

Graph Classes and the Complexity of the Graph Orientation Minimizing the Maximum Weighted Outdegree*

Yuichi Asahiro¹

Eiji Miyano^{2, †}

Hiroataka Ono³

¹ Department of Social Information Systems, Kyushu Sangyo University,
2-3-1 Matsukadai, Higashi-ku, Fukuoka 813-8503, Japan.
Email: asahiro@is.kyusan-u.ac.jp

² Department of Systems Innovation and Informatics, Kyushu Institute of Technology,
680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan.
Email: miyano@ces.kyutech.ac.jp

³ Department of Computer Science and Communication Engineering, Kyushu University,
744 Motoooka, Nishi-ku, Fukuoka 819-0395, Japan.
Email: ono@csce.kyushu-u.ac.jp

Abstract

Given an undirected graph with edge weights, we are asked to find an orientation, i.e., an assignment of a direction to each edge, so as to minimize the weighted maximum outdegree in the resulted directed graph. The problem is called MMO, and is a restricted variant of the well-known minimum makespan problem. As previous studies, it is shown that MMO is in \mathcal{P} for trees, weak \mathcal{NP} -hard for planar bipartite graphs, and strong \mathcal{NP} -hard for general graphs. There are still gaps between those graph classes. The objective of this paper is to show tight thresholds of complexity: We show that MMO is (i) in \mathcal{P} for cactuses, (ii) weakly \mathcal{NP} -hard for outerplanar graphs, and also (iii) strongly \mathcal{NP} -hard for P_4 -bipartite graphs. The latter two are minimal superclasses of the former. Also, we show the \mathcal{NP} -hardness for the other related graph classes, diamond-free, house-free, series-parallel, bipartite and planar.

Keywords: graph orientation, min-max optimization, \mathcal{NP} -hardness, cactus, (outer)planar, (P_4 -)bipartite, series-parallel, house-free, diamond-free.

1 Introduction

1.1 Problem and Summary of Results

Let $G = (V, E, w)$ be an undirected and edge weighted graph, where V , E and w denote the set of nodes, the set of edges and a positive integral weight function $w : E \rightarrow \mathbb{Z}^+$, respectively. An *orientation* Λ of the graph G is a set of an assignment of a direction to each edge $\{u, v\} \in E$, i.e., either (u, v) or (v, u) is contained in Λ . The *weighted outdegree* of u is $\sum_{\substack{\{u, v\} \in E: \\ (u, v) \in \Lambda}} w(\{u, v\})$. In this paper, we consider the problem of finding an orientation such that the maximum weighted outdegree is minimum in the resulted

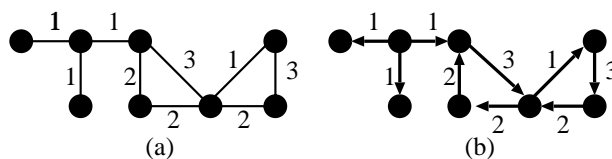


Figure 1: Example of MMO: (a) An edge weighted graph, and (b) an orientation.

directed graph. We call this problem Minimum Maximum Outdegree (MMO). See Fig. 1 for an example of an edge weighted graph and its orientation in which the maximum weighted outdegree is 3 (optimal).

MMO has several applications. For example, such orientations can be used in efficient dynamic data structures for graphs that support fast vertex adjacency queries under a series of edge operations (Brodal & Fagerberg 1999). Also, MMO can be considered a variation of *art gallery problems* (e.g., (Chvátal 1975, O'Rourke 1987)) and the *minimum makespan problem* (e.g., (Lenstra, Shmoys & Tardos 1990)). In particular, we will discuss the minimum makespan problem in the next subsection.

MMO can be solved in polynomial time if all the edge weights are identical (Asahiro, Miyano, Ono, & Zenmyo 2007, Kowalik 2006, Venkateswaran 2004), but it is \mathcal{NP} -hard in general (Asahiro, Miyano, Ono, & Zenmyo 2007, Asahiro, Jansson, Miyano, Ono, & Zenmyo 2007). Even with non-identical weights, the problem can be also solved in polynomial time if the input graph is limited to a tree (Asahiro, Miyano, Ono, & Zenmyo 2007), while for planar bipartite graphs it is still (weakly) \mathcal{NP} -hard.

As many other studies on the computational complexity, it is valuable to consider the frontier between subproblems we know to be solvable in polynomial time and those we know to be \mathcal{NP} -hard. In this paper, we focus on the structure of the input graphs related to the \mathcal{NP} -hardness. Fig. 2 shows the current state of knowledge on the complexity of MMO, including the results in this paper. The figure represents that for example, cactus is a superclass of tree at the bottom. As another example, P_4 -bipartite is a superclass of bipartite and cactus, but bipartite and cactus are not comparable, and so on. All the reductions to show the \mathcal{NP} -hardness are done by simple graphs except outerplanar graphs, which we will explain in a later section. Namely, the weak \mathcal{NP} -hardness of series-parallel graphs is proved with simple graphs,

*This work is partially supported by Grant-in-Aid for Scientific Research on Priority Areas 16092222 and 16092223, and by Grant-in-Aid for Young Scientists (B) 17700022, 18700014 and 18700015.

[†]Currently visiting Dept of Computer Sci and Eng, University of Washington, Seattle, WA 98195-2350, USA.

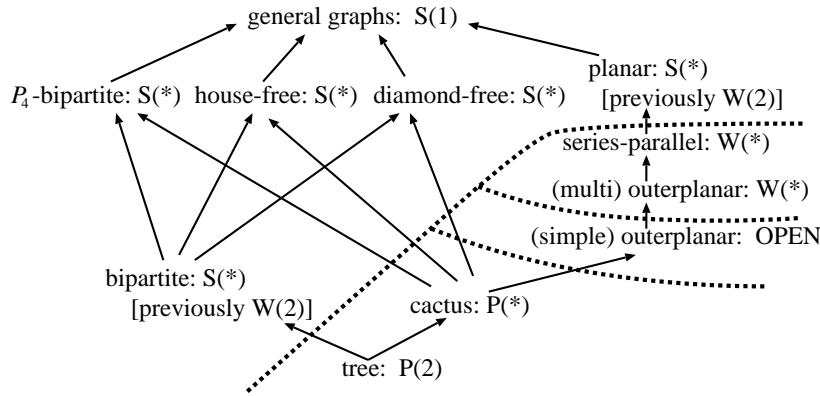


Figure 2: State of knowledge on the complexity of MMO. (W: Weakly \mathcal{NP} -hard, S: Strongly \mathcal{NP} -hard, (1): The results in (Asahiro, Jansson, Miyano, Ono, & Zenmyo 2007), (2): The results in (Asahiro, Miyano, Ono, & Zenmyo 2007), (*): The results in this paper).

but that of outerplanar graphs is proved with multi graphs, and so the complexity for simple outerplanar graphs is still open. Additionally, we propose a pseudo-polynomial time algorithm for series-parallel graphs, which shows the tightness of our weak \mathcal{NP} -hardness result in a sense; MMO for simple outerplanar graphs is either in \mathcal{P} or weakly \mathcal{NP} -hard.

1.2 Related Work

As mentioned before, another aspect of the problem MMO is scheduling; MMO is regarded as a special case of *minimum makespan* or *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} taken to process job $j \in J$ on machine $i \in M$, its goal is to find a job assignment so as to minimize the makespan, i.e., the maximum processing time of any machine. For an undirected graph, let us regard the nodes as the machines and the edges as the jobs. From the viewpoint of scheduling, MMO has the following two restrictions: (i) Each job must be assigned to exactly one of pre-determined two machines, and (ii) the processing time of each job does not depend on the machines.

In (Lenstra, Shmoys & Tardos 1990), a polynomial time 2-approximation algorithm for the general $R||C_{max}$ and its $3/2$ inapproximability are shown. Still there has been gap between these upper and lower bounds; it is one of the well-known open problems (Schuurman & Woeginger 1999). To tackle this kind of situation, it is a natural way to restrict the input as a reasonable subclass: In (Gairing, Lüking, Mavronicolas, & Monien 2004), a polynomial time $2 - 1/k$ -approximation algorithm is proposed, under the assumption that the processing times of jobs are integers and k is the maximum among them. Also, (Asahiro, Jansson, Miyano, Ono, & Zenmyo 2007) considers a further restricted problem in which the processing time of each job is either 1 or k , and then proposes a polynomial time $2 - 2/(k+1)$ -approximation algorithm for $k \geq 3$, and shows that $3/2$ inapproximability still holds for this restricted case even with $k = 2$. In brief summary, the approximation ratios of those algorithms are slightly smaller than two, and the same ($3/2$) lower bound is shown for the restricted case. However, any tight bound between $3/2$ and 2 has not been found for about two decades. The contribution of this paper, from the viewpoint of scheduling, is to make clear what kind of structure of the instances is really difficult to solve.

