

# The MA-ordering max-flow algorithm is not strongly polynomial for directed networks

Akiyoshi Shioura

*Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan*

Received 4 February 2003; received in revised form 5 May 2003; accepted 5 May 2003

## Abstract

Quite recently, Fujishige (Oper. Res. Lett. 31 (2003) 176–178) developed a weakly polynomial-time algorithm for the maximum flow problem by applying the maximum adjacency (MA) ordering technique to directed networks. In this note, we show that the algorithm is not strongly polynomial by giving a real-valued instance for which the algorithm does not terminate. © 2003 Elsevier B.V. All rights reserved.

**Keywords:** Maximum flow; MA ordering; Weakly polynomial; Strongly polynomial

## 1. Introduction

Quite recently, Fujishige [2] applied maximum adjacency (MA) ordering technique to the maximum flow problem on directed graphs and developed a new weakly polynomial-time algorithm, which we refer to as the MA ordering algorithm (see also [3]). The most distinguished feature is that the MA ordering algorithm makes augmentations by using non-path flows while ordinary augmenting path algorithms use path flows in each augmentation phase.

Then, it is natural to wonder whether the weakly polynomial bound for the MA ordering algorithm can be strengthened to a strongly polynomial bound. In this note, we show that the MA ordering algorithm is not a strongly polynomial algorithm. For this, we use the fact that a strongly polynomial algorithm terminates in finite time even for instances with

irrational data (cf. [4,5]). Indeed, Queyranne [5] showed the maximum-capacity augmenting path algorithm of Edmonds–Karp [1] is not strongly polynomial by constructing a real-valued instance for which the algorithm does not terminate. The MA ordering algorithm presents a similar behavior to the maximum-capacity augmenting path algorithm for some instances, and therefore a slightly modified version of the instance in [5] can be used for this purpose, as shown in Section 3.

## 2. The MA ordering algorithm

In this section we review the MA ordering algorithm for the maximum flow problem. See [2,3] for details.

Let  $\mathcal{N} = (G, s, t, c)$  be an instance of the maximum flow problem, where  $G = (V, A)$  is a directed graph with a vertex set  $V$  and an arc set  $A$ , the vertices  $s \in V$  and  $t \in V$  are source and sink vertices, respectively, and  $c : A \rightarrow \mathbf{R}_+$  is a capacity function taking

---

E-mail address: [shioura@dais.is.tohoku.ac.jp](mailto:shioura@dais.is.tohoku.ac.jp) (A. Shioura).

nonnegative real values. A function  $\varphi: A \rightarrow \mathbf{R}_+$  is called a flow in  $\mathcal{N}$  if it satisfies the capacity constraints  $0 \leq \varphi(a) \leq c(a)$  ( $a \in A$ ) and the flow conservation constraints  $\partial\varphi(v) = 0$  ( $v \in V \setminus \{s, t\}$ ), where

$$\partial\varphi(v) = \sum_{a=(v,w) \in A} \varphi(a) - \sum_{a=(w,v) \in A} \varphi(a) \quad (v \in V).$$

For a flow  $\varphi$  in  $\mathcal{N}$  the value of  $\varphi$  is given by  $\partial\varphi(s) (= -\partial\varphi(t))$ . A maximum flow is a flow of maximum value.

Given a flow  $\varphi$  in  $\mathcal{N}$ , a residual network  $\mathcal{N}_\varphi = (G_\varphi, s, t, c_\varphi)$  with an underlying graph  $G_\varphi = (V, A_\varphi)$  and a capacity function  $c_\varphi: A_\varphi \rightarrow \mathbf{R}_+$  is defined by

$$A_\varphi = A_\varphi^+ \cup A_\varphi^-,$$

$$A_\varphi^+ = \{a \mid a \in A, \varphi(a) < c(a)\},$$

$$A_\varphi^- = \{\bar{a} \mid a \in A, \varphi(a) > 0\}$$

( $\bar{a}$  : a reorientation of  $a$ ),

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

We now explain the MA ordering algorithm. The algorithm starts with the zero flow  $\varphi = \mathbf{0}$ . In each iteration, the algorithm constructs a residual network  $\mathcal{N}_\varphi$  associated with the current flow  $\varphi$ . Then, the algorithm computes an MA ordering w.r.t.  $\mathcal{N}_\varphi$  which is an ordering  $\{v_0, v_1, \dots, v_k\}$  of some of the vertices in  $V$  such that  $v_0 = s$ ,  $v_k = t$ , and for  $j = 1, 2, \dots, k$  the following inequality holds:

$$\begin{aligned} & \sum \{c(v_i, v_j) \mid (v_i, v_j) \in A_\varphi, i < j\} \\ & \geq \sum \{c(v_i, w) \mid (v_i, w) \in A_\varphi, i < j\} \\ & \quad (w \in V \setminus \{v_0, v_1, \dots, v_{j-1}\}). \end{aligned}$$

