f(x, r) \in I_{P_2}$.
2. For any $x \in I_{P_1}$ and for any rational $r > 1$, if $SOL_{P_1}(x) \neq \emptyset$ then $SOL_{P_2}(f(x, r)) \neq \emptyset$.
3. For any $x \in I_{P_1}$, for any rational $r > 1$, and for any $y \in SOL_{P_2}(f(x, r))$, $g(x, y, r) \in SOL_{P_1}(x)$.
4. f and g are computable by two algorithms whose running time is polynomial for any fixed rational r.
5. For any instance $x \in I_{P_1}$, for any rational $r > 1$, and for any $y \in SOL_{P_2}(f(x, r))$, $R_{P_2}(f(x, r), y) \leq r$ implies $R_{P_1}(x, g(x, y, r)) \leq 1 + \alpha(r - 1)$.

Here, I_P is the set of instances of P, $SOL_P(x)$ is the set of feasible solutions of instance x of problem P, and $R_P(x, y)$ is the performance ratio of y with respect to x for problem P. The triple (f, g, α) is said to be an **AP-reduction** from P_1 to P_2.

Lemma 7. *The transformation algorithm that reduces feedback arc set problem to the rectilinear capacity inverse minimum cost flow problem is an AP-reduction in which f and g do not depend on performance ratio r and $\alpha = 1$.*

Proof: The transformation algorithm preserves the cycles and the weights of the arcs. Therefore, the feasible solutions of the two problem instances correspond to each other one-to-one. Moreover, the algorithm runs in polynomial time and the corresponding feasible solutions can be generated polynomially. The objective function value is also preserved during the transformation. Hence, it fulfills all the conditions of being an AP-reduction.

□

Since the feedback arc set problem is $\mathcal{APX}$-hard and AP-reducible to the capacity inverse minimum cost flow problem under unit-weight rectilinear norm, approximating R-IMCF is at least as difficult as approximating FAS. Thus, we can conclude the following result.

Corollary 8. *Capacity inverse minimum cost flow problem is $\mathcal{APX}$-hard under unit-weight rectilinear norm.*

4 IMCF Under Chebyshev Norm (L_∞)

In this section, we show that the inverse problem under L_∞ norm (subsequently abbreviated with C-IMCF) is polynomially solvable using a simple greedy algorithm. At each iteration of the algorithm we select a negative cost cycle and remove an arc from this cycle. For deletion we choose an arc in the cycle which is not forbidden to be deleted and which has a minimum capacity.

Greedy Algorithm

1. Initialize the set of deleted arcs $A_D = \emptyset$.
2. Choose a negative cost cycle C using any negative cycle detection algorithm.
   IF there exists no negative cycle, STOP.
   Output: $\hat{u}_{ij} = u_{ij}$ for $(i, j) \notin A_D$, else $\hat{u}_{ij} = 0$ with objective value $\max_{(i,j)\in A_D} u_{ij}$
3. Find $(k, l) = \arg\min_{(i,j)\in C\setminus A_R} u_{ij}$, then set $A_D = A_D \cup \{(k, l)\}$. GO TO Step-2.

Theorem 9. *The greedy algorithm solves the capacity inverse minimum cost flow problem under L_∞-norm optimally in $\mathcal{O}(nm^2)$ time.*

Proof: In each iteration we have to detect a negative cycle which takes $\mathcal{O}(mn)$ time [1]. Since we delete only one arc in each iteration, the algorithm will terminate after at most m iterations. Hence, the worst case running time is $\mathcal{O}(nm^2)$.

