J Comb Optim (2011) 22: 78–96
DOI 10.1007/s10878-009-9276-z

Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree

Yuichi Asahiro · Jesper Jansson · Eiji Miyano · Hirotaka Ono · Kouhei Zenmyo

Published online: 5 November 2009

Abstract Given a simple, undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{Z}^+$, we consider the problem of orienting all edges in E so that the maximum weighted outdegree among all vertices is minimized. It has previously been shown that the unweighted version of the problem is solvable in polynomial time while the weighted version is (weakly) NP-hard. In this paper, we strengthen these results as follows: (1) We prove that the weighted version is strongly NP-hard even if all edge

An extended abstract of this article was presented in *Proceedings of the Third International Conference on Algorithmic Aspects in Information and Management* (AAIM 2007), Lecture Notes in Computer Science, Vol. 4508, pp. 167–177, Springer-Verlag Berlin Heidelberg, 2007.

This work is partially supported by KAKENHI No. 16092223, No. 17700022, No. 18700015, No. 20500017 and No. 21680001, the Special Coordination Funds for Promoting Science and Technology, Asahi glass foundation and Inamori foundation.

Y. Asahiro
Department of Information Science, Kyushu Sangyo University, Higashi-ku, Fukuoka 813-8503, Japan
e-mail: asahiro@is.kyusan-u.ac.jp

J. Jansson
Ochanomizu University, Bunkyo-ku, Tokyo 112-8610, Japan
e-mail: Jesper.Jansson@ocha.ac.jp

E. Miyano · K. Zenmyo
Department of Systems Design and Informatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan

E. Miyano
e-mail: miyano@ces.kyutech.ac.jp

K. Zenmyo
e-mail: kouhei@theory.ces.kyutech.ac.jp

H. Ono (✉)
Department of Informatics, Kyushu University, Nishi-ku, Fukuoka 819-0395, Japan
e-mail: ono@csce.kyushu-u.ac.jp

weights belong to the set $\{1, k\}$, where k is any fixed integer greater than or equal to 2, and that there exists no pseudo-polynomial time approximation algorithm for this problem whose approximation ratio is smaller than $(1 + 1/k)$ unless P = NP; (2) we present a new polynomial-time algorithm that approximates the general version of the problem within a ratio of $(2 - 1/k)$, where k is the maximum weight of an edge in G; (3) we show how to approximate the special case in which all edge weights belong to $\{1, k\}$ within a ratio of 3/2 for $k = 2$ (note that this matches the inapproximability bound above), and $(2 - 2/(k + 1))$ for any $k \geq 3$, respectively, in polynomial time.

Keywords Graph orientation · Degree · Approximation algorithm · Inapproximability · Maximum flow · Scheduling

1 Introduction

Let $G = (V, E, w)$ be a simple, undirected, edge-weighted graph, where V, E and w denote the set of vertices of G, the set of edges of G, and a positive integral weight function $w : E \to \mathbb{Z}^+$, respectively. An *orientation* Λ *of* G is an assignment of a direction to each edge $\{u, v\} \in E$, i.e., $\Lambda(\{u, v\})$ is either (u, v) or (v, u). Given an orientation Λ of G, the *weighted outdegree* of a vertex u is $d^+_\Lambda(u) = \sum_{\substack{\{u,v\}\in E:\\ \Lambda(\{u,v\})=(u,v)}} w(\{u, v\})$.

In this paper, we consider the problem of finding an orientation of an input graph G such that the maximum weighted outdegree among all vertices is minimum, taken over all possible orientations of G. To specify different classes of edge weight functions, we formally define the problem as follows.

Problem:	S-MINIMUM MAXIMUM OUTDEGREE (S-MMO)
Input:	A simple, undirected, edge-weighted graph $G = (V, E, w)$, where w is a positive integral weight function of the form $w : E \to S$ and where S is a set of allowed weights.
Output:	An orientation Λ of G that minimizes $\max_{u\in V}\{d^+_\Lambda(u)\}$.

The most general case of S-MMO with no restrictions on the weight function, except that it must be a positive integral function, is denoted by $\mathbb{Z}^+$-MMO. In this paper, we assume that S is of the form $S = \{1, 2, \ldots, k\}$, where k is a positive integer. (The running times of our algorithms will depend on k.) We also study a special case in which the range of w is restricted to a positive integer set $S = \{1, k\}$ with $k \geq 2$.

Throughout this paper, given an instance of S-MMO, we set $n = |V|$ and $m = |E|$. The weighted outdegree $d^+_\Lambda(u)$ of a vertex u is also called *the outdegree of* u for short. For any orientation Λ of G, the *value of* Λ is defined to be $\max_{u\in V}\{d^+_\Lambda(u)\}$. We use $OPT(G)$ or Δ^* to denote the optimal value for G, i.e., the minimum of $\max_{u\in V}\{d^+_\Lambda(u)\}$ taken over all possible orientations Λ of G. A graph orientation algorithm ALG is called a σ*-approximation algorithm* and ALG's *approximation ratio* is σ if $ALG(G)/OPT(G) \leq \sigma$ holds for every graph G, where $ALG(G)$ is the value of the solution obtained by running ALG on input G.

1.1 Motivation

Graph orientations which minimize the maximum outdegree can be used to construct efficient dynamic data structures for graphs that support fast vertex adjacency queries under a series of edge insertions and edge deletions (Brodal and Fagerberg 1999). Also, S-MMO can be viewed as a variation of *the art gallery problem* (see, e.g., Chvátal 1975; O'Rourke 1987), *load balancing problems*, or *unrelated parallel machine scheduling* (see, e.g., Lenstra et al. 1990; Pinedo 2002). In particular, the polynomial time (in)approximability of the latter problem has been intensively studied. Refer to Sect. 6.1 for a further discussion on the relation between S-MMO and scheduling.

Graph orientation itself is a quite basic, natural, and important problem in graph theory and combinatorial optimization; see, e.g., Chap. 61 of Schrijver (2003) and the short survey in Asahiro et al. (2007). As an example, it is known that any planar graph has an orientation with value at most 3 and an acyclic orientation with value at most 5, and such orientations can be found in linear time (Chrobak and Eppstein 1991). However, most previous studies focus on problems related to orientations satisfying some special graph properties such as high connectivity, small diameter, no cycles, small difference between the indegree and outdegree of each vertex, etc. (Biedl et al. 2005; Fomin et al. 2004; Kára et al. 2005), and very few studies consider orientations which minimize the maximum outdegree (or equivalently, indegree) (Asahiro et al. 2007; Kowalik 2006; Venkateswaran 2004).

1.2 Previous results and summary of new results

Previous work has shown that S-MMO can be solved in polynomial time if all edge weights are identical (Asahiro et al. 2007; Kowalik 2006; Venkateswaran 2004). More precisely, the fastest known algorithm for $\{k\}$-MMO runs in $O(m^{3/2} \cdot \log(\Delta_1^*))$ time, where Δ_1^* denotes the optimal value of $\{1\}$-MMO (Asahiro et al. 2007). On the other hand, S-MMO is (weakly) NP-hard in the general case (Asahiro et al. 2007). For any subgraph H of G, let $V(H)$ and $E(H)$ denote the vertex set of H and the edge set of H, respectively. A $(2 - 1/\lceil L(G) \rceil)$-approximation algorithm for $\mathbb{Z}^+$-MMO with $O(m^2)$ running time was presented in Asahiro et al. (2007), where $L(G)$ is the maximum density among all subgraphs of G, that is, $L(G) = \max_{H \subseteq G} \{\sum_{\{u,v\} \in E(H)} w(\{u, v\})/|V(H)|\}$. No inapproximability results for S-MMO were previously known.

