アルゴリズム 25－1
（1992．1．24）

最小費用流問題に対する Speculative Contraction 法: 実用的なアルゴリズムへ向けて

藤重 悟*, 岩野 和生†, 中野 淳†, 手塚 集†

* 筑波大学社会工学系
† 日本ＩＢＭ 東京基礎研究所

最小費用流問題に対する Orlin の強多項式アルゴリズムの変形である Plotkin-Tardos のアルゴリズムに *speculative contraction* と呼ばれる手法を用いて，実際にインプリメントした際の高速化を図る．Δ scaling phase において Plotkin-Tardos のアルゴリズムは $5n\Delta$ 以上のフローが流れる枝を縮約するが，speculative contraction 法ではこれよりかなり小さい閾値で縮約を実行し，その際に生じる primal infeasibility については後処理で解消する．実験結果によれば疎なグラフに対しては 3.1 倍の高速化が確認され，また speculative contraction 法は primal-dual 法にも計算時間でまさった．

A Speculative Contraction Method for Minimum Cost Flows: Toward a Practical Algorithm

Satoru Fujishige*, Kazuo Iwano†, Jun Nakano†, and Shu Tezuka†

* Institute of Socio-Economic Planning, University of Tsukuba
Tsukuba, Ibaraki 305
† IBM Research, Tokyo Research Laboratory, IBM Japan
5-11 Sanbancho, Chiyoda-ku, Tokyo 102

We propose a new heuristic method, called a *speculative contraction method*, for minimum cost flows based on the Plotkin and Tardos algorithm, a variant of Orlin's algorithm. In a Δ scaling phase, the Plotkin and Tardos algorithm contracts arcs of flow value more than $5n\Delta$. However, the speculative contraction method contracts arcs of flow value much smaller than $5n\Delta$, and corrects the possible primal infeasibility caused by inappropriate contractions at the end. Our experiments show that the new method significantly reduces the running time of the original algorithm. In particular, for sparse graphs, it shows a speed up of 3.1. Moreover, it runs faster than our implementation of the Primal-Dual method. We hope our experimental results on speculative contractions invoke further research toward theoretically and practically faster scaling algorithms.

1 Introduction

The minimum cost flow problem is an important network flow problem and has been widely studied for the last four decades. (See surveys in [2, 10].) In particular, there have been emerging theoretical developments since Tardos invented the first strongly polynomial time algorithm for this problem [18].

Many of recent algorithms are based on the scaling methods on cost or capacity. For example, the cost scaling algorithms include Tardos [18], Galil and Tardos [9], and Goldberg and Tarjan [11], while the capacity (excess) scaling algorithms include Orlin [15], Fujishige [7], Orlin [16], and Plotkin and Tardos [17]. Ahuja, Goldberg, Orlin, and Tarjan devised the currently fastest polynomial time algorithm, $O(nm \log\log U \log(nC))$ time, based on a double scaling method [1]. Among strongly polynomial algorithms Orlin's $O(m \log n(m + n \log n))$ time algorithm [16] is the fastest known algorithm. Here, n (*resp.* m) denotes the number of nodes (*resp.* arcs), and U (*resp.* C) is the maximum capacity (*resp.* cost) in an input network.

Despite recent theoretical developments, very little has been known about the practical efficiency of recent scaling algorithms. DIMACS, the Center for Discrete Mathematics and Theoretical Computer Science, thus, organized the competition of implementing recent network flow algorithms [4].

In the above context, we propose a new heuristic method based on Orlin's algorithm, and perform experiments on its practical efficiency. Plotkin and Tardos' variant [17] of Orlin's algorithm is simple, does not assume the strong connectivity of an input graph, and runs in the same complexity as Orlin's algorithm does. Therefore, in this paper we discuss Plotkin and Tardos' variant (hereafter, we call it PT) instead of Orlin's original algorithm.

