A POLYNOMIAL-TIME BINARY SEARCH ALGORITHM FOR THE MAXIMUM BALANCED FLOW PROBLEM

Akira Nakayama

Otaru University of Commerce

(Received January 20, 1988; Revised March 20, 1989)

Abstract We consider the maximum balanced flow problem of a two-terminal network \( N \), i.e., a maximum flow problem with an additional constraint described in terms of a balancing rate function \( \alpha : A \to \mathbb{R}^+ - \{0\} \), where \( A \) is the arc set of \( N \) and \( \mathbb{R}^+ \) is the set of nonnegative reals. In this paper, we propose a polynomial time algorithm for the maximum balanced flow problem, on condition that all given functions in \( N \) are rational. The proposed algorithm, which is composed of a binary search algorithm and Dinic's maximum flow algorithm with a parameter, requires \( O(\max\{\log(c^*), m\log(n^*), nm\})T(n, m) \) time, where \( c^* = \max\{c(a) : a \in A\} \) for positive integral arc-capacities \( c(a) : a \in A \) and \( n^* = \max\{n(a) : a \in A\} \) for \( \alpha(a) \equiv \zeta(a)/\eta(a) \leq 1 \) such that \( \zeta(a) \) and \( \eta(a) \) are positive integers, and \( T(n, m) \) is the time for the maximum flow computation for a network with \( n \) vertices and \( m = |A| \) arcs.

1. Introduction

Minoux [10] considered the maximum balanced flow problem, i.e., the problem of finding a maximum flow in a two-terminal network such that each arc flow value of the underlying graph is bounded by a fixed proportion of the total flow value from source \( s \) to sink \( t \). The maximum balanced flow problem is motivated by Minoux's research of reliability analysis of communication networks. If a flow from \( s \) to \( t \) is balanced, then it is guaranteed that the value of the blocked arc flow is at most the fixed proportion of the total flow value from \( s \) to \( t \).

Several algorithms [2,3,10,11,13] are proposed for the maximum balanced flow problem. Cui [2,3] showed a simplex and a dual simplex methods without cycling on the underlying graph \( G \) of two-terminal network. When balancing rate functions are constant, Minoux's algorithm [10] and that of Nakayama [11] are proposed. The former needs \( O(p_{\max}^2S(n, m)) \) time, where \( p_{\max} \) is the maximum number of arc disjoint directed paths from source to sink of \( G \) and \( S(n, m) \) is the complexity of the shortest path problem for a network with \( n \) vertices and \( m \) arcs and with a nonnegative arc length function. The latter takes \( O(\min\{m, [1/\tau]\})T(n, m) \) time, where \( \alpha(a) = r \) \( (a \in A) \) for given balancing rate function \( \alpha : A \to \mathbb{R}_+ - \{0\} \) \( (\mathbb{R}_+ \) is the set of nonnegative reals.), some real \( r \) and the arc set \( A \) of \( G \), and \( T(n, m) \) is the time for the maximum flow computation for a two-terminal network with \( n \) vertices and \( m \) arcs, and \( [1/\tau] \) is the maximum integer less than or equal to \( 1/\tau \). For general balancing rate functions, Zimmermann [13] proposed an algorithm with \( O(T(n, m)^2) \) computation time.

On the other hand, Ichimori et al. [7,8] considered the weighted minimax flow problem, and Fujishige et al. [5] pointed out the equivalence of the maximum balanced flow problem and the weighted minimax flow problem. When capacity function \( c : A \to \mathbb{Z}_+ \) and weight function \( w : A \to \mathbb{Z}_+ \) are given for the set \( \mathbb{Z}_+ \) of nonnegative integers, the algorithm [8] takes \( O(T(n, m)P) \) computation time, where
\[ P = \log(\max\{c(a)w(a) : a \in A\}). \] The algorithm [7] runs in \( O(T(n,m)^2) \) time for general weight functions, having the same speed as Zimmermann's.

