A Polynomial Approximation algorithm for the Separable Convex Cost Flow Problem

Masakatsu Higashikubo
Sumitomo Electric Industries,
Kitahama, Chuo-ku, Osaka 541, Japan

Masao Fukushima
Graduate School of Information Science,
Nara Institute of Science and Technology,
Ikoma, Nara 630–01, Japan

Toshihide Ibaraki
Department of Applied Mathematics and Physics,
Faculty of Engineering, Kyoto University,
Sakyo-ku, Kyoto 606, Japan

The minimum cost flow problem has many important applications in various fields. For the problem with a linear cost function, numerous algorithms have been developed, including those strongly polynomial time algorithms, which are recently proposed. If the cost function is separable, convex and quadratic, some polynomial time algorithms are also known. In particular, Minoux's algorithm is based on the scaling technique of Edmonds and Karp, and successively solves the piecewise linear approximation problems using the out-of-kilter method. In this paper we generalize Minoux's algorithm to the problem with a general separable convex cost function, which may not be quadratic. We show that the proposed algorithm finds an ϵ-optimal approximate solution in polynomial time $O\left(mn^2 \log_2 \frac{mKU}{\epsilon}\right)$, where m and n are the numbers of arcs and nodes, respectively, U is the maximum absolute value of capacities, K is a Lipschitz constant of the first derivative of the cost function, and $\epsilon > 0$ represents the allowable difference between the cost function values of the obtained approximate solution and the exact optimal solution. We also report some numerical results with the proposed algorithm.

1 INTRODUCTION

Let $G = (N, A)$ be the given graph, where N is the set of n nodes and A is the set of m arcs. For each arc $a \in A$, we have a flow x_a, integer upper and lower bounds u_a and l_a, respectively, and a real cost function $c_a(x_a)$. We denote the flow vector by $x = (x_a; a \in A) \in R^m$ and the node-arc incidence matrix by $E \in R^{m \times n}$. Then the minimum cost flow problem can be written as

$$\begin{aligned} \text{P}: \quad \text{minimize} \quad & \sum_{a \in A} c_a(x_a) \\ \text{subject to} \quad & Ex = 0 \\ & l_a \leq x_a \leq u_a \quad \text{for all} \quad a \in A. \end{aligned}$$

When the functions c_a are all linear, the problem is a linear minimum cost flow problem, for which various algorithms have been proposed. In particular, the network simplex method [2] is effective enough to solve very large problems. On the other hand, there have recently been presented many algorithms that can solve linear minimum cost flow problem in polynomial time. Edmonds and Karp [3] proposed a polynomial time algorithm using a scaling technique for the first time. Since then, various scaling algorithms have been presented, including capacity scaling algorithms [3, 4], cost scaling algorithms [5, 16], and others [1, 5]. In particular, the strongly polynomial algorithm proposed by Orlin [14] is currently considered the fastest in the sense of computational complexity. There are other algorithms that can solve linear minimum cost flow problems in polynomial time. These include the ellipsoid method [9] and interior point methods [8, 10, 17], which are designed to solve general linear programming problems.

Interior point algorithms have also been extended to solve separable quadratic cost flow problems. Kapoor and Vaidya [7], Ye and Tse [18] independently proposed an extension of Karmarkar's interior point method to the separable convex quadratic cost flow problem. Another polynomial algorithm for quadratic cost problems was proposed by Minoux [12]. This algorithm is based on the scaling technique of Edmonds and Karp [3] and successively solves piecewise linear approximation problems using the out-of-kilter method.

In this paper we generalize Minoux's algorithm to the case of general convex costs.

Under the Lipschitz continuity assumption on the first derivatives of cost functions, we can show that the proposed algorithm finds an ϵ-optimal solution of the minimum convex cost flow problem in polynomial time. We also report some numerical experiments with the proposed method.

This paper is organized as follows. In section 2, we propose successive piecewise linear approximations of problem P, and an algorithm to solve the approximate problems. Moreover we present some properties of approximate problems and the algorithm. In section 3, we show the polynomiality of the algorithm proposed in section 2. Some numerical experiments of the proposed method is given in section 4. Finally, in section 5, we consider the case where the cost function is convex by comparing the results of Minoux [12].

2 SUCCESSIVE PIECEWISE LINEAR APPROXIMATIONS

In this section, we present an algorithm that successively solves piecewise linear approximations of problem P.

