九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree

Asahiro, Yuichi
Department of Information Science, Kyushu Sangyo University

Miyano, Eiji
Department of Systems Design and Informatics, Kyushu Institute of Technology

Ono, Hirotaka
Department of Economic Engineering, Kyushu University

https://hdl.handle.net/2324/25472

出版情報：Discrete Applied Mathematics. 159 (7), pp.498-508, 2011-04-06. Elsevier
バージョン：
権利関係：(C) 2010 Elsevier B.V.

KYUSHU UNIVERSITY

Graph Classes and the Complexity of the Graph Orientation Minimizing the Maximum Weighted Outdegree ⋆,⋆⋆

Yuichi Asahiro [a], Eiji Miyano [b], Hirotaka Ono [c]

[a] *Department of Information Science, Kyushu Sangyo University, Fukuoka 813-8503, Japan.*

[b] *Department of Systems Design and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan.*

[c] *Department of Economic Engineering, Kyushu University, Fukuoka 812-8581, Japan.*

Abstract

Given an undirected graph with edge weights, we are asked to find an orientation, that is, 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 tighter thresholds of complexity: We show that MMO is (i) in $\mathcal{P}$ for cactus graphs, (ii) weakly $\mathcal{NP}$-hard for outerplanar graphs, and also (iii) strongly $\mathcal{NP}$-hard for graphs which are both planar and bipartite. This implies the $\mathcal{NP}$-hardness for P_4-bipartite, diamond-free or house-free graphs, each of which is a superclass of cactus. We also show (iv) the $\mathcal{NP}$-hardness for series-parallel graphs and multi-outerplanar graphs, and (v) present a pseudo-polynomial time algorithm for graphs with bounded treewidth.

Key words: graph orientation, min-max optimization, $\mathcal{NP}$-hardness, cactus, (outer)planar, (P_4-)bipartite, series-parallel

⋆ An extended abstract of this article was presented in Proceedings of Fourteenth Computing: The Australasian Theory Symposium (CATS 2008), Wollongong, NSW, Australia, January 22-25, 2008.[2]

⋆⋆This work is partially supported by KAKENHI (No. 16092222, 16092223, 17700022, 18700014, 18700015, 20500017 and 22700019).

Email addresses: asahiro@is.kyusan-u.ac.jp (Yuichi Asahiro),

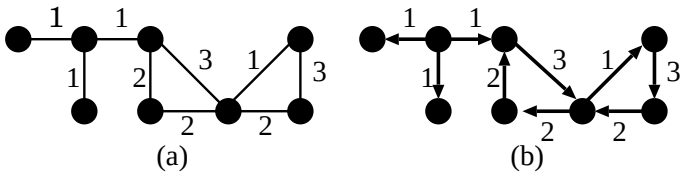

Fig. 1. Example of MMO: (a) An edge weighted graph, and (b) an orientation.

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 $w : E \to \mathbb{Z}^+$, respectively. An *orientation* Λ *of 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 on Λ is $\sum_{(u,v)\in\Lambda} w(\{u, v\})$. In this paper, we consider the problem of finding an orientation such that the maximum weighted outdegree in the resulted directed graph is minimum among all the orientations. 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).

The problem 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 [6]. Also, MMO can be considered a variation of *art gallery problems* (e.g., [7, 18]) and *the minimum makespan problem* (e.g., [15]). In particular, we will discuss the minimum makespan problem in the next subsection.

The problem MMO can be solved in polynomial time if all the edge weights are identical [3, 13, 24], but it is (strongly) $\mathcal{NP}$-hard in general [3, 1]. Even with non-identical weights, the problem can be also solved in polynomial time if the input graph is limited to a tree, while for planar bipartite graphs it is still (weakly) $\mathcal{NP}$-hard [3].

As many other studies on the computational complexity, it is valuable to consider the threshold 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. Figure 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

`miyano@ces.kyutech.ac.jp` (Eiji Miyano), `hirotaka@en.kyushu-u.ac.jp` (Hirotaka Ono).