We can see the minimax transportation problem, studied by Ahuja [1], of finding a feasible flow \( (x(a) : a = (i,j) \in I \times J) \) from \( I \) to \( J \) such that \( \max\{c(a)x(a) : a = (i,j) \in I \times J\} \) is minimum, where \( I \) is a set of origins, \( J \) is a set of destinations and \( c(a) \) is the cost of unit shipment on each arc \( a = (i,j) \in I \times J \). The minimax transportation problem may be regarded as a special version of the weighted minimax flow problem.

The objective of the present paper is to propose a polynomial time algorithm for the maximum balanced flow problem of a two-terminal network \( N \), on condition that all given functions including \( \alpha : A \to R^+ \) in \( N \) are rational. We put \( \alpha(a) = \zeta(a)/\eta(a) \) for some two positive integers \( \zeta(a) \) and \( \eta(a) \). The total complexity is \( O(\max\{\log c^* + m \log \eta^*, nm\}T(n,m)) \), where \( c^* = \max\{c^o(a) : a \in A\} \) for arc-capacities \( c^o(a) \in Z^+ - \{0\} \) \( (a \in A) \), \( \eta^* = \max\{\eta(a) : a \in A\} \). The proposed algorithm, which is composed of a binary search algorithm and Dinic's maximum flow algorithm with a parameter, will be expected to be faster than known algorithms in case that all input data are rational.

### 2. The Maximum Balanced Flow Problem

Let \( G = (V,A) \) be a directed graph where \( V \) is the vertex set and \( A \) is the arc set of \( G \). For two capacity functions \( c^o : A \to R^+ \) and \( c_o : A \to R^+ \), a balancing rate function \( \alpha : A \to R^+ - \{0\} \) and a function \( \beta : A \to R \), consider a two-terminal network \( N = (G = (V,A),c^o,c_o,\alpha,\beta,s,t) \) where \( R^+ \) is the set of nonnegative reals, \( R \) is the set of reals, \( s \) is the source and \( t \) is the sink of \( G \). The maximum balanced flow problem \( (P) \) for network \( N \) is formulated as follows.

\[
(P) : \quad \text{Maximize } f(a^*) \quad \text{subject to} \\
(1) \quad D \cdot f = 0, \\
(2) \quad c_o(a) \leq f(a) \leq c^o(a) \quad (a \in A), \\
(3) \quad f(a) \leq \alpha(a)f(a^*) + \beta(a) \quad (a \in A),
\]

where arc \( a^* = (t,s) \notin A \) is added to \( G \) and \( D \) is the vertex-arc incidence matrix of \( G \). We assume that \( c^o, c_o \) and \( \beta \) are integral, and that \( c^o(a) > \beta(a) \) \( (a \in A) \) and \( \alpha(a) \equiv \zeta(a)/\eta(a) \leq 1 \) \( (a \in A) \) for some positive integers \( \zeta(a) \) and \( \eta(a) \). Define \( \theta \) by

\[
\theta = \prod \{\eta(a) : a \in A\}. \tag{4}
\]

If the function \( f : A^* \to R^+ \) \( (A^* = A \cup \{a^*\}) \) satisfies \( (1) \sim (3) \), then \( f \) is called a balanced flow in network \( N \). Let \( f^* \) be the value maximizing \( f(a^*) \) in \( N \), and define the boundary \( \partial f : V \to R \) of a function \( f : A^* \to R^+ \) in \( N \) by

\[
\partial f(v) = \sum\{f((v,i)) : (v,i) \in A^*\} - \sum\{f((i,v)) : (i,v) \in A^*\}, \tag{5}
\]

where \( v \in V \). Associated with problem \( (P) \), consider the following two problems \( (P^*) \) for network \( N^* = (G = (V,A),c^o,c_o,s,t) \):

---

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Algorithm for Maximum Balanced Flows

\((P^*)\) : Maximize \(g(a^*)\)
subject to (1) and (2), where \(f\) should be replaced by \(g\),

and \((P(y))\) for network \(N(y) = (G = (V, A), (c^0(a, y) : a \in A), e_0, \beta, s, t)\), where \(y\) is a parameter and \(c^0(a, y) = \min\{c^0(a), \alpha(a)y + \beta(a)\}\):

\((P(y))\) : Maximize \(f(a^*)\)
subject to constraint (1) and
(6) \(e_0(a) \leq f(a) \leq c^0(a, y) (a \in A)\).

