Approximation Algorithms for the Graph Orientation Minimizing the Maximum Weighted Outdegree*

Yuichi Asahiro[1] , Jesper Jansson[2,**] , Eiji Miyano[3] , Hirotaka Ono[2], and Kouhei Zenmyo[3]

[1] Department of Social Information Systems, Kyushu Sangyo University, Fukuoka 813-8503, Japan
asahiro@is.kyusan-u.ac.jp
[2] Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka 812-8581, Japan
{jj@tcslab.,ono@}csce.kyushu-u.ac.jp
[3] Department of Systems Innovation and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan
{miyano@,kouhei@theory.}ces.kyutech.ac.jp

Abstract. Given an 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. In this paper (1) we prove that the problem is strongly NP-hard if all edge weights belong to the set $\{1, k\}$, where k is any 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 polynomial time algorithm that approximates the general version of the problem within a factor 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 factor 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.

1 Introduction

1.1 Problems and Summary of Results

Let $G = (V, E, w)$ be a simple, undirected and weighted graph, where V, E and w denote the set of nodes, the set of edges and a positive integral weight function $w : E \to \mathbb{Z}^+$, respectively. Throughout the paper, let $|V| = n$ and $|E| = m$ for the graph. An *orientation Λ of the graph G* is an assignment of a

* This work is partially supported by Grant-in-Aid for Scientific Research on Priority Areas No. 16092223, and by Grant-in-Aid for Young Scientists (B) No. 17700022, No. 18700014 and No. 18700015.
** Supported by JSPS (Japan Society for the Promotion of Science).

M.-Y. Kao and X.-Y. Li (Eds.): AAIM 2007, LNCS 4508, pp. 167–177, 2007.

direction to each edge $\{u, v\} \in E$, i.e., $\Lambda(\{u, v\})$ is either (u, v) or (v, u). The *weighted outdegree* of u is $d^+_\Lambda(u)$, where $d^+_\Lambda(u)$ denotes $\sum_{\substack{\{u,v\}\in E \\ \Lambda(\{u,v\})=(u,v)}} w(\{u, v\})$.

We consider the problem of finding an orientation such that the maximum weighted outdegree is minimum. This basic problem has several applications. For example, such orientations 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 [3]. Also, it can be considered a variation of *art gallery problems* (e.g., [4,11]) and *unrelated parallel machine scheduling* (e.g., [10]). Especially, the polynomial time (in)approximability of the latter problem has been intensively studied, as discussed in the next subsection.

Previous studies show that our problem can be solved in polynomial time if all the edge weights are identical [1,9,15], while it is NP-hard in general [1]. Also, a $(2 - 1/\lceil L(G)\rceil)$-approximation algorithm with $O(m^2)$ running time was presented in [1], where $L(G) = \max_{H\subseteq G} \{\sum_{\{u,v\}\in E(H)} w(\{u, v\})/|V(H)|\}$.

In this paper, we consider the problem from the viewpoint of polynomial time approximability and inapproximability. Our results are summarized as follows:

- We present a $(2-1/k)$-approximation algorithm with running time $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^* + m^2)$, where k, m and Δ^* denote the maximum weight of the edges, the number of the edges and the optimal value, respectively.
- For special cases in which the weight of each edge is either 1 or k, a refined algorithm achieves a better approximation factor, $2 - 2/(k + 2)$, also with running time $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^* + m^2)$.
- We prove that there is no polynomial time approximation algorithm whose factor is smaller than 3/2, unless P=NP. (More precisely, in case where weights of all the edges are either 1 or a positive integer $k \geq 2$, no pseudo-polynomial time algorithm achieves an approximation ratio smaller than $1 + 1/k$.) That is, for $k = 2$, the above algorithm is best possible with respect to the approximation ratio.

Note that the new $2 - 1/k$-approximation ratio in this paper and the previous $2 - 1/\lceil L(G)\rceil$ one in [1] are incomparable; sometimes the former is better than the latter, and vice versa. For example, we have an instance for which the latter algorithm outputs 5/3-factor solution, while the former achieves approximation ratio 1.5 (see Figure 6 in [1]). Due to space limitations, the formal proofs have been omitted in this paper. Please refer to the full paper for a complete version.