In this paper, we study S-MMO from the viewpoint of polynomial-time approximability and inapproximability. First, Sect. 2 introduces some additional notation and terminology needed to describe our results. Then, in Sects. 3.1–3.4, we present four new polynomial-time approximation algorithms named MAJORITY, CYCLE-CANCELING, REFINED CYCLE-CANCELING, and LARGE-k. (Although MAJORITY has the same running time and a worse approximation ratio than CYCLE-CANCELING, we have included the description of MAJORITY because it provides a simple way to illustrate some key ideas used in the design and analysis of CYCLE-CANCELING and REFINED CYCLE-CANCELING.) Section 4 shows how to improve the running times of our first three approximation algorithms. Next, we give a reduction from At-most-3-SAT(2L) in Sect. 5 which proves the strong NP-hardness of

$\{1, k\}$-MMO for $k \geq 2$ and also yields the first non-trivial lower bound on the approximation ratio, under the assumption P $\neq$ NP. Finally, in Sect. 6, we discuss the relation between S-MMO and scheduling and state some open problems. Our new results are summarized below.

- $\{1, \ldots, k\}$-MMO has a $(2 - 1/k)$-approximation algorithm with running time $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^* + m^2)$ [Algorithm CYCLE-CANCELING in Sect. 3.2 together with Corollary 1 in Sect. 4].
- The special case $\{1, k\}$-MMO where $k \geq 3$ has a (slightly better) $(2 - 2/(k + 1))$-approximation algorithm, also with running time $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^* + m^2)$ [Algorithm REFINED CYCLE-CANCELING in Sect. 3.3 together with Corollary 1 in Sect. 4].
- $\{1, k\}$-MMO admits a $1 + n/(2k)$-approximation algorithm which runs in $O(m^{3/2} \cdot \log n)$ time [Algorithm LARGE-k in Sect. 3.4]. This is useful for instances with $k \gg n$.
- $\{1, k\}$-MMO for any fixed $k \geq 2$ is strongly NP-hard [Theorem 6 in Sect. 5].
- For any fixed integer $k \geq 2$, no pseudo-polynomial time algorithm for $\{1, k\}$-MMO achieves an approximation ratio smaller than $1 + 1/k$, unless P = NP [Theorem 7 in Sect. 5]. This implies that there is no polynomial time approximation algorithm for $\mathbb{Z}^+$-MMO with approximation ratio less than $3/2$, unless P = NP. This also means that, for $k = 2$, Algorithm CYCLE-CANCELING is optimal with respect to the approximation ratio.

Note that the $2 - 1/\lceil L(G) \rceil$-approximation ratio from Asahiro et al. (2007) and the new $2 - 1/k$ one are incomparable; sometimes the former is better than the latter, and vice versa. For example, there exist instances where the former algorithm outputs a $5/3$-ratio solution while the latter achieves the ratio $3/2$ (see Fig. 6 in Asahiro et al. 2007).

2 Preliminaries

From here on, we assume that the vertices in G are lexicographically ordered. We denote an undirected edge with endpoints u and v, where $u < v$ in lexicographic order, by $e_{u,v}$ or simply $\{u, v\}$. A directed edge (or *arc*) from a vertex u to a vertex v is written as (u, v). The directed graph defined by an orientation Λ of G is denoted by $\Lambda(G) = (V, \Lambda(E), w)$. A *directed path of length l* from a vertex v_0 to a vertex v_l in $\Lambda(G)$ is a set of arcs $\{(v_{i-1}, v_i) \in \Lambda(E) \mid i = 1, 2, \ldots, l\}$, also represented by the sequence $\langle v_0, v_1, \ldots, v_l \rangle$ for simplicity. In particular, a directed path satisfying $v_l = v_0$ is called a *directed l-cycle*. For any directed path $P = \langle v_0, v_1, \ldots, v_l \rangle$, the directed path obtained by traversing P in its reverse order is denoted by $\overline{P}$, i.e., $\overline{P} = \langle v_l, v_{l-1}, \ldots, v_0 \rangle$.

Next, for any $u \in V$, let $\Gamma(u) = \{v \mid \{u, v\} \in E\}$ denote the set of neighbors of u. For any orientation Λ of G, define *the set of neighbors of u under Λ* as $\Gamma_\Lambda(u) = \{v \mid \{u, v\} \in E \text{ and } \Lambda(\{u, v\}) = (u, v)\}$. We call any vertex u^* whose weighted outdegree is maximum in Λ *critical*, and also say that u^* is a *critical vertex with respect to* Λ. Let $w_{\max}$ be the maximum weight among all edges in E and let W be the total weight

of all edges in E. Every orientation has the following trivial lower bound caused by the maximum weight edges:

Proposition 1 (Asahiro et al. 2007) *For any undirected weighted graph G and any orientation Λ of G, the value of Λ is at least $w_{\max}$.*

Finally, we introduce three basic operations named REVERSE, UP-TO-ROOTS, and SOLVE-1-MMO which will be used later in this paper.

- REVERSE does the following: *Given an orientation Λ of G and a directed path $P = \langle u_0, u_1, \ldots, u_l \rangle$ in $\Lambda(G)$, update Λ by replacing P with $\overline{P}$, i.e., let $\Lambda(e_{u_i,u_{i+1}}) = (u_{i+1}, u_i)$ for $i = 0, \ldots, l-1$.* We call this operation REVERSE-CYCLE if $u_0 = u_l$. Note that if P is a directed cycle and all $w(e_{u_i,u_{i+1}})$'s are equal, then the outdegree of every vertex remains unchanged.
- UP-TO-ROOTS determines an orientation Λ for a given simple, undirected forest G as follows: *First fix an arbitrary root node for each tree in G. Then, for every edge e, orient $\Lambda(e)$ towards the root node of the tree containing e.*[1]
- SOLVE-1-MMO outputs an optimal orientation of a given graph with identical edge weights. SOLVE-1-MMO can be implemented to run in $O(m^{3/2} \cdot \log(\Delta_1^*))$ for $\{k\}$-MMO time (Asahiro et al. 2007). (Here, the log factor comes from a binary search.)

3 Approximation algorithms

We now present the details of our four new approximation algorithms for S-MMO. The first two, MAJORITY and CYCLE-CANCELING, work for any $S = \{1, \ldots, k\}$, whereas the last two, REFINED CYCLE-CANCELING and LARGE-k, are designed for the special case where S is of the form $S = \{1, k\}$.

3.1 Majority voting algorithm

In this subsection, we give a simple 2-approximation algorithm named MAJORITY. Although MAJORITY can be considered a variation of the Lenstra-Shmoys-Tardos algorithm (Lenstra et al. 1990), which is based on LP-rounding and has an approximation ratio of 2, MAJORITY is combinatorial and provides some basic intuition for the algorithms presented in later subsections. Furthermore, according to Corollary 1 in Sect. 4, MAJORITY is much faster than the Lenstra-Shmoys-Tardos algorithm.

Algorithm MAJORITY is presented in Fig. 1. It works as follows. First, replace each edge $e = \{u, v\}$ in G with $w(e)$ edges of weight 1 between u and v, so that an undirected multigraph G' with $W = \sum_{e \in E} w(e)$ edges is obtained. Next, find an optimal orientation Λ' of G'. (In Λ', for each $\{u, v\} \in E$, some replicated edges of $\{u, v\}$ may be oriented from u to v while others are oriented from v to u.) Then, decide an

[1] Observe that $OPT(G) = w_{\max}$ if G is a forest, i.e., a graph that does not contain any cycles. (It is easy to see that the UP-TO-ROOTS operation finds an optimal solution for forests (Asahiro et al. 2007).) Thus, the bound in Proposition 1 is optimal for this case.

Algorithm MAJORITY

1. For graph G, construct G' by replacing each edge e with $w(e)$ edges of weight 1.
2. Find an optimal orientation Λ' of G' by using SOLVE-1-MMO.
3. Decide the orientation Λ of G according to (1) for each edge in G.
4. Return Λ.