Let π_i be a potential for node $i \in N$, and $\theta_a = \pi_j - \pi_i$ be a potential difference (tension) for each arc $a = (i,j) \in A$. Then a kilter diagram [11], or a characteristic curve [15], for each arc $a \in A$ is the set

$$\begin{aligned} &\{(x_a, \theta_a) \mid x_a = u_a \ \text{ and } \ \theta_a \geq g_a(u_a)\} \\ \bigcup\ &\{(x_a, \theta_a) \mid x_a = l_a \ \text{ and } \ \theta_a \leq g_a(l_a)\} \\ \bigcup\ &\{(x_a, \theta_a) \mid \theta_a = g_a(x_a) \ \text{ for } \ l_a \leq x_a \leq u_a\}, \end{aligned}$$

where $g_a(x_a) = \mathrm{d}c_a(x_a)/\mathrm{d}x_a$ (see Fig.1).

In the following, we assume that the first derivatives g_a of the the cost functions are Lipschitz continuous on the interval $[-U, U]$ for all $a \in A$, where $U = \max_{a \in A} \{\, |u_a|, |l_a| \}$. That is, there exists a constant $K > 0$ such that

$$|g_a(x) - g_a(y)| \leq K|x - y| \ \text{ for all } \ x, y \in [-U, U]\,. \tag{2.1}$$

Note that since the arc cost function c_a is assumed to be convex, the kilter diagram is nondecreasing, and that the arc cost function c_a is obtained by the integral

$$c_a(x_a) = \int_{l_a}^{x_a} g_a(\xi)d\xi. \tag{2.2}$$

It is known [15] that a flow x and a potential $\pi = (\pi_i; i \in N) \in R^n$ satisfy the flow conservation equation $Ex = 0$ and (x_a, θ_a) is on the kilter diagram for each arc $a \in A$ if and only if x solves problem P and π solves the dual of P. Therefore our aim is to find a pair (x, π) such that (x_a, θ_a) are on the kilter diagram for all arcs $a \in A$. However, since the cost functions of problem P are convex, the kilter diagram for each arc is a nonlinear curve as shown in Fig.1. Because it is in general not easy to find such (x, π) directly, we consider successive approximations of arc cost function c_a using piecewise linear functions, which graduately tend to the original function c_a. Since the kilter diagrams of such approximate cost functions become the form of staircase, the out-of-kilter method for piecewise linear minimum cost problems [11] can be used to solve the approximate problems.

The kth approximate problem of P, which we call $\mathrm{P}^{(k)}$, is defined in the following manner. Let $p = \lceil \log_2 U \rceil - k + 1$. We define for each arc $a \in A$ the lower and upper bounds $l_a^{(k)}$ and $u_a^{(k)}$ by

$$l_a^{(k)} = 2^p \left\lfloor \frac{l_a}{2^p} \right\rfloor, \quad u_a^{(k)} = 2^p \left\lceil \frac{u_a}{2^p} \right\rceil, \tag{2.3}$$

and the function $g_a^{(k)}$ by

$$g_a^{(k)}(x_a) = \begin{cases} g_a\left(2^p \lceil x_a/2^p \rceil\right) & \text{for} \quad l_a^{(k)} + t2^p < x_a < l_a^{(k)} + (t+1)2^p \\ & \left(t = 0, 1, \cdots, \left(u_a^{(k)} - l_a^{(k)}\right)/2^p - 1\right) \\ \left[g_a\left(x_a\right), g_a\left(x_a + 2^p\right)\right] & \text{for} \quad x_a = l_a^{(k)} + t2^p \\ & \left(t = 1, 2, \cdots, \left(u_a^{(k)} - l_a^{(k)}\right)/2^p - 1\right) \\ \left(-\infty, g_a\left(l_a^{(k)} + 2^p\right)\right] & \text{for} \quad x_a = l_a^{(k)} \\ \left[g_a\left(u_a^{(k)}\right), +\infty\right) & \text{for} \quad x_a = u_a^{(k)}, \end{cases} \tag{2.4}$$

which approximates the first derivatives g_a of arc cost function c_a of problem P. Note that the definition of $g_a^{(k)}$ implies that

$$g_a^{(k)}(x_a) \geq g_a(x_a) \text{ for all } x_a \in \left[l_a^{(k)}, u_a^{(k)}\right], a \in A. \tag{2.5}$$

Fig.2 illustrates an example of approximate functions $g_a^{(k)}$ for the function g_a of Fig.1.