An excess scaling algorithm PT consists of Δ phases starting from $\Delta = U$ to 0, and it reduces Δ by at least a half at the end of each phase. During a Δ phase the algorithm contracts an arc which carries flow exceeding $5n\Delta$. In contrast, we propose a *speculative contraction*, which contracts an arc of flow value much less than $5n\Delta$. Since it may contract an arc inappropriately, it may result in a negative flow value on some arc. Thus, we adjust these negative flow values by using the Primal-Dual algorithm.

According to our experimental results, a speculative contraction method significantly reduces the original running time (we observe a factor of about 3.1 speed up for sparse graphs). Moreover, it runs faster than our implementation of the Primal-Dual method. We hope that our proposal of speculative contractions invoke further research toward scaling algorithms that are faster in theory and practice.

2 Preliminaries

We basically follow Plotkin and Tardos' notations in [17]. Let $G = (V, A)$ be a directed graph with $|V| = n$ nodes and $|A| = m$ arcs. For the sake of simplicity, we assume there is no parallel or opposite arcs. We define two real-valued functions: one is an arc *cost* function $\gamma : A \longrightarrow R$, while the other is a node *demand* function $d : V \longrightarrow R$ such that $\sum_{v \in V} d(v) = 0$. For $v, w \in V$ with $(v, w) \in A$, we define $\gamma(w, v) = -\gamma(v, w)$. We also define a real-valued arc *capacity* function $c : A \longrightarrow R$. We call a tuple $\mathcal{N} = (G, \gamma, d, c)$ a *network*. In particular, when every arc capacity is unconstrained, that is, $c(a) = \infty$ for each $a \in A$, we call $\mathcal{N}$ a *transshipment network*. A non-negative real-valued function f on A is called a *pseudoflow*. For a pseudoflow f, we define the excess of each node v by $ex_f(v) = \sum_{(w,v) \in A} f(w, v) - \sum_{(v,w) \in A} f(v, w) - d(v)$. A pseudoflow f is called a *transshipment* if the excess of every node with respect to f is 0; that is, a flow conservation rule is satisfied at each node. The cost of a pseudoflow f is $\gamma(f) = \sum_{a \in A} \gamma(a) f(a)$.

Given a pseudoflow f, we define a *residual network with respect to* f by $\mathcal{N}_f = (G_f = (V, A_f), \gamma, d, c_f)$

such that $c_f(v,w) = \infty$ for all $(v,w) \in A$. $c_f(v,w) = f(w,v)$ for $(v,w) \in V \times V$ where $(w,v) \in A$. $c_f(v,w) = 0$ otherwise. Let $A_f = \{(v,w) \mid c_f(v,w) > 0, v,w \in V\}$.

Here, we can define the *transshipment problem* as the problem of finding a minimum cost transshipment when given a transshipment network. The following theorem on the optimality of a transshipment is well known. Before introducing the theorem, we need a few more definitions. A *node price function* (or *potential function*) p is a real-valued function defined on V. The *reduced cost function* γ_p *with respect to a price function* p is defined by $\gamma_p(v,w) = \gamma(v,w) + p(v) - p(w)$.

Theorem 2.1 [6] *A transshipment is optimal if and only if there exists a price function p which satisfies (1) the dual feasibility constraints: for each arc $a \in A$, $c_p(a) \geq 0$ and (2) the complementary slackness constraints: for each arc $a \in A$, if $c_p(a) > 0$, then $f(a) = 0$.* □