Fig. 1 Algorithm MAJORITY

orientation Λ of G by majority voting. More precisely, let $f_{u\to v}$ and $f_{v\to u}$ denote the number of edges oriented from u to v and from v to u, respectively, in Λ'. Since G is simple, $f_{u\to v} + f_{v\to u} = w(e_{u,v})$ holds. The orientation Λ of the original G is determined in the following manner: For each $e_{u,v} \in E$, assign

$$\Lambda(e_{u,v}) := \begin{cases} (u, v), & \text{if } f_{u\to v} \geq f_{v\to u}, \\ (v, u), & \text{otherwise.} \end{cases} \tag{1}$$

(By the definitions above, the direction is determined according to the lexicographic order in case of a tie.)

Theorem 1 *For any $S = \{1, \ldots, k\}$, Algorithm* MAJORITY *approximates S-MMO within a ratio of* 2 *and runs in $O(W^{3/2} \cdot \log \Delta^*)$ time.*

Proof First, consider the running time. Steps 1, 2 and 3 take $O(W)$, $O(W^{3/2} \cdot \log \Delta^*)$ and $O(W)$ time, respectively, the total running time of MAJORITY is $O(W^{3/2} \cdot \log \Delta^*)$.

Next, consider the approximation ratio. The outdegree of any vertex u under Λ is $\sum_{v\in\Gamma_\Lambda(u)} w(e_{u,v}) = \sum_{v\in\Gamma_\Lambda(u)} (f_{u\to v} + f_{v\to u})$. Let u^* be a critical vertex with respect to Λ. Then $ALG(G) = \sum_{v\in\Gamma_\Lambda(u^*)} (f_{u^*\to v} + f_{v\to u^*})$, and $OPT(G) \geq OPT(G') \geq \sum_{v\in\Gamma(u^*)} f_{u^*\to v}$; since Λ' is a relaxed orientation of G, the optimal value of G' is a lower bound on the optimal solution of G. Hence,

$$\frac{ALG(G)}{OPT(G)} \leq \frac{ALG(G)}{OPT(G')} \leq \frac{\sum_{v\in\Gamma_\Lambda(u^*)} (f_{u^*\to v} + f_{v\to u^*})}{\sum_{v\in\Gamma(u^*)} f_{u^*\to v}}$$

$$\leq \frac{\sum_{v\in\Gamma_\Lambda(u^*)} 2 \cdot f_{u^*\to v}}{\sum_{v\in\Gamma_\Lambda(u^*)} f_{u^*\to v}} = 2.$$

The last inequality holds since Λ is decided by majority voting. The approximation ratio is 2. □

The analysis is tight, as the example in Fig. 2 demonstrates: The value of the optimal solution for the instance G is k, while a possible output of MAJORITY is $2k$, as shown in Fig. 2(d).

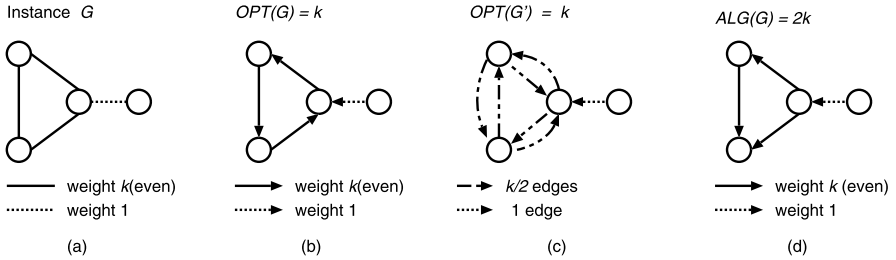

Fig. 2 A worst-case example for MAJORITY: (**a**) an instance G, (**b**) an optimal orientation of G, (**c**) an optimal orientation Λ' of G', and (**d**) a possible output of MAJORITY based on (**c**)

Algorithm CYCLE-CANCELING

1. For graph G, construct G' by replacing each edge e with $w(e)$ edges of weight 1.
2. Find an optimal orientation Λ' of G' by using SOLVE-1-MMO.
3. Decide the (partial) orientation Λ of G according to (2), and obtain $G_{\Lambda'} = (V, F_{\Lambda'})$.
4. If there exists a directed l-cycle in $G_{\Lambda'}$ where $l \geq 3$, apply REVERSE-CYCLE and go to Step 3.
5. For undecided edges of Λ, apply UP-TO-ROOTS to $G_{\Lambda'}$.
6. Return Λ.

Fig. 3 Algorithm CYCLE-CANCELING

3.2 Cycle canceling algorithm

Here, we describe an algorithm named CYCLE-CANCELING which improves MAJORITY; its approximation ratio is $2 - 1/k$. In fact, CYCLE-CANCELING also uses the same basic idea of replacing each weighted edge by a number of unweighted edges and computing an optimal solution for the resulting unweighted multigraph. However, it then decides the orientation of each edge in a different manner.

CYCLE-CANCELING is listed in Fig. 3. In the first and second steps of the algorithm, do as MAJORITY; construct G' (replicate each edge) and then find an optimal orientation Λ'. After that, decide a partial orientation of the original problem by

$$\Lambda(e_{u,v}) := \begin{cases} (u, v), & \text{if } f_{v \to u} = 0, \\ (v, u), & \text{if } f_{u \to v} = 0, \\ -, & \text{otherwise}, \end{cases} \tag{2}$$

where $-$ means "not decided yet". Note that the direction of the edges decided by this operation is essentially the same as the one of Λ'; the value of the orientation does not change.

Next, we introduce a new operation, *cycle cancelation*, which updates the orientation to a more desirable one without changing the outdegree of any vertex. To this end, we construct another undirected graph $G_{\Lambda'} = (V, F_{\Lambda'})$, where $F_{\Lambda'} = \{e_{u,v} \in E \mid f_{u\to v} \neq 0 \text{ and } f_{v\to u} \neq 0 \text{ in } \Lambda'\}$. From $G_{\Lambda'}$, we find an l-cycle with $l \geq 3$, say $C = \langle v_1, v_2, \ldots, v_l, v_1(\equiv v_{l+1})\rangle$, if one exists. (From here on, when we mention l-cycles with $l \geq 3$, we just say "cycles" for simplicity, because we do not consider 2-cycles in this paper.) Let $c = \min\{f_{v_i\to v_{i+1}} \mid i = 1, \ldots, l\}$, which is a positive integer, by the definition of $F_{\Lambda'}$. Then, we go back to G' and Λ' and apply REVERSE-CYCLE with size c to C; since there exist c cycles of $\langle v_1, v_2, \ldots, v_l, v_1(\equiv v_{l+1})\rangle$ on G' under Λ', we can reverse the direction of the edges along the c cycles. It should be noted that the outdegree (or the indegree) of each vertex in the resulting directed graph is equal to the one under Λ'; it is still an optimal orientation in G' and can be updated as Λ'. For this new Λ', we apply (2), then go back to the beginning of this paragraph. Since at least one edge $\{v_i, v_{i-1}\}$ on the cycle C satisfies $f_{v_i\to v_{i+1}} = 0$ by the REVERSE-CYCLE, the new $F_{\Lambda'}$ is strictly smaller than the old $F_{\Lambda'}$; this step ends in at most $m - 2$ iterations.

After a number of (or possibly zero) iterations of the above procedure, $G_{\Lambda'}$ becomes a forest, and we set $\mathcal{F} := G_{\Lambda'}$. Note that all the edges of $\mathcal{F}$ are not decided yet by (2). The cycle cancelation itself implies that there always exists an optimal solution Λ' for the relaxed problem such that Λ' has no cycles in $\mathcal{F}$. Then, we have the simple disjoint tree structure, for which we can apply the UP-TO-ROOTS operation to decide the orientation of all the remaining edges.