Using function $g_a^{(k)}$, we may define the arc cost function $c_a^{(k)}$ for problem $\mathrm{P}^{(k)}$ as

$$c_a^{(k)}(x_a) = \int_{l_a^{(k)}}^{x_a} g_a^{(k)}(\xi) d\xi. \tag{2.6}$$

Now we describe an algorithm for P, which successively finds solutions of the approximate problems $\mathrm{P}^{(k)}$ for $k = 1, 2, \cdots, k_{max}$, where k_{max} is the maximum number of iterations. (In the next section, we shall discuss how to determine the value of k_{max}.)

Algorithm 1.

(a) Let $k := 1$, $x^{(0)} := 0$, and $\pi^{(0)} := 0$.

(b) Find a solution pair $(x^{(k)}, \pi^{(k)})$ of problem $\mathrm{P}^{(k)}$ by the out-of-kilter method with $(x^{(k-1)}, \pi^{(k-1)})$ as an initial solution.

(c) If $k = k_{max}$, stop. Otherwise set $k := k + 1$ and return to (b).

We consider some properties of approximate problems $\mathrm{P}^{(k)}$.

Lemma 1. (1) *When $k \geq \lceil \log_2 U \rceil + 1$, we have $l_a^{(k)} = l_a$ and $u_a^{(k)} = u_a$ for all $a \in A$, i.e., problems $P^{(k)}$ and P have the same lower and upper bounds for each arc $a \in A$ (see Fig.2).*

(2) *For $k \geq \lceil \log_2 U \rceil + 1$, any optimal solution of $P^{(k)}$ satisfies the capacity bound conditions of P.*

(3) *For $k \geq \lceil \log_2 U \rceil + 1$, we have the inequalities*

$$c_a^{(k)}(x_a) \geq c_a(x_a) \geq c_a^{(k)}(x_a) - KU^2 \cdot 2^{-k+2} \quad \textit{for all} \quad x_a \in [l_a, u_a], \tag{2.7}$$

where K is the Lipschitz constant for the first derivative g_a of arc cost function c_a.

Proof. Since $p = \lceil \log_2 U \rceil - k + 1$ by definition, (1) and (2) are clear from the definition (2.3) of $l_a^{(k)}$ and $u_a^{(k)}$. Next we prove (3). The first inequality in (2.7) is obvious from (2.2), (2.5) and (2.6).

To see the second inequality, let us choose $x_a \in [l_a, u_a]$ arbitrarily. Then (2.2)–(2.6) imply

$$\begin{aligned} & c_a^{(k)}(x_a) - c_a(x_a) \\ = & \int_{l_a}^{x_a} \left[g_a^{(k)}(\xi) - g_a(\xi) \right] d\xi \qquad \text{(by (2.2) and (2.6))} \\ \leq & \int_{l_a}^{u_a} \left[g_a^{(k)}(\xi) - g_a(\xi) \right] d\xi \qquad \text{(by (2.5) and } \ x_a \leq u_a) \\ = & \sum_{t=0}^{(u_a - l_a)/2^p - 1} \int_{l_a + t2^p}^{l_a + (t+1)2^p} \left[g_a^{(k)}(\xi) - g_a(\xi) \right] d\xi. \end{aligned} \tag{2.8}$$

For each $t = 0, 1, \cdots, (u_a - l_a)/2^p - 1$, we have from the definition (2.4) of $g_a^{(k)}$

$$g_a^{(k)}(\xi) = g_a^{(k)}(l_a + (t+1)2^p) \ \text{ for } \ \xi \in [l_a + t2^p, l_a + (t+1)2^p)\,. \tag{2.9}$$

Also we have by the monotonicity of g_a,

$$g_a(\xi) \geq g_a(l_a + t2^p) \ \text{ for } \ \xi \in [l_a + t2^p, l_a + (t+1)2^p]\,, \tag{2.10}$$

and by the Lipschitz condition (2.1) for g_a,

$$g_a(l_a + (t+1)2^p) - g_a(l_a + t2^p) \leq K2^p. \tag{2.11}$$

Then (2.9) – (2.11) imply

$$\begin{aligned} & \int_{l_a + t2^p}^{l_a + (t+1)2^p} \left[g_a^{(k)}(\xi) - g_a(\xi) \right] d\xi \\ \leq & \ 2^p \left(g_a(l_a + (t+1)2^p) - g_a(l_a + t2^p) \right) \\ \leq & \ 2^p \cdot K2^p = K2^{2p}. \end{aligned} \tag{2.12}$$