Note that \((P(y))\) can be regarded as a maximum flow problem with parameter \(y\) in capacities \((c^0(a, y) : a \in A)\).

**Proposition 1.** Let \(f^{**}(y)\) be the value maximizing \(f(a^*)\) in network \(N(y)\). If problem \((P)\) is feasible, then we have \(f^* = \max\{y : f^{**}(y) = y\}\).

Define the capacity \(c(A(S))\) of a cut \(A(S) := A^+(S) \cup A^-(S)\) by
\[
c(A(S)) = \sum\{c^0(a) : a \in A^+(S)\} - \sum\{c_0(a) : a \in A^-(S)\},
\]
where for \(S \subset V (s \in S, t \notin S)\), \(A^+(S) = \{(i, j) \in A : i \in S, j \notin S\}\) and \(A^-(S) = \{(i, j) \in A : j \in S, i \notin S\}\). A *minimum cut* is defined to be a cut having the minimum capacity. Then we have:

**Theorem 2 [4].** For any network the maximum flow value from the source to the sink is equal to the capacity of a minimum cut.

Let \(A(S, y)\) be a minimum cut in network \(N(y)\) at \(y\), and
\[
K'(S, y) = \{a \in A^+(S, y) : c^0(a) > \alpha(a)y + \beta(a)\} \quad \text{and} \quad K''(S, y) = A^+(S, y) - K'(S, y).
\]
From theorem 2 we have \(f^{**}(y) = U(S, y)y + W(S, y)\), where
\[
U(S, y) = \sum\{\alpha(a) : a \in K'(S, y)\} \quad \text{and} \quad W(S, y) = \sum\{\beta(a) : a \in K'(S, y)\} + \sum\{c^0(a) : a \in K''(S, y)\} - \sum\{c_0(a) : a \in A^-(S)\}.
\]
\(U(S, y)\) is called slope in \(N(y)\) at \(y\). Define \(b^o\) and \(b_o\) by
(7) \(b^o = \max\{(c^0(a) - \beta(a))/\alpha(a) : a \in A\}\),
(8) \(b_o = \max\{\max\{(c^0(a) - \beta(a))/\alpha(a) : a \in A\}, 0\}\).

**3. Algorithm for the Maximum Balanced Flow Problem**

Consider two functions \(z = f^{**}(y)\) and \(z = y\) in a \((y, z)\)-plane. From proposition 1, if problem \((P)\) is feasible then the optimal value of \((P)\) is the maximum \(y^*\) such that \((y^*, y^*)\) is an intersection point of \(z = f^{**}(y)\) and \(z = y\). The outline of our algorithm is composed of the following two parts 1 and 2, though the detailed description will be shown in subsequent sections:
Part 1: By a binary search algorithm, we find $y_o$ and $y^o$ such that $y_o \leq f^* \leq y^o$ and $y^o - y_o < \gamma$ for some fixed value $\gamma \in \mathbb{R}_+$. 

Part 2: We find $f^*$ by Dinic's maximum flow algorithm with parameter $y$ satisfying $y_o \leq y \leq y^o$.

3.1 Algorithm of Part 1

In later discussion, we assume that problem $(P^*)$ is feasible. Let

\[ \gamma = \frac{1}{(\theta m^2 (m + n + 1)^{2n^2 - 1})^w} \]

where $m = |A|$, $n = |V|$ and $w = 2mn + n^2 - 2m + n - 2$. Algorithm I of Part 1 is as follows.