2 Preliminaries

2.1 Definitions

Let $G = (V, E, w)$ be an edge weighted undirected graph, where V and E are node and edge sets, respectively, and w is a positive integral weight function $w : E \rightarrow \mathbb{Z}^+$. $V(G)$ and $E(G)$ also denote the node set and edge set of the graph G , respectively. We denote the undirected edge whose endpoints are u and v where $u < v$ in lexicographic order by $\{u, v\}$, and denote the directed edge (or arc) from u toward v by (u, v) . An *orientation* Λ of the graph G is a set of an assignment of a direction to each edge $\{u, v\} \in E$, i.e., Λ contains exactly either one of (u, v) and (v, u) . Also $\Lambda(\{u, v\})$ denotes the direction (u, v) or (v, u) of an edge $\{u, v\}$ in Λ .

For a node v , $d_G(v)$ denotes the *degree* of v in G , i.e., $d_G(v) = |\{\{v, u\} \mid \{v, u\} \in E\}|$. The *weighted outdegree* (or, simply *outdegree*) $d_G^+(\Lambda, v)$ of a node v under an orientation Λ of the graph G is defined as the total weight of outgoing arcs of v , i.e.,

$$d_G^+(\Lambda, v) = \sum_{\{u, v\} \in E: (v, u) \in \Lambda} w(\{u, v\}).$$

For simplicity we also use $d(v)$ and $d^+(\Lambda, v)$ instead of $d_G(v)$ and $d_G^+(\Lambda, v)$ if the graph G we are discussing is clear. Then the *cost* of an orientation Λ of a graph G is defined to be $\Delta_\Lambda(G) = \max_{v \in V} \{d_G^+(\Lambda, v)\}$.

A *path* P of length l is denoted by a sequence of nodes such as $P = \langle v_0, v_1, v_2, \dots, v_l \rangle$. Also a *cycle* C of length l is denoted by $C = \langle v_1, v_2, \dots, v_l, v_1 \rangle$. In this paper, a *cycle* always refers a simple cycle, namely, for the cycle C , $v_i \neq v_j$ for any i and j . A node in a cycle is a *gate* if it is adjacent to any node that does not belong to the cycle, so that the degree of the gate is at least three.

A graph is a *cactus* if every edge is part of at most one cycle. The definition of the series-parallel graphs is little bit complicated (p.100 of (Gross & Yellen 2004)):

Definition 1 A series-parallel graph with distinguished terminals l and r is denoted (G, l, r) and is defined recursively as follows:

- The graph consisting of a single edge $\{v_1, v_2\}$ is a series-parallel graph (G, l, r) with $l = v_1$ and $r = v_2$.
- A series operation $(G_1, l_1, r_1) \odot_s (G_2, l_2, r_2)$ forms a series-parallel graph by identifying r_1 with l_2 . The terminals of the new graph are l_1 and r_2 .

- A parallel operation $(G_1, l_1, r_1) \odot_p (G_2, l_2, r_2)$ forms a series-parallel graph by identifying l_1 with l_2 and r_1 with r_2 . The terminals of the new graph are l_1 and r_1 .
- A jackknife operation $(G_1, l_1, r_1) \odot_j (G_2, l_2, r_2)$ forms a series-parallel graph by identifying r_1 with l_2 ; the new terminals are l_1 and r_1 .

The definitions of graph classes except cactus and series-parallel in this paper, s.t., house-free, diamond-free and so on, can be found in (Brandstädt, BangLe, & Spinrad 1987, Gross & Yellen 2004). We would like to note here that tree, bipartite, cactus, house-free, and diamond-free are obviously recognized in polynomial time (See also, e.g., (Kloks, Kratsch, & Müller 2000)). Also, there are efficient recognition algorithms for outerplanar, series-parallel, and planar that run in linear time (Mitchell 1979, Valdes, Tarjan, & Lawler 1982, Hopcroft, & Tarjan 1974). However, it is \mathcal{NP} -hard to recognize P_4 -bipartite graphs (Hoàng, & Le 2001).

2.2 Problem S -MMO and Basic Properties

The problem that we consider in this paper is the minimization of the maximum outdegree of a given undirected graph with edge weights. We formally define our problem as follows.

Problem: S -MINIMUM MAXIMUM OUTDEGREE (S -MMO)

Input: An undirected graph $G = (V, E, w)$, where w is an edge weight function $w : E \rightarrow S$.

Output: An orientation Λ that minimizes $\Delta_\Lambda(G)$.

If we have no restriction on the weight function w (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 $S = \{1, 2, \dots, k\}$.

Let $\Delta^*(G)$ denote the cost of an optimal orientation OPT_G of the graph G , i.e., $\Delta^*(G) = \Delta_{OPT_G}(G)$. We say a graph orientation algorithm is a σ -approximation algorithm if $\Delta_{ALG}(G)/\Delta^*(G) \leq \sigma$ holds for any graph G , where ALG is an orientation obtained by the algorithm for G . Every orientation has the following trivial lower bound caused by the maximum weight w_{\max} of edges (Asahiro, Miyano, Ono, & Zenmyo 2007): For a graph G and any orientation Λ , $\Delta_\Lambda(G) \geq w_{\max}$, so that $\Delta^*(G) \geq w_{\max}$.

The following property of a cactus is very simple but plays a key role to construct the polynomial time algorithm in the next section.

Proposition 2 *In a cactus G in which $d_G(v) \geq 2$ for all $v \in V$, there always exists a cycle with at most one gate.*

Proof: We prove this proposition by contradiction. Suppose that all cycles have at least two gates. Let C be a cycle of length l , $C = \langle v_1, v_2, \dots, v_l, v_1 \rangle$.

Without loss of generality, assume that v_1 is a gate, i.e., there exists a node $x_2 \notin V(C)$ adjacent to v_1 . Since $d(x_2) \geq 2$ by the assumption, there also exists a node x_3 adjacent to x_2 . Similarly, for a node x_i reachable from v_1 , there exists a node x_{i+1} adjacent to x_i . Consider a path P starting from v_1 , $P = \langle v_1, x_2, \dots, x_p \rangle$ for $p \geq 2$. If $x_j = v_h$ for some $2 \leq j \leq p$ and $2 \leq h \leq l$, there exists a cycle $C' = \langle v_1, x_2, \dots, x_j(=v_h), v_{h-1}, \dots, v_2, v_1 \rangle$. The

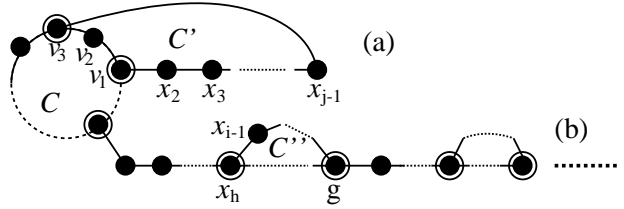


Figure 3: Proof of Proposition 2.

cycles C and C' share the edge $\{v_1, v_2\}$, which contradicts that G is a cactus. (See Fig. 3 (a) in which $h = 3$)

Hence, we assume that such a node x_j does not exist. It turns out to happen $x_i = v_1$ or $x_i = x_h$ for some i and $2 \leq h \leq i-1$, i.e., $C_1'' = \langle v_1, x_2, \dots, x_{i-1}, x_i(=v_1) \rangle$ or $C_2'' = \langle x_h, x_{h+1}, \dots, x_{i-1}, x_i(=x_h) \rangle$, respectively, is a cycle with the gate x_i . Since we assumed that every cycle has at least two gates, there must exist another gate $g \neq v_1$ in C_1'' , or $g \in \{x_{h+1}, \dots, x_{i-1}\}$ in C_2'' , respectively. We can replace C by C_1'' or C_2'' and v_1 by g in the above discussion and then continue. However, since the number of the nodes in G is bounded, eventually a contradiction occurs, namely, the cycle C_1'' or C_2'' has only one gate or G is not a cactus. (See Fig. 3 (b)) \square

3 Polynomial Time Algorithm for Cactuses

In this section, we present a polynomial time algorithm for cactuses. First we introduce a relaxed version (S, T) -MINIMUM MAXIMUM OUTDEGREE ((S, T) -MMO) of the original problem S -MMO and show its several propositions in Sec. 3.1. In Sec. 3.2, we describe an algorithm to solve the decision version (S, T) -MMO(K) of (S, T) -MMO. Finally, the proposed polynomial time algorithm to solve (S, T) -MMO will be given in Sec. 3.3.

