社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－AL－107 (4)
2006／7／3

最大重み付き出次数を最小化するグラフ有向化問題の近似（不）可能性*

朝廣雄一[†] Jesper Jansson[‡] 宮野英次[§] 小野廣隆[‡] 善明紘平[§]

概要

重み付き無向グラフに対して，最大重み付き出次数が最小となるような向き付けを与える問題について，近似可能性・不可能性の観点から調査する．

(In)approximability of Graph Orientation to Minimize the Maximum Weighted Outdegree*

Yuichi Asahiro[†] Jesper Jansson[‡] Eiji Miyano[§] Hirotaka Ono[‡] Kohei Zenmyo[§]

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. It has previously been shown that the unweighted version of the problem is polynomially solvable 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 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 new 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. An *orientation* Λ *of the graph* G is an assignment of 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 $\sum_{\Lambda(\{u,v\})=(u,v)} w(\{u, v\})$.

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,

*This work is partially supported by Grant-in-Aid for Scientific Research on Priority Areas No. 16092223 and No. 14085204, and by Grant-in-Aid for Young Scientists (B) No. 17700022 and No. 18700014.

[†]Department of Social Information Systems, Kyushu Sangyo University, Fukuoka 813-8503, Japan. asahiro@is.kyusan-u.ac.jp

[‡]Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka, Japan. {jj@tcslab., ono}@csce.kyushu-u.ac.jp

[§]Department of Systems Innovation and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan. {miyano@, kouhei@theory.}ces.kyutech.ac.jp

the latter problem has been intensively studied on its polynomial-time (in)approximability, as shown in the next subsection.

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

In this paper, we consider the problem from the viewpoint of (in)approximability. Our results are summarized as follows:

- We present $(2 - 1/k)$-approximation algorithm with the 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 the 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.

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, less diameter, no-cycle and so on [2, 5, 9], 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 the scheduling. For an undirected graph, let us consider the vertices as the machines and the edges as the jobs. Then our orientation problem is regarded as 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 the scheduling, our problem has some restriction, that is, 1) each job can be fulfilled by only 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. Lenstra, Shmoys, and Tardos gave a polynomial time 2-approximation algorithm that is based on the LP-formulation for the general version of $R||C_{max}$ and its 3/2 inapproximability result [10, 14]. Note that the proof of inapproximability uses the assumption that the processing time of each job may vary; it is not applicable to our problem. Namely, our result provides a stronger inapproximability bound to the problem. Our problem is also regarded as a restricted version of scheduling on *identical* unrelated parallel machines, in which each job can be fulfilled by all the machines and the processing time p_{ij} of job j on machine i is identically fixed p_j. This problem has FPTAS [8], which is contrasted with our inapproximability results. Another interesting contrast is about the set of processing times, or the weight set; scheduling on unrelated parallel machines has a polynomial time algorithm for a

weight set $\{p, q\}$ with $q = 2p$, though it is shown that our problem, which is a so-called *restricted variant* of that, is NP-hard even for a weight set $\{p, q\}$ with $q = 2p$.

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. Throughout the paper, let $|V| = n$ and $|E| = m$ for the input graph. 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 $u < v$ 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 $\{(v_{i-1}, v_i) \mid (v_{i-1}, v_i) \in A, i = 1, 2, \ldots, l\}$ 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 a *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 $\Gamma_\Lambda(u) = \{v \mid \{u, v\} \in E \text{ and } \Lambda(\{u, v\}) = (u, v)\}$, 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 2.1 ([1]) For an undirected weighted 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-MMO	
S-MINIMUM MAXIMUM OUTDEGREE	
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 $S = \{1, 2, \ldots, k\}$, which is actually equivalent to $\mathbb{Z}^+$-MMO[1]. We also consider a special case in which the range of w is restricted to a positive integer set $S = \{1, k\}$ 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 also use Δ^* to denote the optimal value.

Here we introduce three basic operations; REVERSE, UP-TO-ROOTS and SOLVE-1-MMO.

[1]This is because we can reduce any weight function to $S = \{1, 2, \ldots, k\}$ just by setting $k = w_{\max}$.

- REVERSE does the following: *Given an orientation Λ of graph G and a directed path $P = \langle u_1, u_2, \ldots, u_l \rangle$ 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 $i = 1, \ldots, l-1$.* Note that the outdegree for each vertex remains the same after the operation if P is a directed cycle. We call this operation REVERSECYCLE if $u_1 = u_l$.

- UP-TO-ROOTS determines an orientation Λ for a given simple undirected graph G of forest, in the following manner: *First fix a root node for each connected component (it is a tree). Then for every edge e, set $\Lambda(e)$ as going up to the root node of its belonging connected component.* 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.