Algorithm I:

Step 1: Put $\text{FLAG0} = \text{FLAG1} = 1$. Find the maximum flow value $g^*$ in network $N^*$. If $g^* \geq b_o$, then we have the optimal value $f^* = g^*$ and stop. Otherwise, put $y_o = g^*$ and $y_o = b_o$.

Step 2: (2.1) If $y^o - y_o < \gamma$, then stop. Otherwise, put $y'' = (y^o + y_o)/2$.

Do $\text{WAIT-A-MINUTE} (y'', y^o, y_o, \text{FLAG0}, N(y))$. If $\text{FLAG0} = 0$ ($y_o$ is renewed.), then go back to (2.1).

(2.2) Do $\text{JUDGE} (y'', y^o, y_o, \text{FLAG1}, N(y))$. If $\text{FLAG1} = 0$, then stop. Otherwise, go back to (2.1).

In algorithm I, WAIT-A-MINUTE ($y'', y^o, y_o, \text{FLAG0}, N(y)$) and JUDGE ($y'', y^o, y_o, \text{FLAG1}, N(y)$) are the following procedures, where two variables $\text{FLAG0}$ and $\text{FLAG1}$ are in $\{0, 1\}$ and $N(y) = (G = (V, A), (c^o(a, y) : a \in A), c_o, s, t)$.

Procedure WAIT-A-MINUTE ($y'', y^o, y_o, \text{FLAG0}, N(y)$):

Calculate the maximum flow value $f^{**} (y)$ of $N(y)$ at $y = y''$. If we have $y'' \leq f^{**} (y'')$ or no flows for $N(y)$, then put $y_o = y''$ and $\text{FLAG0} = 0$.

Otherwise, we put $\text{FLAG0} = 1$.

Procedure JUDGE ($y'', y^o, y_o, \text{FLAG1}, N(y)$):

Find line $z = L(y)$ with slope $U(S, y'')$ for some $S \subset V$ containing point $(y'', f^{**} (y''))$. Then obtain the intersection point $(y', z')$ of $z = L(y)$ and $z = y$.

If $y' > y^o$ or $y' < y_o$, then put $\text{FLAG1} = 0$. Otherwise, renew $y^o$ or $y_o$ as follows:

$y^o = y'$ \hspace{1cm} ($y' \leq y''$),
$y_o = y'$ \hspace{1cm} ($y' > y''$).

$\text{FLAG0}$ shows whether JUDGE ($y'', y^o, y_o, \text{FLAG1}, N(y)$) is carried out or not, while $\text{FLAG1}$ means that if $\text{FLAG1} = 0$, then problem $(P)$ is infeasible.

3.2 Algorithm of Part 2

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Algorithm for Maximum Balanced Flows

Assume that $y^o - y_o < \gamma$ after algorithm I. Before describing algorithm II, change network $N(y)$ into network $N'(y) = (G', A'), (c'(a, y) : a \in A'), s', t')$ as follows.

10. $V' = V \cup \{s', t'\}$, $A' = A^* \cup A^+ \cup A^-,$
11. $A^+ = \{(s', v) : v \in V, \partial c_o(v) < 0\}$, $A^- = \{(v, t') : v \in V, \partial c_o(v) > 0\}$,
12. $c'(a, y) = c^o(a, y) - c_o(a)$ $(a \in A^*)$,
13. $c'((s', v), y) = -\partial c_o(v)$ $((s', v) \in A^+),$
14. $c'(y, t') = \partial c_o(v)$ $((v, t') \in A^-),$

where $c_o(a^*) = c''(a^*, y) = y$, $s'$ is the source and $t'$ is the sink of $N'(y)$. Then we have the following proposition.