3.1 Relaxed Problem (S, T) -MMO

We relax S -MMO to a problem whose input graph has node weights as well as edge weights. Before describing the problem formally, we define some notations analogously to those for edge weighted graphs in Sec. 2.1.

Let $G = (V, E, f, w)$ be a node and edge weighted undirected graph, where V and E are node and edge sets, respectively, and f and w are positive integral weight functions $f : V \rightarrow \mathbb{Z}^+$ and $w : E \rightarrow \mathbb{Z}^+$. The *weighted outdegree* (or, simply *outdegree*) $d_G^+(\Lambda, v)$ of a node v under an orientation Λ of the graph G is modified to the weight of v itself plus the total weight of outgoing arcs of v , i.e.,

$$d_G^+(\Lambda, v) = f(v) + \sum_{\{u, v\} \in E: (v, u) \in \Lambda} w(\{u, v\}).$$

Definitions of the others, e.g., degree, orientation, cost of an orientation, etc., are the same as before. Then the new problem is defined as follows.

Problem: (S, T) -MINIMUM MAXIMUM OUTDEGREE ((S, T) -MMO)

Input: An undirected graph $G = (V, E, f, w)$, where f is a node weight function $f : V \rightarrow T$, and w is an edge weight function $w : E \rightarrow S$.

Output: An orientation Λ that minimizes $\Delta_\Lambda(G)$.

Theorem 3 Consider a node and edge weighted graph $G = (V, E, f, w)$, an edge weighted graph $G^0 = (V, E, w)$, and a constant c . If $f(v) = c$ for all $v \in V$, then $\Delta_\Lambda(G) = \Delta_\Lambda(G^0) + c$ for any orientation Λ . \square

From the above theorem, we obtain the following corollary in a straightforward way, and so \mathcal{NP} -hardness results for S -MMO (by the previous studies and in this paper as well) are directly applied to (S, T) -MMO.

Corollary 4 For a node and edge weighted graph $G = (V, E, f, w)$, (S, T) -MMO is equivalent to S -MMO, if $f(v) = 0$ for all $v \in V$. \square

For simplicity, we denote (S, c) -MMO to represent $(S, \{c\})$ -MMO, that is $f(v) = c$ for all nodes. For a pair of graphs $G = (V, E, f, w)$ and $G' = (V', E', f', w')$, G' is a *subgraph* of G if $V' \subseteq V$, $E' \subseteq E$, and $w'(e) = w(e)$ for all $e \in E'$. A subgraph G' of G is called a *proper subgraph* of G if an additional condition $f'(v) = f(v)$ for all $v \in V'$ is satisfied. Note that in this paper we will only see subgraphs satisfying that $f'(v) \geq f(v)$ for all nodes. Here we extend the definition of the orientation: An orientation Λ of a graph G may contain (u, v) or (v, u) for $\{u, v\} \notin E(G)$. This extension does not affect the value of (out)degrees by definition. When we have to deal with $w(\{u, v\})$ for $\{u, v\} \notin E(G)$, we just consider $w(\{u, v\}) = 0$.

In the following, we state four propositions 5, 6, 7, and 8. These propositions are utilized in order to develop the polynomial time algorithms for cactuses. Proposition 5 shows a relationship between optimal costs for two graphs only node weight functions of which are different. Propositions 6 and 7 are on the optimal costs for proper subgraphs of a graph. Then in Proposition 8, we take a look at the optimal costs for non-proper subgraphs. Since they are not difficult to show, we omit the proofs for these propositions.

Proposition 5 Consider two graphs $G = (V, E, f, w)$ and $G' = (V, E, f', w)$ such that $f(v) \leq f'(v)$ for all $v \in V$. Then $\Delta^*(G) \leq \Delta^*(G')$ holds. \square

Proposition 6 For a graph G , its proper subgraph $G' = G - e$ for $e \in E(G)$, and a pair of orientations Λ of G and Λ' of G' , s.t., $\Lambda' = \Lambda \setminus \{\Lambda(e)\}$, the following three conditions are satisfied:

- (i) $\Delta_\Lambda(G') = \Delta_{\Lambda'}(G')$,
- (ii) $\Delta_\Lambda(G) \geq \Delta_{\Lambda'}(G')$, and
- (iii) $\Delta^*(G) \geq \Delta^*(G')$. \square

Proposition 7 For a graph G , its proper subgraph $G' = G - v$ for $v \in V(G)$, and a pair of orientations Λ of G and Λ' of G' , s.t., $\Lambda' = \Lambda \setminus \{\Lambda(\{v, u\}) \mid \{v, u\} \in E(G)\}$, the following three conditions are satisfied:

- (i) $\Delta_\Lambda(G') = \Delta_{\Lambda'}(G')$,
- (ii) $\Delta_\Lambda(G) \geq \Delta_{\Lambda'}(G')$, and
- (iii) $\Delta^*(G) \geq \Delta^*(G')$. \square

Proposition 8 Consider a graph $G = (V, E, f, w)$ and its edge $e = \{u, v\}$, s.t., $f(u) + w(e) > K$ for a constant K . If $\Delta^*(G) \leq K$, then $(v, u) \in OPT_G$ and also $\Delta^*(G) = \Delta^*(G')$ for the subgraph $G' = (V, E', f', w')$, where $E' = E \setminus \{e\}$, $f'(v) = f(v) + w(e)$, $f'(x) = f(x)$ for all $x \in V' \setminus \{v\}$, and $w'(e) = w(e)$ for all $e \in E'$. \square

3.2 Decision Problem (S, T) -MMO(K)

In this section, we consider a decision version (S, T) -MMO(K) of (S, T) -MMO and present a polynomial time algorithm to solve it, which is the main part of the algorithm to solve $\{1, \dots, k\}$ -MMO for cactuses.

Problem: (S, T) -MMO(K)

Input: An undirected graph $G = (V, E, f, w)$, where f is a node weight function $f : V \rightarrow T$, and w is an edge weight function $w : E \rightarrow S$.

Question: Is there an orientation Λ such that $\Delta_\Lambda(G) \leq K$?

Remind that any orientation has cost at least the maximum edge weight w_{\max} , so it is assumed to be $K \geq w_{\max}$. Again, $(S, 0)$ -MMO(K) is considered as a decision version of S -MMO.

We first introduce three procedures **OutAll**, **FixEdge**, and **OrientCycle**, which are used in the proposed algorithm **AlgCactus**. The first procedure **OutAll**(G, Λ, v) (Fig. 4) determines orientations for all edges connecting to a node v , and then remove v and the edges from the (current) graph G . The second procedure **FixEdge**(G, K, Λ, e) (Fig. 5) determines an orientation for an edge e and then remove e from the (current) graph G , which is based on Proposition 8. The last procedure **OrientCycle**(G, Λ, C) (Fig. 6) determines an orientation for a cycle C having at most one gate. Fig. 7 shows a detailed description of the whole algorithm **AlgCactus**. The correctness and time complexity of the algorithm **AlgCactus** are shown in the two lemmas below in this section.

Fig. 8 depicts an example execution of **AlgCactus** for a graph (Fig. 8(a)) with $K = 3$, where nodes and edges drawn by dotted lines are removed and the numbers in boxes represent node weights greater than 0. First, **OutAll** is applied to the node s (Fig. 8(b)), so that there is no node and edge satisfying the condition (1) or (2) in **AlgCactus**. Next, **OrientCycle** is applied to the cycle C (Fig. 8(c)) in which the node t is the gate, and then the edge $\{t, u\}$ is processed by **FixEdge** (Fig. 8(d)). Eventually, we obtain an final (optimal) orientation by applying, say, **OutAll** to the node t of the graph in Fig. 8(d), and then **FixEdge** to the remaining edge.

Lemma 9 The algorithm **AlgCactus** outputs correct answers for (S, T) -MMO(K).