Note that the arc set  $A'_\varphi = \{(v_i, v_j) \mid (v_i, v_j) \in A_\varphi, 0 \leq i < j \leq k\}$  forms an acyclic subgraph  $H_\varphi = (V, A'_\varphi)$  of  $G_\varphi$ . Put  $\delta = \min_{1 \leq j \leq k} \sum \{c(v_i, v_j) \mid (v_i, v_j) \in A_\varphi, i < j\}$ . By the definition of MA ordering, it is easy to see that there exists a flow  $\psi$  in  $\mathcal{N}_\varphi$  with the value  $\delta$  such that  $\psi(a) > 0$  implies  $a \in A'_\varphi$ . If  $\delta > 0$ , then the algorithm uses such  $\psi$  to augment the

current flow  $\varphi$  in  $\mathcal{N}$  as follows:

$$\varphi(a) \leftarrow \begin{cases} \varphi(a) + \psi(a) & (a \in A_\varphi^+ \text{ and } \psi(a) > 0), \\ \varphi(a) - \psi(a) & (\bar{a} \in A_\varphi^- \text{ and } \psi(\bar{a}) > 0), \\ \varphi(a) & (\text{otherwise}). \end{cases}$$

If  $\delta = 0$ , then the current flow  $\varphi$  is a maximum flow, and the algorithm terminates.

If each capacity  $c(a)$  is an integer, the MA ordering algorithm terminates in  $O(n \log nU)$  iterations and in  $O(n(m + n \log n) \log nU)$  time, where  $n = |V|$ ,  $m = |A|$ , and  $U$  denotes the maximum capacity [2]. Hence, the MA ordering algorithm is weakly polynomial.

### 3. A bad instance

In this section, we show a real-valued instance for which the MA ordering algorithm does not terminate.

Consider the network shown in Fig. 1. The source and sink vertices are  $s$  and  $t$ , respectively. We call three arcs  $(4,5)$ ,  $(10,9)$ ,  $(15,14)$  the *key arcs*. Capacity of each arc is indicated in the figure. Define  $r = (\sqrt{5} - 1)/2$ . The symbols  $S_1$ ,  $S_2$ , and  $S_3$  represent the values  $(1 + r)/2$ ,  $\frac{1}{2}$ , and  $r/2$ , respectively. These numbers satisfy the identities  $S_1 - r = r^3 S_1$ ,  $S_2 - r^2 = r^3 S_2$ ,  $S_3 - r^3 = r^3 S_3$ , and the inequalities  $1 > S_1 > r > S_2 > r^2 > S_3 > r^3 > S_1 - r$ , etc., which are needed to understand the behavior of the MA ordering algorithm on the network.

Suppose that the MA ordering algorithm is applied to this network with the initial feasible flow  $\varphi^{(0)} = \mathbf{0}$ . In the first iteration, the algorithm computes an acyclic subgraph  $H_{\varphi^{(0)}}$  of the residual network  $\mathcal{N}_{\varphi^{(0)}}$ , which is nothing but an  $s$ - $t$  path  $P_1$  given by

$$\begin{aligned} P_1 = \{ & (s, 1), (1, 4), (4, 5), (5, 8), (8, 10), (10, 9), \\ & (9, 12), (12, 15), (15, 14), (14, 16), (16, t) \}. \end{aligned}$$

Hence, the algorithm augments  $\delta = r$  units of flow along the path  $P_1$  to obtain a new flow  $\varphi^{(1)}$ .

In the second iteration, the algorithm computes an acyclic subgraph  $H_{\varphi^{(1)}}$  shown in Fig. 2. Since  $H_{\varphi^{(1)}}$  contains a unique  $s$ - $t$  path  $P_2$  given by

$$\begin{aligned} P_2 = \{ & (s, 2), (2, 5), (5, 4), (4, 6), (6, 9), (9, 10), \\ & (10, 13), (13, 15), (15, 14), (14, 17), (17, t) \}, \end{aligned}$$

the algorithm augments  $\delta = r^2$  units of flow along  $P_2$ .