**Proposition 3** [9]. We have a feasible flow in $N(y)$ satisfying $c_o(a^*) = c^o(a^*, y) = y$ if and only if we have a maximum flow $(f'(a, y) : a \in A')$ from $s'$ to $t'$ in $N'(y)$ such that $f'(a, y) = c'(a, y)$ $(\forall a \in A^+)$. □

Let $q(y)$ and $q'(y)$ be linear functions of $y$, and $\Gamma = [r, r'] \subset \mathbb{R}$ be a closed interval. If either $q(y) \leq q'(y)$ $(\forall y \in \Gamma)$ or $q(y) \geq q'(y)$ $(\forall y \in \Gamma)$ then $q(y)$ and $q'(y)$ are comparable in $\Gamma$. Define ROUTINE $(q(y), q'(y), \Gamma, Y)$ as follows, where $Y$ is a variable.

**Procedure ROUTINE** $(q(y), q'(y), \Gamma, Y)$:

If $q(y)$ and $q'(y)$ are comparable in $\Gamma$, then put $Y = -1$. Otherwise, obtain the solution $Y \in \mathbb{R}$ of equation $q(y) = q'(y)$ $(y \in \Gamma)$.

Now we show algorithm II of Part 2.

Algorithm II:

Step 1: Put $\text{FLAG}_0 = \text{FLAG}_1 = 1$. Calculate a maximum flow for network $N'(y)$ by Dinic's maximum flow algorithm: Construct layered network $L$ of $N'(y)$ and find a maximal flow of $L$.

(1.1) Renew $L$ and denote new layered network by $L$ again. If we attain a maximum flow $(f'(a, y) : a \in A')$, then go to Step 2. Otherwise, find a maximal flow of $L$:

(1.1.1) Find a flow-augmenting path $Q(y)$ of $L$ and choose two arc-capacities $q(y)$ and $q'(y)$ of $Q(y)$. (Note that $q(y)$ and $q'(y)$ are linear functions of $y$.)

(1.1.2) Do ROUTINE $(q(y), q'(y), [y_o, y^o], Y)$. If $Y = -1$, then go to (1.1.3). Otherwise, do WAIT-A-MINUTE $(Y, y^o, y_o, \text{FLAG}_0, N(y))$.

If $\text{FLAG}_0 = 0$, then go to (1.1.3). Otherwise, do JUDGE $(Y, y^o, y_o, \text{FLAG}_1, N(y))$. If $\text{FLAG}_1 = 0$, then stop.

(1.1.3) If we calculated the minimum arc capacity of $Q(y)$, do the flow
augmentation of \( Q(y) \). Otherwise, find other two arc-capacities \( q(y) \) and \( q'(y) \) of \( Q(y) \) and go to (1.1.2). If we have a maximal flow of \( L \), then go to (1.1) of Step 1. Otherwise, go to (1.1.1).

Step 2: If we attain a maximum flow \( (f'(a, y) : a \in A') \) such that \( f'(a, y) = c'(a, y) \) for all \( a \in A^+ \), then we have the optimal value \( f^* = \max \{ y : y \in [y_o, y^\circ] \} \) and stop. Otherwise, \( (P) \) is infeasible.

4. The Validity and Complexity

The following proposition is easy to see:

**PROPOSITION 4.** If problem \( (P) \) is feasible and we have not found the optimal value \( f^* \) after algorithm I, then we have \( y_o \leq f^* \leq y^\circ \). □

The residual network \( N''(y) = (G'' = (V'', A''), (e''(a, y) : a \in A''), s', t') \) with respect to a flow \( (f(a, y) : a \in A') \) in network \( N'(y) \) is defined as

\[
\begin{align*}
V'' &= V', \quad A'' = A'_1 \cup A'_2, \\
e''(a, y) &= e'(a, y) - f(a, y) \quad (a \in A'_1), \\
e''(a^-, y) &= f(a, y) \quad (a^- \in A'_2),
\end{align*}
\]

where \( A'_1 = \{ a \in A' : f(a, y) < c'(a, y) \} \) and \( A'_2 = \{ a^- : a^- \) is the reversed arc of \( a \in A' \) with \( f(a, y) > 0 \}. \)