Proof: Let the final orientation constructed by **AlgCactus** be Λ_f that is constructed regardless of the outputs of **AlgCactus**, 'Yes' or 'No,' for the input graph. Note that orientations for some edges may not be determined in Λ_f when the algorithm outputs 'No.' The algorithm **AlgCactus** determines a part of the orientation Λ_f and constructs a subgraph by removing nodes and edges step by step. We prove that such a constructed subgraph is sufficient to be considered in order to obtain correct answers, i.e., all the nodes removed from the input graph have outdegree at most K under Λ_f , and the optimal cost of such a subgraph is at most that of the input graph.

(Step 1: OutAll) Let two graphs before and after an application of **OutAll** be G_1 and H_1 , respectively. Also Λ_1 denotes the current orientation at the end of Step 1. By definition, **OutAll** does not change the value of $f(u)$ for any node u , and so H_1 is a proper subgraph of G_1 . Therefore, $\Delta^*(G_1) \geq \Delta^*(H_1)$ holds from Proposition 7. Since $\Lambda_f \supseteq \Lambda_1$ and orientations for all the edges connecting to the node v that

Procedure `OutAll`(G, Λ, v)

Input: A graph $G = (V, E, f, w)$, a (partial) orientation Λ , and a node $v \in V$.

Step 1: Add (v, u) to Λ for all $\{u, v\} \in E$.

Step 2: Remove the node v and its connecting edges from G .

Figure 4: Procedure `OutAll`**Procedure** `FixEdge`(G, K, Λ, e)

Input: A graph $G = (V, E, f, w)$, a constant K , a (partial) orientation Λ and an edge $e = \{u, v\} \in E$.

Step 1: If $f(u) + w(e) > K$, then add (v, u) to Λ , and set $f(v) = f(v) + w(e)$. Otherwise, add (u, v) to Λ , and set $f(u) = f(u) + w(e)$.

Step 2: Remove the edge e from G .

Figure 5: Procedure `FixEdge`**Procedure** `OrientCycle`(G, Λ, C)

Input: A graph $G = (V, E, f, w)$, a (partial) orientation Λ , and a cycle $C = \langle v_1, v_2, \dots, v_l, v_1 \rangle$ having at most one gate.

Step 1: If C has no gate, then

(a): Orient the edges of C in one direction along C , i.e., add $(v_1, v_2), (v_2, v_3), \dots, (v_l, v_1)$ to Λ .

Otherwise, i.e., C has exactly one gate, say, v_1 , execute the following:

(b): If $w(\{v_1, v_2\}) < w(\{v_1, v_l\})$, then add $(v_1, v_2), (v_2, v_3), \dots, (v_l, v_1)$ to Λ and set $f(v_1) = f(v_1) + w(\{v_1, v_2\})$.

(c): Otherwise, add $(v_1, v_l), (v_l, v_{l-1}), \dots, (v_2, v_1)$ to Λ and set $f(v_1) = f(v_1) + w(\{v_1, v_l\})$.

Step 2: Remove all the nodes and edges of C except the gate from G .

Figure 6: Procedure `OrientCycle`

is removed by `OutAll` are already determined in Λ_1 , $d_G^+(\Lambda_f, v) = d_G^+(\Lambda_1, v) \leq K$.

We can consider other orientations than Λ_1 in relation to the edges connecting to v , for example, one edge $\{v, x\}$ is oriented inward as (x, v) . By such an orientation, we obtain a graph H'_1 in which $f(x)$ is equal to $f(x)$ in G_1 plus $w(\{x, v\})$, although $f(x)$ in H_1 equals to that in G_1 . The difference between H_1 and H'_1 is only the weight $f(x)$, and from Proposition 5, $\Delta^*(H'_1) \geq \Delta^*(H_1)$ holds. Hence, if $\Delta^*(G_1) \leq K$, then $\Delta^*(H_1) \leq K$ also holds. (It may hold that $\Delta^*(H'_1) > \Delta^*(G_1)$) Therefore, it is suffi-

Algorithm `AlgCactus`(G, K)

Input: A cactus $G = (V, E, f, w)$ and a constant K .

Output: Yes (and an orientation Λ), or No .

Step 0: Set $\Lambda := \emptyset$, and $G' := G$.

Step 1: If there exists a node $u \in V(G')$, s.t.,

$$f(u) + \sum_{\{u,v\} \in E(G')} w(\{u,v\}) \leq K, \quad (1)$$

then execute `OutAll`(G', Λ, u).

Step 2: If there exists an edge $e = \{u, v\} \in E(G')$, s.t.,

$$f(u) + w(e) > K, \quad (2)$$

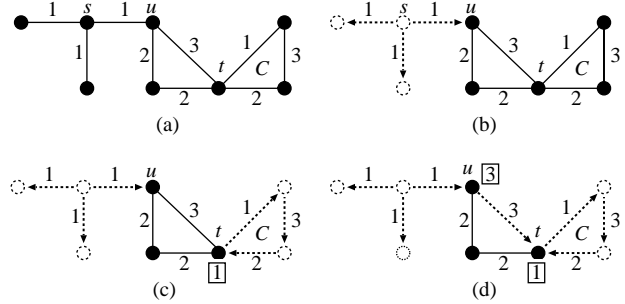
then execute `FixEdge`(G', K, Λ, e).

Step 3: Repeat Steps 1 and 2, until there is neither a node nor an edge satisfying the conditions (1) or (2).

Step 4: If $f(v) > K$ for some node $v \in V(G')$, then output 'No' and halt.

Step 5: Remove isolated nodes (if exist) from G' . If G' is empty, output 'Yes' (and Λ) and halt.

Step 6: Find a cycle C having at most one gate. Execute `OrientCycle`(G', Λ, C), and then return to Step 1.

Figure 7: Algorithm `AlgCactus`Figure 8: Example execution of `AlgCactus`: (a) Input graph, (b) application of `OutAll` to the node s , (c) application of `OrientCycle` to the cycle C , and (d) application of `FixEdge` to the edge $\{t, u\}$.

cient to consider only H_1 , to solve (S, T) -MMO(K).

(**Step 2: FixEdge**) Let the two graphs at the beginning and the end of Step 2 be $G_2 (= H_1$ above) and H_2 , respectively. From Proposition 8, we can see that if $\Delta^*(G_2) \leq K$, then $\Delta^*(G_2) = \Delta^*(H_2)$. Note that no node is removed at Step 2 of `AlgCactus`. Therefore, again it is sufficient to consider only H_2 , to solve (S, T) -MMO(K).

(**Step 3: Repeating Steps 1 and 2**) By repeating Steps 1 and 2, we finally obtain a graph H , which is a subgraph of the input graph G . From the above discussions on Steps 1 and 2, we observe that $\Delta^*(G) \geq \Delta^*(H)$. Therefore, since all the nodes al-

ready removed have outdegree at most K under the current orientation and also under the final orientation Λ_f , what we need to do is to consider the optimal cost for H to solve (S, T) -MMO(K).

(Steps 4 and 5: Halting criteria) If there exists a node v in H having $f(v) > K$, then it is apparent that $\Delta^*(H) > K$ and so $\Delta^*(G) > K$. Therefore we answer 'No.' The rest of the case is that every node v in H has $f(v) \leq K$. Even if an isolated node is removed, the (current) orientation Λ is not modified at all, and the outdegree of the removed node does not change under Λ_f . If the graph is turned to be empty after removing all the isolated nodes, its optimal cost is trivially zero. Namely, $\Delta^*(H) = \max_{v \in V(H)} \{f(v)\} \leq K$. Also since all orientations for all edges have already determined in Λ , Λ is the final orientation Λ_f . In addition to that the removed nodes at Steps 1 have outdegree at most K under $\Lambda_f (= \Lambda)$ as mentioned above. Therefore we can conclude that the answer is 'Yes.'

(Step 6: OrientCycle) G_6 and H_6 denote the two graphs at the beginning and the end of Step 6, respectively. All the nodes have degree at least 2 in G_6 , because, otherwise a contradiction occurs: All isolated nodes, that is, the nodes having degree 0 are removed in Step. 5. Suppose that there exists a node u in G_6 such that $d_{G_6}(u) = 1$, and let the edge connecting to u be e . Since the degree of u is one, $f(u) + \sum_{\{u,v\} \in E(G_6)} w(\{u,v\}) = f(u) + w(e)$ holds, which means that either of the conditions (1) and (2) is always satisfied. However, this contradicts that the fact that G_6 is obtained after repeatedly applying Steps 1 and 2 until there does not exist such a node (Step 3). From this observation and Proposition 2, there always exists a cycle C having at most one gate. Let a cycle with at most one gate be $C = \langle v_1, \dots, v_l, v_1 \rangle$ ($l \geq 2$).

