Applied Mathematical Modelling 35 (2011) 5276–5285

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

An $O(mn\log(nU))$ time algorithm to solve the feasibility problem

Mehdi Ghiyasvand

Department of Mathematics, Faculty of Science, Bu-Ali Sina University, Hamedan, Iran

A R T I C L E   I N F O

Article history:
Received 6 November 2010
Received in revised form 9 April 2011
Accepted 15 April 2011
Available online 4 May 2011

Keywords:
Network flows
The infeasibility problem
The infeasibility algorithm

A B S T R A C T

The phase I maximum flow and most positive cut methods are used to solve the feasibility problem. Both of these methods take one maximum flow computation. Thus the feasibility problem can be solved using maximum flow algorithms. Let n and m be the number of nodes and arcs, respectively. In this paper, we present an algorithm to solve the feasibility problem with integer lower and upper bounds. The running time of our algorithm is $O(mn\log(nU))$, where U is the value of maximum upper bound. Our algorithm improves the $O(m^2\log(nU))$-time algorithm in [12]. Hence the current algorithm improves the running time in [12] by a factor of n. Sleator and Goldberg's algorithm is one of the well-known maximum flow algorithms, which runs in $O(mn\log n)$ time, see [5]. Under similarity assumption [11], our algorithm runs in $O(mn\log n)$ time, which is the running time of Sleator and Goldberg's algorithm. The merit of our algorithm is that, in the case of infeasibility of the given network, it not only diagnoses infeasibility but also presents some information that is useful to modeler in estimating the maximum cost of adjusting the infeasible network.

1. Introduction

Let $G = (N,A)$ be *a directed network* by a set N of n *nodes* and a set A of m directed *arcs*. We associate with each arc $(i,j) \in A$ an *upper bound* u_{ij} that denotes the maximum amount that can flow on the arc and a *lower bound* l_{ij} that denotes the minimum amount that must flow on the arc. We also represent the *flow* on an arc $i \to j \in A$ by x_{ij}. Let U be the maximum upper bound. Most of the network flow problems (for example the minimum cost flow and maximum flow problems) require to identify a flow x satisfying the following constrains:

$$\sum_{j\in N} x_{ji} - \sum_{j\in N} x_{ij} = 0, \quad \forall\, i \in N, \quad (conservation) \tag{1}$$

$$l_{ij} \leqslant x_{ij} \leqslant u_{ij}, \quad \forall\, i \to j \in A. \quad (boundedness) \tag{2}$$

Diagnosing whether there exists a flow x satisfying the conservation and boundedness constraints called *the feasibility problem*. A flow satisfying the constraints (1) and (2) called *a feasible flow* and a network that has a feasible flow called *a feasible network*. There are two methods to solve the feasibility problem: The phase I max flow and most positive cut procedures.

The phase I max flow procedure solves the feasibility problem using a maximum flow algorithm on an auxiliary network (see [1, p.169] and [2]). Methods to compute a most positive cut are presented in [3,2,4]; they are the tree search, phase I max-flow, and minimum flow methods, respectively. The tree search runs in $O(n^2)$ time, and the phase I max-flow and minimum flow take one maximum flow computation. The tree search needs a technical assumption to work correctly, namely,

E-mail address: mghiyasvand@basu.ac.ir

0307-904X/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2011.04.027

that the set of arcs with $l_{ij} < u_{ij}$ is a acyclic. This assumption is greatly violated in the applications, so it is not feasible for us to use the tree search. Therefore the best running times to solve the feasibility problem is equal to the running time of the best maximum flow algorithms.

The well-known algorithms on the maximum flow problem runs in $O(mn\log n)$-time (see [5]), $O(mn\log(n^2/m))$-time (see [6]), $O(nm\log_{m/n\log n} n)$-time (see [7]), $O(mn\log_{m/n}n + n^2\log^{2+\epsilon}n)$-time for any constant $\epsilon > 0$ (see [8]), $O(min\{n^{2/3},m^{1/2}\}m\text{-}\log(n^2/m)\log U)$-time (see [9]), and $O(nm\ \log(2 + n\sqrt{\log U}/m))$-time (see [10]).

In this paper, we consider a network G with integer lower and upper bounds and present a new algorithm to solve the feasibility problem, which runs in $O(mn\log(nU))$-time. Our algorithm is a scaling algorithm that uses the phase I maximum flow procedure in each phase and diagnoses the case of infeasibility by finding a positive cut. Similarity assumption [11] says that the bounds are at most a fixed power of n, namely $\log U = O(\log n)$. This assumption usually makes sense in practice and leads to lower asymptotic running times. Thus, under the similarity assumption, our algorithm runs in $O(mn\log n)$-time, which is the running time of Sleator and Tarjan's algorithm, see [5]. Although our algorithm does not improve the running time of Sleator and Tarjan's algorithm, but it not only diagnoses the feasibility or infeasibility of a network flow, but also helps us to estimate the maximum cost for repairing infeasible network.

An $O(m^2\log(nU))$-time algorithm to solve the feasibility problem has been presented in [12] using a variation of Minty's lemma [13]. The framework of the current scaling algorithm is the same as [12]: the both of them, in each phase, obtain a δ-feasible flow using a 2δ-feasible flow. But in the current paper, to obtain a δ-feasible flow, we use an efficient and well known maximum flow algorithm instead of Minty's lemma. We change lower and upper bounds on arcs and prove that a δ-feasible flow can be computed by solving a maximum flow problem with a maximum value at most m. The result of this modification is that we obtain a δ-feasible flow in $O(nm)$-time instead of $O(m^2)$-time. In the worst case analysis, the number of arcs is at most $(n-1) + (n-2) + \cdots + 1 = n(n-1)/2 = O(n^2)$, so the current algorithm improves the running time of the algorithm in [12] by a factor of n.