Theorem 2 *For any $S = \{1, \ldots, k\}$, Algorithm* CYCLE-CANCELING *approximates S-MMO within a ratio of $(2 - \frac{1}{k})$ and runs in $O(W^{3/2} \cdot \log \Delta^* + m^2)$ time.*

Proof We first consider the running time of CYCLE-CANCELING. Steps 1 and 2 have the same time complexity as MAJORITY, i.e., $O(W^{3/2} \cdot \log \Delta^*)$ time. Each iteration of Step 3 takes $O(m)$ time, and each iteration of Step 4 takes $O(m)$ time by the depth first search, and these steps can be iterated at most $m - 2$ times. Step 5 takes $O(m)$ time. In total, the running time is $O(W^{3/2} \cdot \log \Delta^* + m^2)$.

Next, we analyze the approximation ratio. Let u^* be any critical vertex in G with respect to Λ, i.e., a vertex with maximum weighted outdegree in Λ. We shall prove that $d^+_\Lambda(u^*) \leq (2 - \frac{1}{k}) \cdot OPT(G)$. First of all, note that $OPT(G) \geq k$ by Proposition 1 and also that $OPT(G) \geq OPT(G') = d^+_{\Lambda'}(x^*) \geq d^+_{\Lambda'}(u^*)$, where x^* is any critical vertex with respect to Λ'. Let $\mathcal{F}^*$ be the forest of rooted trees produced by UP-TO-ROOTS in Step 5. There are two possible cases to consider after the iterations of Steps 3 and 4:

1. u^* is a root in $\mathcal{F}^*$:[2] In this case, we immediately have $d^+_\Lambda(u^*) \leq d^+_{\Lambda'}(u^*)$ because zero or more of u^*'s outgoing edges in Λ' are reversed to obtain Λ, but none of its incoming edges in Λ' is reversed in Step 5. Then, recall that $d^+_{\Lambda'}(u^*) \leq OPT(G)$ by the above.

[2]This case also handles the possibility that u^* is an isolated vertex in $G_{\Lambda'}$.

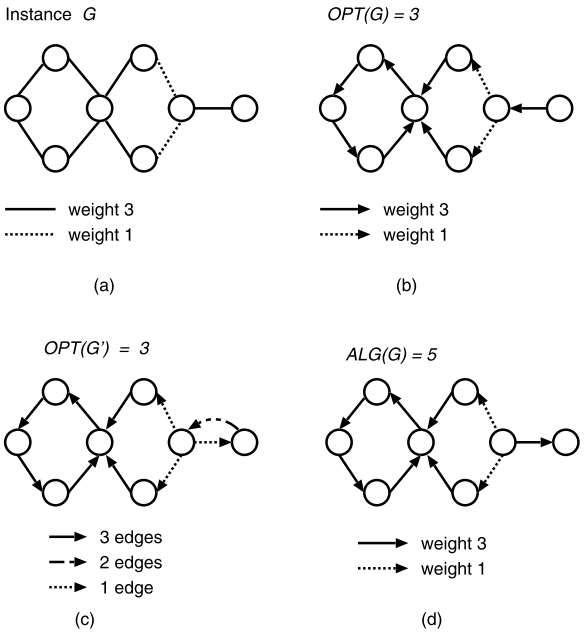

Fig. 4 A worst-case example for CYCLE-CANCELING: (**a**) an instance G, (**b**) an optimal orientation of G, (**c**) an optimal orientation Λ' of G', and (**d**) a possible output of CYCLE-CANCELING based on (**c**)

2. u^* is not a root in $\mathcal{F}^*$: In this case, let p denote the parent of u^* and $\mathcal{C}$ the set of children of u^* in $\mathcal{F}^*$, respectively. Clearly, we have $d^+_\Lambda(u^*) = d^+_{\Lambda'}(u^*) + f_{p\to u^*} - \sum_{v\in\mathcal{C}} f_{u^*\to v} \le d^+_{\Lambda'}(u^*) + f_{p\to u^*}$, which yields

$$\frac{d^+_\Lambda(u^*)}{OPT(G)} \le \frac{d^+_{\Lambda'}(u^*) + f_{p\to u^*}}{OPT(G)} \le \frac{d^+_{\Lambda'}(u^*)}{d^+_{\Lambda'}(u^*)} + \frac{f_{p\to u^*}}{k} \le 1 + \frac{k-1}{k} = 2 - \frac{1}{k},$$

where the last inequality holds since $f_{p\to u^*} + f_{u^*\to p} \le k$ and $f_{u^*\to p} \ge 1$.

In both cases, $d^+_\Lambda(u^*)$ is within the desired bound. □

Figure 4 shows a worst-case example of CYCLE-CANCELING for an instance of $\{1, 3\}$-MMO whose approximation ratio is $5/3 = 2 - 1/3$. This construction can be modified in a straightforward way to produce worst-case examples for general k, which means that the analysis of Theorem 2 is tight.

Remark By Theorem 2, the approximation ratio of CYCLE-CANCELING for $k = 2$ is $3/2$. This is actually the best possible in polynomial time for $k = 2$ (unless P = NP), as we shall prove in Sect. 5.

3.3 Refined cycle canceling algorithm

We now consider the special case of S-MMO in which $S = \{1, k\}$ for $k \ge 3$, and show that it can be approximated even more efficiently than by Algorithm CYCLE-CANCELING. The new algorithm is called REFINED CYCLE-CANCELING and is outlined in Fig. 5. The main idea is to show that if all edge weights in G are either 1 or k,

Algorithm REFINED CYCLE-CANCELING

1.–4. Execute Steps 1 to 4 of CYCLE-CANCELING.
5. While there exists a leaf node u connecting to v such that $f_{u\to v} \geq f_{v\to u}$ in $\mathcal{F} = (V, F)$, let $\Lambda(e_{u,v}) := (u, v)$ and remove $e_{u,v}$ from F.
6. For undecided edges of Λ, apply UP-TO-ROOTS to $\mathcal{F}$.
7. Return Λ.

Fig. 5 Algorithm REFINED CYCLE-CANCELING

a slight modification to CYCLE-CANCELING allows us to compute a stronger lower bound on an optimal solution which then yields an improved approximation ratio.

As mentioned in the previous section, the cycle cancelation itself provides an optimal solution for the relaxed problem with a tree property. Here, we focus on Step 5 of CYCLE-CANCELING, in which the naive application of UP-TO-ROOTS with arbitrary roots gives a worst-case example (as shown in Fig. 4); this causes the approximation ratio to be $2 - 1/k$. The reason is that some vertices having large outdegrees under the orientation Λ' are not suitable for being roots; if such a vertex is set to be a root, its outdegree will distribute to its neighbors so that the neighbors have large outdegrees under Λ compared to under Λ'. To avoid this situation, Algorithm REFINED CYCLE-CANCELING proceeds as follows.

First execute Steps 1 to 4 of CYCLE-CANCELING, and obtain a forest $\mathcal{F}$. If there exists a leaf node u in $\mathcal{F}$ such that $f_{u\to v} \geq f_{v\to u}$ holds for its neighbor v, we fix the orientation of $e_{u,v}$ as (u, v) and remove $e_{u,v}$ from $\mathcal{F}$ (i.e., $\Lambda(e_{u,v}) := (u, v)$ and $\mathcal{F} = (V, F)$ with $F := F \setminus \{e_{u,v}\}$). Repeat this operation until no leaf node u satisfies $f_{u\to v} \geq f_{v\to u}$ where v is the neighbor node of u. Then, the algorithm applies UP-TO-ROOTS.