Case (a): C has no gate. In this case, Step. 1(a) of **OrientCycle** is applied to the cycle C . Since there is no node in C satisfying the condition (2), every node in C has outdegree at most K under the orientation determined in Step. 1(a) of **OrientCycle** and also under the final orientation Λ_f . Since C has no gate, C is a maximal connected component, so that C and H_6 does not share any nodes and hence $\Delta^*(G_6) \geq \Delta^*(H_6)$. [End of Case(a)]

Case (b): C has exactly one gate. Let the gate be v_1 without loss of generality, and suppose $w(\{v_1, v_2\}) \leq w(\{v_1, v_l\})$ (The discussion for the case $w(\{v_1, v_2\}) > w(\{v_1, v_l\})$ is similar). In this case, Step. 1(b) of **OrientCycle** is applied to C .

Consider a node $v \in \{v_2, \dots, v_l\}$. Since v is not a gate, $d(v) = 2$ holds. Let the two edges connecting to v be $e_1 = \{u, v\}$ and $e_2 = \{t, v\}$. For v and e_1, e_2 , neither conditions (1) nor (2) does not hold. Hence, if the optimal cost is at most K , we cannot orient e_1 and e_2 as (v, u) and (v, t) at the same time by the condition (1) in order to obtain an orientation whose cost is at most K . Also both of $f(v) + w(e_1)$ and $f(v) + w(e_2)$ are at most K by the condition (2). This situation is true for all the nodes in C except the gate v_1 . Therefore, under the orientation Λ_f , the outdegree of every node in C except v_1 is at most K .

There are two other possibilities for the orientation of C in order to construct a final orientation whose cost is at most K : (I) $\Lambda_I \supseteq \{(v_1, v_l), (v_l, v_{l-1}), \dots, (v_2, v_1)\}$, which is in the reverse direction of that by **OrientCycle**, and (II) $\Lambda_{II} \supseteq \{(v_1, v_2), (v_2, v_3), \dots, (v_{i-1}, v_i), (v_{i+1}, v_i), (v_{i+2}, v_{i+1}), \dots, (v_l, v_{l-1}), (v_l, v_1)\}$ for some $i \neq 1$, in which both of the two edges connecting to the gate v_1 in C are oriented outward. By the conditions (1) and (2), another orientation has the cost greater than

K . Let $H_6^{(I)}$ and $H_6^{(II)}$ denote the subgraphs that can be obtained by those orientations Λ_I and Λ_{II} , respectively, i.e., by removing the nodes in C except the gate v_1 , and increasing $f(v_1)$ by $w(\{v_1, v_l\})$ and $w(\{v_1, v_2\}) + w(\{v_1, v_l\})$, respectively. From Proposition 5, $\Delta^*(H_6^{(I)}) \geq \Delta^*(H_6)$ and $\Delta^*(H_6^{(II)}) \geq \Delta^*(H_6)$ hold, since $f(v_1)$ in H_6 is smaller than those in $H_6^{(I)}$ and $H_6^{(II)}$. Therefore, it is sufficient to consider the graph H_6 to solve (S, T) -MMO(K).

[End of Case (b)]

From the above discussions, by Step 6, the removed nodes have outdegree at most K under Λ_f and for the resulted graph H_6 , $\Delta^*(G_6) \geq \Delta^*(H_6)$ holds. In conjunction with the discussions above, the nodes removed so far at Steps 1, 2, 5 and 6 have outdegree at most K under the orientation Λ_f , and also $\Delta^*(G) \geq \Delta^*(H_6)$ holds. Then, in order to solve (S, T) -MMO(K) for the input graph G , what we need to do in the rest is to solve (S, T) -MMO(K) for the graph H_6 by returning to Step 1. \square

The following proposition gives the time complexity of the algorithm **AlgCactus**.

Proposition 10 *AlgCactus runs in $O(|E|^2)$ time.*

Proof: At Steps 1, 2, and 6, orientation of at least one edge is determined. Therefore the total number of processing those steps, and thus Steps 3, 4, and 5 also, are bounded above by $O(|E|)$. Since each step can be done by scanning nodes and edges in $O(|E|)$ time, the total running time is $O(|E|^2)$. \square

Although we omit the proof, the running time of **AlgCactus** can be reduced with a careful preprocessing:

Lemma 11 *The algorithm AlgCactus runs in $O(|E|)$ time with preprocessing done in $O(|V| \log |V|)$ time.* \square

3.3 Polynomial Time Algorithm

In this section, we show that $\{1, \dots, k\}$ -MMO is solvable in polynomial time by proving an upper bound of optimal costs of orientations for cactuses:

Lemma 12 *For any cactus G , $\Delta^*(G) \leq f_{\max} + 2w_{\max}$ for (S, T) -MMO, where f_{\max} and w_{\max} are the maximum weights of nodes and edges, respectively.*

Proof: The proof is constructive. First we apply Steps 0 through 5 of **AlgCactus** except for Step 4 to G with $K = f_{\max} + 2w_{\max}$, by which the removed nodes have outdegree at most $f_{\max} + 2w_{\max}$ under the final orientation, and remaining nodes have outdegree 0 under the current orientation at the end of Step 5. Then we modify Step 6 of **AlgCactus** as follows and apply it.

Step 6': Find a cycle $C = \langle v_1, v_2, \dots, v_l, v_1 \rangle$ having at most one gate.

- If C does not have a gate, add $(v_1, v_2), (v_2, v_3), \dots, (v_l, v_1)$ to Λ .
- Otherwise, i.e., C has exactly one gate, say, v_l . Add (v_1, v_l) and $(v_1, v_2), (v_2, v_3), \dots, (v_{l-1}, v_l)$ to Λ .

Then remove C except the gate and return to Step 1.

By this modified Step 6', we observe that

- Every remaining node v has outdegree $f(v)$ under the current orientation at the end of Step 6'.
- If C has the gate v_l , v_1 has outdegree at most $f(v_1) + 2w_{\max}$ under the final orientation.
- The nodes removed at Step 6' except v_1 and the gate v_l have outdegree at most $f_{\max} + w_{\max}$ under the final orientation.

Repeating the procedures, all the nodes are removed from the graph at last, and all the removed nodes have outdegree at most $f_{\max} + 2w_{\max}$ under the final orientation. Therefore, $\Delta^*(G) \leq f_{\max} + 2w_{\max}$ holds. \square

Remind that we assume that node and edge weight functions f and w are integral functions in this paper. Therefore, we can obtain optimal orientations for (S, T) -MMO by solving $O(\log(f_{\max} + w_{\max}))$ times the (S, T) -MMO(K) in a binary search manner on K for $w_{\max} \leq K \leq f_{\max} + 2w_{\max}$ from the above lemma. Based on the Lemma 11, (S, T) -MMO is solvable in polynomial time $O(|V| \log |V| + |E| \log(f_{\max} + w_{\max}))$ for cactuses (Note that the preprocessing for **AlgCactus** has to be done only once). In a straightforward way, we obtain the following theorem for $\{1, \dots, k\}$ -MMO ($\equiv (\{1, \dots, k\}, 0)$ -MMO):

Theorem 13 $\{1, \dots, k\}$ -MMO is solvable in polynomial time $O(|V| \log |V| + |E| \log k)$ for cactuses. \square

4 Pseudo-polynomial Time Algorithm for Series-Parallel Graphs

In this section, we describe the main idea of a pseudo-polynomial time algorithm solving $\{1, \dots, k\}$ -MMO for series-parallel graphs. The algorithm is a dynamic programming-based one, which utilizes a decomposition tree (Valdes, Tarjan, & Lawler 1982) defined by the series, parallel and jackknife operations. It is known that determining whether a given graph $G = (V, E)$ is a series-parallel graph can be done in linear time (Wimer & Hedetniemi 1988, Borie, Parker & Tovey 2002). Moreover, we can also obtain a decomposition tree T of G in linear time if G is a series-parallel graph.

For an arbitrary series-parallel graph (G, l, r) , where l and r are left and right terminals, respectively, and two values $w_l \in \{0, 1, \dots, w_G(l)\} \stackrel{\text{def}}{=} \sum_{\{l, u\} \in E(G)} w(\{l, u\})$ and $w_r \in \{0, 1, \dots, w_G(r)\} \stackrel{\text{def}}{=} \sum_{\{r, u\} \in E(G)} w(\{r, u\})$, we define

$$WSP(G, l, r, w_l, w_r) = \min_{\Lambda} \max_{v \in V(G)} \left\{ d_G^+(\Lambda, v) \mid \begin{array}{l} d_G^+(\Lambda, l) = w_l, \\ d_G^+(\Lambda, r) = w_r \end{array} \right\},$$

where Λ is an orientation for G .

