AN ALGORITHM FOR THE MAXIMUM BALANCED FLOW PROBLEM

Akira NAKAYAMA

Abstract

We consider the maximum balanced flow problem, which is a maximum flow problem with an additional constraint described in terms of a balancing rate function. In the present paper, we propose an algorithm for the maximum balanced flow problem by employing the binary search and a parametric method. Our algorithm requires $O((\lceil \log M \rceil + 2M)\, T(n, m))$time, where $T(n, m)$ is the time for the maximum flow computation for a two terminal network with n vertices and m arcs, and M is the maximum flow value of the maximum flow problem without the additional constraint (i.e., usual maximum flow problem), and $\lceil \log M \rceil$is the maximum integer less than or equal to $\log M$.

1. Introduction

M. Minoux [9] 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 the source s to the sink t. The maximum balanced flow problem is motivated by Minoux's research of reliability analysis of communication networks. If a flow from source s to sink 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. For example, consider a telephone network with its source and sink corresponding to two cities A and B, respectively. When a telephone line joining two adjacent spots breaks down, telephone routes through the broken line from A to B are blocked, but if the telephone routing considered as a flow from the source to the sink is balanced, then the number of the blocked routes is at most the fixed proportion of the total number of current routes from A to B. It is shown that few telephone lines break at the same time in Minoux's research [9]. If we have a maximum balanced flow in this network, a reliable telephone routing from A to B will be obtained.

Several algorithms [2], [6], [7], [9], [10] and [11] are proposed for the maximum balanced flow problem. W.-T. Cui [2] showed a variant of the dual simplex method without cycling on the underlying graph of a two terminal network, but his algorithm runs possibly in non-polynomial time. When a given balancing rate function is constant, Minoux's algorithm [9] and that of A. Nakayama [10] are known. The former requires $O((P_{\max})^2 S(n, m))$ computation time, where $P_{\max}$ is the maximum number of arc disjoint directed paths from source s to sink t in the underlying graph 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 $O(\min(m, \lfloor 1/r \rfloor)\ T(n, m))$ computation time, where r is a constant value of the constant balancing rate function and $T(n, m)$ is the time for the maximum flow computation for a two terminal network with n vertices and m arcs and $\lfloor 1/r \rfloor$ is the maximum integer less than or equal to $1/r$. For a general balancing rate function, U. Zimmermann [11] proposed a polynomial time algorithm with $O((T(n, m))^2)$ computation time.

On the other hand, T. Ichimori, H. Ishii and T. Nishida [6, 7] considered the weighted minimax flow problem, and S. Fujishige, A. Nakayama and W.-T. Cui [4] have pointed out the equivalence of the

maximum balanced flow problem and the weighted minimax flow problem. The algorithm [7] by Ichimori and Nishida assumes that for the set R of reals and the arc set A of the underlying graph $G=(V, A)$, the given positive capacity function $c: A\to R$ and weighted function $w: A\to R$ are integral, and it takes $O(T(n, m)P)$ time, where P is defined by $P=\lceil \log(\max\{c(a)w(a): a\in A\})\rceil$. The algorithm [6] by Ichimori, Ishii and Nishida has the computational complexity $O((T(n, m))^2)$ for general balancing rate functions, which is the same speed as Zimmermann's.

The objective of the present paper is to propose an algorithm which requires $O((\lceil \log M\rceil+2M)T(n, m))$ computation time,where M is the maximum flow value from the source to the sink of the maximum flow problem without the additional constraint described in terms of a balancing rate function (i.e., usual maximum flow problem), where we assume that a capacity function is rational. Our algorithm combines the binary search algorithm in [7] with the algorithm in [1] or [10] by paying attention to a particular monotonicity derived from a sequence of the capacities of cuts in the given network.

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 a capacity function $c: A\to R_+$, a balancing rate function $\alpha: A\to R_+-\{0\}$, consider the two terminal network $N=(G=(V, A), c, \alpha, s, t)$ where R_+ is the set of nonnegative reals, s is the source and t is the sink of G. Then the maximum balanced flow problem (P) by Minoux is formulated as follows.

(P): Maximize $f(a^*)$

subject to

(1) $Df=0$,

(2) $0\leqq f(a)\leqq c(a)\quad (a\in A)$,

(3) $f(a)\leqq \alpha(a)f(a^*)\quad (a\in A)$,

where $a^*=(t, s)\notin A$ is the arc added to the underlying graph G and D is the vertex-arc incidence matrix of G. Without loss of generality, we assume that $c(a)$ is an integer for any arc a of G.