1.2 Related Work

Graph orientation itself is a quite basic, natural and important problem in graph theory and combinatorial optimization (see Chapter 61 of [13]). However, most of the studies consider the problems of finding an orientation with lower outdegree satisfying some special graph properties, such as high connectivity, small diameter, no-cycle and so on [2,5,8], and very few studies consider just the minimization of the maximum outdegree (or indegree) [1,15].

As mentioned in the previous subsection, another aspect 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 our orientation problem can be regarded as a special case of the job assignment problem, in which the minimization of the maximum outdegree means to minimize the finishing time of all the jobs [12]. From the viewpoint of scheduling, our problem has some restriction, that is, 1) each job must be assigned to exactly one of pre-determined two machines, and 2) the processing time of each job does not depend on the machines. Therefore, our problem is a special case of *scheduling on unrelated parallel machines* ($R||C_{max}$ in the now-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 a job scheduling so as to minimize the makespan, i.e., the maximum processing time of any machine. In [10], Lenstra, et al. gave a polynomial time 2-approximation algorithm that is based on the LP-formulation for the general $R||C_{max}$ and its 3/2 inapproximability result (see also [14].)

Note that the 3/2 inapproximability result of Lenstra, et al. *cannot* be directly applied to the restricted assignment variant in which every job can be processed on a *constant number* of machines. In our problem, each job associated with an edge can be assigned only to one of the *two* machines associated with the two nodes of the edge, which means that their proof is not applicable to our case. Also note that their proof of inapproximability uses the assumption that the processing time of each job may vary depending on which machine it is processed on. Thus, our result provides a stronger inapproximability bound to the problem.

2 Preliminaries

2.1 Definitions

Let $G = (V, E, w)$ be a simple, undirected, weighted graph, where V, E, and w denote a set of vertices, a set of edges, and an integral weight function, $w : E \to \mathbb{Z}^+$, respectively. Let $w_{\max}$ and W be the maximum weight of edges and the total weight of edges, respectively. We denote the undirected edge whose endpoints are u and v where $u < v$ in lexicographic order by $e_{u,v}$, or simply $\{u, v\}$, and denote the directed edge (or *arc*) from u toward v, by (u, v). An *orientation* Λ of the undirected graph G is an assignment of direction to each edge $\{u, v\} \in E$, i.e., (u, v) or (v, u). A *directed path* P *of length* l from a vertex v_0 to a vertex v_l in a directed graph $G = (V, A, w)$ is a set $\{(v_{i-1}, v_i) \mid (v_{i-1}, v_i) \in A, i = 1, 2, \ldots, l$ and $v_i \neq v_j$ for any i and $j\}$ of arcs, which is also denoted by a sequence $\langle v_0, v_1, \ldots, v_l\rangle$ for simplicity. For the path P, the path of its reverse order is denoted by $\overline{P}$, i.e., $\overline{P} = \langle v_l, v_{l-1}, \ldots, v_0\rangle$. Especially, a directed path P satisfying $v_l = v_0$ is called an *l-directed cycle*.

Let $d^+_\Lambda(v)$ and $d^-_\Lambda(v)$ under an orientation Λ denote the total weight of outgoing arcs and that of incoming arcs of a vertex v in the weighted directed graph $G(V, A, w)$, which we call the *weighted outdegree* and the *weighted indegree* of v, respectively. Throughout the paper, we use the words "outdegree" and "indegree" to represent these weighted degrees. Then the *cost* of an orientation Λ for a graph G is defined to be $\Delta_\Lambda(G) = \max_{v \in V}\{d^+_\Lambda(v)\}$. For an undirected graph

$G = (V, E)$ and a node $u \in V$, we define $\Gamma(u) = \{v \mid \{u, v\} \in E\}$, the set of neighbors of u. Given an orientation Λ of G, we define $\Gamma_\Lambda(u) = \{v \mid \{u, v\} \in E$ and $\Lambda(\{u, v\}) = (u, v)\}$, the set of neighbors of u on G under Λ.

Every orientation has the following trivial lower bound caused by the maximum weight of edges:

Proposition 1. ([1]) For a graph G and any orientation Λ, $\Delta_\Lambda(G) \geq w_{\max}$. □

2.2 Problem and Basic Operations

The problem that we consider in this paper is the minimization of the maximum outdegree of a given undirected weighted simple graph. To specify the class of weight function of the graph, we formally define our problem as follows.