In a decomposition tree, let us assume that a (sub)tree T_a is composed of its subtrees T_b and T_c by an operation series, parallel, or jackknife, where T_a, T_b and T_c correspond to (G_a, l_a, r_a) , (G_b, l_b, r_b) and (G_c, l_c, r_c) , respectively. Roughly speaking, for series, parallel, and jackknife operations, the following equations (3), (4), and (5) hold, respectively:

$$WSP(G_a, l_a, r_a, w_l, w_r) = \min_{w_b, w_c} \max \left\{ \begin{array}{l} WSP(G_b, l_b, r_b, w_l, w_b), \\ WSP(G_c, l_c, r_c, w_c, w_r), \end{array} \right\} \quad (3)$$

Algorithm AlgSP(G)

Input: A series-parallel graph $G = (V, E, w)$.

Output: $\Delta^*(G)$.

Step 0: Construct a decomposition tree T for G , and let l and r be two terminals of G .

Step 1: For all $w_l = 0, 1, \dots, w_G(l)$ and $w_r = 0, 1, \dots, w_G(r)$, compute $WSP(G, l, r, w_l, w_r)$ in a recursive manner by equations (3), (4) and (5).

Step 2: Output $\min_{w_l, w_r} WSP(G, l, r, w_l, w_r)$.

Figure 9: Algorithm AlgSP

$$WSP(G_a, l_a, r_a, w_l, w_r) = \min_{\substack{w_{bl} + w_{cl} = w_l, \\ w_{br} + w_{cr} = w_r}} \max \left\{ \begin{array}{l} WSP(G_b, l_b, r_b, w_{bl}, w_{br}), \\ WSP(G_c, l_c, r_c, w_{cl}, w_{cr}), \end{array} \right\} \quad (4)$$

$$WSP(G_a, l_a, r_a, w_l, w_r) = \min_{\substack{w_{br} + w_{cl} = w_r, \\ w_{cr} = w_{br} + w_{cl}}} \max \left\{ \begin{array}{l} WSP(G_b, l_b, r_b, w_l, w_{br}), \\ WSP(G_c, l_c, r_c, w_{cl}, w_{cr}), \end{array} \right\} \quad (5)$$

The above equations (3), (4) and (5) show a principle of optimality, which yields an algorithm based on the dynamic programming. Fig. 9 shows the algorithm. Now we discuss the time complexity of AlgSP. As mentioned above, Step 0 is done in $O(|E|)$ time. In Step 1, we keep $w_G(l) \times w_G(r)$ WSP values for each (G, l, r) , and if we have all WSP values for its two children, the evaluation of equations (3), (4) and (5) can be done in $w_{G_b}(r_b) \times w_{G_c}(l_c)$, $w_{G_a}(l_a) \times w_{G_a}(r_a)$ and $w_{G_a}(r_a) \times w_{G_c}(r_c)$ time, respectively. All of these are bounded by $k^2|V|^2$. The number of recursions is at most $|E|$, so this step is done in $O(|E|k^2|V|^2)$. Step 2 can be done also in $O(k^2|V|^2)$ time. Therefore the total running time of AlgSP is $O(k^2|E||V|^2)$, which is pseudo-polynomial for the input size. More details will appear in journal version.

Theorem 14 $\{1, \dots, k\}$ -MMO is solvable in pseudo-polynomial time $O(k^2|E||V|^2)$ for series-parallel graphs. \square

5 \mathcal{NP} -hardness

In this section, we show the \mathcal{NP} -hardness of $\{1, \dots, k\}$ -MMO for restricted graph classes: outerplanar, series-parallel, planar, bipartite, P_4 -bipartite, diamond-free, and house-free. We again note that outerplanar, P_4 -bipartite, diamond-free, and house-free are minimal superclasses of cactus (Brandstädt, BangLe, & Spinrad 1987). The following theorem shows the weak \mathcal{NP} -hardness of $\{1, \dots, k\}$ -MMO for (multi) outerplanar graphs, but its proof is quite easy.

Theorem 15 $\{1, \dots, k\}$ -MMO is weakly \mathcal{NP} -hard for (multi) outerplanar graphs.

Proof: The proof is by a polynomial time reduction from the weakly \mathcal{NP} -hard problem PARTITION

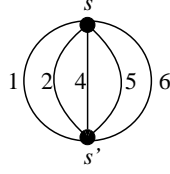


Figure 10: Proof of Theorem 15.

([SP12] on p.223 of (Garey & Johnson 1979)): Given a set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers, determine if there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = \sum_{s_i \in S \setminus S'} s_i$.

We construct an edge weighted graph $G = (V, E, w)$ from an instance of PARTITION. Let the instance of PARTITION be $S = \{s_1, s_2, \dots, s_n\}$. The node set V consists of two nodes, $V = \{s, s'\}$. The edge set E contains n multiple edges e_1, e_2, \dots, e_n connecting between the nodes s and s' , where the weight of each edge e_i is equal to s_i , i.e., $w(e_i) = s_i$. The graph G is clearly outerplanar. Let us define $W = \sum_{s_i \in S} s_i/2$. This reduction is obviously done in polynomial time. See Fig. 10 for an example of the case $S = \{1, 2, 4, 5, 6\}$.

We consider that the situation $s_i \in S'$ (or $s_i \notin S'$) corresponds to orient the edge e_i from s to s' (or s' to s) in G . If there is a set $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = \sum_{s_i \in S \setminus S'} s_i = W$, then both of the outdegrees of s and s' in G is equal to W under the corresponding orientation, which is an optimal orientation. Otherwise, either of them has outdegree greater than W . \square

The \mathcal{NP} -hardness of $\{1, \dots, k\}$ -MMO for series-parallel graphs is again proved by a reduction from PARTITION. Since the constructed graph in the above proof is also a series-parallel graph, the \mathcal{NP} -hardness for series-parallel graphs also holds straightforward. However, the constructed graph in the above proof is a multigraph, and thus, the \mathcal{NP} -hardness has been proved only for multigraphs. The objective of the following theorem is to show the \mathcal{NP} -hardness for simple graphs; however it is not applicable to outerplanar graphs.

Theorem 16 $\{1, \dots, k\}$ -MMO is weakly \mathcal{NP} -hard for series-parallel graphs.

Proof: From an instance $S = \{s_1, s_2, \dots, s_n\}$ of PARTITION, we construct an edge weighted graph $G = (V, E, w)$. The node set V is divided into two types: (i) Subset nodes s and s' , and (ii) Item nodes v_i and v'_i for each s_i . The total number of nodes is $2n + 2$. Let us define $W = \sum_{s_i \in S} s_i/2$. The edge set E contains $3n$ edges, $\{s, v_i\}$'s, $\{v_i, v'_i\}$'s and $\{v'_i, s'\}$'s for $1 \leq i \leq n$. As for the weights of the edges, $w(\{s, v_i\}) = w(\{v'_i, s'\}) = s_i$, and $w(\{v_i, v'_i\}) = W$ for $1 \leq i \leq n$. This reduction is done in polynomial time. See Fig. 11 for an example of the case $S = \{1, 2, 4, 5, 6\}$ again.

We prove that there is an orientation Λ of G such that $\Delta_\Lambda(G) \leq W$ if and only if there exists a set $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = W$ in the following.

(If part) Suppose that there exists a subset S' such that $\sum_{s_i \in S'} s_i = W$. Consider the following orientation Λ according to S' : If $s_i \in S'$, then (s, v_i) , (v_i, v'_i) , and (v'_i, s') are in Λ ; otherwise (s', v'_i) , (v'_i, v_i) , and (v_i, s) are in Λ . Under this orientation Λ , $d^+(\Lambda, s) = d^+(\Lambda, s') = W$. Also, if $s_i \in S'$, then $d^+(\Lambda, v_i) = W$ and $d^+(\Lambda, v'_i) = s_i$ hold; otherwise

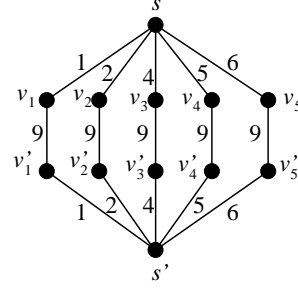


Figure 11: Proof of Theorem 16.