While Algorithm CYCLE-CANCELING simply applies UP-TO-ROOTS operations to the obtained forests, REFINED CYCLE-CANCELING decides the orientation of edges connected to leaves according to the values of f's for the leaves and their parents, and then applies UP-TO-ROOTS. Note that this modification does not depend on S and does not make the solution worse, though it might be difficult to show that it has an improved approximation ratio. We can, however, show a better approximation ratio for the special case $S = \{1, k\}$.

Theorem 3 *For any $S = \{1, k\}$ where $k \geq 3$, Algorithm* REFINED CYCLE-CANCELING *approximates S-MMO within a ratio of $(2 - \frac{2}{k+1})$ and runs in $O(W^{3/2} \cdot \log \Delta^* + m^2)$ time.*

Proof It is easy to see that adding Steps 5 and 6 to Algorithm CYCLE-CANCELING in Sect. 3.2 does not increase the asymptotic running time. Therefore, the running time is $O(W^{3/2} \cdot \log \Delta^* + m^2)$.

To analyze the approximation ratio of REFINED CYCLE-CANCELING, we proceed similarly as in the proof of Theorem 2. Let u^* be any critical vertex in G with respect to Λ, and let $\mathcal{F}^*$ be the forest of rooted trees produced by UP-TO-ROOTS in Step 6.

Recall that $OPT(G) \geq k$ and $OPT(G) \geq OPT(G') \geq d^+_{\Lambda'}(u^*)$. There are two main cases:

1. u^* is a node which satisfies the condition in Step 5: Then, since $f_{p\to u^*} \leq \frac{k}{2}$ for the parent p of u^*,

$$\frac{d^+_{\Lambda}(u^*)}{OPT(G)} \leq \frac{d^+_{\Lambda'}(u^*) + f_{p\to u^*}}{OPT(G)} \leq \frac{d^+_{\Lambda'}(u^*)}{d^+_{\Lambda'}(u^*)} + \frac{f_{p\to u^*}}{k} \leq 1 + \frac{k/2}{k} = \frac{3}{2}.$$

2. u^* is a node which did not satisfy the condition in Step 5:
   (a) If u^* is a root in $\mathcal{F}^*$, then $d^+_{\Lambda}(u^*) \leq d^+_{\Lambda'}(u^*) \leq OPT(G)$ as before, and we are done.
   (b) If not, consider the tree T in $\mathcal{F}^*$ that contains u^*. Let p be the parent of u^* in T and let $\langle u_1, u_2, \ldots, u_\ell \rangle$ be the path between any two leaves u_1 and u_ℓ in the undirected version of T. Since u_1 and u_ℓ satisfy $f_{u_1\to u_2} < f_{u_2\to u_1}$ and $f_{u_\ell\to u_{\ell-1}} < f_{u_{\ell-1}\to u_\ell}$, there must exist an intermediate node u_i such that $f_{u_{i-1}\to u_i} < f_{u_i\to u_{i-1}}$ and $f_{u_i\to u_{i+1}} \geq f_{u_{i+1}\to u_i}$. Next, because all edges in T have weight k, we know that $f_{v\to w} + f_{w\to v} = k$ for every edge $\{v, w\}$ in T, which means that $f_{u_i\to u_{i-1}} > k/2$ and $f_{u_i\to u_{i+1}} \geq k/2$. Thus, the outdegree of u_i is at least $f_{u_i\to u_{i-1}} + f_{u_i\to u_{i+1}} > k$, i.e., $OPT(G') \geq k + 1$. Plugging in this stronger lower bound gives us

$$\frac{d^+_{\Lambda}(u^*)}{OPT(G)} \leq \frac{d^+_{\Lambda'}(u^*) + f_{p\to u^*}}{OPT(G)} \leq \frac{d^+_{\Lambda'}(u^*)}{d^+_{\Lambda'}(u^*)} + \frac{f_{p\to u^*}}{k+1}$$
$$\leq 1 + \frac{k-1}{k+1} = 2 - \frac{2}{k+1}.$$

Since $2 - \frac{2}{k+1} \geq 3/2$ for $k \geq 3$, the approximation ratio is $2 - \frac{2}{k+1}$ for $k \geq 3$. It should be noted that the approximation ratio of REFINED CYCLE-CANCELING for $k = 2$ is $3/2$ (same as CYCLE-CANCELING) because then Step 6 is not executed. □

Figure 6 shows a worst-case example of REFINED CYCLE-CANCELING for $\{1, 3\}$-MMO. Since this example is also extendable to general $\{1, k\}$-MMO, the analysis of Theorem 3 is tight.

3.4 Approximation algorithm for large k

This subsection presents a simple approximation algorithm named LARGE-k for $\{1, k\}$-MMO which is suitable when $k \gg n$. Its approximation ratio is $1 + \frac{n}{2k}$ and its running time does not depend on k or W. The algorithm is described in Fig. 7.

The next theorem states the approximation ratio of Algorithm LARGE-k.

Theorem 4 *For any $S = \{1, k\}$, Algorithm* LARGE-k *approximates S-MMO within a ratio of $(1 + \frac{n}{2k})$ and runs in $O(m^{3/2} \cdot \log n)$ time.*

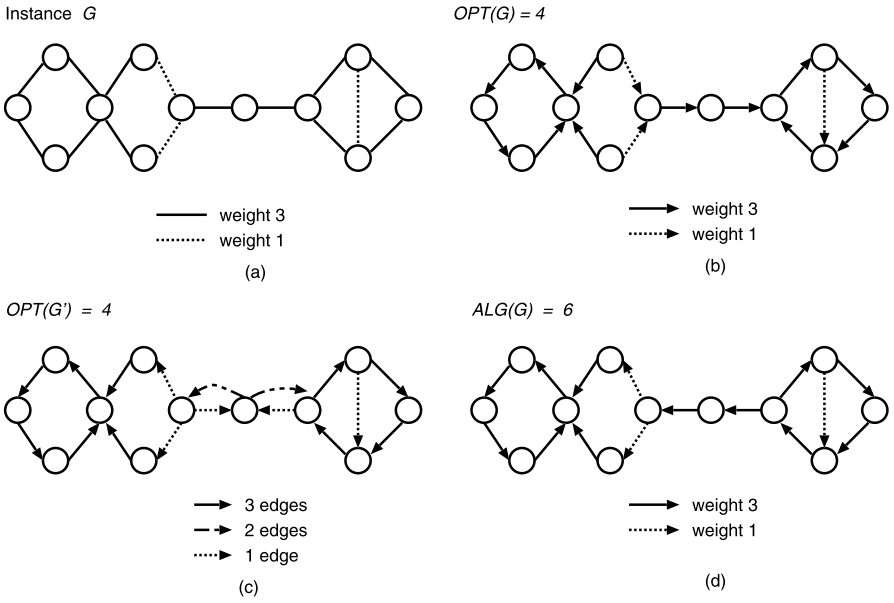

Fig. 6 A worst-case example for REFINED CYCLE-CANCELING: (**a**) an instance G, (**b**) an optimal orientation of G, (**c**) an optimal orientation Λ' of G', and (**d**) a possible output of REFINED CYCLE-CANCELING based on (**c**)

Algorithm LARGE-k

1. For the given graph G, construct two graphs $G_1 = (V, E_1)$ and $G_k = (V, E_k)$, where E_1 and E_k are the sets of edges with weight 1 and k, respectively.
2. Apply operation SOLVE-1-MMO to G_1 and G_k independently, and let Λ'_1 and Λ'_k be the returned optimal solutions.
3. Let Λ be the composite orientation of Λ'_1 and Λ'_k for the whole graph G.
4. Return Λ.