Let \( N''_i(y) = (G''_i = (V''_i, A''_i), (e''_i(a, y) : a \in A''_i), s', t') \) be \( i \)-th residual network as to a maximal flow \( (f_{i-1}(a, y) : a \in A''_{i-1}) \) of \( N''_{i-1}(y) \), where \( N''_1(y) = N'(y) \). Let \( L''_i(y) \) be the layered network of \( N''_i(y) \), and \( Q(y) \) be a flow augmenting path of \( L''_i(y) \). The flow augmentation of \( Q(y) \) is called path-flow augmentation of \( L''_i(y) \).

**PROPOSITION 5.** Let \( n(i) \) be the number of path-flow augmentations of \( L''_i(y) \). Then we have \( n(i) \leq m' - i + 1 \) for \( m' = \lvert A' \rvert \).

(Proof) Let \( \Xi_i \) be a set of the paths joining \( s' \) and \( t' \) of \( L''_i(y) \). We see that each path in \( \Xi_i \) has the same length, say, \( p(i) \). Then we have

\[
p(i) + n(i) - 1 \leq \lvert A(L''_i(y)) \rvert \leq m',
\]

where \( A(L''_i(y)) \) is the arc set of \( L''_i(y) \). From \( i \leq p(i) \), we have \( n(i) \leq m' - i + 1 \). □

**PROPOSITION 6.** Let \( (f, j(a, y) : a \in A(L''_i(y))) \) be a flow of \( L''_i(y) \) obtained after \( j \) path-flow augmentations of \( L''_i(y) \). Then we have:

\[
\begin{align*}
f, j(a, y) &= \sum \{ \kappa^a(e) e''(e, y) : e \in A(L''_i(y)) \} \quad (\kappa^a(e) \in Z, \ a \in A(L''_i(y))), \\
\max \{ \lvert \kappa^a(e) \rvert : e \in A(L''_i(y)) \} &\leq 2^{j-1}. \\
\end{align*}
\]

If \( f, j(a, y) < c''(a, y) \), then we have \( \kappa^a_e(0) = 0 \ (e \in A(L''_i(y))) \), where \( Z \) is the set of integers, \( Z_+ \) is the set of nonnegative integers and
$c''_i(e,y) \in \mathbb{Z}_+ - \{0\}$ is the capacity of arc $e$ in $N''_i(y)$.

(Proof) We can prove (18) and (19) by induction on $j$. We note here that if $f_i(a,y) = c''_i(a,y)$ for some $a \in A(L''_i(y))$ and some $k \leq j$, then we have:

$f_i(a,y) = c''_i(a,y) \quad (k \leq d \leq j)$. □

**Proposition 7.** Let $(c''_1(a,y) : a \in A''_1)$ be capacity of the $i$-th residual network $N''_i(y)$, where $i \geq 2$. Then we have:

$$c''_i(a,y) = \sum \{ \psi^a_{e}(e,y) : e \in A' \} \quad (\psi^a_{e}(e) \in \mathbb{Z}, \; a \in A''_i),$$

$$\max \{| \psi^a_{e}(e) | : e \in A' \} \leq (m' + 1)^{i-2} 2^{u(i)},$$

where $u(i) = (i - 1)(2m' - i)/2$ and $m' = |A'|$.