We can now introduce Plotkin and Tardos' algorithm (PT) as shown in Figures 1 and 2. Algorithm PT takes a transshipment network $\mathcal{N}$ as an input. Notice that we define $S_f(\Delta) = \{v \in V \mid ex_f(v) > \Delta\}$ and $T_f(\Delta) = \{v \in V \mid ex_f(v) < -\Delta\}$. Algorithm PT consists of excess scaling phases starting from the maximum absolute value of node demands and ending at 0. In the Δ-phase, the algorithm repeats augmentations until $S_f(\frac{n-1}{n}\Delta) \cup T_f(\frac{n-1}{n}\Delta) = \emptyset$, it contracts arcs of flow value more than $5n\Delta$ by $CONTRACT(5n\Delta)$, and finally it resets Δ by at least a half. The procedure $AUGMENT(S,T,\Delta)$ first finds a shortest path tree starting from nodes in S with respect to a reduced cost γ_p, and then augments flow from nodes in S to nodes in T along with the disjoint minimum cost paths in the tree. At the end of algorithm PT, the procedure *UNFOLD* reverses the contraction process to unfold the network and compute the resulting arc flows. Since every node is involved in augmentations at most $O(\log n)$ times before being contracted, there are at most $O(n \log n)$ phases. Therefore, the algorithm PT attains a strongly polynomial time, $O(n \log n(m + n \log n))$. See the detailed discussion in [17].

3 Speculative Contractions

In this section we introduce our heuristic method called a *speculative contraction*, and briefly discuss our implementation.

As observed in [16] and [17], once an arc carries a flow of more than $5n\Delta$ at a Δ phase, a flow on the arc will never vanish in the succeeding scaling phases. Therefore, algorithm PT contracts such an arc, and reduces the size of a graph. In [17], $5n\Delta$ is determined by the maximum possible flow value to a single arc. That is, at each phase there are at most $2n$ augmentations of value Δ, whose summation over the succeeding phases is $4n\Delta$, and at the end of the algorithm there is at most $n\Delta$ flow coming from the contracted node. However, it is unlikely that flow of the estimated $5n\Delta$ units gathers to a single node. Therefore, we propose a *speculative contraction*, which contracts an arc of flow value much less than $5n\Delta$, and we may expect this heuristic method to work well for some value $\beta\Delta$ (e.g. $2\Delta, 4\Delta, 8\Delta, \ldots$).

One drawback of speculative contractions is that they may result in an infeasible flow f: that is, for some arc a, $f(a)$ may be negative. In order to resolve the infeasibility, we use the implementation shown in Figure 3. First, we run algorithm PT with a contraction threshold value $\beta\Delta$. Let f be the obtained flow function. Notice that there may exist negative arc flow values. We denote them by the set $A_{f-} = \{a \in A \mid f(a) < 0\}$. We now define a new network $\mathcal{N}_g = \{G_g, \gamma, d_g, c_g\}$ where a flow g is defined as $g(a) = f(a)$ for $a \notin A_{f-}$ and 0 for $a \in A_{f-}$, and a new demand function d_g is defined as $d_g(v) = \sum_{(v,w)\in A_{f-}} f(v,w) - \sum_{(w,v)\in A_{f-}} f(v,w)$ for each $v \in V$. Lastly, we solve new minimum cost flow problem $\mathcal{N}_g$ by the Primal-Dual method.

Notice that there is a temptation to regard a network $\mathcal{N}_g$ as a residual graph which appears in the middle of execution of PT, and resume the execution of PT for $\mathcal{N}_g$ with an increased contraction threshold value to find an optimal flow. However, this method may not work due to the following facts. In order to resume an execution of PT from a Δ-phase, it is required that every residual capacity should be an integral multiple of Δ. This condition may not be sustained by $\mathcal{N}_g$. Let Δ be the maximum absolute value of excesses in $\mathcal{N}_g$, and let c_{min} be the minimum non-zero absolute value of residual capacities. Then c_{min} may be too small to be an integral multiple of Δ. Therefore, we find an optimal solution by regarding $\mathcal{N}_g$ as a new instance of the capacitated minimum cost flow problem. Thus, we use the Primal-Dual method for solving $\mathcal{N}_g$. In fact, Step (2.2) of *SPECULATIVE* can be replaced with other minimum cost flow algorithms like the network simplex method.