To prove the correctness of the algorithm, suppose A^* is an optimal set of arcs to be deleted for C-IMCF and $A^* \neq A_D$. Hence,

$$\max_{(i,j)\in A^*} u_{ij} \leq \max_{(i,j)\in A_D} u_{ij}$$

Moreover, by construction of the greedy algorithm there exists a negative cycle C for which

$$\arg\max_{(i,j)\in A_D} u_{ij} =: (i^*, j^*) \in C \quad \text{and} \quad u_{i^*j^*} = \min_{(i,j)\in C\setminus A_R} u_{ij}$$

Then,

$$\max_{(i,j)\in A^*} u_{ij} \leq \max_{(i,j)\in A_D} u_{ij} \leq u_{ij} \quad \forall\, (i,j) \in C\setminus A_R$$

We also conclude from Proposition 1 that A^* has to contain at least one arc from each negative cycle. So, there exists an arc $(k,l) \in A^* \cap C$.

$$u_{kl} \leq \max_{(i,j)\in A^*} u_{ij} \leq \max_{(i,j)\in A_D} u_{ij} \leq u_{kl} \tag{3}$$

Consequently, all the inequalities in (3) hold with equality and the solution of the greedy algorithm is optimum.

□

Although the solution of the greedy algorithm is optimal, it may not be always good enough in practice. The weakness of using only Chebyshev norm and the greedy algorithm is that some arcs might be deleted from $G(\hat{x})$ unnecessarily. In other words, the solution found by the greedy algorithm does in general not have the minimum number of arcs to be deleted. To overcome this drawback, we use a multicriteria approach in which we exploit a lexicographic bicriteria objective function instead of the L_∞ norm as single criterion. In this approach, we first minimize the maximum capacity of the arcs to be deleted, then we minimize the number of arcs to be deleted. Hence

$$\text{lexmin} \left\{ \begin{array}{l} \max_{(i,j)\in A_D} u_{ij} \\ |A_D| \end{array} \right.$$

generates the minimum cardinality set A_D that optimally solves C-IMCF.

The subproblem to be solved in this lexicographical problem is a rectilinear capacity inverse minimum cost flow problem with an upper bound on the capacity of the arcs to be deleted. It is not difficult to see that the rectilinear problem without upper bound constraint is a special case of the constrained problem. We only need to set an upper bound equal to the capacity of the maximum capacity arc in the graph. Since the rectilinear problem without the constraint is $\mathcal{NP}$-hard (Section 2), so is the problem with the upper bound constraint.

In the rest of this section, we propose a 2-phase approximation algorithm for solving the bicriteria inverse problem, which is based on an idea of Demetrescu and Finocchi [9] for the weighted FAS problem.

Algorithm FAS [9]: The algorithm consists of two phases. First, it looks for a simple cycle $\mathcal{C}$ and, if such a cycle exists, identifies an arc in $\mathcal{C}$ having minimum weight, say ϵ. Then, the weight of all the arcs in $\mathcal{C}$ is decreased by ϵ and the arcs whose weight becomes zero are removed. The first phase terminates when the graph becomes acyclic. In the second phase, the algorithm tries to add back some of the deleted arcs to the graph paying attention that no cycles are reintroduced.

The idea of the FAS algorithm can be adapted to approximate an efficient solution for the lexicographic bicriteria problem after making some modifications. Our version of the algorithm runs in the following way. In the first phase, we find an optimal solution for the Chebyshev inverse problem by applying the greedy algorithm. In this phase we also keep track of the the **cycle index**, i.e. number of cycles an arc is included in. This number is clearly a lower bound on the number of cycles containing a certain arc. In phase 2 we start with the feasible set of deleted arcs found by the greedy phase and apply two methods to reduce the number of arcs deleted while maintaining the feasibility:

1. We choose an arc (i, j) with the minimum cycle index and reinsert it to the graph if no new negative cycle is introduced.

2. If arc (i, j) cannot be reinserted to the graph, we find a negative cycle including arc (i, j). We select a new arc (k, l) on this cycle with the same or higher cycle index and which has a capacity not greater than the optimal solution of the Chebyshev problem. Then we delete the new arc (k, l) from the residual graph if it remains negative cycle free. We call this operation **SWAP**.