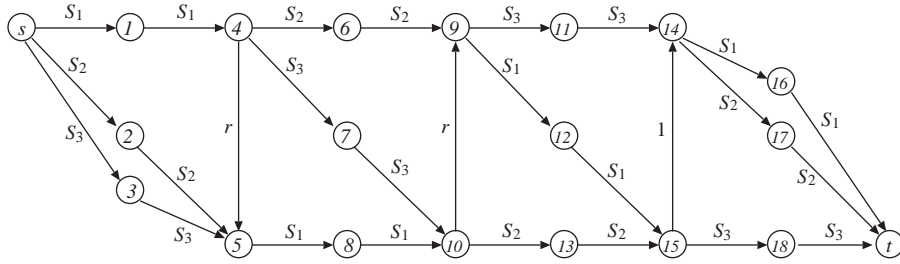
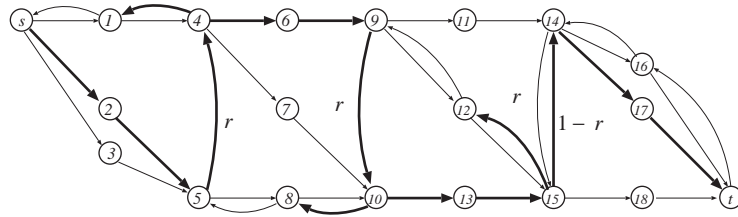
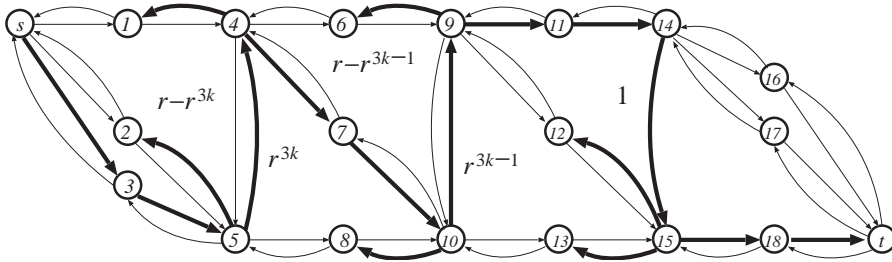


Fig. 1. A bad instance for the MA ordering algorithm.


Fig. 2. A residual network  $\mathcal{N}_{\varphi(1)}$  and an acyclic subgraph  $H_{\varphi(1)}$ . Arcs in  $H_{\varphi(1)}$  are drawn by thick arrows. Residual capacities of key arcs and their reverse arcs are indicated in the figure.

Fig. 3. A residual network  $\mathcal{N}_{\varphi(3k-1)}$  and an acyclic subgraph  $H_{\varphi(3k-1)}$ .

The cyclic pattern occurs from the third iteration. In the  $3k$ th iteration with  $k \geq 1$ , the algorithm computes an acyclic subgraph  $H_{\varphi(3k-1)}$  shown in Fig. 3 containing a unique  $s$ - $t$  path  $P_3$  given by

$$P_3 = \{(s, 3), (3, 5), (5, 4), (4, 7), (7, 10), (10, 9), \\ (9, 11), (11, 14), (14, 15), (15, 18), (18, t)\},$$

and augments  $\delta = r^{3k}$  units of flow along  $P_3$ .

Similarly in the  $(3k+1)$ st iteration with  $k \geq 1$ , the algorithm computes an acyclic subgraph  $H_{\varphi(3k)}$  shown in Fig. 4 and augments  $\delta = r^{3k+1}$  units of flow along  $P_1$ ; in the  $(3k+2)$ nd iteration with  $k \geq 1$ , the algorithm computes an acyclic subgraph  $H_{\varphi(3k+1)}$  shown in

Fig. 5 and augments  $\delta = r^{3k+2}$  units of flow along  $P_2$ . The flow values on the key arcs and the arcs with upper bounds  $S_1$ ,  $S_2$ , and  $S_3$  change as noted in Table 1. In the  $l$ th iteration, the algorithm augments the flow by  $r^l$ , and the resulting flow value is  $r(1-r^l)/(1-r)$ . Hence, the flow value converges to  $r/(1-r)$ , which is equal to the minimum cut capacity  $S_1 + S_2 + S_3 = 1+r$ .