Then by (2.8) and (2.12)

$$\begin{aligned} c_a^{(k)}(x_a) - c_a(x_a) & \leq \frac{u_a - l_a}{2^p} \cdot K2^{2p} \\ & = (u_a - l_a) \cdot K2^p \\ & \leq KU2^{p+1}. \end{aligned}$$

Since $p = \lceil \log_2 U \rceil - k + 1$, we have the second inequality of (2.7). This completes the proof. $\square$

Recall that, at each iteration of Algorithm 1, we use the out-of-kilter method to solve the approximate problem $\mathrm{P}^{(k)}$ with $\left(x^{(k-1)}, \pi^{(k-1)}\right)$ as an initial solution. (In particular, (x, π)=(0,0) is chosen as the initial solution to solve problem $\mathrm{P}^{(1)}$.) Then we may show the following properties.

Lemma 2. (1) *For all k, the obtained solutions $x^{(k)}$ of $P^{(k)}$ maintain the flow conservation equation $Ex = 0$ and are integer multiples of 2^p, where $p = \lceil \log_2 U \rceil - k + 1$.*
(2) *For each k, $P^{(k)}$ can be solved in $O(mn^2)$ time.*

This lemma is a direct modification of the results obtained by Minoux [12] for the quadratic case. In fact, the present lemma can be proved in essentially the same manner as Property 3 in [12], so we omit the proof here.

3 POLYNOMIALITY OF THE ϵ-APPROXIMATION ALGORITHM

In this section we show that an ϵ-optimal solution of problem P can be obtained in polynomial time by Algorithm 1 described in section 2.

Let x^* be an optimal solution of problem P. Then a feasible flow x of P is said to be *ϵ-optimal* if

$$c(x) - c(x^*) \leq \epsilon,$$

where $c(x)$ is the cost function of problem P, i.e.

$$c(x) = \sum_{a \in A} c_a(x_a).$$

We shall denote the cost functions of the approximate problems $\mathrm{P}^{(k)}$ by

$$c^{(k)}(x) = \sum_{a \in A} c_a^{(k)}(x_a).$$

Then we have the following lemma.

Lemma 3. *For any $k \geq \lceil \log_2 U \rceil + 1$, $x^{(k)}$ is a feasible solution for problem P and satisfies*

$$0 \leq c(x^{(k)}) - c(x^*) \leq \delta^{(k)}, \tag{3.13}$$

where $\delta^{(k)} = mKU^2 \cdot 2^{-k+2}$.

Proof. By Lemma 1(2), when $k \geq \lceil \log_2 U \rceil + 1$, $x^{(k)}$ satisfies the upper and lower bound conditions for P. Moreover, by Lemma 2(1), $x^{(k)}$ satisfies the flow conservation equation $Ex = 0$ for each k. Therefore $x^{(k)}$ is feasible to problem P.

The first inequality of (3.13) is obvious by the optimality of x^* and the feasibility of $x^{(k)}$. Next we show the second inequality. From Lemma 1(3) we have the inequalities (2.7). By taking summation of (2.7) for all arcs $a \in A$ and letting $\delta^{(k)} = mKU^2 \cdot 2^{-k+2}$, we get

$$c^{(k)}(x) \geq c(x) \geq c^{(k)}(x) - \delta^{(k)}. \tag{3.14}$$

Substituting $x = x^{(k)}$ in (3.14) yields

$$c^{(k)}(x^{(k)}) \geq c(x^{(k)}), \tag{3.15}$$

while substituting $x = x^*$ in (3.14) gives

$$c(x^*) \geq c^{(k)}(x^*) - \delta^{(k)}. \tag{3.16}$$

On the other hand, since $x^{(k)}$ is optimal to $\mathrm{P}^{(k)}$, we have

$$c^{(k)}(x^*) \geq c^{(k)}(x^{(k)}). \tag{3.17}$$

Then (3.16) and (3.17) imply

$$c(x^*) \geq c^{(k)}(x^{(k)}) - \delta^{(k)}. \tag{3.18}$$

Finally, from (3.15) and (3.18) we have the desired inequality. This completes the proof. □

The next theorem gives a proper value of the maximum number of iterations, k_{max}, for Algorithm 1 to obtain an ϵ-optimal solution of problem P.