Below we provide a pseudo-code of the algorithm, which we call **bicriteria approximation algorithm:**

- 1^{st} **Phase:** Apply Greedy Algorithm to find u_{max} and a feasible arc setA_D
- 2^{nd} **Phase:**
  - **FOR ALL DELETED ARCS**
    - **INSERTION 1:**
      - Choose a deleted arc (i, j) with the minimum cycle index among the non-processed ones
      - **IF** inserting (i, j) causes no cycles, set $A(\hat{x}) = A(\hat{x}) \cup \{(i, j)\}$ and mark (i, j) processed.
      - **ELSE** go to SWAP
    - **SWAP:**
      - Find a negative cycle C that contains the arc (i, j)
      - Find a new arc $(k, l) \in C$ such that cycle index$(k, l) \geq$ cycle index(i, j), $A_D \backslash \{(i, j)\} \cup \{(k, l)\}$ is feasible, and $u_{kl} \leq u_{max}$
      - IF (k, l) exists, set $A_D = A_D \backslash \{(i, j)\} \cup \{(k, l)\}$ and mark (k, l) processed.
      - **ELSE** go to INSERTION 1
  - **FOR ALL DELETED ARCS**
    - **INSERTION 2:**
      - Choose a deleted arc (i, j) with the minimum cycle index
      - **IF** inserting (i, j) does not cause cycles, set $A(\hat{x}) = A(\hat{x}) \cup \{(i, j)\}$

Obviously, the algorithm has a polynomial running time. It has been already shown that the first phase has a worst case running time of $\mathcal{O}(nm^2)$. In the second phase the most time consuming operation is SWAP. Testing the feasibility of a new arc in SWAP requires $\mathcal{O}(nm)$ time. This test is applied in the worst case for m arcs and SWAP runs at most m times. So the worst case running time of the second phase is $\mathcal{O}(nm^3)$.

5 Computational Experiments

In the empirical analysis of our heuristic, a crucial part is the generation of suitable test cases. Since the bicriteria problem is $\mathcal{NP}$-hard, it is desirable to have input residual graphs for which the optimal number of arcs to be deleted is known. For this purpose we exploit the method described by Saab [17] for feedback arc set problems after making necessary changes.

Avg. Degree	# Nodes	Optimal # Deleted Arcs
4	500	90
	1000	150
	2000	210
8	500	120
	1000	200
	1500	250
16	500	180
	1000	250

Table 1: Graphs generated by the modification of Saab's method [17]

# Nodes	# Arcs	Index	Optimal	Bicriteria Approx.
96	528	1	21	21
		2	26	28
		3	31	33
		4	33	33
		5	44	46
160	912	1	14	14
		2	18	18
		3	29	29
		4	31	32
		5	46	49
200	1340	1	22	22
		2	29	31
		3	33	35
		4	42	43
		5	47	56
		6	61	65
500	3975	1	8	8
		2	20	21
		3	20	22
		4	31	34
		5	40	44

Table 2: Optimal test cases generated from RMFGEN instances [11]

The original algorithm plants arc disjoint cycles by using the relationship between vertex ordering and feedback arc sets. Graphs including only arc disjoint cycles are certainly inappropriate for testing the bicriteria algorithm, because the set of deleted arcs generated in the greedy phase is optimum. In the new version of the algorithm we plant arc disjoint cycle groups such that the

optimum number of deleted arcs is equal to the number of these cycle groups. Within these cycle groups, all the cycles have exactly one arc in common (leftward arc in vertex ordering). Moreover, we allow subgroups inside these cycle groups. We force one of the leftward arcs to have the highest capacity amongst the leftward arcs but to be the minimum capacity arc of its cycle group. The rest of the capacities are randomly generated such that the desired optimality is preserved. Random negative costs are assigned to the arcs.

We implemented the bicriteria approximation algorithm using C++ and the Boost Graph Library [18], and tested on graphs constructed by the modified version of Saab's algorithm. In Table 1 we provide the average node degree, number of nodes and the optimal number of deleted arcs for these graphs. We generated for each size 3 instances, so 24 graphs in total. We observed that all these test cases are solved to optimality by the bicriteria approximation algorithm. The reason for this lies in the structure of the graphs generated. In these graphs all the cycle groups constructed contain at least one negative cost 2-cycle or 3-cycle. This indicates that as the number of arc disjoint cycles increases and the length of negative cycles decreases the better the algorithm performs independent of the size and degree of the graph.