If the function $f: A\cup\{a^*\}\to R_+$ satisfies (1) and (2), then f is called a feasible flow or a flow in network N. If a flow f also satisfies (3), then f is called a balanced flow. Let f^* be the value of maximum $f(a^*)$, and we call f^* the maximum balanced flow value or simply the optimal value.

Associated with the problem (P), consider the following problem (P^*) for network $N^*=(G=(V, A), c, s, t)$:

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

And we also consider the following parametric problem $(P(y))$ with a parameter y for network $N(y)=(G=(V, A), c, \alpha, y, s, t)$:

$(P(y))$: Maximize $f(a^*)$
subject to constraints (1), (2) and
(4) $f(a)\leqq\alpha(a)y \qquad (a\in A)$.

Let $f^{**}(y)$ be the value of maximum $f(a^*)$ in the network $N(y)$ and M be the value of maximum $g(a^*)$ in the network N^*. Then for the respective optimal values f^* and $f^{**}(y)$ of the problems (P) and $(P(y))$ we have:

Proposition 1: $f^*=\max\{y: f^{**}(y)=y\}$. □

From Proposition 1 and $f^{**}(0)=0$, we always have the optimal value of the original problem (P). Concerning the property of the function $f^{**}(y)$ of y, we have:

Proposition 2: $f^{**}(y)$ is a monotone non-decreasing, continuous, piece-wise linear and concave function. □

Any vertex partition (S, S^c) with $s\in S, t\in S^c$ is called a cut, where S^c is the complement of S. Then the capacity $c(S, S^c)$ of a cut (S, S^c) is defined as

(5) $c(S, S^c)=\sum\{c(a): a\in A^+(S)\}$,

where $A^+(S)=\{a=(i, j)\in A: i\in S, j\in S^c\}$. A minimum cut is defined to be a cut having the minimum capacity. Then we have the following theorem known as max-flow min-cut theorem.

Theorem 3[3]: For any network the maximum flow value from the source to the sink is equal to the capacity of a minimum cut. □

Let $(S, S^c; y)$ be a minimum cut in the network $N(y)$. Then from Theorem 3 and a minimum cut $(S, S^c; y)$ in the network $N(y)$, we have

(6) $f^{**}(y)=y\sum\{\alpha(a): a\in K'(S, y)\}+\sum\{c(a): a\in K''(S, y)\}$,

where $K'(S, y)=\{a\in A^+(S): c(a)>\alpha(a)y\}$ and $K''(S, y)=A^+(S)-K'(S, y)$. We define the slope-part $U(S, y)$ and the constant-part $W(S, y)$ of a minimum cut $(S, S^c; y)$ by

(7) $U(S, y)=\sum\{\alpha(a): a\in K'(S, y)\}$,

$W(S, y)=\sum\{c(a): a\in K''(S, y)\}$.

Note that the constant-part $W(S, y)$ is an integer by our assumption.

The function $f^{**}(y)$ is not differentiable in y, but for any $y>0$, we have the left differential coefficient $\sigma^-(y)$ defined as

(8) $\sigma^-(y)=\lim_{\Delta y\to 0-}\dfrac{f^{**}(y+\Delta y)-f^{**}(y)}{\Delta y}$.

Smilarly, we define the right differential coefficient $\sigma^+(y)$. $\sigma^+(y)$ $(\sigma^-(y))$ is called the right (left) slope of the function $f^{**}(y)$ at y. Then we have:

Proposition 4: $\sigma^-(y)=\max\{U(S, y): U(S, y)$ is the slope-part obtained from a minimum cut $(S, S^c; y)$ in the network $N(y)\}$. □

We define b^0 by

(9) $b^0=\max\{c(a)/\alpha(a): a\in A\}$.

Note that the value $c(a)/\alpha(a)$ is obtained from the equality case in the inequality $c(a)\geqq\alpha(a)y$.

3. Algorithm for the Maximum Balanced Flow Problem

Consider the two functions $z=f^{**}(y)$ and $z=y$ in a (y, z)-plane. From Proposition 1 we see that the optimal value is the maximum y^* such that (y^*, y^*) is an intersection point of the functions $z=f^{**}(y)$ and $z=y$. The outline of our algorithm to find such y^* is composed of the following two parts, though the detailed description will be shown in subsequent sections:

(10) Part 1: Let Y be a given real value and f^* the optimal value. By a binary method, we find y_0 and y^0 such that $y_0\leqq f^*<y^0$ and $y^0-y_0<Y$.