Theorem 1. *For an arbitrary $\epsilon > 0$, Algorithm 1 finds an ϵ-optimal solution for P in $k_{max} = \lceil \log_2 (4mKU^2/\epsilon) \rceil$ iterations.*

Proof. Our aim is to find the least value of k such that

$$c(x^{(k)}) - c(x^*) \leq \epsilon. \tag{3.19}$$

From (3.13) in Lemma 3, we have

$$c(x^{(k)}) - c(x^*) \leq \delta^{(k)} \quad \text{for all} \quad k \geq \lceil \log_2 U \rceil + 1,$$

where $\delta^{(k)} = mKU^2 2^{-k+2}$. So it is enough to find k such that

$$\delta^{(k)} \leq \epsilon,$$

that is,

$$mKU^2 2^{-k+2} \leq \epsilon.$$

This implies

$$k \geq \log_2 \frac{4mKU^2}{\epsilon}.$$

Therefore, (3.19) holds for any k such that $k \geq k_{max}$, where

$$k_{max} = \left\lceil \log_2 \frac{4mKU^2}{\epsilon} \right\rceil.$$

This completes the proof. □

Now we establish the polynomial complexity of Algorithm 1.

Theorem 2. *Algorithm 1 finds an ϵ-optimal solution of P in $O\left(mn^2 \log_2 \frac{mKU}{\epsilon}\right)$ time.*

Proof. Combining Lemma 2(2) and Theorem 1, we have the polynomial complexity

$$O\left(mn^2\right) \times \left\lceil \log_2 \frac{4mKU^2}{\epsilon} \right\rceil = O\left(mn^2 \log_2 \frac{mKU}{\epsilon}\right).$$

This completes the proof. □

Moreover we consider the case where problem P has integrality restrictions on the flows. That is, we consider the integer minimum cost flow problem IP. The following theorem, which corresponds to Theorem 1 in Minoux [13], shows that Algorithm 1 can solve problem IP in polynomial time. The proof of the theorem is similar to that of [13] and hence omitted here.

Theorem 3. *Algorithm 1 finds an optimal integer solution of IP in $O\left(mn^2 \log_2 U\right)$ time.*

4 NUMERICAL EXPERIMENTS

In this section, we report some numerical experiments with the proposed method. All computer codes were written in FORTRAN 77 and run on a SUN SPARC1 IRX. We

generated test problems as follows. We first randomly generated a network with a predetermined density. Then we give the upper and lower bounds for each arc by generating random integers uniformly distributed in the intervals [0,10] and [0,50], respectively. Finally we determine the cost function $c_a(x_a)$ for each arc $a \in A$ as $c_a(x_a) = \alpha x_a^3$, where α is a random integer in the interval [1,10]. This implies that the Lipschitz constant K for $g_a'(x_a)$ ($= c_a''(x_a)$) is 60. The numbers in the following tables are the averages of five test examples for each problem size.

First we examine the effectiveness of the scaling technique. That is, we compare the proposed scaling method with the out-of-kilter method directly applied to a piecewise linear approximation problem with a given accuracy. Table 1 summarizes the results of numerical experiments for two problem sizes $(n, m) = (30, 271)$ and $(n, m) = (50, 750)$. In Table 1, the value of ϵ is computed as $\epsilon = mKU2^{p+1}$, and #potential and #flow respectively denote the numbers of times potentials and flows are updated. Fig.3 and Fig.4 illustrate how the average CPU time grows with the attained accuracy by the proposed scaling method and the direct out-of-kilter method, respectively. The obtained results show the effectiveness of scaling in terms of the CPU time and the numbers of potential and flow updates in the out-of-kilter method. In particular, we may observe that the relative efficiency of the proposed scaling method increases with the required accuracy, as observed by Minoux [12] for his algorithm for quadratic costs. Note that ϵ is given by $\epsilon = mKU2^{p+1}$, we see from Fig.3 that the CPU time of the proposed scaling method is almost linear with respect to $\log_2 \frac{1}{\epsilon}$. This exhibits the complexity of the proposed method shown in Theorem 2.

Next we examine how the computation becomes costly as the number of arcs increases. For this purpose, we have generated some test problems with a fixed number of nodes and various numbers of arcs. The results are summarized in Table 2 and Fig.5, in which the density is given by $m/(n(n-1)/2)$, and the desired accuracy is set to 2^{-8}. In particular, Fig.5 suggests that the rate of increase in the computational cost is slightly higher than linear, which is also supported by the complexity result shown in Theorem 2.