Fig. 7 Algorithm LARGE-k

Proof The running time is $O(m^{3/2} \cdot \log n)$ because SOLVE-1-MMO is called twice in Algorithm LARGE-k, and both $\log(\Delta_1^*(G_1))$ and $\log(\Delta_1^*(G_k))$ are $O(\log n)$. Next,

$$\frac{ALG(G)}{OPT(G)} = \frac{\max_{v\in V}\{d^+_{\Lambda'_1}(v) + d^+_{\Lambda'_k}(v)\}}{OPT(G)} \leq \frac{\max_{v\in V}\{d^+_{\Lambda'_1}(v)\} + \max_{v\in V}\{d^+_{\Lambda'_k}(v)\}}{OPT(G_k)}$$
$$= \frac{OPT(G_1) + OPT(G_k)}{OPT(G_k)} \leq \frac{n/2}{k} + 1,$$

since $OPT(G) \geq OPT(G_k) \geq k$ and since $n/2$ is a trivial upper bound for $OPT(G_1)$ derived from the complete graph with n vertices. □

4 Polynomial-time computation of {1}-MMO for G'

In this section, we develop a technique for making Algorithms MAJORITY, CYCLE-CANCELING, and REFINED CYCLE-CANCELING *polynomial-time* algorithms. Recall from the previous section that in these algorithms, we solve {1}-MMO for the graph G', which is generated from G by replacing each edge e with $w(e)$ edges of weight 1, as a sub-procedure. Although {1}-MMO for $G = (V, E)$ can be solved in $O(|E|^{3/2} \log |V|)$ time by the algorithm of Asahiro et al. (2007), {1}-MMO(G') requires $O(W^{3/2} \cdot \log \Delta^*)$ time, which is *pseudo-polynomial* time (that is, it is not necessarily polynomial in the length of the input). However, the information actually needed by MAJORITY, CYCLE-CANCELING, and REFINED CYCLE-CANCELING is not the orientation itself but the values $f_{u\to v}$ and $f_{v\to u}$. This section explains how to compute these values in polynomial time. The modified algorithm is presented in Fig. 10.

To find the values $f_{u\to v}$ and $f_{v\to u}$ efficiently, instead of explicitly constructing G' and applying SOLVE-1-MMO, we first solve a *relaxed version* of S-MMO where the orientation of any edge may be fractional, meaning that its weight may be distributed among both directions as (positive) non-integers. For example, an edge $\{u, v\}$ in G with weight 6 might be oriented as (u, v) with weight 3.6 and (v, u) with weight 2.4. The optimal solution to relaxed S-MMO can be obtained by solving a series of maximum directed flow problems as follows. Given the graph $G = (V, E, w)$ and a positive integer Δ_{tmp}, construct a flow network $\mathcal{N}_G = (V_\mathcal{N}, A_\mathcal{N}, cap)$, where $V_\mathcal{N} = V \cup E \cup \{s, t\}$, $A_\mathcal{N} = \{(s, e) \mid e \in E\} \cup \{(e, v_i), (e, v_j) \mid e = \{v_i, v_j\} \in E\} \cup \{(v, t) \mid v \in V\}$, and

$$cap(a) = \begin{cases} w(e), & \text{if } a = (s, e), \\ w(e), & \text{if } a = (e, v), \\ \Delta_{tmp}, & \text{if } a = (v, t). \end{cases}$$

Figures 8 and 9 show an example of a graph G and its corresponding network $\mathcal{N}_G$. Since $\mathcal{N}_G$ has only integral capacities, the flow integrality theorem (Cormen et al. 1990) ensures that the maximum flow value in $\mathcal{N}_G$ is an integer. It is straightforward to transform a maximum flow solution of $\mathcal{N}_G$ into a solution for relaxed S-MMO with value at most Δ_{tmp}. By applying a binary search on Δ_{tmp}, we can thus obtain an optimal solution for the relaxed version of the problem.

Next, we need to ensure that the obtained optimal orientation for relaxed S-MMO is always integral. (In general, an optimal orientation for relaxed S-MMO may not be integral even though the maximum flow value Δ_{opt} itself is an integer.) Therefore, in the description of Algorithm MODIFIED SOLVE-1-MMO(G') in Fig. 10, Steps 2, 3 and 4 have been added to ensure the integral flow property; the next paragraph explains in detail how this works. Note that we can skip Steps 2 to 4 if we employ a maximum flow algorithm in Step 1 which always outputs an integral optimal solution (such as the one by Goldberg and Rao 1998); however, we include these steps in the description of the algorithm for completeness so that it works for any selected maximum flow algorithm in Step 1.

The idea is simple. If we obtain a non-integral flow after Step 1, we adjust it to a solution of 1-MMO(G'), in which both $f_{u\to v}$ and $f_{v\to u}$ should be integral for any

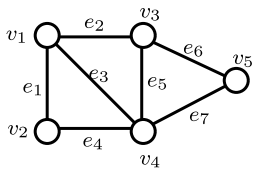

Fig. 8 A graph G

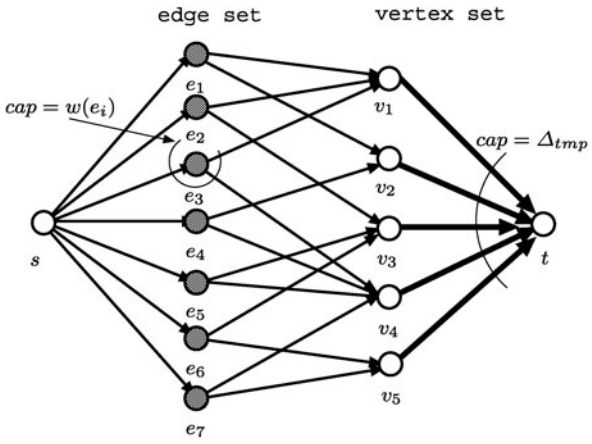

Fig. 9 The network $\mathcal{N}_G$ constructed from G in Fig. 8

Algorithm MODIFIED SOLVE-1-MMO(G')

1. Find an optimal orientation for relaxed S-MMO on G by solving maximum directed flow problems in $\mathcal{N}_G$ while doing a binary search to find Δ^*. Set $f_{u\to v}$ and $f_{v\to u}$ for every $\{u, v\} \in E$.
2. Construct $G^* = (V, E^*)$, where $E^* = \{(u, v) \mid f_{u\to v} - \lfloor f_{u\to v} \rfloor > 0\}$. If $E^* = \emptyset$, goto Step 5.
3. For a directed l-cycle C in G^* where $l \geq 3$, apply REVERSE-CYCLE to C with size $\min_{(u,v)\in C}\{f_{u\to v} - \lfloor f_{u\to v} \rfloor\}$ to update $f_{u\to v}$ for every $(u, v) \in C$, and goto Step 2. If no cycle in G^*, goto Step 4.
4. For G^*, apply UP-TO-ROOTS as described above.
5. Return Λ' as $f_{u\to v}$ and $f_{v\to u}$ for all $\{u, v\} \in E$.