(11) Part 2: For the value y^0 obtained in Part 1, put $y^1=y^0$. Calculate the maximum flow value $f^{**}(y^1)$. If $y^1=f^{**}(y^1)$, then we have the optimal value $f^*=y^1$ and stop. Otherwise, find the line L defined in (6). Then we obtain the intersection point (y^2, y^2) of the two lines L and $z=y$. Compute $f^{**}(y^2)$, and if $y^2=f^{**}(y^2)$, then we have the optimal value $f^*=y^2$ and stop. Otherwise, we repeat Part 2 until we have $y^2=f^{**}(y^2)$.

3.1 Algorithm of Part 1.

The following proposition is easily obtained from (1), (2), (4) and Proposition 2.

Proposition 5: For the parametric problem $(P(y))$ we have:

(12) If there exist distinct values y' and y'' such that $y'>f^{**}(y')$ and $y''>f^{**}(y'')$ for the parametric problem $(P(y))$, then we have no optimal value between the values y' and y''.

(13) For any $y \geqq 0$, we have a (feasible) flow of the parametric problem $(P(y))$. □

Let Y be a given real number and f^* the optimal value, and the algorithm of Part 1 is as follows.

Algorithm of Part 1.

Step 1: (1.1) Find the maximum flow value M in network N^*. If $M \geqq b^0$, then we have the optimal value $f^* = M$ and stop.

(1.2) Calculate the maximum flow $f^{**}(y)$ of the network $N(y)$ for $y = M$. If $f^{**}(y) = y$, then stop. (We have the optimal value $f^* = M$.) Otherwise, put $y^0 = M$ and $y_0 = 0$.

Step 2: (2.1) If $Y > y^0 - y_0$, then stop.

(2.2) Put $y'' = (y^0 + y_0)/2$. Find the maximum flow value $f^{**}(y'')$, its slope-part $U(S, y'')$ and constant-part $W(S, y'')$ in the network $N(y'')$. Renew y^0 or y_0 as follows.

$y^0 = y''$ $(y'' > f^{**}(y''))$,

$y_0 = y''$ $(y'' \leqq f^{**}(y''))$.

Then go back to (2.1) of Step 2.

3.2 Algorithm of Part 2.

After the algorithm of Part 1 was over, we have y_0 and y^0 such that $y^0 - y_0 < Y$ and $y_0 \leqq f^* < y^0$ if we have not found the optimal value f^* yet. Now we state the algorithm of Part 2.

Algorithm of Part 2.

Step 1: (1.1) Put $y' = y^0$, where y^0 is the value obtained in the algorithm of Part 1.

(1.2) Calculate the maximum flow value $f^{**}(y')$, its slope-part $U(S, y')$ and its constant-part $W(S, y')$.

(1.3) If $y' = f^{**}(y')$, then we have the optimal value $f^* = y'$

and stop.

(1.4) Put $y' \leftarrow W(S, y')/(1-U(S, y'))$ and go back to (1.2) of Step 1.

Note that we have the slope-part $U(S, y')<1$.

4. Validity and Complexity.

For the algorithm of Part 1, the following proposition is guaranteed.

Proposition 6: By the algorithm of Part 1, we have the optimal value y such that $y^0>y\geqq y_0$ if we have not found the optimal value yet.

(Proof) It is easy to see that in Step 1, we have either the optimal value or a value y such that $y^0>y\geqq y_0$. Consider Step 2. Let $f^{**}(y'')$ be the maximum flow value in the network $N(y'')$ for $y''=(y^0+y_0)/2$ in (2.2). If $y''\leqq f^{**}(y'')$, then we have the optimal value y^* such that $y''\leqq y^*<y^0$. (Note that y'' is not always the optimal value when $y''=f^{**}(y'')$.) If $y''>f^{**}(y'')$, then from Proposition 5 we have no optimal solution y such that $y''\leqq y<y^0$. □

Let $Q(y)=(y, f^{**}(y))$ be the point on the graph of the function $f^{**}(y)$ at y. A point $Q(y'')$ is called the corner point of the function $z=f^{**}(y)$ if $f^{**}(y)$ is not differentiable at $y=y''$. A (line) segment $L_S(h', h'')$ on the graph $z=f^{**}(y)$ is $z=dy+b(h'\leqq y\leqq h'')$, where the two points $Q(h')$ and $Q(h'')$ are the adjacent corners. Then the following proposition can be obtained from Theorem 3 and Proposition 4.