Finally we have compared the proposed method with a dual-based Newton method presented in [6]. As a result, we have found that the latter method could solve most problems much faster than the proposed method. For instance, the problems with

(n, m) =(30,174), (30,348), (50,490), and (50,980) were solved in about 20%, 10%, 8%, and 4%, respectively, of the CPU times spent by the proposed method.

Table 1. Effectiveness of the scaling technique

(a)$n = 30, m = 144$

		Proposed scaling method		Direct out-of-kilter method	
accuracy		CPU	#potential	CPU	#potential
2^p	ϵ	(sec.)	/#flow	(sec.)	/#flow
2^0	13218.31	0.8	248/79	2.0	639/180
2^{-1}	6551.20	0.9	276/85	4.0	1298/364
2^{-2}	3267.04	1.0	309/91	8.5	2685/726
2^{-3}	1630.36	1.1	347/97	17.0	5321/1433
2^{-4}	814.44	1.2	386/103	34.6	10726/2868
2^{-5}	407.06	1.4	417/110	70.7	21638/5744
2^{-6}	203.49	1.5	455/116	143.9	43130/11467
2^{-7}	101.73	1.6	490/123	295.8	86429/22932
2^{-8}	50.86	1.8	541/129	593.5	173075/45842

(b)$n = 50$,$m = 370$

		Proposed scaling method		Direct out-of-kilter method	
accuracy		CPU	#potential	CPU	#potential
2^p	ϵ	(sec.)	/#flow	(sec.)	/#flow
2^0	29706.00	4.8	581/176	12.0	1479/364
2^{-1}	14747.21	5,3	651/189	25.3	3141/723
2^{-2}	7345.37	6.0	739/203	53.0	6658/1450
2^{-3}	3666.68	6.7	825/218	109.4	13664/2908
2^{-4}	1831.74	7.3	908/231	223.1	27538/5802
2^{-5}	915.44	8.0	993/245	451.6	55231/11543
2^{-6}	457.62	8.8	1097/255	943.0	113702/22990
2^{-7}	228.78	9.5	1176/272	1914.0	222721/46248
2^{-8}	114.38	10.3	1273/283	3865.1	445139/92529

$\epsilon = mKU2^{p+1}$

#potential $\cdots$ the number of times potentials are updated

#flow $\cdots$ the number of times flows are updated

Table 2. Checking the proposed complexity in term of the number of arcs.

	$n = 20$			$n = 30$		
density (%)	m	CPU (sec.)	#potential /#flow	m	CPU (sec.)	#potential /#flow
20	38	0.2	210/31	87	0.8	422/70
40	76	0.5	280/80	174	2.3	594/160
60	114	0.9	350/110	261	4.3	709/229
80	152	1.6	432/159	348	6.9	820/305
100	190	2.0	502/185	435	9.9	889/367

	$n = 40$			$n = 50$		
density (%)	m	CPU (sec.)	#potential /#flow	m	CPU (sec.)	#potential /#flow
20	156	2.4	724/134	245	5.4	1029/199
40	312	6.4	914/241	490	14.7	1306/353
60	468	12.8	1154/372	735	27.5	1532/490
80	624	20.2	1297/471	980	44.3	1823/666
100	780	28.4	1410/557	1225	65.6	1925/805

density $= m/(n(n-1)/2) \times 100$

#potential $\cdots$ the number of times potentials are updated

#flow $\cdots$ the number of times flows are updated

5 CONCLUDING REMARKS

In this paper we have generalized Minoux's algorithm to the case of general convex costs. We have shown that the proposed algorithm finds an ϵ-optimal solution of the minimum convex cost flow problem in polynomial time $O\left(mn^2 \log_2 \frac{mKU}{\epsilon}\right)$. We have also reported some numerical experiments with the proposed method, and verified its computainal complexity. The computational results show that the proposed method significantly improves upon a direct application of the out-of-kilter method to piecewise linear approximation problems. From a practical viewpoint, however, it turns out that the proposed method is by no means the best, as exemplified by the comparison with the method of [12].