This paper consists of four sections in addition to Introduction. Some applications of the feasibility problem are mentioned in Section 1.1. Section 2 explains the most positive cut and phase I maximum flow procedures. In Section 3, the framework of our feasibility algorithm is shown, which is a scaling algorithm with $O(\log(nU))$ phases. Our method in each phase of the algorithm is shown in Section 4, which runs in $O(nm)$-time. An example for describing the algorithm is presented in Section 5. Finally, Section 6 concludes the paper.

1.1. Applications of the feasibility problem

In this section, we show some applications of the feasibility problem.

(a) Distribution system: associate with each node $i \in A$ an integer number $b(i)$ representing its supply/demand. If $d(i) > 0$, node i is a demand node; if $d(i) < 0$, node i is a supply node with demand of $-b(i)$; and if $d(i) = 0$, node i is a transshipment node. Assume $\sum_{i\in N} d(i) = 0$. The general case of the feasibility problem requires a flow x satisfying the following constraints:

$$\sum_{j\in N} x_{ji} - \sum_{j\in N} x_{ij} = d(i), \quad \forall\ i \in N,$$
$$l_{ij} \leqslant x_{ij} \leqslant u_{ij}, \quad \forall\ i \to j \in A.$$

The following distribution system illustrates how the feasible flow problem arises in practice. Suppose that some merchandise are available at some seaports and is desired by other ports. We know the stock of merchandise available at the ports, the amount required at the other ports, and the minimum and maximum quantity of merchandise that can be shipped on a particular sea route. We wish to know whether we can satisfy all of the demands by using the available supplies. In our algorithm, we suppose that $d = 0$. If we have $d_i \neq 0$ for some node i, then it is easy to reduce this case to the case $d = 0$ by adding a new node o with $d_o = 0$. Then for each node i with $d_i > 0$, we add an arc $i \to o$ with $l_{io} = u_{io} = d_i$ and for each node i with $d_i < 0$, we add an arc $o \to i$ with $l_{oi} = u_{oi} = -d_i$. Finally, we reset all $d_i = 0$. It is obvious that the new network is feasible if and only if the original network is feasible. Therefore we can use our algorithm in the new network to solve the feasibility problem in the original network.

(b) The minimum cost flow problem: supposing that each arc $i \to j \in A$ has a cost c_{ij} that denotes the cost per unit flow on the arc. The minimum cost flow problem is an optimization model formulated as follows:

$$\text{Minimize}\left\{\sum_{i\to j\in A} c_{ij}x_{ij} \mid x \textit{ is a feasible flow}\right\}.$$

Therefore the minimum cost flow problem needs to solve the feasibility problem.

2. The most positive cut and phase I maximum flow procedures

In this section, we explain two well-known most positive cut and phase I maximum flow procedures to solve the feasibility problem. The positive cut procedure uses the cuts of network to solve the feasibility problem. If $S, T \subset N$ form a nontrivial partition of N (i.e., $S, T \neq \phi$, $S \cap T = \phi$, and $S \cup T = N$), then we define the *cut* (S,T) as $(S,T) = \{i \to j \in A | i \in S \text{ and } j \in T\}$. *The value of (S,T)* is defined as:

$$V(S,T) = \sum_{\{i \to j | i \to j \in (S,T)\}} l_{ij} - \sum_{\{i \to j | i \to j \in (T,S)\}} u_{ij}. \tag{3}$$

Theorem 1 [14]. *A network with constrains (1) and (2) is feasible if and only if for every cut (S,T), we have $V(S,T) \leqslant 0$.* □

Theorem 1 says a network is feasible if it does not have any cut (S,T) with $V(S,T) > 0$, which called a *positive cut*. This theorem suggests a method to solve the feasibility problem. A cut (S^*,T^*) is called a *most positive cut* if it maximizes $V(S,T)$ over all cuts. Thus $V(S^*,T^*) \leqslant 0$ if and only if the network is feasible. A tree search method to computing a most positive cut was presented in [3], which needs a assumption that the set of arcs with $l_{ij} < u_{ij}$ is a acyclic. Because this assumption is greatly violated, the tree search method is not used in practice. Then two algorithms to compute a most positive cut are presented in [4] and [2], they showed that a most positive cut is computed using a maximum flow computation.

The phase I max flow procedure (see [1,2]) first chooses an initial flow $\hat{x}$ satisfying the boundedness and computes $e_i = \sum_j \hat{x}_{ji} - \sum_j \hat{x}_{ij}$. The auxiliary network is constructed as follow: a new source node s and arcs $s \to k$ for all nodes k that $e_k > 0$ is added to the network. The capacity of such arcs will be e_k. Also a new sink node t and arcs $l \to t$ for all nodes l that $e_l < 0$ and capacity $-e_l$ is added. For each arc $i \to j$ in the original network, two arcs $i \to j$ and $j \to i$ with upper bounds $r_{ij} = u_{ij} - \hat{x}_{ij}$ and $r_{ji} = \hat{x}_{ij} - l_{ij}$ are introduced. Then the maximum flow from s to t in the auxiliary network saturates all the source and sink arcs if and only if the original network is feasible. Therefore the phase I maximum flow procedure takes one maximum flow computation.