Procedure *Plotkin-Tardos*
(1) $\Delta \leftarrow \max_{v \in V} |d(v)|$;
(2) **while** $\Delta \neq 0$ **do begin**
(2.1) **while** $S_f(\frac{n-1}{n}\Delta) \cup T_f(\frac{n-1}{n}\Delta) \neq \emptyset$ **do begin**
(2.2) **if** $S_f(\frac{n-1}{n}\Delta) \neq \emptyset$
(2.3) **then** $AUGMENT(S_f(\frac{n-1}{n}\Delta), T_f(\frac{1}{n}\Delta), \Delta)$;
(2.4) **else** $AUGMENT(S_f(\frac{1}{n}\Delta), T_f(\frac{n-1}{n}\Delta), \Delta)$;
end
(3) $CONTRACT(5n\Delta)$;
if f is zero on all uncontracted arcs
(4) **then** $\Delta \leftarrow \max_{v \in \tilde{V}} |ex_f(v)|$, where $\tilde{V}$ denotes the set of contracted nodes;
(5) **else** $\Delta \leftarrow \Delta/2$;
end
(6) $f \leftarrow UNFOLD$;

Figure 1: Algorithm *Plotkin-Tardos* [17].

Procedure *AUGMENT (S,T,Δ)*
(1) $\pi(s) \leftarrow 0$ for each $s \in S$; Find a shortest path tree starting from nodes in S with respect to a reduced cost γ_p in G_f. Let $\pi(v)$ be the obtained cost in the tree for each $v \in V$.
(2) $p(v) \leftarrow p(v) + \pi(v)$ for each $v \in V$.
(3) For each $s \in S$, if a subtree of the shortest path tree containing s includes a node $t \in T$, move Δ units of flow from s to t along with a tree path.

Figure 2: Algorithm *AUGMENT*.

Procedure *SPECULATIVE*
(1) Run *Plotkin-Tardos* with a contaction threshold $\beta\Delta$.
Let f be the obtained flow, and $A_{f-} = \{a \in A \mid f(a) < 0\}$.
(2) **if** $A_{f-} \neq \emptyset$ **then begin**
(2.1) Create a netwrok $\mathcal{N}_g$;
(2.2) Run Primal-Dual for $\mathcal{N}_g$;
end

Figure 3: Algorithm *SPECULATIVE*

Hereafter, we denote *SPECULATIVE* (*resp. Plotkin-Tardos*) with a contraction threshold $\beta\Delta$ by *SPECULATIVE*($\beta\Delta$) (*resp.* $PT(\beta\Delta)$). Notice that the Primal-Dual method, which we denote by PD, can be regarded as *SPECULATIVE*(0).

We wish to investigate the following issues on the speculative contraction method: (1) How practical and effective is this new method? (2) How can we determine an appropriate contraction threshold? (3) What are the theoretical issues regarding this method? We consider theses questions in the next section.

4 Experimental Results and Observations

4.1 Specifications

Instance generators: We employ an instance generator, *netgen* [14] for creating transportation networks and transshipment networks based on random seeds.

Types of networks: We prepare the following families of networks:

The density/size family: We use *netgen* to produce a family of (n, m) transshipment networks where n (*resp.* m) is the number of nodes (*resp.* arcs). We pick $n = 200, 400, 800, 1600$, and $m = 4n, n\log_2 n, n^2/16, n^2/4$.

The total-supply family [4]**:** We use *netgen* to produce a family of $(n, m, total)$ transportation networks where $(n, m, total) = (nodes, arcs, totalsupply)$. We use $m = 8n$, $total = 10^3, 10^5, 10^7, 10^9$ for $n = 200, 400$.

The supply/demand ratio family [4]**:** We use *netgen* to produce a family of (n, m, s, t) transportation networks where $(n, m, s, t) = (nodes, arcs, sources, sinks)$. We use $m = 8n$, and vary (s, t) as described later for $n = 200, 400$.

Performance measurement: We use the following measurements: CPU times of the whole execution, the execution of PT, and major subroutines. We use the **-pg** option of the C compiler in order to obtain the CPU time of each subroutine call.