Therefore, the MA ordering algorithm requires infinitely many iterations, and is not a strongly polynomial algorithm. We note that Fujishige's scaling variant in [2] is developed for integer-valued instances and requires infinitely many iterations for real-valued instances, as in the case of the ordinary capacity scaling max-flow algorithm. It is an interesting question

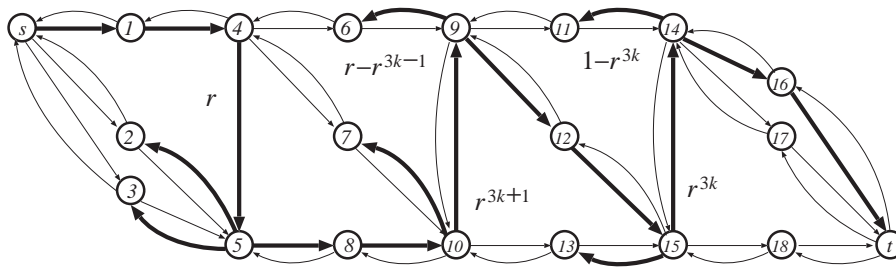
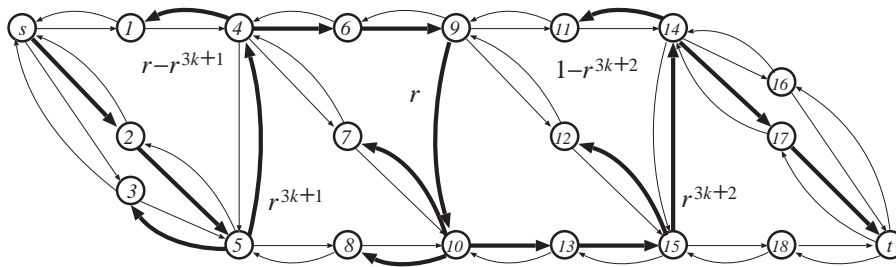
Fig. 4. A residual network  $\mathcal{N}_{\varphi(3k)}$  and an acyclic subgraph  $H_{\varphi(3k)}$ .Fig. 5. A residual network  $\mathcal{N}_{\varphi(3k+1)}$  and an acyclic subgraph  $H_{\varphi(3k+1)}$ .

Table 1

Change of flow when the MA ordering algorithm is applied to the bad instance

Iter.	Flow at the beginning of the iteration					
	Key arcs			Arcs with upper bound		
	(4, 5)	(10, 9)	(15, 14)	$S_1$	$S_2$	$S_3$
1	0	0	0	0	0	0
2	$r$	$r$	$r$	$S_1(1 - r^{3k})$	$S_2(1 - r^{3k})$	$S_3(1 - r^{3k-3})$
$3k$	$r^{3k}$	$r - r^{3k-1}$	1	$S_1(1 - r^{3k})$	$S_2(1 - r^{3k})$	$S_3(1 - r^{3k})$
$3k + 1$	0	$r - r^{3k+1}$	$1 - r^{3k}$	$S_1(1 - r^{3k+3})$	$S_2(1 - r^{3k})$	$S_3(1 - r^{3k})$
$3k + 2$	$r^{3k+1}$	$r$	$1 - r^{3k+2}$	$S_1(1 - r^{3k+3})$	$S_2(1 - r^{3k+3})$	$S_3(1 - r^{3k})$
$3k + 3$	$r^{3k+3}$	$r - r^{3k+2}$	1			

whether other variants of the MA ordering algorithm could be strongly polynomial, which is left for further research.

## Acknowledgements

The author thanks Satoru Fujishige and Shiguo Isotani for their comments on this manuscript. This work is supported by Grant-in-Aid of the Ministry of

Education, Culture, Sports, Science and Technology of Japan.

## References

- [1] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. ACM 19 (1972) 248–264.
- [2] S. Fujishige, A maximum flow algorithm using MA ordering, Oper. Res. Lett. 31 (2003) 176–178.

- [3] S. Fujishige, S. Isotani, New maximum flow algorithms by MA ordering and scaling, *Proceedings of the Third Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*, 2003, pp. 186–193.
- [4] S.T. McCormick, A. Shioura, Minimum ratio canceling is oracle polynomial for linear programming, but not strongly polynomial, even for networks, *Oper. Res. Lett.* 27 (2000) 199–207.
- [5] M. Queyranne, Theoretical efficiency of the algorithm ‘Capacity’ for the maximum flow problem, *Math. Oper. Res.* 5 (1980) 258–266.