(Proof) We use induction on $i$. From proposition 6, we have (20) and (21) for $i = 2$. Suppose that we carried out $J$ path-flow augmentations to find a maximal flow $(f_i, f_i(e,y) : e \in A(L''_i(y)))$ of $L''_i(y)$. From proposition 6 we have

$$f_i(a,y) = \sum \{ \kappa^a_{e}(e) c''_1(e,y) : e \in A(L''_1(y)) \} \quad (\kappa^a_{e}(e) \in \mathbb{Z}).$$

Then we have $$\max \{| \kappa^a_{e}(e) | : e \in A(L''_i(y)) - \tilde{F}_2 \} \leq 2^{i-1} \quad (a \in A(L''_i(y))).$$

From (22) ~ (24), inductive assumption, $|A(L''_i(y))| \leq m'$ and $J \leq m' - i + 1$, we have (20) and (21) replacing $i$ by $i + 1$. Note that

$$1 + m'(m' + 1)^{i-2} 2^{u(i+1)} \leq (m' + 1)^{i-1} 2^{u(i+1)}. \quad \Box$$

**Proposition 8.** Let $\rho(i) = (m' + 1)^{i-2} 2^{u(i)}$ in (21). Then we have:

$$\rho(i) \leq \rho(n - 1) = (m + n + 1)^{n-3} 2^{n-1} \quad (2 \leq i \leq n - 1),$$

where $v = (n - 2)(2m + n + 1)/2$.

(Proof) Let $p$ be the length of the shortest directed path from $s'$ to $t'$ of network $N''(y)$. From $p \geq 3, i \leq |V'|-1, |V'| = n + 2, m' \leq m + n$ and proposition 7, we have (25). □
PROPOSITION 9. If \( Y \neq -1 \) in \( \text{WAIT-A-MINUTE}(Y, y^o, y_o, \text{FLAG}0, N(y)) \), then we have \( Y = \tau \theta / \chi \) for some \( \chi \in \{ z \in \mathbb{Z}_+ : 0 < z \leq \theta m^{2^m+n} \rho(n-1) \} \) and some \( \tau \in \mathbb{Z}_+ \).

(Proof) Consider \( i \)-th layered network \( L''_i(y) \). Assume that we are going to do \( J \)-th path-flow augmentation. From (10) \sim (14) and proposition 7 we see that the solution \( Y \) is obtained from linear equation of \( Y \) such that

\[
\sum \{ \kappa_i^1(e) \alpha(e) : e \in A \} y + \tau_1 = \sum \{ \kappa_i^2(e) \alpha(e) : e \in A \} y + \tau_2,
\]

where \( \kappa_i^d(e) \in \mathbb{Z}_+ \), \( \kappa_i^d(e) \leq \rho(i)2^i \) and \( \tau_d \in \mathbb{Z} \) for \( d = 1, 2 \). From (4) we have

\[
\kappa'(a) \in \mathbb{Z}_+ - \{0\} \quad (a \in A) \text{ such that } \alpha(a) = \kappa'(a)/\theta \leq 1. \text{ Let}
\]

\[
\chi = \sum \{ \kappa_i^1(e) \kappa'(e) : e \in A \} - \sum \{ \kappa_i^2(e) \kappa'(e) : e \in A \}.
\]

Assuming \( \tau_2 = \tau_1 \geq 0 \) we have \( Y = \tau_2 \theta / \chi \). From (27), propositions 5 and 8 and \( \kappa'(e) \leq \theta \) (\( e \in A \)), we have \( \chi \leq \theta m^{2^m+n} \rho(n-1) \). \( \Box \)

PROPOSITION 10. \( \text{WAIT-A-MINUTE}(Y, y^o, y_o, \text{FLAG}0, N(y)) \) is carried out at most once for \( Y \neq -1 \).