3 Approximation Algorithms

In this section, we present four 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 multiplication 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 $2 - 2/(k+1)$. The last one (in Section 3.5) is also an algorithm for $\{1, k\}$-MMO. It is based on a totally different idea and achieves the approximation ratio of $1 + n/(2k)$, which is useful for instances with $k \gg n$.

In Section 3.3, we show how to improve the running times of the first three approximation algorithms to $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^* + m^2)$ time.

3.1 Majority Voting Algorithm

We present a basic 2-approximation algorithm in this subsection. The idea of the algorithm is as follows: We replace each edge $e = \{u, v\}$ in G with $w(e)$ unweighted edges 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 multiplicated 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[2]. By using these, we decide the orientation Λ of the original G by the following manner: For $e_{u,v} \in E$,

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

Note that in the case of a tie the direction is determined according to a lexicographic order.

We call this algorithm MAJORITY. Although this algorithm itself is just a variation of Lenstra-Shmoys-Tardos algorithm [10], it provides some basic ideas for the algorithms presented in the following subsections.

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 by (1).
4. Return Λ.

[2]This is just for the simplicity of the explanation. Actually, all the arguments in this paper can be extended to graphs with multiple edges.

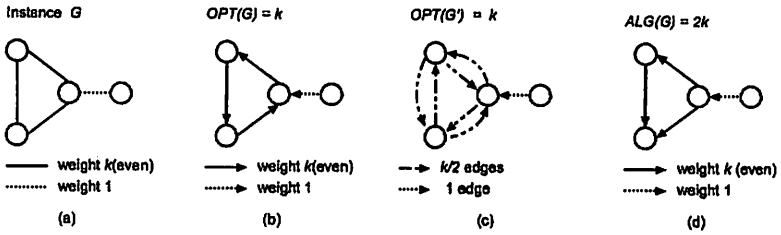

Fig. 1: A worst-case example for MAJORITY: (a) an instance G, (b) an optimal solution for G, (c) an optimal solution Λ' for G', and (d) the output based on (c).

Theorem 3.1 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. □

The analysis is tight, as the example in Figure 1 illustrates.

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 orientation Λ of G by (2) and obtain $G_{\Lambda'} = (V, F_{\Lambda'})$, as described later.
4. If there exists a directed cycle in $G_{\Lambda'}$, apply REVERSECYCLE and go to 3.
5. Do UP-TO-ROOTS for undecided edges in Λ.
6. Return Λ.

In the first and second steps of the algorithm, do as MAJORITY; construct G' (multiplicate each edge) and then find an optimal orientation Λ'. After that we decide the orientation of the original problem by

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

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 a cycle, say $C = \langle v_1, v_2, \ldots, v_l, v_1 (\equiv v_{l+1}) \rangle$, if exists. 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 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. It should be noted 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 3.2 For $S = \{1, \ldots, k\}$, Algorithm CYCLE-CANCELING approximates S-MMO within a factor of $(2 - \frac{1}{k})$ and runs in $O(W^{3/2} \log \Delta^*)$ time. □

Figure 2 shows a worst-case example of CYCLE-CANCELING for an instance of $S = \{1, 3\}$, whose approximation ratio is $5/3 = 2 - 1/3$. It is easy to see that the similar construction produces worst-case examples for general k, which means that the analysis of Theorem 3.2 is tight.

Furthermore, a modified version of this algorithm can obtain a better approximation if the weight set of the graph is $\{1, k\}$.

Alg. REFINED CYCLE-CANCELING

1-4. (Same as CYCLE-CANCELING).
4'. 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.
5. For undecided edges of Λ, apply UP-TO-ROOTS to $\mathcal{F}$.
6. Return Λ.

Theorem 3.3 For any $S = \{1, k\}$, 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. □

Remark: According to Theorem 3.2, 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.

3.3 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 $w(e)$ weight 1 edges, as a subprocedure. 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 network flow technique[3]. 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. Since the relaxed optimal solution may be also fractional in general, we adjust the fractional solution of the problem to an integral solution by using a similar technique as CYCLE-CANCELING; finding a relaxed optimal orientation with forest structure by REVERSECYCLE, and then applying UP-TO-ROOTS operation. Since the solution is an approximated solution of 1-MMO(G') but the objective value is larger than the optimal value by at most 1, we can find an optimal solution of MMO(G') by at most n applications of the REVERSE operation.