Problem:	S-Minimum Maximum Outdegree (S-MMO)
Input:	An undirected graph $G = (V, E)$ and a weight function $w : E \to S$, where S is a set of weights.
Output:	An orientation Λ that minimizes $\max\{d^+_\Lambda(u) \mid u \in V\}$.

Namely, if we have no restriction about the weight function (just it should be a positive integral function), our problem is $\mathbb{Z}^+$-MMO. In this paper, we mainly consider the problem for the case of $S = \{1, 2, \ldots, k\}$. We also consider a special case in which the range of w is restricted to $S = \{1, k\}$ with $k \geq 2$.

Let OPT denote an optimal orientation. We say a graph orientation algorithm is a σ-approximation algorithm if $ALG(G)/OPT(G) \leq \sigma$ holds for any undirected graph G, where $ALG(G)$ is the objective value of a solution obtained by the algorithm for G, and $OPT(G)$ is that of an optimal solution. In the following we use $OPT(G)$ or Δ^* to denote the optimal value.

Here we introduce three basic operations; Reverse, Up-To-Roots and Solve-1-MMO.

- Reverse does the following: *Given an orientation Λ of graph G and a directed path $P = \langle u_0, u_2, \ldots, u_l \rangle$ in G under Λ, update Λ by replacing P with $\overline{P}$, i.e., let $\Lambda(e_{u_i,u_{i+1}}) = (u_{i+1}, u_i)$ for $i = 0, \ldots, l-1$.* Note that the outdegree for each vertex remains the same after the operation if P is a directed cycle and $w(e_{u_i,u_{i+1}})$'s are all identical. We call this operation ReverseCycle if $u_0 = u_l$.
- Up-To-Roots determines an orientation Λ for a given simple forest G, in the following manner: *First fix an arbitrary root for each connected component of G (it is a tree). Then for every edge e, orient $\Lambda(e)$ towards the root of the tree containing e.* Note that for a forest with weighted edges Up-To-Roots operation returns an optimal solution, whose value is $w_{\max}$ [1].
- Solve-1-MMO outputs an optimal orientation Λ for a given undirected graph G with identical weights. It is shown in [1] that the running time of Solve-1-MMO is $O(m^{3/2} \cdot \log(\Delta^*/k))$ for $\{k\}$-MMO, in which the log factor comes from the binary search.

3 Approximation Algorithms

In this section, we present three pseudo-polynomial time approximation algorithms for the S-MMO problem. The first and the second algorithms (in Sections 3.1 and 3.2) work for S-MMO with $S = \{1, 2, \ldots, k\}$, both of which are based on the replication of weighted edges, and their approximation ratios are 2 and $2 - 1/k$, respectively. The third algorithm (in Section 3.3) for $\{1, k\}$-MMO is a refined version of the second one, and its approximation ratio is $2 - 2/(k+1)$ for $k \geq 3$. In Section 3.4, we show how to improve the running times of the three approximation algorithms to polynomial time.

3.1 Majority Voting Algorithm

We first present a basic 2-approximation algorithm, named Majority. Although Majority can be considered a variation of Lenstra-Shmoys-Tardos algorithm [10] (LST, for short), which is based on the LP-rounding and has approximation factor 2, Majority is combinatorial and provides basic ideas for the algorithms presented later. Also it is much faster than LST, by Corollary 1.

The idea of the algorithm is as follows: We replace each edge $e = \{u, v\}$ in G with $w(e)$ edges of weight 1 between u and v, and then we obtain an undirected multi-graph G' with $W = \sum_{e \in E} w(e)$ edges. We find an optimal MMO orientation Λ' for G', and then we decide an orientation of each weighted edge on G according to Λ' by the majority voting manner; in Λ', for each $e_{u,v} \in E$, some of replicated edges of $e_{u,v}$ are oriented from u to v and the others from v to u. Let us denote the number of edges from u to v (resp., from v to u) in Λ' by $f_{u \to v}$ (resp., $f_{v \to u}$). Since we assume the original graph is simple, $f_{u \to v} + f_{v \to u} = w(e_{u,v})$ holds. By using these, we decide the orientation Λ of the original G by the following manner: For $e_{u,v} \in E$,

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

In the case of a tie the direction is determined according to a lexicographic order. We call this algorithm Majority.

