2003年日本オペレーションズ・リサーチ学会
春季研究発表会

New Maximum Flow Algorithms by MA Orderings and Scaling

Osaka University FUJISHIGE Satoru
Osaka University *ISOTANI Shigueo

1. Introduction

Maximum adjacency (MA) ordering has effectively been applied to graph connectivity problems by Nagamochi and Ibaraki [3]. We show an application of MA ordering to the maximum flow problem to get a new polynomial-time algorithm. For a flow network with n vertices, m arcs, and integral arc capacities $c(a)$ $(\leq U)$ our MA ordering algorithm [2] finds a maximum flow by $O(n \log nU)$ augmentations, in $O(n(m + n\log n)\log nU)$ time. We also propose a scaling version of our algorithm together with its variants. The scaling algorithms require $O(mn \log U)$ time. Moreover, we give some computational results.

2. Definitions

Let $\mathcal{N} = ((V, A), s^+, s^-, c)$ be a flow network, where (V, A) is a directed graph with a vertex set V and an arc set A, $s^+ \in V$ a source, $s^- \in V$ a sink, and $c : A \to \mathbb{Z}_+$ a capacity function taking on nonnegative integers. A function $\varphi : A \to \mathbb{Z}_+$ is a *flow* in $\mathcal{N}$ and the *value* of the maximum flow is denoted by $\hat{v}(\varphi)$.

Given a flow φ in $\mathcal{N}$, a *residual network* $\mathcal{N}_\varphi = ((V, A_\varphi), s^+, s^-, c_\varphi)$ is defined by $A_\varphi = A^+_\varphi \cup A^-_\varphi$, where $A^+_\varphi = \{a \mid a \in A, \varphi(a) < c(a)\}$ and $A^-_\varphi = \{\bar{a} \mid a \in A,\ 0 < \varphi(a)\}$ $(\bar{a}:$ a reorientation of $a)$; and

$$c_\varphi(a) = \begin{cases} c(a) - \varphi(a) & (a \in A^+_\varphi) \\ \varphi(\bar{a}) & (a \in A^-_\varphi). \end{cases}$$

3. A Maximum Flow Algorithm Using MA Orderings

Suppose that we are given a flow φ in $\mathcal{N}$. For any flow ψ in the residual network $\mathcal{N}_\varphi$ such that $a \in A^+_\varphi$ and $\bar{a} \in A^-_\varphi$, where $\bar{a}$ is a reorientation of a, imply $\psi(a) = 0$ or $\psi(\bar{a}) = 0$, we define a flow $\varphi \oplus \psi$ in the original network $\mathcal{N}$ by

$$\varphi \oplus \psi(a) = \begin{cases} \varphi(a) + \psi(a) & (a \in A^+_\varphi,\ \psi(a) > 0) \\ \varphi(a) - \psi(\bar{a}) & (\bar{a} \in A^-_\varphi,\ \psi(\bar{a}) > 0) \\ \varphi(a) & (\text{otherwise}). \end{cases}$$

The value of the new flow $\varphi \oplus \psi$ in $\mathcal{N}$ increases by the value of ψ in $\mathcal{N}_\varphi$. While ordinary augmenting path algorithms choose an appropriate flow ψ along a single directed path for each augmentation, we will make an augmentation by a multiple-path flow ψ found by an MA ordering. An MA ordering from s^+ to s^- in $\mathcal{N}$ is obtained as follows.

Procedure MA-Ordering($\mathcal{N}_\varphi$)

Step MA0: Put $i \leftarrow 0$ and $b(u) \leftarrow 0$ for each $u \in V$. Also put $W \leftarrow \{s^+\}$ and $v_0 \leftarrow s^+$. For each $u \in V$ let L_u be an empty list.

Step MA1: For each $w \in V \setminus W$ with $(v_i, w) \in A_\varphi$ put $b(w) \leftarrow b(w) + c_\varphi(v_i, w)$ and add arc (v_i, w) to list L_w.

Step MA2: Let v_{i+1} be a vertex that attains the maximum of $b(w)$ $(w \in V \setminus W)$. If $v_{i+1} = s^-$, then return $(v_0{=}s^+, v_1, \cdots, v_{i+1}{=}s^-)$, b, and L_u $(u \in V)$, and otherwise put $W \leftarrow W \cup \{v_{i+1}\}$ and $i \leftarrow i + 1$ and go to Step MA1.

Now our MA ordering algorithm for maximum flows is described as follows.

A Maximum Flow Algorithm

Step 0: Put $\varphi(a) \leftarrow 0$ for each $a \in A$.

Step 1: Perform MA-Ordering($\mathcal{N}_\varphi$). Let k be a positive integer such that $v_k = s^-$. Put $\delta \leftarrow \min\{b(v_j) \mid j = 1, 2, \cdots, k\}$. If $\delta = 0$, then return φ (a maximum flow) and otherwise put $\beta(s^-) \leftarrow \delta$ and $\beta(u) \leftarrow 0$ for each $u \in V \setminus \{s^-\}$.

Step 2: For each $a \in A_\varphi$ put $\psi(a) \leftarrow 0$.
For $i = k,\ k-1,\ \cdots,\ 1$ do the following:
For each arc (u, v_i) in list L_{v_i}
$\psi(u, v_i) \leftarrow \min\{\beta(v_i), c_\varphi(u, v_i)\}$
$\beta(v_i) \leftarrow \beta(v_i) - \psi(u, v_i)$
$\beta(u) \leftarrow \beta(u) + \psi(u, v_i)$

Step 3: Put $\varphi \leftarrow \varphi \oplus \psi$ and go to Step 1.