$d^+(\Lambda, v_i) = s_i$ and $d^+(\Lambda, v'_i) = W$ hold. Therefore, since all the nodes have outdegree at most W under Λ , $\Delta_\Lambda(G) \leq W$ holds.

(Only If part) This part is shown by proving that if there exists an orientation Λ of G such that $\Delta_\Lambda(G) \leq W$, there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = W$. Suppose that such an orientation Λ exists.

Since $w(\{v_i, v'_i\}) = W$ for all i 's, we observe that either of v_i and v'_i has outdegree at least W under any orientation. If $(v_i, v'_i) \in \Lambda$, (s, v_i) is also in Λ , because, otherwise $d^+(\Lambda, v_i) > W$ that contradicts the assumption $\Delta_\Lambda(G) \leq W$. Similarly, if $(v'_i, v_i) \in \Lambda$, then $(s', v'_i) \in \Lambda$, too. Let J denote the set of indices i 's such that $(s, v_i) \in \Lambda$. The outdegree of s under Λ is $d^+(\Lambda, s) = \sum_{i \in J} w(\{s, v_i\}) = \sum_{i \in J} s_i$. For an index $j \notin J$, the edge $\{s, v_j\}$ is oriented as $(v_j, s) \in \Lambda$, and thus the edge $\{v_j, v'_j\}$ is oriented as (v'_j, v_j) , because, otherwise the outdegree of v_j is greater than W . Since $(v'_j, v_j) \in \Lambda$, (s', v'_j) is also in Λ . Then, it holds that $d^+(\Lambda, s') \geq \sum_{i \notin J} w(\{s', v'_i\}) = \sum_{i \notin J} s_i = 2W - d^+(\Lambda, s)$. Since $\Delta_\Lambda(G) \leq W$, both of $d^+(\Lambda, s) \leq W$ and $d^+(\Lambda, s') \leq W$ must hold, by which we have $d^+(\Lambda, s) = d^+(\Lambda, s') = W$ and then $\sum_{i \in J} s_i = W$. \square

Now we go to the strong \mathcal{NP} -hardness proofs. We show that $\{1, k\}$ -MMO for bipartite or planar graphs is \mathcal{NP} -hard. Both proofs are based on polynomial time reductions from variants of SAT problem: Given a set $U = \{x_1, \dots, x_n\}$ of Boolean variables and a CNF formula $\phi = \bigwedge_{c_i \in C} c_i$, where C is a set of clauses over U , determine if there exists a truth assignment for ϕ .

Before explaining the reduction, we introduce several variants of SAT. At-Most-3SAT(2L) is a restriction of 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-3SAT(2L) is \mathcal{NP} -complete by using problem [LO1] on p. 259 of (Garey & Johnson 1979).

We call a CNF formula *planar* if graph $G(\phi) = (V, E)$, where $V = U \cup C$ and E contains exactly edges $\{x, c\}$ such that either x or \bar{x} belongs to the clause c for $x \in U$ and $c \in C$. It is known that Planar SAT (or 3SAT), where an input CNF is restricted to be planar, remains \mathcal{NP} -complete (Lichtenstein 1982).

We call a CNF formula *monotone* if each clause contains either only negative literals or only positive literals, and it is known that Monotone SAT, where an input CNF is restricted to be monotone, remains \mathcal{NP} -complete (Gold 1978) (Also see [LO2] on p.259 of (Garey & Johnson 1979)). Monotone At-Most-3SAT(2L) is a restriction of Monotone SAT where each clause includes at most three literals and each literal (not variable) appears at most twice in a for-

mula. We can see that Monotone At-Most-3SAT(2L) is also \mathcal{NP} -complete by the following reduction: Let $c_i = \bigvee_{x_a \in P_i} x_a \vee \bigvee_{x_b \in N_i} \bar{x}_b$ be a clause of an arbitrary At-Most-3SAT(2L) instance ϕ , where P_i (resp., N_i) is the set of positive (resp., negative) literals in c_i . We define a new variable x_{c_i} for each c_i . Then a new monotone formula $\phi' = \bigwedge_{c_i \in C} (x_{c_i} \vee \bigvee_{x_a \in P_i} x_a) \wedge (\bar{x}_{c_i} \vee \bigvee_{x_b \in N_i} \bar{x}_b)$ has a truth assignment if and only if ϕ has a truth assignment. Furthermore, ϕ' is still an instance of At-Most-3SAT(2L) because the numbers of appearances of original literals are same as ϕ and new literals x_{c_i} 's and \bar{x}_{c_i} 's appear exactly once for each. Hence Monotone At-Most-3SAT(2L) is (strongly) \mathcal{NP} -complete.

We first show the strong \mathcal{NP} -hardness of $\{1, k\}$ -MMO for bipartite graphs.

Theorem 17 *For any integer $k \geq 2$, $\{1, k\}$ -MMO is strongly \mathcal{NP} -hard for bipartite graphs.*

Sketch of Proof: We only give a polynomial time reduction from Monotone At-Most-3SAT(2L), and omit the proof of its correctness. Suppose that a formula ϕ of Monotone At-Most-3SAT(2L) with n variables $\{x_1, \dots, x_n\}$ and m clauses $\{c_1, \dots, c_m\}$ is given. We call a clause *positive* (resp., *negative*) if it contains only positive (resp., negative) literals. For ϕ , we construct a graph G_ϕ including two gadgets that mimic (a) literals and (b) clauses, and also (c) a special gadget. (a) Each literal gadget consists of two nodes labeled by x_i and \bar{x}_i and one edge $\{x_i, \bar{x}_i\}$ between them, corresponding to variable x_i of ϕ . The weight of $\{x_i, \bar{x}_i\}$ 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 $c_1 = x \vee y \vee z$ is appeared in ϕ , then node c_1 is connected to nodes x , y and z . (See Fig. 12.) (c) The special gadget is a cycle of $2k$ nodes, say s_1, s_2, \dots, s_{2k} , and $2k$ edges where each edge of the cycle has weight k . If a positive (resp., negative) clause consists of i variable(s), then it is connected to nodes $s_1, s_3, \dots, s_{2(k-i)-1}$ (resp., $s_2, s_4, \dots, s_{2(k-i+1)}$) in the special gadget by edges of weight 1. Hence, the degree of every clause node is exactly $k + 1$. Note that G_ϕ is bipartite, since nodes associated with positive (resp., negative) clauses are connected only to positive (negative) literal nodes or s_i nodes with odd (resp., even) i in the special gadget, and vice versa. Also, this construction can be done in polynomial time.

For this bipartite G_ϕ , we can show that the following holds: (i) If ϕ is satisfiable, $\Delta^*(G_\phi) \leq k$. (ii) If ϕ is not satisfiable, $\Delta^*(G_\phi) \geq k + 1$ (The detailed proof is omitted). \square

By the proof of Theorem 17, we obtain the following corollary.

Corollary 18 *Even for bipartite graphs, $\{1, k\}$ -MMO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $1 + 1/k$ unless $\mathcal{P} = \mathcal{NP}$.* \square

Since neither the graph house nor diamond is bipartite, a bipartite graph is also a house-free and diamond-free graph. Also a bipartite graph is a P_4 -bipartite graph by definition, we obtain the following corollary, too.

Corollary 19 *$\{1, k\}$ -MMO is strongly \mathcal{NP} -hard for P_4 -bipartite, house-free, and diamond-free. Moreover,*

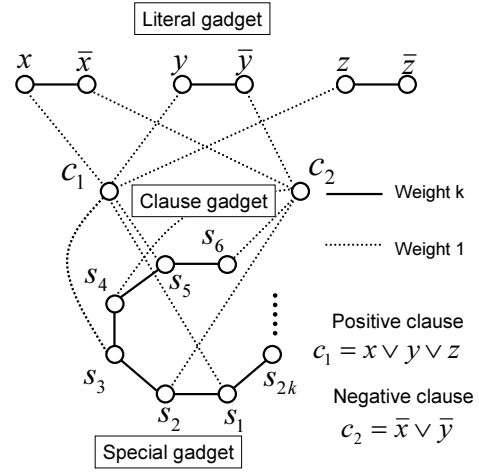


Figure 12: Proof of Theorem 17.

for these graph classes, $\{1, k\}$ -MMO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $1 + 1/k$ unless $\mathcal{P} = \mathcal{NP}$. \square

Next we show the strong \mathcal{NP} -hardness of $\{1, k\}$ -MMO for planar graphs.