Data structure: We use a binary heap for implementing Dijkstra's shortest path algorithm.

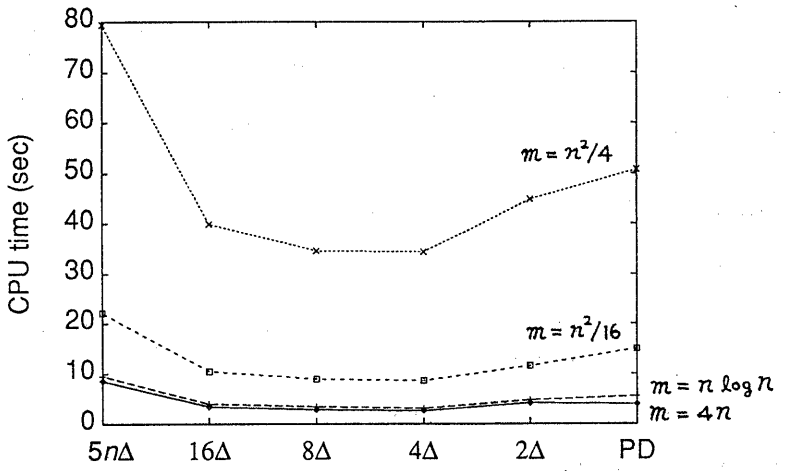
Contraction threshold: For each network, we run *Primal-Dual*, and also run *SPECULATIVE* for contraction thresholds of $5n\Delta, 16\Delta, 8\Delta, 4\Delta$, and 2Δ.

4.2 Experimental Results

The density/size family: Table 1 shows results of a test family generated by *netgen* for $n = 800$ and $m = 4n$, $n\log_2 n$, $n^2/16$. Each entry consists of the total CPU time without I/O in seconds, a percentage of the CPU time spent for post-processing PD, and a speed up factor of *SPECULATIVE*($\beta\Delta$), (that is, CPU-time(*SPECULATIVE*($\beta\Delta$))/CPU-time($PT(5n\Delta)$)). Table 2 shows average speed up ratios obtained from complete test families generated by *netgen* for $n = 200, 400, 800$, and 1600. Figure 4 illustrates the CPU times for networks of size $n = 400$ for different contraction thresholds.

As Figure 4 and Tables 1 and 2 show, our speculative contraction method outperforms both of the original Plotkin-Tardos' algorithm and the Primal-Dual method. The average speed up of *SPECULATIVE*(4Δ) to the Plotkin-Tardos (*resp.* Primal-Dual) is about 3.1 (*resp.* 1.5) for sparse graphs and about 2.4 (*resp.* 1.9) for dense graphs.

Figure 4: The density family ($n = 400$)



Surprisingly enough, this *optimal contraction threshold value* $\beta\Delta$, which makes our speculative contraction method fastest, seems to be a constant between 4Δ and 8Δ regardless of network density, size, supply/demand ratio, total supply, and network generators. We hope further theoretical research may explain this phenomenon.

Notice that as $CPU(PD)\%$ in Table 1 shows the above speed up comes mainly from the speed up of $PT(\beta\Delta)$, not from the post-processing of PD. In particular, for some small $\beta\Delta$, $PT(\beta\Delta)$ can obtain an optimal solution without involving a post-processing of PD. For example, $PT(16\Delta)$ can find optimal solutions for $n = 800$ with $m = 3200, 5347$.

When we increase the network size from $n = 200$ to 1600 while keeping the density same as $m = 4n$, we observe that the logarithm of CPU time grows almost linearly with respect to the logarithm of n (see Figure 5 in [8]). This means that the CPU time is proportional to some fixed power of n (say, $CPU = cn^{f(\beta\Delta)}$). By linear fitting, we find that $f(8\Delta) = 1.77$, and it seems to be slightly smaller than $f(5n\Delta) = 1.97$ and $f(PD) = 2.08$.