3. The feasibility algorithm

In this section, the framework of the feasibility algorithm is described, which is the framework of the scaling algorithm in [12]. A flow x is called a δ-feasible flow (see [12]) if it satisfies in (1) and

$$x_{ij} - u_{ij} \leqslant \delta, \quad l_{ij} - x_{ij} \leqslant \delta.$$

A network G with lower and upper bounds $l_{ij} - \delta$ and $u_{ij} + \delta$ is called *the δ-network*. Thus the δ-network is feasible if there a δ-feasible flow. For describing our feasibility algorithm, we need the next theorem.

Theorem 2 ([12]). *$x = 0$ is U-feasible. Moreover if l and u are integer and, for a $\delta < 1/m$, δ-network is feasible, then the network G is feasible.* □

Using Theorem 2, we have the framework of the *feasibility algorithm* shown in Fig. 1. The algorithm starts with $\delta = U$ and $x = 0$. In each phase, it tries to find a δ-feasible flow using the input 2δ-feasible flow. Theorem 2 says the feasibility algorithm has $O(log(nU))$ phases. Minty's lemma [13] is used in [12] to obtain a δ-feasible flow, in order to do each phase in $O(m^2)$-time. The next section shows how each phase of algorithm can be done in $O(mn)$-time using a maximum flow computation with the maximum value at most m.

```
The feasibility algorithm :
  Begin
    Let $x := 0$ and $\delta := U$;
    Do until $\delta < \frac{1}{m}$ ;
    Begin
      Let $\delta := \delta/2$;
      Producing a $\delta$-feasible flow;
    End;
  End.
```

Fig. 1. The feasibility algorithm.

4. Producing a δ-feasible flow using a 2δ-feasible flow

In this section, an input 2δ-feasible flow x is transformed to a δ-feasible flow if there is a δ-feasible flow. For each arc $i \to j \in A$, define lower and upper bounds λ_{ij} and μ_{ij} as follows

$$\lambda_{ij} = \left\lceil \frac{l_{ij} - \delta - x_{ij}}{\delta} \right\rceil, \qquad \mu_{ij} = \left\lfloor \frac{u_{ij} + \delta - x_{ij}}{\delta} \right\rfloor. \tag{4}$$

The network $G = (N,A)$ with lower and upper bounds λ and μ, called *the network* $G(\delta,x,\lambda,\mu)$. Like [12], we define the sets $G = G^- \cup G^+$, $B = B^+ \cup B^-$ and R by the following way (see Fig. 2):

$$\begin{aligned}
G^+ &= \{i \to j \in A \mid u_{ij} + \delta < x_{ij} \leqslant u_{ij} + 2\delta\},\\
G^- &= \{i \to j \in A \mid u_{ij} < x_{ij} \leqslant u_{ij} + \delta\},\\
B^+ &= \{i \to j \in A \mid l_{ij} - 2\delta \leqslant x_{ij} < l_{ij} - \delta\},\\
B^- &= \{i \to j \in A \mid l_{ij} - \delta \leqslant x_{ij} < l_{ij}\}, \quad \text{and}\\
R &= \{i \to j \in A \mid l_{ij} \leqslant x_{ij} \leqslant u_{ij}\}.
\end{aligned}$$

The following lemma presents the values λ_{ij}'s and μ_{ij}'s for some $i \to j \in A$.

Lemma 3. *For each arc $i \to j \in A$,*

(a) if $i \to j \in B^+$, then $\lambda_{ij} = 1$,
(b) if $i \to j \in B^-$, then $\lambda_{ij} = 0$,
(c) if $i \to j \in G^+$, then $\mu_{ij} = -1$,
(d) if $i \to j \in G^-$, then $\mu_{ij} = 0$.

Proof. Supposing that $i \to j \in B^+$ (resp., $i \to j \in B^-$), by the definition of the set B^+ (resp., B^-), we have $0 < l_{ij} - \delta - x_{ij} < \delta$ (resp., $0 < x_{ij} - (l_{ij} - \delta) < \delta$), which means $\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \rceil$ is equal to 1 (resp., 0). Thus, by (4), $\lambda_{ij} = 1$ (resp., $\lambda_{ij} = 0$).

Now we prove (c) and (d). If $i \to j \in G^+$ (resp., $i \to j \in G^-$), the definition of the set G^+ (resp., G^-) says $0 < x_{ij} - (u_{ij} + \delta) < \delta$ (resp., $0 < u_{ij} + \delta - x_{ij} < \delta$), which means $\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \rfloor$ is equal to -1 (resp., 0), so (4) says $\mu_{ij} = -1$ (resp., $\mu_{ij} = 0$). □

Lemma 4. *For each $i \to j \in A$, we have*

(a) $u_{ij} < x_{ij} + \delta \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor$,
(b) $x_{ij} + \delta \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil < l_{ij}$.

Proof. We have $\frac{u_{ij}+\delta-x_{ij}}{\delta} < \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor + 1$, so $u_{ij} + \delta - x_{ij} < \delta \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor + \delta$, which concludes (a).