Theorem 20 *For any integer $k \geq 2$, $\{1, k\}$ -MMO is strongly \mathcal{NP} -hard for planar graphs.*

Sketch of Proof: We use a reduction similar to that in Theorem 17. Instead of Monotone At-Most-3SAT, we use the reduction from Planar 3SAT. Again, we only show the reduction and proof is omitted.

Suppose Planar 3SAT instance ϕ and its planar drawing are given. For such an instance, we construct a graph G'_ϕ including gadgets associated with (a) variables and (b) clauses, and (c) special gadgets. (a) A variable gadget of x consists of $3l$ nodes and $3l$ edges, where l is the number of appearances of x in ϕ . For convenience, we assume that variable x appears in clauses c_1, c_2, \dots, c_l , and in the given planar drawing c_i 's are drawn in this order (Fig. 13, top). Then we prepare $2l$ nodes labeled by $x^{(i)}$ and $\bar{x}^{(i)}$, and l nodes labeled by $d^{(i)}$. This labeling corresponds to the ordering of c_i 's. For these nodes, we put edges $\{x^{(i)}, \bar{x}^{(i)}\}$ with weight k , $\{\bar{x}^{(i)}, d^{(i)}\}$ with weight 1 and $\{d^{(i)}, x^{(i+1)}\}$ with weight 1, for $i = 1, 2, \dots, l$ ($l + 1 \equiv 1$). Note that a variable gadget itself is planar. (b) Each clause gadget is one node labeled by c_j , corresponding to clause c_j of ϕ (same as the proof of Theorem 17). We connect clause gadgets to nodes of variable gadgets as follows: Again, assume that a variable x appears in clauses c_1, c_2, \dots, c_l . In the variable gadget of x , we prepared $2l$ nodes, $x^{(i)}, \bar{x}^{(i)}$ for $i = 1, \dots, l$, whose numbering corresponds to the index of c_j 's. Then, we connect edges according this numbering; if x (resp., \bar{x}) appears in c_j , then put edge $\{c_j, x^{(j)}\}$ (resp., $\{c_j, \bar{x}^{(j)}\}$) with weight 1 (Fig. 13, bottom). Since ϕ is 3CNF, c_j is connected to at most three nodes in variable gadgets. (c) A special gadget is a cycle of $k + 1$ nodes and $k + 1$ edges where each edge of the cycle has weight k . We prepare a special gadget for each clause gadget and for each node $d^{(i)}$ in a variable gadget. 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 its special gadget by edges of weight 1. For each node $d^{(i)}$, it is connected to $k - 1$ nodes in its own special gadget by

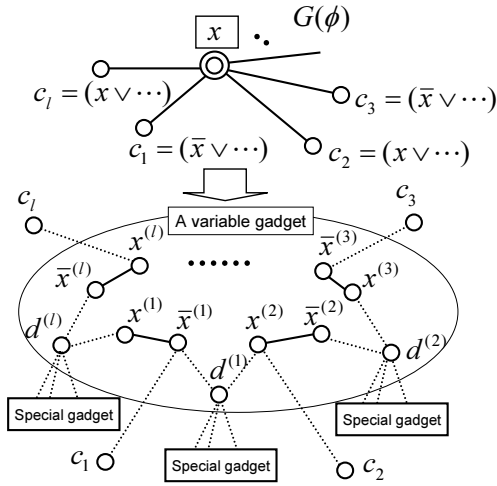


Figure 13: Proof of Theorem 20.

edges of weight 1. Hence, the degree of every clause node or every node $d^{(i)}$ is exactly $k + 1$.

Note that G'_ϕ is planar because we can consider G'_ϕ is obtained by replacing each variable node of planar $G(\phi)$ with the corresponding variable gadget, which does not violate its planarity.

We can say that (i) If ϕ is satisfiable, $\Delta^*(G'_\phi) \leq k$.

(ii) If ϕ is not satisfiable, $\Delta^*(G'_\phi) \geq k + 1$. (The detailed proof is omitted.) \square

By the proof of Theorem 20, again we obtain the following corollary.

Corollary 21 *Even for planar graphs, $\{1, k\}$ -MMO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $1 + 1/k$ unless $\mathcal{P} = \mathcal{NP}$.* \square

6 Conclusion

We have discussed about the complexity of MMO for several graph classes. The results are shown in Figure 2. Except others, we would like to note here about outerplanar graphs. In this paper, we show the weak \mathcal{NP} -hardness for “multi” outerplanar graphs, however the complexity for “simple” outerplanar graphs is still unknown. Since we have developed a pseudo-polynomial time algorithm for series-parallel graphs, the complexity of MMO for “simple” outerplanar graphs is either \mathcal{P} or weakly \mathcal{NP} -hard, which is one of the further research topics.

References

Asahiro, Y., Jansson, J., Miyano, E., Ono, H., & Zenmyo, K. (2007), Approximation algorithms for the graph orientation minimizing the maximum Weighted outdegree in ‘Proc. 3rd International Conference on Algorithmic Aspects in Information and Management, Lecture Notes in Computer Science’, Vol. 4508, pp. 167–177.

Asahiro, Y., Miyano, E., Ono, H., & Zenmyo, K. (2007), ‘Graph orientation algorithms to minimize the maximum outdegree’, *International Journal of Foundations of Computer Science*, **18**(2), pp. 197–215.

Borie, R., Parker, R., & Tovey, C. (2002), ‘Solving problems on recursively constructed graphs’, *Technical Report TR-2002-04, Dept. Comp. Sci., University of Alabama*.

Brandstädt, A., BangLe, V., & Spinrad, J. P. (1987), *Graph Classes: A Survey*, SIAM.

Brodal, G. S., & Fagerberg, R. (1999), Dynamic representations of sparse graphs, in ‘Proc. 6th Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science’, Vol. 1663, pp. 342–351.

Chvátal, V. (1975), ‘A combinatorial theorem in plane geometry’, *J. Combinatorial Theory, series B*, **18**, pp. 39–41.

Gairing, M., Lücking, T., Mavronicolas, M., & Monien, B. (2004), Computing Nash equilibria for scheduling on restricted parallel links, in Proc. 36th ACM Symposium on Theory of Computing, pp. 613–622.

Garey, M., & Johnson, D. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., 1979.

Gold, E. M. (1978), Complexity of automaton identification from given data, *Information and Control*, **37**(3), pp. 302–320.

Gross, J. L., & Yellen, J.(eds) (2004), *Handbook of Graph Theory*, CRC Press.

Hoàng, C.T., & Le, V.B. ‘ P_4 -free colorings and P_4 -bipartite graphs’, *Discrete Mathematics and Theoretical Computer Science*, **4**, pp. 109–122.

Hopcroft, J.E., & Tarjan, R.E. (1974), ‘Efficient planarity testing’, *J. ACM*, **21**, pp. 549–568.

Kowalik, L. (2006), Approximation scheme for lowest outdegree orientation and graph density measures, in ‘Proc. 17th International Symposium on Algorithms and Computation, Lecture Notes in Computer Science’, Vol.4288, pp. 557–566.

Kloks, T., Kratsch, D., & Müller, H. (2000), ‘Finding and counting small induced subgraphs efficiently,’ *Information Processing Letters*, **74**(3-4), pp.115–121

Lenstra, J. K., Shmoys, D. B., & Tardos, É. (1990), ‘Approximation algorithms for scheduling unrelated parallel machines’, *Mathematical Programming*, **46**(3), 259–271, 1990.

Lichtenstein, D. (1982), ‘Planar formulae and their uses’, *SIAM Journal on Computing*, **11**(2), pp. 329–343.

Mitchell, S.L. (1979), ‘Linear algorithms to recognize outerplanar and maximal outerplanar graphs’, *Information Processing Letters*, **9**, pp. 229–232.

O’Rourke, J. (1987), *Art Gallery Theorems and Algorithms*, Oxford University Press.

Schuurman, P., & Woeginger, G. J. (1999), ‘Polynomial time approximation algorithms for machine scheduling: Ten open problems,’ *J. Scheduling*, **2**, pp. 203–213.

Venkateswaran, V. (2004), Minimizing maximum in-degree, *Discrete Applied Mathematics*, **143**(1-3), pp. 374–378.

Valdes, J., Tarjan, R.E., & Lawler, E.L. (1982), ‘The recognition of series-parallel digraphs’ *SIAM J. Computing*, **11**, pp. 298–313.

Wimer, T.V. & Hedetniemi, S.T. (1988), ‘K-terminal recursive families of graphs’ *Congressus Numerantium*, **63**, pp. 161–176.