Proposition 7: Let $L_S(h, h')$ and $L_S(h', h'')$ be the segments on the graph $z=f^{**}(y)$ such that $z=dy+b(h\leqq y\leqq h')$ and $z=d'y+b'(h'\leqq y\leqq h'')$.

(14) If $h<y<h'$, then we have $U(S, y)=\sigma^-(y)$.

(15) At the corner point $Q(h')$ we have $d'\leqq U(S, y)\leqq d$. □

Let $U(S^i, y^i)$ ($W(S^i, y^i)$) be the slope-part (the constant-part) of a minimum cut of the network $N(y^i)$ at i-th repetition of (1.2) ~ (1.4) of Step 1 in the part 2 algorithm, respectively. Our aim here is to show the constant-part $W(S^i, y^i)$ is an integer and a decreasing function as to i. First, we can prove the following proposition by Proposition 7 and the same method as in [10].

Proposition 8: The slope-part $U(S^i, y^i)$ is an increasing function with respect to the number i of repetitions of the part 2 algorithm.□

Second, for the constant-part $W(S^i, y^i)$ we have:

Proposition 9: The constant-part $W(S^i, y^i)$ is an integer and a decreasing function as to the number i of iterations of the part 2 algorithm.

(Proof) From our assumption and (7), we see that the constant-part $W(S, y)$ of the network $N(y)$ is an integer for any y ($y_0 \leqq y < y^0$).

Suppose that $y^1 > f^{**}(y^1)$ for some y^1 and that we have not found the optimal value yet. From (6) ~ (8), we have the line $L(y^1)$ with the slope-part $U(S^1, y^1)$ containing the point $Q(y^1)$ as follows.

(16) $L(y^1)$: $z = U(S^1, y^1)y + W(S^1, y^1)$.

Note that we have $f^{**}(y^1) = U(S^1, y^1)y^1 + W(S^1, y^1)$. From $U(S^1, y^1) < 1$, we can find the intersection point $R^2 = (y^2, y^2)$ of the two lines $L(y^1)$ and $z = y$.

Then we have

(17) $y^2 = W(S^1, y^1)/(1 - U(S^1, y^1))$.

Again, calculate the maximum flow value $f^{**}(y^2)$, the slope-part $U(S^2, y^2)$ and the constant-part $W(S^2, y^2)$. If $y^2 = f^{**}(y^2)$, then we have the optimal value y^2, so assume $y^2 > f^{**}(y^2)$. Then we have the line $L(y^2)$ with slope-part $U(S^2, y^2)$ containing the point $Q(y^2)$ as follows.

(18) $L(y^2)$: $z = U(S^2, y^2)y + W(S^2, y^2)$.

Note that we have $f^{**}(y^2)=U(S^2, y^2)y^2+W(S^2, y^2)$. From $y^2>f^{**}(y^2)$, the notes of (16) and (18) we have:

(19) $U(S^1, y^1)y^2+W(S^1, y^1)>U(S^2, y^2)y^2+W(S^2, y^2)$.

From Proposition 8, we have $W(S^1, y^1)>W(S^2, y^2)$. □

Let us reconsider the equation of (6) at i-th repetition of Step 1 of the part 2 algorithm and rewrite as follows, i. e.,

(20) $f^{**}(y^i)=U(S^i, y^i)(y^i-y_0)+(U(S^i, y^i)y_0+W(S^i, y^i))$.

By letting $x^i=y^i-y_0$, we have, in place of the function $f^{**}(y^i)$, the function $F^{**}(x^i)$ such that

(21) $F^{**}(x^i)=U(S^i, x^i+y_0)x^i+(U(S^i, x^i+y_0)y_0+W(S^i, x^i+y_0))$,

where $x^i\geqq y_0$. From Propositions 2, 8 and 9, we have

(22) $0\leqq U(S^i, x^i+y_0)<U(S^{i+1}, x^{i+1}+y_0)<1$,

(23) $W(S^i, x^i+y_0)-W(S^{i+1}, x^{i+1}+y^0)\geqq 1$.