We also have $\left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil < \frac{l_{ij}-\delta-x_{ij}}{\delta} + 1$, or $\delta \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil < l_{ij} - \delta - x_{ij} + \delta$, which concludes (b). □

By the definitions, if $B^+ \cup G^+ = \emptyset$, then the input 2δ-feasible flow x is also a δ-feasible flow and the current phase is finished. Otherwise the feasibility algorithm searched a δ-feasible flow using the network $G(\delta,x,\lambda,\mu)$. Consider the phase I max flow corresponding to $G(\delta,x,\lambda,\mu)$ with the following initial flow $\hat{x}$:

$$\hat{x}_{ij} = \begin{cases} 1 & \textit{if} \quad i \to j \in B^+,\\ -1 & \textit{if} \quad i \to j \in G^+,\\ 0 & \textit{otherwise}. \end{cases} \tag{5}$$

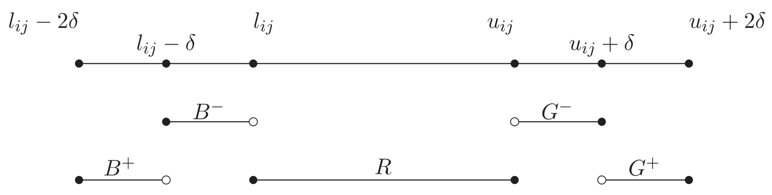

Fig. 2. Showing B^+, B^-, R, G^+, and G^-.

Note that, in $G(\delta,x,\lambda,\mu)$, the lower and upper bounds are λ and μ, respectively. Thus in the phase I max flow procedure corresponding to $G(\delta,x,\lambda,\mu)$, two arcs $i \to j$ and $j \to i$ have the following upper bounds

$$r_{ij} = \mu_{ij} - \hat{x}_{ij} \quad \text{and} \quad r_{ji} = \hat{x}_{ij} - \lambda_{ij}.$$

Supposing that $\tilde{x}$ is a maximum flow in this phase I max flow procedure. The flow x^o in $G(\delta,x,\lambda,\mu)$ corresponding to $\tilde{x}$ is computed by the following

$$x^o_{ij} = \hat{x}_{ij} + \tilde{x}_{ij} - \tilde{x}_{ji}, \quad \forall\, i \to j \in A. \tag{6}$$

Note that, if $\tilde{x}$ saturates the source and sink arcs in the phase I max flow procedure, then $G(\delta,x,\lambda,\mu)$ is feasible and x^o is a feasible flow in it. Otherwise $G(\delta,x,\lambda,\mu)$ is infeasible. The flow x' in the δ-network corresponding x^o is defined by

$$x'_{ij} = x_{ij} + \delta x^o_{ij}, \quad \forall\, i \to j \in A, \tag{7}$$

where x is the input 2δ-network flow. If $G(\delta,x,\lambda,\mu)$ is feasible, then x' is a δ-feasible flow and the current phase is finished. Fig. 3 shows the procedure of producing a δ-feasible flow. Note that if $G(\delta,x,\lambda,\mu)$ is infeasible, then the maximum flow $\tilde{x}$ does not saturates the source and sink arcs. The following definition and lemmas discuss about the case that $G(\delta,x,\lambda,\mu)$ is infeasible.

Definition 1. Supposing that $G(\delta,x,\lambda,\mu)$ is infeasible. Define Z as the set of nodes that can be reached by node s in the phase I max flow after sending the maximum flow $\tilde{x}$, and $\overline{Z} = N - Z$.

Lemma 5. *For each $i \to j \in (Z,\overline{Z})$, we have $x'_{ij} > u_{ij}$.*

Proof. Consider an arbitrary arc $i \to j \in (Z,\overline{Z})$. By Definition 1, the set Z is computed after sending the maximum flow $\tilde{x}$, so

$$\tilde{x}_{ij} = r_{ij}, \quad \text{and} \tag{8}$$
$$\tilde{x}_{ji} = 0. \tag{9}$$

Note that, r_{ij} is the residual capacity of the arc $i \to j$ in the phase I max flow procedure. The arc $i \to j$, corresponding to the 2δ-feasible flow x, belongs to B^+, B^-, R, G^+, or G^-. Thus we have the following cases:

Case (5-1): $i \to j \in B^+$.
In this case, by (5), $\hat{x}_{ij} = 1$, so $r_{ij} = \mu_{ij} - 1 = \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor - 1$. Thus, by (6), (8) and (9), we have $x^o_{ij} = \hat{x}_{ij} + \tilde{x}_{ij} - \tilde{x}_{ji} = 1 + \left(\left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor - 1 \right) - 0 = \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor$, so, by (7), $x'_{ij} = x_{ij} + \delta \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor$. Hence, by Lemma 4(a), $x'_{ij} > u_{ij}$.

```
Producing a δ-feasible flow
  Begin
    Define $B^+, B^-, G^+, G^-$ and $R$ corresponding to $x$;
    If $B^+ \cup G^+ = \phi$ then write "$x$ is a $\delta$-feasible flow" and break;
    Else for each $i \to j \in A$ define
      $\lambda_{ij} = \lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \rceil, \quad \mu_{ij} = \lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \rfloor$;
    Define the network $G(\delta, x, \lambda, \mu)$ ;
    Solve the phase I max flow using the initial flow $\widehat{x}$ defined by (5);
    Let $\widetilde{x}$ be a maximum flow corresponding to the phase I max flow;
    Let $x^o_{ij} = \widehat{x}_{ij} + \widetilde{x}_{ij} - \widetilde{x}_{ji}$, for each $i \to j \in A$;
    If $\widetilde{x}$ saturates the source and sink arcs in the phase I max flow;
      Begin
        Let $x'_{ij} = x_{ij} + \delta x^o_{ij}$, for each $i \to j \in A$;
        Write "$x'$ is a $\delta$-feasible flow" and break;
      End;
    Else write "the network $G(N, A)$ is infeasible" and break;
  End.
```

Fig. 3. The procedure of producing a δ-feasible flow using a 2δ-feasible flow.