Algorithm Majority

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

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

Proof. Since Steps 1, 2 and 3 take $O(W)$, $O(W^{3/2} \log \Delta^*)$ and $O(W)$ time, respectively, the running time of Majority is $O(W^{3/2} \log \Delta^*)$, in total. The approximation factor 2 is immediately obtained by the result of [10]. □

3.2 Cycle Canceling Algorithm

Here, we describe a new algorithm named Cycle-Canceling, which improves Majority; the approximation ratio is $2 - 1/k$.

Algorithm Cycle-Canceling

1. For graph G, construct G' by replacing each edge e with $w(e)$ edges.
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, $G_{\Lambda'} = (V, F_{\Lambda'})$ as described later.
4. If $G_{\Lambda'}$ has an l-directed cycle with $l \geq 3$, apply ReverseCycle and go to 3.
5. For undecided edges of Λ, apply Up-To-Roots.
6. Return Λ.

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 we decide the 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 same as the one of Λ'; the cost of the orientation does not change.

Here, we introduce a new operation, *cycle cancellation*, which updates the orientation to more desirable orientation without changing the outdegrees of all the nodes. 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 exists. (From here, when we mention l-cycles with $l \geq 3$, we just use "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 $F_{\Lambda'}$. We then go back to G' and Λ' and apply ReverseCycle 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. Note that the outdegree (or the indegree) of each node 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 the equation (2), then go back to the beginning of this paragraph. Since at least one edge on the cycle C satisfies $f_{v_i\to v_{i+1}} = 0$ by the ReverseCycle, the new $F_{\Lambda'}$ is strictly smaller than the old $F_{\Lambda'}$; this step ends in at most $m-2$ iterations.

After the several (or possibly no) iterations of the above procedure, $G_{\Lambda'}$ becomes a forest, and set $\mathcal{F} := G_{\Lambda'}$. Note that all the edges of $\mathcal{F}$ are not decided yet by (2). The cycle cancellation 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 nice tree structure, for which we can apply Up-To-Roots operation that decides the orientation of all the remaining edges.

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

Proof. We first consider the running time. Steps 1 and 2 require the same time complexity as Majority, i.e., $O(W^{3/2} \log \Delta^*)$ time. Each iteration of Steps 3 takes $O(m)$ time, and also each iteration of Steps 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 $O(W^{3/2} \log \Delta^* + m^2)$.

Next, we analyze the approximation factor. Let u^* be any critical node in G with respect to Λ, i.e., a node with maximum weighted outdegree under Λ. We now 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 node 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 $\mathcal{F}^*$: [1] 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. 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

$$d^+_\Lambda(u^*) = d^+_{\Lambda'}(u^*) + f_{p \to u^*} - \sum_{v \in \mathcal{C}} f_{u^* \to v} \leq d^+_{\Lambda'}(u^*) + f_{p \to u^*},$$

which yields

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

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

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

Note that the analysis of Theorem 2 is tight; we can construct a worst-case example of Cycle-Canceling for $\{1, 3\}$-MMO (see the full-length version of our paper).

Remark: According to Theorem 2, the approximation factor of Algorithm 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 see in Section 4.

[1] This case also handles the possibility that u^* is disconnected from all other vertices in $G_{\Lambda'}$.

3.3 Refined Cycle Canceling Algorithm

We now consider the special case of S-MMO in which $S = \{1, k\}$ for $k \geq 3$, and show that it can be approximated more efficiently than by Theorem 2. The key idea is to show that if all edge weights in G are either 1 or k, a slight modification to Algorithm Cycle-Canceling allows us to compute a stronger lower bound on an optimal solution which then yields an improved approximation factor.

As mentioned in the previous section, the cycle cancellation itself provides an optimal solution for the relaxed problem with a tree property. Here, we focus on Step 5 of the algorithm Cycle-Canceling, in which the naive application of Up-To-Roots with arbitrary roots gives a worst-case example; this causes the approximation ratio to be $2 - 1/k$. Its reason is that some nodes having large outdegree under the orientation Λ' are not suitable for being root; if such a node is set to be a root, its outdegree will distribute to its neighbors, so that the neighbors have large outdegree under Λ compared to that under Λ'. To avoid such a bad situation, we introduce a simple procedure.

In the algorithm, do the same operations as Cycle-Canceling until Step 4, 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 $F := F \setminus \{e_{u,v}\}$). We 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 we apply Up-To-Roots.

Algorithm Refined Cycle-Canceling

1-4. (Same as Cycle-Canceling).
4'. While there exists a leaf 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.
5. For undecided edges of Λ, apply Up-To-Roots to $\mathcal{F}$.
6. Return Λ.