Let $W'(S^i, x^i)=U(S^i, x^i+y_0)y_0+W(S^i, x^i+y_0)$ and M be the maximum flow value of the network N^*. Then we have the following proposition.

Proposition 10: If $U(S^{i+1}, x^{i+1}+y_0)-U(S^i, x^i+y_0)\leqq 1/(2M)$, then we have

$$W'(S^i, x^i)-W'(S^{i+1}, x^{i+1})\geqq 1/2.$$

(Proof) By the definition of $W'(S^i, x^i)$, we have

(24) $W'(S^i, x^i)-W'(S^{i+1}, x^{i+1})=(U(S^i, x^i+y_0)$
$-U(S^{i+1}, x^{i+1}+y_0))y_0+(W(S^i, x^i+y_0)-W(S^{i+1}, x^{i+1}+y_0))$.

From the assumption of this proposition, we have

(25) $U(S^i, x^i+y_0)-U(S^{i+1}, x^{i+1}+y_0)\geqq -1/(2M)$.

From (23), (24) and (25), we have

(26) $W'(S^i, x^i)-W'(S^{i+1}, x^{i+1})\geqq 1-y_0/(2M)$.

Since $M\geqq y_0$, we have our result. □

Concerning the total complexity of the algorithms of Part 1 and Part 2, we have:

Proposition 11: The over all computational complexity of the algorithm composed of Parts 1 and 2 is

$$O((\lceil \log M - \log Y \rceil + 2M + \lceil 2Y \rceil) T(|V|, |A|)),$$

where $T(|V|, |A|)$ is the time for the maximum flow computation for a two terminal network with $|V|$ vertices and $|A|$ arcs and for a real r, $\lceil r \rceil$ is the maximum integer less than or equal to r.

(Proof) First we consider the complexity of the algorithm of Part 1. Step 1 requires $O(T(|V|, |A|))$ time. As to Step 2, one cycle from (2.1) to (2.2) of Step 2 takes $O(T(|V|, |A|))$ time. Let j be the number of the repetitions of Step 2. We see that such j is the minimum number k such that $M/2^k \leqq Y$, where M is the maximum flow value of the network N^* and Y is the fixed width. Hence the computational complexity of the algorithm of Part 1 is $O(\lceil \log M - \log Y \rceil T(|V|, |A|))$. Second we consider the complexity of the algorithm of Part 2. We divide the rate of increase of the slope-part into the following two cases, i. e.,

(27) $U(S^i, x^i + y_0) - U(S^{i+1}, x^{i+1} + y_0) \leqq 1/(2M)$,

(28) $U(S^i, x^i + y_0) - U(S^{i+1}, x^{i+1} + y_0) > 1/(2M)$.

If we have the case (27) every time Step 1 of the part 2 algorithm is repeated, then from Proposition 10 we require at most k repetitions of the maximum flow computation such that $k/2 \geqq Y$. Such k is $\lceil 2Y \rceil$. If we have the case (28) every time Step 1 of the part 2 algorithm is repeated, then from (22) we need at most $2M$ maximum flow computations. Note that the slope-part $U(S^i, x^i + y_0)$ and the constant-part $W(S^i, x^i + y_0)$ are monotone functions with respect to the number i. Thus, the over-all computational complexity of our algorithms is

$$O((\lceil \log M - \log Y \rceil + 2M + \lceil 2Y \rceil) T(|V|, |A|)). \quad \square$$

The complexity of the composed algorithm depends on the fixed value Y, so we would like to minimize its complexity from computational point of view. Let $H(Y) = \lceil \log M - \log Y \rceil + 2M + \lceil 2Y \rceil$. Then we see that

the minimum of $H(Y)$ equals $H(1)$ as the following proposition shows.

Proposition 12: $\min\{H(Y): 0<Y<M\}=H(1)=\lceil \log M \rceil+2M+2$.

(Proof) Let us divide the fixed value Y into two cases.

Case 1: $Y\geqq 1$; We can put Y and M as follows.

(29) $Y=2^i+j+\theta$,

where $i=\max\{q: 2^q\leqq Y<2^{q+1}\}$, $j=\lceil Y\rceil-2^i$ and $\theta=Y-\lfloor Y\rfloor$.