Case (5-2): $i \to j \in B^-$ or $i \to j \in R$.
In this cases, (5) says $\hat{x}_{ij} = 0$, so $r_{ij} = \mu_{ij} - 0 = \mu_{ij}$. Hence (6), (8) and (9) conclude $x^o_{ij} = 0 + \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor - 0 = \left\lfloor \frac{u_{ij}+\delta-x_{ij}}{\delta} \right\rfloor$. Thus similar to the case (5-1), by (7) and 4(a), we get $x'_{ij} > u_{ij}$.

Case (5-3): $i \to j \in G^+$.
In this case, by (5) and Lemma 3(c), we have $\hat{x}_{ij} = -1$ and $\mu_{ij} = -1$. Thus $r_{ij} = \mu_{ij} - \hat{x}_{ij} = -1 - (-1) = 0$. Hence (6), (8) and (9) conclude $x^o_{ij} = -1 + 0 - 0 = -1$, which means, by (7), $x'_{ij} = x_{ij} - \delta$. The arc $i \to j$, corresponding to the flow x, belongs to the set G^+. Thus the arc $i \to j$, corresponding to the flow x', belongs to the set G^-, which means $x'_{ij} > u_{ij}$.

Case (5-4): $i \to j \in G^-$.
By (5) and Lemma 3(d), we get $\hat{x}_{ij} = 0$ and $\mu_{ij} = 0$, so $r_{ij} = \mu_{ij} - \hat{x}_{ij} = 0$. In a similar way to the case (5-3), it is shown that $x'_{ij} = x_{ij}$, which means the arc $i \to j$, corresponding to the flow x', belongs to the set G^-, so $x'_{ij} > u_{ij}$. □

Lemma 6. *For each $i \to j \in (\overline{Z}, Z)$, we have $x'_{ij} < l_{ij}$.*

Proof. Consider an arbitrary arc $i \to j \in (\overline{Z}, Z)$. By Definition 1, the set Z is computed after sending the maximum flow $\tilde{x}$, so

$$\tilde{x}_{ij} = 0, \quad \textit{and} \tag{10}$$

$$\tilde{x}_{ji} = r_{ji}. \tag{11}$$

The arc $i \to j$, corresponding to the 2δ-feasible flow x, belongs to B^+, B^-, R, G^+, or G^-. Thus we have the following cases:>

Case (6-1): $i \to j \in B^+$.
In this case, by (5) and Lemma 3(a), we have $\hat{x}_{ij} = 1$ and $\lambda_{ij} = 1$. Thus $r_{ji} = \hat{x}_{ij} - \lambda_{ij} = 1 - (1) = 0$. Hence (6), (10) and (11) conclude $x^o_{ij} = \hat{x}_{ij} + \tilde{x}_{ij} - \tilde{x}_{ji} = 1 + 0 - 0 = 1$, which means, by (7), $x'_{ij} = x_{ij} + \delta$. The arc $i \to j$, corresponding to the flow x, belongs to the set B^+, so it, corresponding to the flow x', belongs to the set B^-, which means $x'_{ij} < l_{ij}$.

Case (6-2): $i \to j \in B^-$.
By (5) and Lemma 3(b), we get $\hat{x}_{ij} = 0$ and $\lambda_{ij} = 0$, so $r_{ji} = \hat{x}_{ij} - \lambda_{ij} = 0$. In a similar way to the case (6-1), it is shown that $x'_{ij} = x_{ij}$, which means the arc $i \to j$, corresponding to the flow x', belongs to the set B^-, so $x'_{ij} < l_{ij}$.

Case (6-3): $i \to j \in G^+$.
In this case, by (5), $\hat{x}_{ij} = -1$, so $r_{ji} = \hat{x}_{ij} - \lambda_{ij} = -1 - \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil$. Thus, by (6), (10) and (11), we have $x^o_{ij} = \hat{x}_{ij} + \tilde{x}_{ij} - \tilde{x}_{ji} = -1 + 0 - \left(-1 - \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil\right) = \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil$, so, by (7), $x'_{ij} = x_{ij} + \delta \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil$. Thus, by Lemma 4(b), $x'_{ij} < l_{ij}$.

Case (6-4): $i \to j \in G^-$ and $i \to j \in R$.
In this cases, (5) says $\hat{x}_{ij} = 0$, so $r_{ji} = \hat{x}_{ij} - \lambda_{ij} = -\lambda_{ij}$. Hence (6), (10) and (11) conclude $x^o_{ij} = \hat{x}_{ij} + \tilde{x}_{ij} - \tilde{x}_{ji} = 0 + 0 - (-\lambda_{ij}) = \left\lceil \frac{l_{ij}-\delta-x_{ij}}{\delta} \right\rceil$. Therefore in a similar way to the case (6-3), by (7) and 4(b), we get $x'_{ij} < l_{ij}$. □

The next theorem uses Lemmas 5 and 6 and says if $G(\delta, x, \lambda, \mu)$ is infeasible, then the original network $G(N,A)$ is infeasible.