We examine the complexity of our algorithm. Note that the time required for Procedure MA-Ordering is $O(m + n\log n)$ by adapting Dijkstra's shortest path algorithm with the Fibonacci heap. Then, Step 1 requires $O(m + n\log n)$ time. And also note that Step 2 and Step 3 require $O(m)$ time. We consider how many times the cycle of Steps 1~3 is repeated.

Lemma 1 *Suppose that* $\delta > 0$ *in* Step 1. *Then* ψ *computed in* Step 2 *has a value not less than* $(\hat{v}(\varphi^*) - \hat{v}(\varphi))/n$, *where* φ^* *is a maximum flow in* $\mathcal{N}$. □

Lemma 1 shows that, denoting by $\varphi^{(i)}$ the flow φ computed at the end of the ith execution of Step 3, we have $\hat{v}(\varphi^*) - \hat{v}(\varphi^{(i+1)}) \leq (1 - \frac{1}{n})(\hat{v}(\varphi^*) - \hat{v}(\varphi^{(i)}))$. This implies that every $O(n)$ iterations of Steps 1~3 at least halve the difference $\hat{v}(\varphi^*) - \hat{v}(\varphi)$. Since initially $\hat{v}(\varphi^*) - \hat{v}(\varphi) \leq nU - 0$ where U denotes the maximum arc capacity in $\mathcal{N}$ and since φ computed while executing our algorithm is integer-valued, our algorithm finds a maximum flow by repeating Steps 1~3 $O(n \log nU)$ times. Hence, we have

Lemma 2 *Our MA-ordering max-flow algorithm finds a maximum flow by repeating* Steps 1~3 $O(n \log nU)$ *times and requires in total* $O(n(m + n \log n) \log nU)$ *time.* □

4. Scaling Algorithms

A scaling version of our algorithm is given as follows [2]. Starting from $\delta = U$, instead of performing MA-Ordering we expand W to $W \cup \{v\}$ for $v \in V \setminus W$ if $b(v) \geq \delta$. When W can not be expanded, we replace δ by $\lfloor \delta/2 \rfloor$ and continue the algorithm until $\delta = 1$. Other part of the scaling algorithm is exactly the same as our original one. The scaling algorithm requires $O(mn \log U)$ time without using sophisticated data structures such as the Fibonacci heap.

As a variant of the scaling algorithm, instead of putting $\delta \leftarrow \lfloor \delta/2 \rfloor$ when we cannot expand W in a current scaling phase, we put $\delta \leftarrow \max\{\lfloor \sigma\gamma \rfloor, 1\}$, where σ is an appropriately chosen constant such that $0 < \sigma < 1$ and $\gamma = \max\{b(v) \mid v \in V \setminus W\}$. The algorithm terminates after finishing a scaling phase with $\delta = 1$. Note that if we put $\sigma = 1$, we get an algorithm without explict scaling.

5. Computacional Results

We use a DELL Precision Workstation 330 with an Intel Pentium 4, CPU 1.80GHz, 512 MB and running Linux RedHat. Program DF implements Dinitz's algorithm, Program H_PRF Goldberg and Tarjan's algorithm using highest label first criterion, and Program Q_PRF Goldberg and Tarjan's algorithm using a queue to select active vertices. The three programs are the same as used by Cherkassky and Goldberg in their paper [1].

Employing the adjacency list representation of input graphs, we implemented both the original version of our algorithm using MA orderings and its scaling versions. The modified implicit scaling version FS1 runs faster than other variants of our algorithm. Our MA-ordering algorithm shows rather poor performance but there is not very substantial difference in efficiency among the variants.

We used the generator GENRMF available from DIMACS Challenge. It creates networks with b grid-like frames of size $(a \times a)$. The number of vertices is a^2b and that of arcs $5a^2b - 4ab - a^2$. Figure 1 shows results for a network with 2^x vertices and with parameters $a = 2^{x/4}$ and $b = 2^{x/2}$ (*Genrmf-long*). Our algorithm FS1 was faster than Dinitz's algorithm but was slower than Goldberg and Tarjan's.

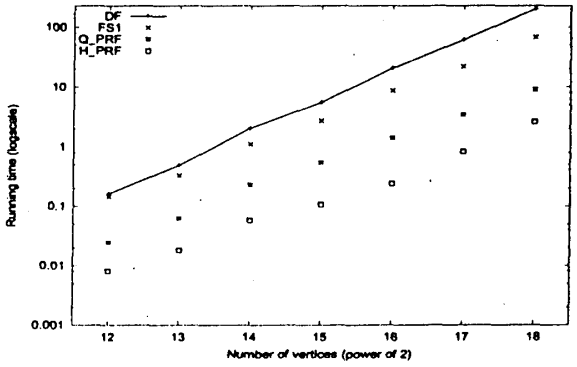

Figure 1: Computational results for *Genrmf-long*.

6. Concluding Remarks

We have proposed a new polynomial-time maximum flow algorithm with MA orderings and its scaling versions. Furthermore, our computational results showed that ours ran faster than Dinitz's for the generated Genrmf-long family data. Our proposed algorithms thus seem to be worth further investigation.

References

[1] B. V. Cherkassky and A. V. Goldberg: On implementing the push-relabel method for the maximum flow problem. *Algorithmica* **19** (1997) 390–410.

[2] S. Fujishige: A maximum flow algorithm using MA orderings. *OR Letters* (to appear).

[3] H. Nagamochi and T. Ibaraki: Graph connectivity and its augmentation: applications of MA orderings. *Discrete Applied Math.* **123** (2002) 447–472.