(30) $M=2^{i'}+j'$,

where $i'=\max\{q: 2^q\leqq M<2^{q+1}\}$ and $j=M-2^{i'}$. From (29) and (30), we have

(31) $H(Y)\geqq(i'-i)+2M+2(2^i+j)$.

Note that $i'\geqq i$. It is easy to see that

(32) $-i+2(2^i+j)\geqq 2$ (i, j are non-negative integers).

Case 1.1: $j'=0$; From $\lceil \log M\rceil=i'$, we have $H(1)=i'+2M+2$. From (31) and (32), We have $H(Y)\geqq H(1)$.

Case 1.2: $j'\geqq 1$; From $\lceil \log M\rceil=i'+1$, we have $H(1)=i'+2M+3$. From

$-i+2(2^i+j)\geqq 3$ $(i\geqq 1)$, we have $H(Y)\geqq H(1)$ if $i\geqq 1$.

Assume that we have $i=0$ and $\theta>0$. Note that we have $j=0$ from (29).

Case 1.2.1: $j'=0$; From (30), we have $H(Y)\geqq i'+2M+3>i'+2M+2=H(1)$.

Case 1.2.2: $j'>0$; From (29) and (30), we have

(33) $j'-\theta>0$.

From (33), we have $H(Y)\geqq i'+2M+4>i'+2M+3=H(1)$.

Case 2: $Y<1$; If $Y>1/2$, then we have $H(Y)\geqq H(1)$ from $\log Y<0$. Otherwise, we have, from the monotonicity of the logarithm function,

(34) $\log Y\leqq \log Y'$ $(Y\leqq Y')$.

From (34) we have

(35) $\log M-\log Y \geqq \log M-\log Y' \quad (Y \leqq Y')$.

From the monotonicity of the function $f(x)=\lceil x \rceil$, we have

(36) $\lceil \log M-\log Y \rceil \geqq \lceil \log M-\log Y' \rceil \quad (Y \leqq Y')$.

From (36), we have

(37) $H(Y) \geqq H(1/2)=\lceil \log M \rceil+2M+2=H(1) \quad (Y \leqq 1/2)$.

Hence we have $H(Y) \geqq H(1) \quad (0<Y<M)$. □

From Propositions 11 and 12, we have the following theorem.

Theorem 13: The over all computationl complexity of the composed algorithm is $O((\lceil \log M \rceil+2M)T(|V|, |A|))$. □

REFERENCES

[1] Ahuja, R. K.:Algorithms for the Minimax Transportation Problem. *Naval Research Logistics Quarterly.* Vol. 33 (1986), 725-739.

[2] Cui, W.-T.: An algorithm for the Maximum Balanced Flow Problem. *Second Year Essay, Doctoral Program in Socio-Economic Planning,* University of Tsukuba, 1986.

[3] Ford, L. R. Jr. and Fulkerson, D. R.: *Flows in Networks.* Princeton University Press. Princeton, N. J., 1962.

[4] Fujishige, S., Nakayama, A. and Cui, X.-T.: On the Equivalence of the Maximum Balanced Flow Problem and the Weighted Minimax Flow Problem. *Operations Research Letters.* Vol. 5, No. 4 (1986), 207-209.

[5] Hu, T. C.: *Combinatorial Algorithms.* Addison-Wesley Publishing Company, 1982.

[6] Ichimori, T., Ishii, H. and Nishida, T.: Weighted Minimax Real-valued Flows. *Journal of the Operations Research Society of Japan.* Vol. 24, No.1 (1981), 52-60.

[7] Ichimori, T. and Nishida, T.: Finding the Weighted Minimax Flow in a Polynomial Time. *Journal of the Operations Research Society of Japan.* Vol. 23, No. 3 (1980), 268-271.

[8] Lawler, E. L.: *Combinatorial Optimization—Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[9] Minoux, M.: Flots Équilibres et Flots avec Sécurité. E.D.F.-*Bulletin de la Direction des Études et Recherches, Série* C-Mathématiques, Informatique, Vol. 1 (1976), 5-16.

[10] Nakayama, A.: A Polynomial Algorithm for the Maximum Balanced Flow Problem with a Constant Balancing Rate Function. *Journal of the Operations Research Society of Japan.* Vol. 29, No. 4 (1986), 400-410.

[11] Zimmermann, U.: Duality for Balanced Submodular Flows. Preprint No. 89, Fachbereich Mathematik, Universitat Kaiserslautern, 1985.