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

Proof. It is easy to see that adding Step 4' to Algorithm Cycle-Canceling in Section 3.2 does not increase the asymptotic running time. Therefore, the running time is $O(W^{3/2} \log \Delta^* + m^2)$.

To analyze the approximation factor of Refined Cycle-Canceling, we proceed similarly as in the proof of Theorem 2. Let u^* be any critical node in G with respect to Λ, and let $\mathcal{F}^*$ be the forest of rooted trees produced by Up-To-Roots in Step 5. 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 4': 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 4':
   (a) If u^* is a root in $\mathcal{F}^*$ then $d^+_{\Lambda}(u^*) \leq d^+_{\Lambda'}(u^*) \leq OPT(G)$ and we are done as before.
   (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 is $2 - \frac{2}{k+1}$ for $k \geq 3$ in total. Note that the approximation ratio of Refined Cycle-Canceling for $k = 2$ is $3/2$ (same as Cycle-Canceling) because Step 5 is not executed. □

The analysis of Theorem 3 is also tight; we can construct a worst-case example of Refined Cycle-Canceling for $\{1, 3\}$-MMO.

3.4 Polynomial Time Computation of 1-MMO of G'

In this subsection, we show the technique of making Algorithms Majority, Cycle-Canceling and Refined Cycle-Canceling into polynomial time algorithms. Recall that in these algorithms, we have to solve 1-MMO for G', which is generated from G by replacing each edge e with $w(e)$ edges of weight 1, as a sub-procedure. Hence, as described in Section 3.1, the algorithm requires $O(W^{3/2} \cdot \log \Delta^*)$ time only to obtain an optimal solution of 1-MMO. However, the information that algorithms Majority, Cycle-Canceling and Refined Cycle-Canceling need is not the orientation itself but the values $f_{u \to v}$ and $f_{v \to u}$, which can be computed in polynomial time.

The idea is as follows: Instead of explicitly constructing G' and applying Solve 1-MMO, we solve a relaxed version of the problem by using a maximum network flow technique. The relaxed version means that for each edge, its orientation may be fractional. For example, edge $e = \{u, v\}$ with weight 2 may be oriented as (u, v) with weight 1.5 and (v, u) with weight 0.5. Although the relaxed optimal solution can contain fractional flows in some edges, the integral maximum flow problem is known to have an optimal solution of integral flows (flow integrality) and some standard algorithms find such solutions indeed (for example, [7] presents $O(m \min\{m^{1/2}, n^{2/3}\} \log(n^2/m) \log U)$-time algorithm, where U is the maximum capacity size). Thus, the solution can be regarded as an optimal solution of 1-MMO for G'. Although we omit the detail, the problem can be solved by computing $O(\log \Delta^*)$ times the maximum flow for a network of $m + n$ vertices and $3m$ arcs with the maximum capacity k, which leads the following.

Theorem 4. We can compute the $f_{u\to v}$ and $f_{v\to u}$ values of all the edges for 1-MMO of G' in $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^*)$ time. □

Corollary 1. (a) The running time of Algorithm MAJORITY can be improved to $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^*)$ time, and also (b) the running time of Algorithms CYCLE-CANCELING and REFINED CYCLE-CANCELING can be improved to $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^* + m^2)$ time. □

4 Inapproximability Results

It is shown that S-MMO is weakly NP-hard [1], but no result about the inapproximability is shown. In this section, we provide a proof of the strong NP-hardness of S-MMO, which also gives inapproximability results. More precisely, we give a reduction from a variation of 3-SAT problem, At-most-3-SAT(2L), to $\{1, k\}$-MMO. 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 be easily proved that At-most-3-SAT(2L) is NP-hard by using problem [LO1] on p. 259 of [6].