Fig. 10 Algorithm MODIFIED SOLVE-1-MMO(G')

$\{u, v\} \in E$. For this purpose, we use REVERSE-CYCLE and UP-TO-ROOTS again. From the obtained solution Λ^* of relaxed S-MMO(G), we construct a directed graph $G^* = (V, E^*)$, where $E^* = \{(u, v) \mid f_{u\to v} - \lfloor f_{u\to v} \rfloor > 0\}$. Note that if G^* contains no edge, Λ^* is an integral optimal solution for relaxed S-MMO(G), that is, an optimal solution for 1-MMO(G'). Since G^* is a bidirectional graph, G^* contains a directed l-cycle with $l \geq 3$, is a (bidirectional) forest, or empty. If G^* contains a directed l-cycle C, we can update Λ^* so as to delete C by applying REVERSE-CYCLE to C with size $c = \min_{(u,v)\in C}\{f_{u\to v} - \lfloor f_{u\to v} \rfloor\}$ as in Sect. 3.2. By a similar argument,

we obtain a forest after at most m applications of REVERSE-CYCLE. Then we apply UP-TO-ROOTS that makes Λ^* integral; that is, from u on the forest to its parent p, update $f_{u\to p} := \lceil f_{u\to p} \rceil$ and $f_{p\to u} := \lfloor f_{p\to u} \rfloor$. This does not increase the value of Λ^* by the following reason: For node u on the forest, let $f(u) = \sum_{v\in\Gamma(u)} \lfloor f_{u\to v} \rfloor$, and p be the parent of u on the forest. Then the weighted outdegree of u under Λ^* before applying UP-TO-ROOTS is $f(u) + \sum_{v\in\Gamma(u)} (f_{u\to v} - \lfloor f_{u\to v} \rfloor) \le \Delta_{opt}$. Due to the integrality of $f(u)$ and Δ_{opt}, we have $f(u) \le \Delta_{opt} - 1$. After the UP-TO-ROOTS, the weighted outdegree of u becomes at most $f(u) + (f_{u\to p} - \lfloor f_{u\to p} \rfloor) + (f_{p\to u} - \lfloor f_{p\to u} \rfloor) = f(u) + 1 \le \Delta_{opt}$. By these, we can obtain an optimal orientation of 1-MMO(G') from any optimal orientation of relaxed S-MMO(G).

Finally, we consider the time complexity of Algorithm MODIFIED SOLVE-1-MMO(G'). Step 1 can be done in $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^*)$ time by a maximum flow algorithm (Goldberg and Rao 1998) while doing a binary search for Δ^*. Steps 2, 3 and 4 are not needed in this case (if executed, they would take $O(m^2)$ time) because the adopted maximum flow algorithm always returns an integral flow. Thus, the computations take $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^*)$ time in total.

Theorem 5 *Algorithm* MODIFIED SOLVE-1-MMO(G') *computes the* $f_{u\to v}$ *and* $f_{v\to u}$ *values of all the edges for* $\{1\}$*-MMO of* G' *in* $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^*)$ *time.*

Corollary 1 *The running time of algorithm* MAJORITY *can be improved to* $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^*)$. *Also, the running times of algorithms* CYCLE-CANCELING *and* REFINED CYCLE-CANCELING *can be improved to* $O(m^{3/2} \cdot \log n \cdot \log k \cdot \log \Delta^* + m^2)$.

Remark We can obtain a strongly polynomial-time algorithm by adopting another maximum flow algorithm such as King et al. (1994) instead of Goldberg and Rao (1998) in Algorithm MODIFIED SOLVE-1-MMO(G'). Then the running times of MAJORITY, CYCLE-CANCELING, and REFINED CYCLE-CANCELING become $O(m^3 \cdot \log_{2+1/\log n} n)$.

5 Inapproximability results

It was shown in Asahiro et al. (2007) that S-MMO is NP-hard by a reduction from the PARTITION problem, which has a pseudo-polynomial time algorithm. This implies that S-MMO was only known to be weakly NP-hard. Also, no previous results about the inapproximability of S-MMO exist. In this section, we provide a proof of the strong NP-hardness of S-MMO which also yields inapproximability results. More precisely, we give a reduction from a variation of the 3-SAT problem, At-most-3-SAT(2L), to $\{1, k\}$-MMO for any fixed integer $k \ge 2$.

At-most-3-SAT(2L) is a restriction of 3-SAT where each clause includes at most three literals and each literal (not variable) appears at most twice in a formula. It can easily be proved that At-most-3-SAT(2L) is NP-hard by using problem [LO1] on p. 259 of Garey and Johnson (1979).

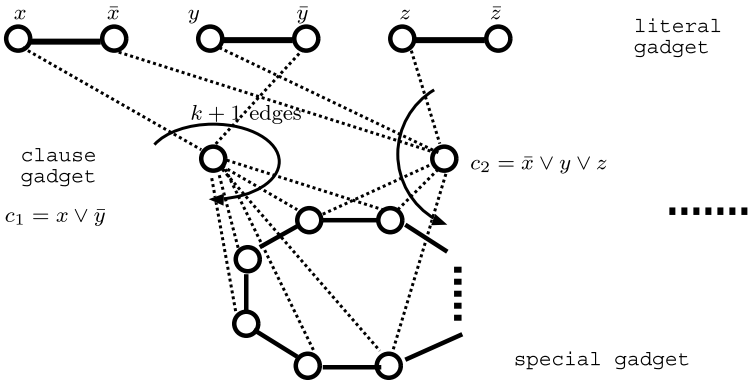

Fig. 11 Reduction from At-Most-3-SAT(2L) to $\{1, k\}$-MMO

The reduction from At-most-3-SAT(2L) to $\{1, k\}$-MMO is as follows. Given a formula ϕ of At-most-3-SAT(2L) with g variables $\{v_1, \ldots, v_g\}$ and h clauses $\{c_1, \ldots, c_h\}$, we construct a graph G_ϕ including gadgets that mimic (a) literals, (b) clauses, and (c) a special gadget:

(a) Each literal gadget consists of two vertices labeled by v_i and $\overline{v_i}$ and one edge $\{v_i, \overline{v_i}\}$ between them, corresponding to variable v_i of ϕ. The weight of $\{v_i, \overline{v_i}\}$ is k.
(b) Each clause gadget is one vertex (called a *clause vertex*) labeled by c_j, corresponding to clause c_j of ϕ. The clause vertex c_j is connected by edges of weight 1 to at most three vertices in the literal gadgets that have the same labels as the literals in the clause c_j. For example, if $c_1 = x \vee \overline{y}$ appears in ϕ, then vertex c_1 is connected to vertices x and $\overline{y}$. See Fig. 11.
(c) The special gadget is a cycle of k vertices and k edges where each edge of the cycle has weight k.[3] For each clause, if it consists of one (two or three, respectively) variable(s), then its clause vertex is connected to k ($k-1$ or $k-2$, respectively) arbitrary vertices in the special gadget by edges of weight 1. Hence, the degree of every clause vertex is exactly $k+1$.

Lemma 1 For the above construction of G_ϕ, the following holds:

(i) If ϕ is satisfiable, then $OPT(G_\phi) \leq k$.
(ii) If ϕ is not satisfiable, then $OPT(G_\phi) \geq k+1$.

Proof To prove (i), suppose there exists a satisfying truth assignment for ϕ. From the assignment, we construct an orientation of G_ϕ with value $OPT(G_\phi) \leq k$. If $v_i = \mathit{true}$ in the assignment, the edge $\{v_i, \overline{v_i}\}$ is oriented from $\overline{v_i}$ to v_i; otherwise, from v_i to $\overline{v_i}$. So far, the outdegree of every vertex associated with the literals of *true* and *false* assignments is 0 and k, respectively. We call the vertices associated with literals of

[3] In case $k = 2$, we prepare a cycle of 3 vertices as an exception to keep the simple property of the graph.

true (resp., *false*) assignments *true* (resp., *false*) *vertices*. (For example, in Fig. 11, if the variable $x = false$ in the truth assignment then the upper leftmost vertex x is a false vertex and the second leftmost vertex $\overline{x}$ is called a true vertex.) For every clause vertex c_j, we select one edge connected to a true vertex and orient it towards c_j, and orient the remaining k edges away from c_j. This orientation of the edges does not increase the outdegree of false vertices or any extra true vertex; it is still at most k. Since every literal appears at most twice in ϕ, the outdegree of true vertices in the literal gadgets is at most two. Finally, edges belonging to the special gadget can be oriented cyclically. Thus, the maximum outdegree of G_ϕ is at most k.