Theorem 7. *If $G(\delta, x, \lambda, \mu)$ is infeasible, then $G(N,A)$ is infeasible.*

Proof. Consider the sets Z and $\overline{Z}$ defined by Definition 1. Lemmas 5 and 6 conclude

$$x'_{ij} > u_{ij}, \quad \forall\ i \to j \in (Z, \overline{Z}), \tag{12}$$

$$x'_{ij} < l_{ij}, \quad \forall\ i \to j \in (\overline{Z}, Z). \tag{13}$$

Define $Z' = Z - \{s\}$ and $\overline{Z'} = \overline{Z} - \{t\}$, so if $i \to j \in (Z', \overline{Z'})$, then $i \to j \in (Z, \overline{Z})$ and if $i \to j \in (\overline{Z'}, Z')$, then $i \to j \in (\overline{Z}, Z)$, which means, by (12) and (13),

$$x'_{ij} > u_{ij}, \quad \forall\ i \to j \in (Z', \overline{Z'}),$$

$$x'_{ij} < l_{ij}, \quad \forall\ i \to j \in (\overline{Z'}, Z').$$

Thus we have

$$\sum_{i \to j \in (\overline{Z'}, Z')} x'_{ij} - \sum_{i \to j \in (Z', \overline{Z'})} x'_{ij} < \sum_{i \to j \in (\overline{Z'}, Z')} l_{ij} - \sum_{i \to j \in (Z', \overline{Z'})} u_{ij}. \tag{14}$$

The flow x' is produced by sending flow along paths from node s to node t, so x' satisfies in the conservation constraint (1) (because the nodes s and t are not in the node set N and they are defined in the phase I max flow). Therefore

$$\sum_{i\to j\in(\overline{Z'},Z')} x'_{ij} - \sum_{i\to j\in(Z',\overline{Z'})} x'_{ij} = 0,$$

which, by (14), means $(\overline{Z'},Z')$ is a positive cut in $G(N,A)$. Therefore, by Theorem 1, we conclude $G(N,A)$ is infeasible. □

Infeasibility $G(\delta,x,\lambda,\mu)$ not only says the $G(N,A)$ is infeasible but also gives some information about the maximum cost of repairing the infeasible flow. In Example 1, it was shown how the infeasibility of the 5-network and the feasibility of the 10-network estimate at most changes in lower and upper bounds to produce a feasible network. The following lemma helps to compute the running time of the feasibility algorithm.

Lemma 8. *The maximum flow value of the phase I max flow in the procedure of producing a δ-feasible flow is at most m.*

Proof. The initial flow $\hat{x}$ for the phase I max flow is defined as (5) and we have $e_i = \sum_j \hat{x}_{ji} - \sum_j \hat{x}_{ij}$, so

$$\sum_i \{i|e_i > 0\} \leqslant m.$$

By the definition of the phase I max flow procedure, the sum of the capacities of the source arcs is $\sum_i\{i|e_i > 0\}$. Thus the maximum flow value is at most m. □

Theorem 9. *The feasibility algorithm runs in $O(mn\log(nU))$.*

Proof. The running time of the procedure of producing a δ-feasible flow is dominated by a maximum flow computation in phase I max flow procedure. The shortest augmenting path algorithm [1] runs in $O(mn)$ time to solve a maximum flow problem if the algorithm is guaranteed to perform $O(m)$ augmentations (see [1], p. 220, lines -7 to -11). Each augmentation sends at least one unit and, by Lemma 8, maximum flow is at most m, so we have at most m augmentations. Thus each phase runs in $O(mn)$ time using the shortest augmentation path algorithm. Theorem 2 says the infeasibility algorithm has at most $O(\log(mU)) = O(\log(nU))$ phases, which means it runs in $O(mn\log(nU))$ time. □

5. Example

In this Section, our algorithms is described using the example network mentioned in Fig. 4. In this figure, $U = 20$, so $x = 0$ is a 20-feasible flow. Consider $\delta = \frac{20}{2} = 10$, Fig. 5 shows the 10-network and the network $G(10,x,\lambda,\mu)$ is shown in Fig. 6. We have $l_{36} - 20 = -8 < x_{36} = 0 < 2 = l_{36} - 10$ and $l_{41} - 20 = -5 < x_{41} = 0 < 5 = l_{41} - 10$, so $3\to 6 \in B^+$ and $4 \to 1 \in B^+$. For other arcs $i\to j$'s, we have $i \to j \in R$. By (5), the initial flow $\hat{x}$ should be defined by $\hat{x}_{41} = 1$, $\hat{x}_{36} = 1$ and $\hat{x}_{ij} = 0$, for other arc $i\to j$'s. Hence in the phase I max flow, we have $e_1 = 1$, $e_2 = 0$, $e_3 = -1$, $e_4 = -1$, $e_5 = 0$ and $e_6 = 1$. The network corresponding to this phase I max flow procedure is shown in Fig. 7.

A maximum flow $\tilde{x}$ corresponding to the network in Fig. 7 is $\tilde{x}_{s1} = \tilde{x}_{12} = \tilde{x}_{23} = \tilde{x}_{3t} = \tilde{x}_{s6} = \tilde{x}_{65} = \tilde{x}_{54} = \tilde{x}_{4t} = 1$ and $\tilde{x}_{ij} = 0$, for other $i \to j$'s (note that this flow saturates the source and sink arcs). The flow x^o in $G(10,x,\lambda,\mu)$ corresponding to the maximum flow $\tilde{x}$ is computed by $x^o_{ij} = \hat{x}_{ij} + \tilde{x}_{ij} - \tilde{x}_{ji}$, which is

$$x^o_{41} = 1+0-0 = 1,\quad x^o_{36} = 1+0-0 = 1,\quad x^o_{12} = 0+1-0 = 1,\quad x^o_{23} = 0+1-0 = 1,\quad x^o_{25} = 0+0-0 = 0,$$
$$x^o_{53} = 0+0-0 = 0,\quad x^o_{56} = 0+0-1 = -1\quad \text{and}\quad x^o_{54} = 0+1-0 = 1.$$

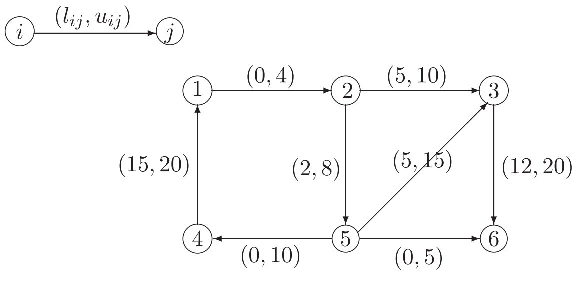

Fig. 4. An example network.