Given a formula ϕ of At-most-3-SAT(2L) with n variables $\{v_1, \ldots, v_n\}$ and m clauses $\{c_1, \ldots, c_m\}$, 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 nodes 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 node labeled by c_j, corresponding to clause c_j of ϕ. The clause gadget c_j is connected to at most three nodes in the literal gadgets that have the same labels as the literals in the clause c_j, by edges of weight 1. For example, if $c_1 = x \vee \overline{y}$ is appeared in ϕ, then node c_1 is connected to nodes x and $\overline{y}$. (See Figure 1.) (c) The special gadget is a cycle of k nodes and k edges where each edge of the cycle

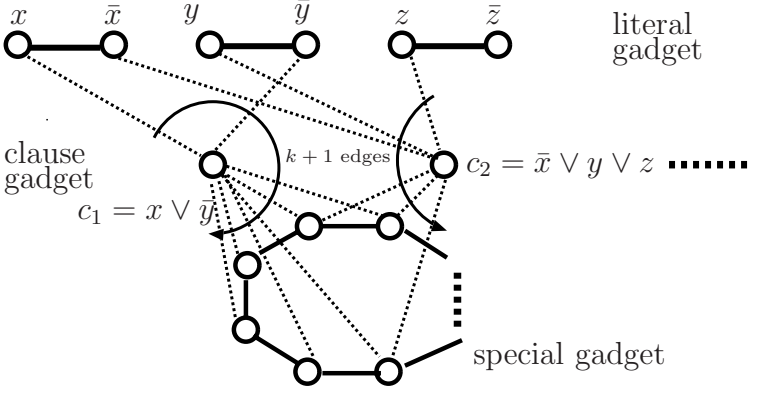

Fig. 1. Reduction from At-Most-3-SAT

has weight k.[2] If a clause consists of one (two or three, resp.,) variable(s), then it is connected to k (arbitrary $k-1$ or $k-2$, resp.,) nodes in the special gadget by edges of weight 1. Hence, the degree of every clause node is exactly $k+1$.

We can prove the following:

Lemma 1. For the above construction of G_ϕ, the followings hold: (i) If ϕ is satisfiable, $OPT(G_\phi) \leq k$. (ii) If ϕ is not satisfiable, $OPT(G_\phi) \geq k+1$. □

From Lemma 1, we immediately obtain the following theorem.

Theorem 5. $\{1, k\}$-MMO is strongly NP-hard. Consequently, $\mathbb{Z}^+$-MMO is also strongly NP-hard. □

Also the (in)satisfiability gap of Lemma 1 yields the following theorem.

Theorem 6. $\{1, k\}$-MMO (resp., $\mathbb{Z}^+$-MMO) has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $1 + 1/k$ (resp., $3/2$), unless P=NP. □

References

1. Y. Asahiro, E. Miyano, H. Ono, and K. Zenmyo, Graph orientation algorithms to minimize the maximum outdegree, *Proceedings of Computing: the Twelfth Australasian Theory Symposium (CATS 2006)*, pp. 11–20, 2006.
2. T. Biedl, T. Chan, Y. Ganjali, M. T. Hajiaghayi and D. R. Wood, Balanced vertex-orderings of graphs, *Discrete Applied Mathematics*, 48 (1), pp. 27–48, 2005.
3. G. S. Brodal and R. Fagerberg, Dynamic Representations of Sparse Graphs, *Proc. WADS1999, LNCS 1663*, pp. 342–351, 1999.
4. V. Chvátal, A combinatorial theorem in plane geometry, *J. Combinatorial Theory, series B*, 18, pp. 39–41, 1975.
5. F. V. Fomin, M. Matamala and I. Rapaport, Complexity of approximating the oriented diameter of chordal graphs, *J. Graph Theory*, 45 (4), pp. 255–269, 2004.
6. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., New York, 1979.
7. A. V. Goldberg and S. Rao, Beyond the flow decomposition barrier, *J. ACM*, 45(5)), pp. 783-797, 1998.
8. J. Kára, J. Kratochvíl, and D. R. Wood, On the complexity of the balanced vertex ordering problem, *Proc. COCOON2005, LNCS 3595*, pp. 849–858, 2005.
9. L. Kowalik, Approximation Scheme for Lowest Outdegree Orientation and Graph Density Measures, *Proc. ISAAC2006, LNCS 4288*, pp. 557–566, 2006.
10. J. K. Lenstra, D. B. Shmoys and Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, 46 (3), 259–271, 1990.
11. J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.
12. M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, 2nd Ed., 2002.
13. A. Schrijver, *Combinatorial Optimization*, Springer, 2003.
14. P., Schuurman and G. J. Woeginger, Polynomial time approximation algorithms for machine scheduling: Ten open problems, *J. Scheduling*, 2, pp. 203–213, 1999.
15. V. Venkateswaran, Minimizing maximum indegree, *Discrete Applied Mathematics*, 143 (1-3), pp. 374–378, 2004.

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