Finally we remark that the separable quadratic convex cost problem QP can be regarded as a special case of general convex cost problem P. For problem QP, Minoux [12] shows that an feasible flow x satisfying the following inequality may yield an exact optimal solution of QP:

$$|x - x^*| \leq \frac{1}{2\Delta^2}, \tag{5.20}$$

where x^* is the optimal solution of QP, $\Delta = K^m(3m+n-1)^{3m+n-1}$, and $|\cdot|$ denotes the Euclidean norm. Moreover, it is shown in [12] that (5.20) holds whenever the following inequality is satisfied.

$$c(x) - c\,(x^*) \leq \frac{1}{2}\left(\frac{1}{2\Delta^2}\right)^2. \tag{5.21}$$

Therefore if we apply Algorithm 1 with $\epsilon = \frac{1}{2}\left(\frac{1}{2\Delta^2}\right)^2$, then we may find an exact solution of problem QP. By Theorem 1, the required number of iterations then becomes

$$1 + \lceil \log_2 mK \rceil + 2\lceil \log_2 U \rceil + 4\lceil \log_2 \Delta \rceil\,. \tag{5.22}$$

Note that, in the case of quadratic costs, the Lipschitz constant K represents the twice of the maximum coefficient in the second order terms of cost functions.

On the other hand, Minoux [12] shows that the number of iterations for his algorithm to satisfy (5.20) is at least

$$1 + \lceil \log_2 mK \rceil + \lceil \log_2 U \rceil + 4\lceil \log_2 \Delta \rceil\,. \tag{5.23}$$

Comparing (5.22) and (5.23), we may observe that Algorithm 1, when applied to the quadratic cost problem QP, has essentially the same computational complexity as the algorithm of Minoux [12].

References

[1] F. Barahona and E. Tardos, Note on Weintraub's minimum-cost circulation algorithm. *SIAM J. Comput.* **18** (1989) 579–583.

[2] G. Bradley, G. Brown, and G. Graves, Design and implementation of large scale primal transshipment algorithms. *Management Sci.* **24** (1977) 1–34.

[3] J. Edmonds and R. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Computing Machinery* **19** (1972) 248–264.

[4] Z. Galil and E. Tardos, An $O(n^2(m+n\log n)\log n)$ minimum cost flow algorithm. *J. Assoc. Computing Machinery* **35** (1988) 374–386.

[5] A. Goldberg and R. Tarjan, Finding minimum-cost circulations by canceling negative cycles. *J. Assoc. Computing Machinery* **36** (1989) 873–886.

[6] S. Ibaraki, M. Fukushima, and T. Ibaraki, Dual-based Newton methods for nonlinear minimum cost flow problems. *J. Operations Res. Soc. Jpn.* **34** (1991) 263–286.

[7] S. Kapoor and P. Vaidya, Fast algorithm for convex quadratic programming and multicommodity flows. In *Proceedings of the Eighteenth Annual ACM Symposium of the Theory of computing* (1986) 147–159.

[8] N. Karmarkar, A new polynomial time algorithm for linear programming. *Combinatorica* **4** (1984) 373–395.

[9] L. Khachian, Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics* **20** (1980) 53–72.

[10] M. Kojima, S. Mizuno, and A. Yoshise, An $O(\sqrt{n}L)$ iteration potential reduction algorithm for linear complementarity problems. *Math. Prog.* **50** (1991) 331–342.

[11] E. Lawler, *Combinatorial Optimization : Network and Matroids*, Holt, Rinehart and Winston, New York (1976).

[12] M. Minoux, A polynomial algorithm for minimum quadratic cost flow problems. *Eur. J. Operational Res.* **18** (1984) 377–387.

[13] M. Minoux, Solving integer minimum cost flows with separable convex cost objective polynomially. *Math. Prog. Study* **26** (1986) 237–239.

[14] J. Orlin, A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the Twentieth Annual ACM Symposium of the Theory of computing* (1988) 377–387.

[15] R. Rockafellar, *Network Flows And Monotropic Optimization*, John Wiley, New York (1984) 312–387.

[16] E. Tardos, A strongly polynomial minimum cost circulation algorithm. *Combinatorica* **53** (1985) 247–255.

[17] P. Vaidya, An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)L)$ arithmetic operations. In *Proceedings of the Nineteenth Annual ACM Symposium of the Theory of computing* (1987) 29–38.

[18] Y. Ye and E. Tse, An extension of Karmarkar's projective algorithm for the convex quadratic programming. *Math. Prog.* **44** (1989) 157–179.