Table 1: The density family for $n = 800$

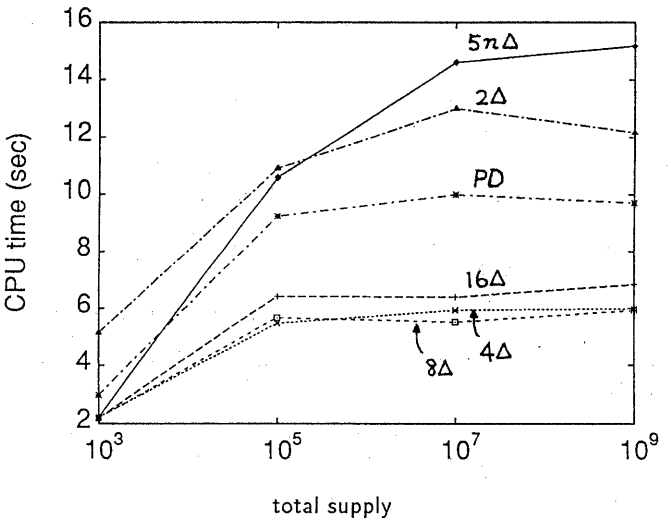
m	Threshold	$5n\Delta$	16Δ	8Δ	4Δ	2Δ	PD
3200	CPU(total)	33.45	11.07	9.94	10.06	17.84	16.77
	CPU(PD)%	0%	3%	8%	27%	64%	100%
	Speed up	1.00	.33	.30	.30	.53	.50
5347	CPU(total)	40.56	14.36	12.64	13.58	23.44	24.40
	CPU(PD)%	0%	2%	8%	28%	61%	100%
	Speed up	1.00	.35	.31	.33	.58	.60
40000	CPU(total)	157.82	65.49	58.00	61.42	93.64	125.87
	CPU(PD)%	0%	1%	5%	20%	51%	100%
	Speed up	1.00	.42	.37	.39	.59	.80

The total supply family: Figure 5 shows the growth rate of the total CPU time when we fix a network

Table 2: Average speed up ratios for the density/size families

m	$5n\Delta$	16Δ	8Δ	4Δ	2Δ	PD
$4n$	1.00	.37	.32	.32	.51	.49
$n \log_2 n$	1.00	.40	.37	.36	.60	.53
$n^2/16$	1.00	.52	.47	.42	.55	.80

Figure 5: The total supply family ($n = 400$)



configuration (that is, $n = 400$, the supply/demand ratio is $(n/2, n/2)$) and change the total supply as 10^3, 10^5, 10^8, 10^9. As a result, CPU times of $SPECULATIVE$ for 4Δ, 8Δ, and 16Δ grow much slower and saturate earlier than those of $PT(5n\Delta)$ and PD.

The supply/demand ratio family: We first choose $(s, t) = (n/8, n/8)$, $(n/8, n/2)$, $(n/2, n/8)$, $(n/2, n/2)$. The contraction thresholds 4Δ and 8Δ work well, whose speed up ratios range from 1.6 to 3.3 to the Plotkin-Tardos and do not vary very much by changing supply/demand ratio. We also vary the number of sources (*resp.* sinks) from 50, to 250 by 50 while keeping the number of sinks (*resp.* sources) the same as 150 for networks of $n = 400$. We observe that the CPU-time increases up to 1.9 times as the number of sinks increases, while it remains roughly the same within 1.1 times when the number of sources increases. This difference seems to be caused by the asymmetry of the implementation of our algorithm regarding to sources and sinks.

5 Concluding Remarks

We have proposed a speculative contraction method for minimum cost flows based on the Plotkin-Tardos algorithm, a variant of Orlin's algorithm. According to our experiments, very small contraction thresholds like 4Δ and 8Δ outperform the original Plotkin and Tardos' algorithm with a $5n\Delta$ contraction threshold and the Primal-Dual method. We observed an average speed up factor to the Plotkin-Tardos algorithm (*resp.* the Primal-Dual method) of 3.1 (*resp.* 1.5) for sparse graphs and 2.4 (*resp.* 1.9) for dense graphs.