Next, we tested the bicriteria algorithm for different types of graphs in which the negative cost cycles are irregularly distributed. For this purpose we generated minimum cost flow problems and corresponding feasible, but non-optimal solutions. The major difficulty here was to identify the optimal solution of each test case. In order to overcome this difficulty we modeled the problem as a set covering problem. We used the set of negative cost cycles identified in the greedy phase to initialize the constraint set of the set covering problem. Then, we calculated the optimal solution by iteratively solving the set covering problem and adding each time a new negative cost cycle to the constraint set if the graph was not negative cycle free. Following Caprara et al. [8], we implemented the proposed set covering scheme using CPLEX 9.2 and C++.

# Nodes	# Arcs	Index	Optimal	Bicriteria Approx.
200	1308	1	38	43
	1500	2	49	60
	2000	3	19	26
	2200	4	39	46
	2900	5	30	31
300	3174	1	42	45
	4519	2	46	51
	5168	3	49	51
	6075	4	37	44
350	4508	1	56	61
	6000	2	53	56
	9000	3	58	60

Table 3: Optimal test cases generated from NETGEN instances [16]

The minimum cost flow problems were generated on 2 different types of graphs. The first group contains grid structured RMFGEN graphs [11]. These graphs are sparse with a single source and a single sink and they may contain 2-cycles. The second group consists of the well-known NETGEN networks. These graphs are transportation networks in which half of the nodes are sources and the other half are sinks having varying average node degrees.

Table 2 and Table 3 illustrate the approximation results for RMFGEN and NETGEN graphs, respectively. For 33% of the RMFGEN test cases the bicriteria algorithm could compute the

optimal solution. Moreover, the highest relative error is less than 20%. On the other hand, for NETGEN test cases the bicriteria algorithm could not find any optimal solutions and the highest relative error is almost 40%. Although the number of test cases is limited, one can observe that the performance of the bicriteria algorithm depends on the structure of the graph. As expected, for graphs containing shorter negative cost cycles, the algorithm computes better approximations.

# Nodes	# Arcs	Index	Set Covering Approx.	Bicriteria Approx.
200	1340	1	141	141
		2	80	86
		3	126	126
		4	233	229
		5	149	170
500	3975	1	155	141
		2	101	101
		3	40	39
		4	78	77
		5	55	55

Table 4: Approximated test cases generated from RMFGEN instances [11]

# Nodes	# Arcs	Index	Set Covering Approx.	Bicriteria Approx.
200	1308	1	134	130
	1500	2	82	97
	2000	3	107	113
	2200	4	281	291
	2900	5	87	96

Table 5: Approximated test cases generated from NETGEN instances [16]

Apparently, the set covering scheme cannot always compute an optimal solution in a reasonable time, which we accepted as 10 minutes. For these cases, we modified the set covering scheme so that we can find a feasible solution that serves as an approximation. The pseudo-code of the set covering approximation is as follows.

1. Apply the Greedy Algorithm to find u_{max} and a set of negative cost cycles $\mathcal{C}$
2. Initialize a set covering problem S_1 for the residual graph $G(x) = (N, A(x))$ with the cycle set $\mathcal{C}_1 = \mathcal{C}$
3. **Iteration i:**
   - Solve set covering problem S_i and find a set of deleted arcs D_i
   - **If** $G_i(x) = (N, A(x) \backslash D_i)$ is negative cycle free, STOP: output D_i
   - **Else if** Time < 5 min. and C_{new} is a negative cost cycle in $G_i(x) = (N, A(x) \backslash D_i)$ **then** set $\mathcal{C}_{i+1} = \mathcal{C}_i \cup C_{new}$. Go to iteration $i + 1$.
   - **Else** set $G(x) = (N, A(x) \backslash D_i)$, initialize $\mathcal{C}$, go to step 2.

Table 4 and 5 compare the approximation values of two algorithms for different graphs. In most of the cases, bicriteria approximation performs as well as the set covering approximation. Moreover, the bicriteria approximation has an obvious time advantage over the set covering approximation. The shortest running time for set covering approximation is 5 minutes whereas this is the highest running time for the bicriteria approximation scheme.