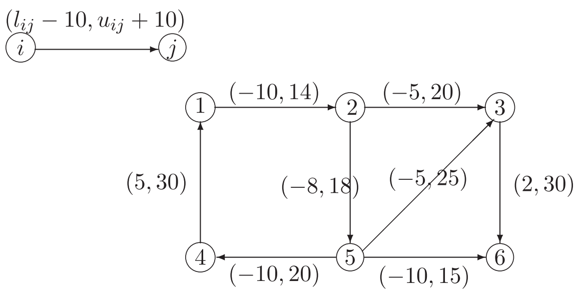

Fig. 5. The 10-network.

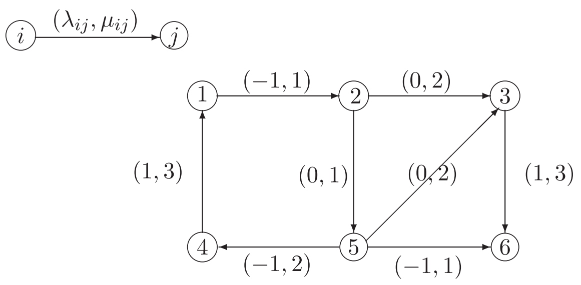

Fig. 6. The network $G(10,x,\lambda,\mu)$.

Therefore, in 10-network, flow x' (which is computed by $x'_{ij} = x_{ij} + \delta x^o_{ij}$) is the following:

$$x'_{41} = 0 + 10(1) = 10, \quad x'_{36} = 0 + 10(1) = 10, \quad x'_{12} = 0 + 10(1) = 10, \quad x'_{23} = 0 + 10(1) = 10,$$
$$x'_{25} = 0 + 10(0) = 0, \quad x'_{53} = 0 + 10(0) = 0, \quad x'_{56} = 0 + 10(-1) = -10, \quad x'_{54} = 0 + 10(1) = 10.$$

In the phase I max flow procedure corresponding to 5-network, the value of maximum flow $\tilde{x}$ is zero, because there is no directed path from node s to node t (see Figs. 8–10). Thus the source and sink arcs are not saturated by the maximum flow $\tilde{x}$, which means, by Theorem 7, $G(N,A)$ is infeasible. The set Z (the nodes that can be reached by node s after sending the maximum flow $\tilde{x}$) and $\overline{Z} = N - Z$ are

$$Z = \{s, 1, 6\}, \quad \overline{Z} = \{t, 2, 3, 4, 5\}.$$

Therefore $Z' = \{1,6\}$ and $\overline{Z'} = \{2,3,4,5\}$. The value of $(\overline{Z'}, Z')$ is

$$\sum_{i \to j \in (\overline{Z'}, Z')} l_{ij} - \sum_{i \to j \in (Z', \overline{Z'})} u_{ij} = l_{41} + l_{36} + l_{56} - u_{12} = 15 + 12 + 0 - 4 = 23 > 0,$$

which means it is a positive cut in the original network $G = (N,A)$. Therefore the example network in Fig. 4 is infeasible.

Now we show how the information of the feasibility algorithm is used to estimate the maximum cost for repairing this infeasible network. By the feasibility of the 10-network and the infeasibility of the 5-network, if we decrease lower bounds and increase upper bounds by 10 units, then the produced network is a feasible flow. In Fig. 4, we have $m = 8$, so we need to

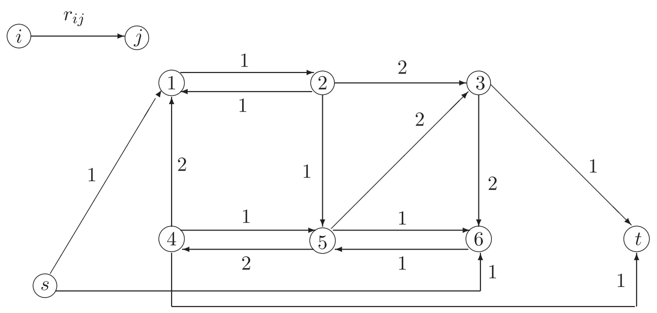

Fig. 7. The phase I max flow network corresponding to 10-network.

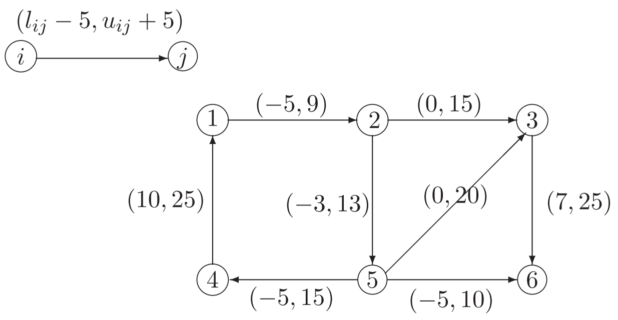

Fig. 8. The 5-network.