(Proof) Assume that \( \text{WAIT-A-MINUTE}(Y, y^o, y_o, \text{FLAG}0, N(y)) \) is carried out twice for \( Y = y_1 \) and \( y_2 \), where \( y_1 \neq y_2 \), \( y_1 \neq -1 \) and \( y_2 \neq -1 \). From proposition 9, we have

\[
y_i = \tau_i \theta / \chi_i \quad (\tau_i \in \mathbb{Z}_+, \chi_i \in \{ z \in \mathbb{Z}_+ : 0 < z \leq \theta m^{2^m+n} \rho(n-1) \})
\]

where \( i = 1, 2 \). From (9) and proposition 8, we have

\[
|y_1 - y_2| \geq \frac{\theta}{(\theta m^{2^m+n} \rho(n-1))^2} = \gamma.
\]

From \( |y_1 - y_2| \leq y^o - y_o < \gamma \) and (29), we have a contradiction. \( \Box \)

Concerning the total complexity of algorithms \( I \) and \( II \), we have:

PROPOSITION 11. The total computational complexity of algorithms \( I \) and \( II \) is

\[
O(\max\{ \log c^*, m \log \eta^*, nm \} T(n, m)),
\]

where \( c^* = \max\{ c^o(a) : a \in A \} \), \( \eta^* = \max\{ \eta(a) : a \in A \} \) and \( T(n, m) \) is the time for the maximum flow computation for a two-terminal network with \( n \) vertices and \( m \) arcs.

(Proof) Consider algorithm \( I \). We have \( O(T(n, m)) \) time for each step 2. Let \( k \) be the number of repetitions of Step 2. From \( g^*/2^k < \gamma \) algorithm \( I \) takes

\[
O(\max\{ \log g^*, \log \theta, mn \} T(n, m)) \text{ time, where } g^* \text{ is the maximum flow value of network } N^*.
\]

From proposition 10 and [6] algorithm \( II \) requires \( O(n^2 m + T(n, m)) \) time. From \( g^* \leq mc^* \) and \( \theta \leq (\eta^*)^m \), we have this proposition. \( \Box \)

Now we show an example of our algorithm:

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
EXAMPLE: Consider network \( N = (G = (V, A), c^o, c^a, \alpha, \beta, s, t) \) with \( a^* = (t, s) \) in Fig.1, where \( a^* \not\in A \), \( V = \{s, 1, 2, t\} \) and \( A = \{a_i : 1 \leq i \leq 5\} \). The ordered triple attached to each \( a \in A \) is \((c^o(a), c^a(a), \alpha(a)y + \beta(a))\). We have \( b^o = 0, b^a = 20, \gamma = 12, \theta = 24 \) and \( \gamma = 1/(24 \times 25 \times 100 \times 2^{48}) \). In Fig.2 we have \( z = y \) and \( z = f^{**}(y) \). After Step 1 of algorithm I we have \( y^o = 12 \) and \( y^o = 0 \). Going to Step 2 we calculate value \( f^{**}(y) \) of network \( N(y) \) for \( y = (12 + 0)/2 = 6 \). From \( f^{**}(6) = 17/2 > 6 \), we put \( y^o = 6 \) and go to (2.1). Repeating Step 2, we finally have \( y^o = 9 + 1/3 \) and \( y^o = 9 + \xi \) \((\xi = (1 - 1/2^{63})/3)\).

![Fig.1](image1.png)  ![Fig.2](image2.png)

We have network \( N'(y) \) in Fig.3 and the layered networks \( L''_1(y) \) in Figs. 4-6, where the linear function of \( y \) beside each arc in each figure is the arc-capacity. From \( 1 \leq y - 3 \) \((y \in [9 + \xi, 28/3])\), we have \( L''_2(y) \) in Fig.5. Solving \( 1 - y/12 = 2y/3 - 6 \) in Fig.6, we have the optimal value \( f^* = 28/3 \).

![Fig.3](image3.png)
Acknowledgements

The author wishes to thank referees for pointing out a few errors in the earlier draft of this paper. He also thanks Professor Satoru Fujishige of University of Tsukuba for giving valuable suggestions on this paper.

References


Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Algorithm for Maximum Balanced Flows


Akira NAKAYAMA : Department of Management Sciences, Faculty of Commerce, Otaru University of Commerce, Otaru, Hokkaido, 047, Japan

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.