Looking at the computational test results we can conclude that it is reasonable to use the bicriteria approximation algorithm when the running time is critical or the graph has a simple structure with several 2 and 3-cycles. However, for small and moderate size graphs with complex structure we suggest to employ a set covering formulation implemented with CPLEX if it is essential to find an exact solution and there are no constraints on CPU time.

6 Conclusions and Future Work

In this paper, we analyzed the capacity inverse minimum cost flow problem for rectilinear and Chebyshev norms. For the former case, we proved that the problem is $\mathcal{NP}$-hard and $\mathcal{APX}$-hard by a reduction from the feedback arc set problem to our problem. On the other hand, the latter case is polynomially solvable by a greedy algorithm. However, since the greedy algorithm might delete some arcs unnecessarily, we consider a bicriteria problem in which we lexicographically minimize the maximum capacity of the arcs deleted and the number of deleted arcs.

The computational results show that the proposed heuristic performs especially well on graphs with simple structure when the cycle lengths are short. But we conjecture that the heuristic does not achieve a bounded approximation ratio since the performance depends highly on the problem structure. Therefore, we work currently on possible improvements of this heuristic so that we can achieve a bounded approximation ratio. We also work on approximation algorithms for the rectilinear IMCF$_u$ with the goal of establishing a constant approximation ratio. Another important aspect of the problem is to improve the polyhedral description of the problem and to take advantage of duality in this context. Since the dual of min cost flows are min cost tensions, results on inverse tension problems which are currently developed will be helpful. A generalization of this approach from network flows and tensions to flows in regular matroids [5] will give some insights into dealing with inverse linear programs having totally unimodular constraint matrices.

It seems that in addition to the application in radiation therapy sketched in the introduction, capacity inverse min cost flow problems may have potential for other applications. These are currently explored as well.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, New Jersey, 1993.

[2] R.K Ahuja and J.B. Orlin. Inverse optimization. *Operations Research*, 49:771–783, 2001.

[3] R.K Ahuja and J.B. Orlin. Combinatorial algorithms of inverse network flow problems. *Networks*, 40:181–187, 2002.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Germany, 1999.

[5] R.E. Burkard and H.W. Hamacher. Minimal cost flows in regular matroids. *Mathematical Programming Studies*, (14):32–47, 1981.

[6] D. Burton and P.L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53:45–61, 1992.

[7] D. Burton and P.L. Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63:1–22, 1994.

[8] A. Caprara, M. Fischetti, and P. Toth. Algorithms for the set covering problem. *Annals of Operations Research*, 89:353–371, 2000.

[9] C. Demetrescu and I. Finocchi. Combinatorial algorithms for feedback problems in directed graphs. *Information Processing Letters*, 86:129–136, 2003.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to Theory of NP-completeness*. W.H. Freeman, 1979.

[11] D. Goldfarb and M.D. Grigoriadis. A computational comparison of the dinic and network simplex methods for maximum flow. *Annals of Operations Research*, 13:83–123, 1988.

[12] H.W. Hamacher and K.-H. Küfer. Inverse radiation therapy planning - a multiple objective optimization approach. *Discrete Applied Mathematics*, 118:145–161, 2002.

[13] C. Heuberger. Inverse optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, 8:329–361, 2004.

[14] Viggo Kann. *On the Approximability of NP-Complete Optimization Problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Sweden, 1992.

[15] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972. R.E. Miller and J.W. Thatcher, editors.

[16] D. Klingman, A. Napier, and J. Stutz. NETGEN: a program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20:814–820, 1974.

[17] Y. Saab. A fast and effective algorithm for the feedback arc set problem. *Journal of Heuristics*, 7:235–250, 2001.

[18] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002.

[19] J. Zhang and L. Liu. Inverse maximum flow problems under the weighted hamming distance. *Journal of Combinatorial Optimization*, 12:395–408, 2006.

[20] J. Zhang and Z. Liu. Calculating some inverse linear programming problems. *Journal of Computational and Applied Mathematics*, 72:261–273, 1996.

[21] J. Zhang and Z. Liu. A general model of some inverse combinatorial optimization problems and its solution method under l_∞ norm. *Journal of Combinatorial Optimization*, 6:207–227, 2002.