Next, we prove (ii) by showing that if G_ϕ has an orientation whose maximum outdegree is at most k then ϕ is satisfiable by constructing the satisfying truth assignment. If an edge in the ith literal gadget v_i is oriented from v_i to $\overline{v_i}$ then we assign $v_i = false$; otherwise, $v_i = true$. Since every clause vertex is connected to the literal gadgets and special gadgets by $k+1$ edges, and every edge between a clause gadget and the special gadget must be oriented towards the special gadget (if not, the maximum outdegree of the special gadget would be at least $k+1$), it follows that for each clause vertex c_j, there must be at least one edge directed towards c_j from a vertex v in a literal gadget, and v must therefore be a true vertex. This means that the above truth assignment satisfies all clauses in ϕ. □

From Lemma 1, we immediately obtain:

Theorem 6 *$\{1, k\}$-MMO for any fixed $k \geq 2$ is strongly NP-hard.*

Corollary 2 *$\mathbb{Z}^+$-MMO is strongly NP-hard.*

In addition, the (in)satisfiability gap of Lemma 1 directly yields the next theorem and corollary.

Theorem 7 *$\{1, k\}$-MMO, where $k \geq 2$ is fixed, has no pseudo-polynomial time algorithm with approximation ratio less than $1 + 1/k$, unless $P = NP$.*

Corollary 3 *$\mathbb{Z}^+$-MMO has no pseudo-polynomial time algorithm with approximation ratio less than $3/2$, unless $P = NP$.*

6 Concluding remarks

6.1 Relation to scheduling

As mentioned in Sect. 1.1, one application of the minimization of the maximum outdegree is scheduling. For an undirected graph, let us consider the vertices as the machines and the edges as the jobs. Then S-MMO can be regarded as a special case of the job assignment problem (Pinedo 2002) in which the minimization of the maximum outdegree means to minimize the finishing time of all the jobs. From the viewpoint of scheduling, our problem has some restrictions: (1) each job must be assigned to exactly one of two predetermined machines, and (2) the processing time of

each job does not depend on the machines. Therefore, S-MMO is a special case of *scheduling on unrelated parallel machines*, or $R||C_{max}$ in standard notation: given a set J of jobs, a set M of machines, and the time $p_{ij} \in \mathbb{Z}^+$ taken to process job $j \in J$ on machine $i \in M$, its goal is to find an assignment of all jobs to the machines so as to minimize the makespan, i.e., the maximum processing time of any machine. Lenstra et al. (1990) gave a polynomial-time 2-approximation algorithm based on the LP-formulation for the general version of $R||C_{max}$ and a ratio 3/2 inapproximability result (see also Schuurman and Woeginger 1999). Alternatively, S-MMO can be regarded as a variant of *scheduling on identical parallel machines*, in which each job can be processed by any of the machines and the processing time p_{ij} of job j on machine i is fixed to be p_j, independent of i. This problem has an FPTAS (Horowitz and Sahni 1976), which contrasts with our inapproximability results for S-MMO. Another interesting difference concerns the set of processing times, or the weight set: $R||C_{max}$ has a polynomial-time algorithm for the special case where the weight set is $\{p, q\}$ with $q = 2p$, but S-MMO remains NP-hard even if all edge weights belong to $\{p, q\}$ with $q = 2p$ (this is possible because S-MMO corresponds to $R||C_{max}$ with weight set $\{p, q, \infty\}$).

Observe that the 3/2-inapproximability result of Lenstra et al. (1990) cannot be applied directly to the restricted assignment variant in which every job can be processed on a *constant number* of machines. In S-MMO, each job associated with an edge can be assigned only to one of the *two* machines associated with the two vertices of that edge. Moreover, their inapproximability proof requires the assumption that the processing time of each job may vary depending on which machine it is processed on. Thus, their inapproximability result does not apply to our case, and in this sense, our result provides a stronger inapproximability bound.

6.2 Open problems

Several open problems remain. One concerns the gap between the polynomial-time approximability and inapproximability of $\{1, k\}$-MMO. For $k = 2$, they coincide, but in the current result, the gap between $2 - 2/(k+1)$ and $1 + 1/k$ increases for larger k. On the other hand, for very large k, it is easy to get a better approximation ratio, as shown in Sect. 3.4. To further investigate that relationship would be interesting. Another topic is to design faster strongly polynomial-time approximation algorithms with a good approximation ratio.

Also, what is the time complexity of $\{k\}$-MMO? SOLVE-1-MMO was shown in Asahiro et al. (2007) to solve $\{k\}$-MMO in $O(m^{3/2} \cdot \log(\Delta_1^*))$ time, but we believe that faster methods may be possible, e.g., by avoiding the binary search. A faster algorithm for $\{k\}$-MMO would immediately imply a faster implementation for Algorithm LARGE-k in Sect. 3.4, for example.

Finally, are there any graph classes besides forests which admit polynomial-time exact solutions? On the negative side, it seems that the techniques in Sect. 5 can be extended to prove that $\{1, k\}$-MMO (and thus also $\mathbb{Z}^+$-MMO) remain hard to approximate even if restricted to planar graphs or if restricted to bipartite graphs. We are currently working on resolving this issue. The problem also seems NP-hard for series-parallel graphs, which would imply that $\mathbb{Z}^+$-MMO on bounded treewidth graphs is NP-hard since series-parallel graphs have treewidth at most 2.

References

Asahiro Y, Miyano E, Ono H, Zenmyo K (2007) Graph orientation algorithms to minimize the maximum outdegree. Int J Found Comput Sci 18(2):197–216

Biedl T, Chan T, Ganjali Y, Hajiaghayi MT, Wood DR (2005) Balanced vertex-orderings of graphs. Discrete Appl Math 48(1):27–48

Brodal GS, Fagerberg R (1999) Dynamic representations of sparse graphs. In: Proc WADS1999. LNCS, vol 1663, pp 342–351

Chrobak M, Eppstein D (1991) Planar orientations with low out-degree and compaction of adjacency matrices. Theor Comput Sci 86(2):243–266

Chv́atal V (1975) A combinatorial theorem in plane geometry. J Comb Theory, Ser B 18:39–41

Cormen T, Leiserson C, Rivest R (1990) Introduction to algorithms. MIT Press, Cambridge

Fomin FV, Matamala M, Rapaport I (2004) Complexity of approximating the oriented diameter of chordal graphs. J Graph Theory 45(4):255–269

Garey M, Johnson D (1979) Computers and intractability: A guide to the theory of NP-completeness. W H Freeman, New York

Goldberg AV, Rao S (1998) Beyond the flow decomposition barrier. J ACM 45(5):783–797

Horowitz E, Sahni S (1976) Exact and approximate algorithms for scheduling nonidentical processors. J ACM 23(2):317–327

Kára J, Kratochvíl J, Wood DR (2005) On the complexity of the balanced vertex ordering problem. In: Proc COCOON2005. LNCS, vol 3595, pp 849–858

King V, Rao S, Tarjan R (1994) A faster deterministic maximum flow algorithm. J Algorithms 17:447–474

Kowalik L (2006) Approximation scheme for lowest outdegree orientation and graph density measures. In: Proc ISAAC2006. LNCS, vol 4288, pp 557–566

Lenstra JK, Shmoys DB, Tardos E (1990) Approximation algorithms for scheduling unrelated parallel machines. Math Program 46(3):259–271

O'Rourke J (1987) Art gallery theorems and algorithms. Oxford University Press, Oxford

Pinedo M (2002) Scheduling: Theory, algorithms, and systems, 2nd edn. Prentice-Hall, Englewood Cliffs

Schrijver A (2003) Combinatorial optimization. Springer, Berlin

Schuurman P, Woeginger GJ (1999) Polynomial time approximation algorithms for machine scheduling: Ten open problems. J Sched 2:203–213

Venkateswaran V (2004) Minimizing maximum indegree. Discrete Appl Math 143(1–3):374–378