[3]Although we omit the detail due to the space limitation, a similar technique is explained in [1]

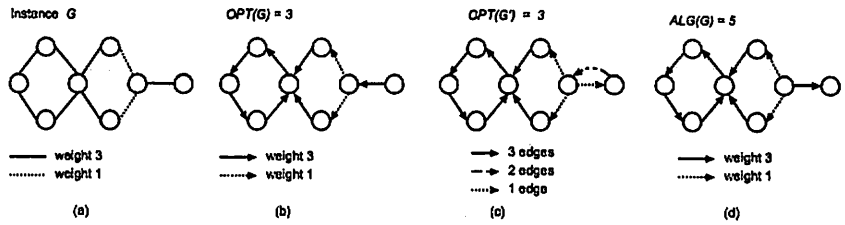

Fig. 2: A-worst case example for CYCLE-CANCELING: (a) an instance G, (b) an optimal orientation for G, (c) an optimal orientation Λ' for G', and (d) the output based on (c).

Alg. MODIFIED SOLVE 1-MMO(G')

1. Find a fractional optimal orientation of G.
2. For $e = \{u, v\}$, if both $f_{u\to v}$ and $f_{v\to u}$ are integral, then decide the values.
3. For edges whose $f_{u\to v}$ (or $f_{v\to u}$) is fractional, apply REVERSECYCLE to eliminate fractional part $f_{u\to v}$ ($f_{v\to u}$) (Goto 2). If no cycle, apply UP-TO-ROOTS. (Refer the orientation as Λ').
4. If there exists a directed path from u to v where $d^+_{\Lambda'}(u) \geq d^+_{\Lambda'}(v) + 2$, apply REVERSE to the path, update Λ' and Goto 4.
5. Return Λ' as $f_{u\to v}$ and $f_{v\to u}$.

Step 1 can be done in $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^*)$ time by a maximum flow algorithm [1, 7]. Steps 2 and 3 require $O(m^2)$ time and Step 4 requires $O(nm)$ time. Thus in total, this computation can be done in $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^* + m^2)$ time.

Theorem 3.4 We can compute the $f_{u\to v}$ and $f_{v\to u}$ values of all the edges for 1-MMO of G' in $O(m^{3/2} \cdot \log m \cdot \log k \cdot \log \Delta^* + m^2)$ time. □

Corollary 3.5 The running times of algorithms MAJORITY, CYCLE-CANCELING and REFINED CYCLE-CANCELING can be improved to $O(m^{3/2} \cdot \log m \cdot \log k \log \Delta^* + m^2)$ time. □

4 Inapproximability Results

In [1], it is shown that S-MMO is NP-hard by a reduction from well-known PARTITION problem, which has a pseudo-polynomial time algorithm. This implies that S-MMO was shown only to be weakly NP-hard. Also 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. The proof is by a reduction from a variation of 3-SAT problem, At-most-3-SAT(2).

At-most-3-SAT(2) 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 predicate. It can be easily proved that At-most-3-SAT(2) is NP-hard by using problem [LO1] on p. 259 of [6].

Given a predicate ϕ of At-most-3-SAT(2) 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 4.) (c) The special gadget is a cycle of k nodes[4] where all the weights of k edges are k. If a clause consists of one (two or three, respectively) variable(s), then it is connected to k

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

(arbitrary $k-1$ or $k-2$, respectively) nodes in the special gadget by edges of weight 1. Hence, the degree of every clause node is exactly $k+1$.

Here, we can show the following lemma (the proof is omitted.)

Lemma 4.1 For the above construction of G_ϕ, the followings hold: (i) If ϕ is satisfiable, $OPT(G_\phi) \leq k$ holds. (ii) If ϕ is not satisfiable, $OPT(G_\phi) \geq k+1$ holds. □

From Lemma 4.1, we immediately obtain the following theorem.

Theorem 4.2 $\{1,k\}$-MMO is strongly NP-hard. Therefore, $\mathbb{Z}^+$-MMO is strongly NP-hard. □

Also the (in)satisfiability gap of Lemma 4.1 yields the following theorem.

Theorem 4.3 $\{1,k\}$-MMO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $1+1/k$, unless P=NP. Namely, $\mathbb{Z}^+$-MMO has no polynomial time algorithm whose approximation ratio is smaller than $3/2$, unless P=NP. □

References

[1] Y. Asahiro, E. Miyano, H. Ono, and K. Zenmyo, "Graph orientation algorithms to minimize the maximum outdegree", *Proceedings of 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] E. Horowitz and S. Sahni, Exact and approximate algorithms for scheduling nonidentical processors, *J. ACM*, 23(2), pp. 317–327, 1976.

[9] 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.

[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.