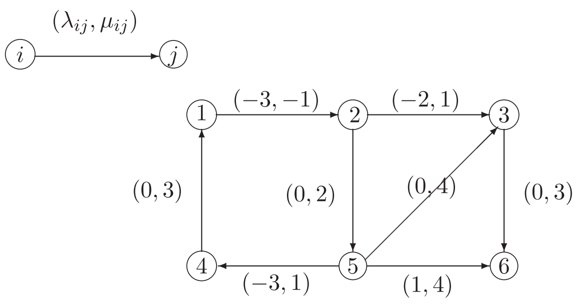

Fig. 9. The $G(5,x,\lambda,\mu)$ network.

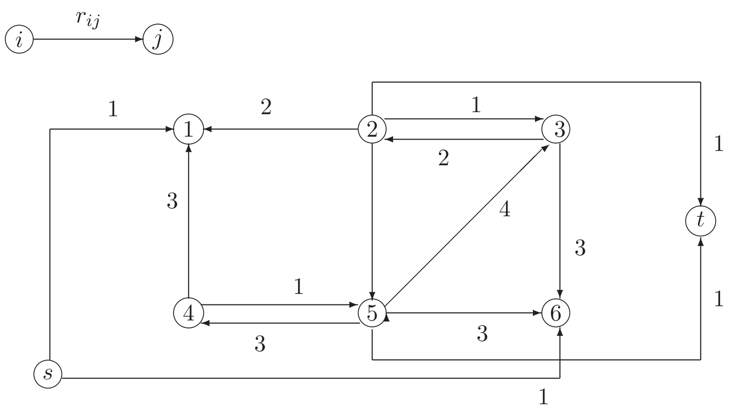

Fig. 10. The phase I max flow network corresponding to 5-network.

change the bounds 2 (10)(m) = 160 units. Of course, we can have a lower estimate using the 10-feasible flow. The 10-feasible flow is $x_{41} = x_{36} = x_{12} = x_{23} = x_{54} = 10$, $x_{25} = x_{53} = 0$, and $x_{56} = -10$. By Fig. 4, $(l_{23},u_{23}) = (5,10)$ and $(l_{54},u_{54}) = (0,10)$, so it is not necessary that the bounds of arcs 2 → 3 and 5 → 4 are changed when the 10-feasible flow is used. Fig. 4 and the 10-feasible flow say that we should increase u_{12} by 6 units and decrease l_{14}, l_{36}, l_{56}, l_{25}, and l_{53} by 5, 2, 10, 2, and 5 units, respectively. Thus, when the 10-feasible flow is used, the changes will be 6 + 5 + 2 + 10 + 2 + 5 = 30, which is 130 units lower than the case we decrease lower bounds and increase upper bounds by 10 units. This information can help to modeler to estimate the maximum cost of repairing.

6. Conclusion

In this paper, the $O(m^2 log(nU))$-time algorithm in [12] to solve the feasibility problem (with integer lower and upper bounds), is improved to $O(mn log(nU))$, which is an improvement by a factor of n in the worst case analysis. The infeasibility problem is solved using maximum flow algorithms. One of the well-known algorithms to solve the maximum flow problem is the $O(mn \log n)$-time algorithm of Sleator and Tarjan, see [5]. Under the similarity assumption [11], namely $\log U = O(\log n)$

(this assumption usually makes sense in practice), our time bound is $O(mn\log n)$, which is the time bound of Sleator and Tarjan's algorithm. The merit of our algorithm is that, in the case of infeasibility, it presents some information so that modeler can estimate the maximum cost of repairing. Thus, our algorithm not only improves the running time of the algorithm presented in [12] by a factor n, but also presents an estimation of the maximum cost for adjusting an infeasible network.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[2] S.T. McCormick, T.R. Ervolina, Computing maximum mean cuts, Discrete Appl. Math. 52 (1994) 53–70.
[3] R. Hassin, The minimum cost flow problem: a unifying approach to dual algorithms and a new tree-search algorithm, Math. programm. 25 (1983) 228–239.
[4] M. Ghiyasvand, A new approach for computing a most positive cut using the minimum flow algorithms, Appl. Math. Comput. 176 (2006) 27–36.
[5] D.D. Sleator, R.E. Tarjan, A data structure for dynamic trees, J. Comput. Syst. Sci. 24 (1983) 362–391.
[6] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem, J. ACM 35 (1988) 921–940.
[7] V. King, S. Rao, R. Tarjan, A faster deterministic maximum flow algorithm, J. Algorithms 17 (1994) 447–474.
[8] S. Phillips, J. Westbrook, Online load balancing and network flow, in: Proceedings of the 25th Annual ACM symposium on theory of comput., 1993, pp. 402–411.
[9] A.V. Goldberg, S. Rao, Beyond the flow decomposition barrier, J. ACM 45 (1998) 753–797.
[10] R.K. Ahuja, J.B. Orlin, R.E. Tarjan, Improved time bounds for the maximum flow problem, SIAM J. Comput. 18 (1989) 939–954.
[11] H.N. Gabow, Scaling algorithms for network problems, J. Comput. Syst. Sci. 31 (1985) 148–168.
[12] H. Salehi Fathabadi, M. Ghiyasvand, A new Algorithm for solving the feasibility problem of a network flow, Appl. Math. Comput. 192 (2007) 429–438.
[13] G.L. Minty, On the axiomatic foundations of the theories of directed linear graphs, electrical networks and programming, J. Math. Mech. 15 (1966) 485–520.
[14] A.J. Hoffman, Some recent applications of the theory of linear inequalities to extremal combinatorial analysis, in: R. Bellman, M. Hall (Eds.), American Mathematical Society (1960) 113–127.