One surprising thing is that this optimal contraction threshold seems to be invariant with respect to the density, size, supply/demand ratio, total supply, and types of networks. We hope our experimental results on speculative contractions will invoke further theoretical and practical research toward practical scaling algorithms.

References

[1] R. K. Ahuja, A. V. Goldberg, J. B. Orlin, and R. E. Tarjan, Finding Minimum-cost Flows by Double Scaling, *Technical Report CS-TR-164-88*, Department of Computer Science, Princeton University, 1988.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows, in G. L. Nemhauser et al., Eds., *Handbooks in OR & MS, Vol.1*, Elsevier Science Publishers B.V. (North-Holland), pp.211–369, 1989.

[3] D. P. Bertsekas and P. Tseng, Relaxation methods for minimum cost ordinary and generalized network flow problems, *Operations Research*, 36, pp.93–114.

[4] DIMACS, The First DIMACS International Algorithm Implementation Challenge: Network Flows and Matching, Call for Participation, General Information, and The Core Experiments. DIMACS, NJ., 1990.

[5] J. Edmonds and R. M. Karp, Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *Journal of ACM*, 19, pp.248–264, 1972.

[6] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[7] S. Fujishige, A Capacity-rounding Algorithm for the Minimum Cost Circulation Problem: A Dual Framework of the Tardos Algorithm, *Mathematical Programming*, 35, pp.298–309, 1986.

[8] S. Fujishige, K. Iwano, J. Nakano and S. Tezuka, A Speculative Contraction Method for Minimum Cost Flows: Toward a Practical Algorithm, *IBM Research Report RT 0061*, 1991.

[9] Z. Galil and É. Tardos, An $O(n^2(m+n\log n)\log n)$ Min-cost Flow Algorithm, *Proc. 27th Annual Symposium on the Foundation of Computer Science*, pp.136–146, 1986.

[10] A. V. Goldberg, É. Tardos, and R. E. Tarjan, Network Flow Algorithms, in *Paths, Flows, and VLSI-Layout*, Algorithmica and Combinatorics 9, B. Korte et al., Eds., Springer-Verlag, New York, NY, pp.101–164, 1990.

[11] A. V. Goldberg and R. E. Tarjan, Finding Minimum-Cost Circulations by Successive Approximation, *Technical Report CS-TR-106-87*, Dept. Computer Science, Princeton University, July 1987.

[12] A. V. Goldberg and R. E. Tarjan, Finding Minimum-Cost Circulations by Canceling Negative Cycles, *Journal of ACM*, Vol.36, No.4, pp.873–886, 1989.

[13] M.D. Grigoriadis, An Efficient Implementation of the Network Simplex Method, *Mathematical Programming Study*, 26, pp.83–111, 1986.

[14] D. Klingman, A. Napier, and J. Stutz, NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems, *Management Science*, Vol.20, No.4, pp.814–821, 1974.

[15] J. B. Orlin, Genuinely Polynomial Simplex and Non-simplex Algorithms for the Minimum Cost Flow Problem, *Technical Report No.1615-84*, Sloan School of Management, M. I. T., Cambridge, MA., 1984.

[16] J. B. Orlin, A Faster Strongly Polynomial Minimum Cost Flow Algorithm, *Journal of ACM*, Vol.35, No.2, pp.374–386, 1988.

[17] S. A. Plotkin and É. Tardos, Improved Dual Network Simplex, *Proceeding of the first annual ACM-SIAM Symposium on Discrete Algorithms*, pp.367–376, 1990.

[18] É. Tardos, A Strongly Polynomial Minimum Cost Circulation Algorithm, *Combinatorica*, 5, pp.247–255, 1985.