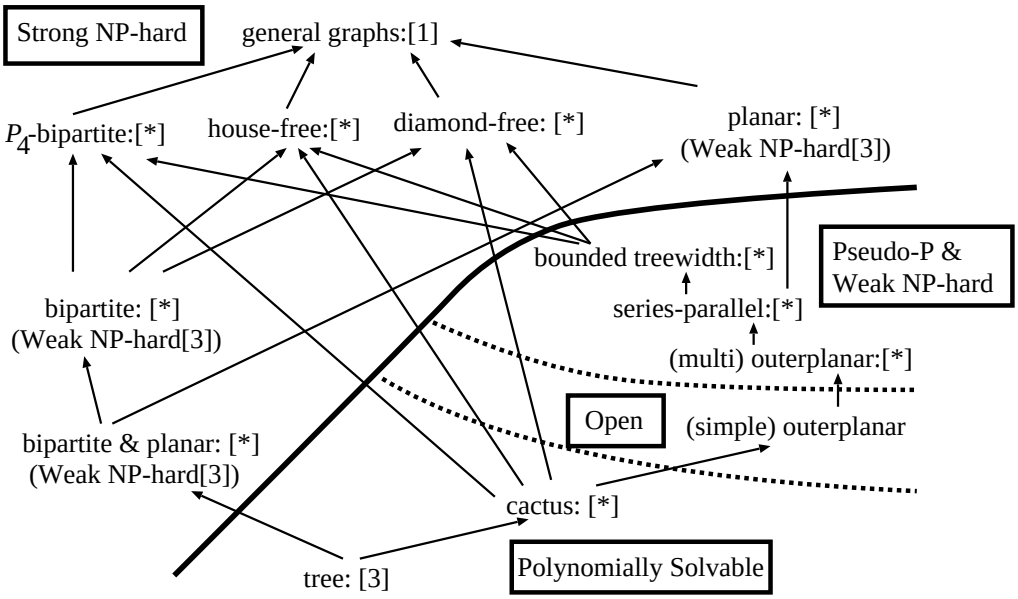

Fig. 2. Graph classes and the complexity of MMO; a number in brackets is a reference number. [*] represents the results in this paper (or its preliminary version [2]).

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, but that of outerplanar graphs is proved with multi graphs, and so the complexity for simple outerplanar graphs is still open.

1.2 Related Work

As mentioned in the previous subsection, 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 [15], a polynomial time 2-approximation algorithm for the general $R||C_{max}$ and its 3/2 inapproximability are shown. Still there has been a gap between

these upper and lower bounds; it is one of the well-known open problems[20]. To tackle this kind of situation, it is a natural way to restrict the input as a reasonable subclass: In [8], 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, [1] 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 which kinds of graphical structures of the instances are really difficult to solve.

2 Preliminaries

2.1 Definitions

Let $V(G)$ and $E(G)$ denote the node and edge sets of graph G, respectively. Also, $N_G(v)$ denotes the set of adjacent nodes to v in G, i.e., $N_G(v) = \{\{v, u\} \mid \{v, u\} \in E(G)\}$. For a node v, $d_G(v)$ denotes the *degree* of v in G, i.e., $d_G(v) = |N_G(v)|$. 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.,

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

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, $P = \langle v_0, v_1, v_2, \ldots, v_\ell\rangle$. Also a *cycle* C of length ℓ is denoted by $C = \langle v_1, v_2, \ldots, v_\ell, v_1\rangle$. In this paper, a *cycle* always refers to a simple cycle, namely, for the cycle C, $v_i \neq v_j$ for any distinct 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 graph* if every edge is a part of at most one cycle. A graph is an *outerplanar graph* if it has a crossing-free embedding in the plane such that all vertices are on the same face. A graph G is a minor of graph H if G can be obtained from H by a series of vertex deletions, edge deletions and/or edge contractions. A graph G is a *series-parallel graph* if K_4 is not

a minor of G. Definitions of other graph classes shown in Figure 2 such as bipartite graphs, house-free graphs and so on, can be found in [5, 10]. In Section 4, we only introduce the class of *graphs with bounded treewidth*, which generalizes series-parallel graphs. We would like to note here that house-free and diamond-free graphs are recognized in polynomial time (See also, e.g., [14]). Also, there are linear time recognition algorithms for tree, bipartite, cactus, outerplanar, series-parallel, and planar graphs [17, 25, 12], though it is $\mathcal{NP}$-hard to recognize P_4-bipartite graphs [11].

2.2 Problem S-MMO

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 $w : E \to S$.
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 $S = \{1, 2, \ldots, k\}$.

Let $\Delta^*(G)$ denote the cost of an optimal orientation OPT_G of the graph G, i.e., $\Delta^*(G) = \Delta_{OPT_G}(G)$. 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 [3]: For a graph G and any orientation Λ, $\Delta_\Lambda(G) \geq w_{\max}$, so that $\Delta^*(G) \geq w_{\max}$.

3 Polynomial Time Algorithm for Cactus graphs

We present a polynomial time algorithm for cactus graphs in this section. In Sec. 3.1, first we introduce a generalized version (S, T)-Minimum Maximum Outdegree ((S, T)-MMO) of the original problem S-MMO, and then show several propositions for (S, T)-MMO. In Sec. 3.2, we describe an algorithm to solve the decision version (S, T)-MMO(K) of (S, T)-MMO. Finally, the proposed algorithm to solve (S, T)-MMO will be given in Sec. 3.3.

3.1 Generalized Problem (S,T)-MMO

We generalize 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 $f : V \to \mathbb{Z}^+$ and $w : E \to \mathbb{Z}^+$. When we explicitly mention the weight functions of a graph G, we may use f_G and w_G instead of f and w. 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.,

$$d^+_G(\Lambda, v) = f(v) + \sum_{(v,u)\in\Lambda} w(\{u, v\}).$$

Definitions of the others, e.g., degree, orientation, and cost of an orientation 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 $f : V \to T$, and w is an edge weight function $w : E \to S$.
Output: An orientation Λ that minimizes $\Delta_\Lambda(G)$.

For (S,T)-MMO, the following proposition obviously holds.

Proposition 1 *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 $v \in V$.* □

For a pair of graphs $G = (V, E, f, w)$ and $G' = (V', E', f', w')$, G' is an *identical edge weight subgraph* (*e-subgraph*) of G if $V' \subseteq V$, $E' \subseteq E$, and $w'(e) = w(e)$ for all $e \in E'$. An e-subgraph G' of G is called an *identical edge and node weight subgraph* (*en-subgraph*) of G if an additional condition $f'(v) = f(v)$ for all $v \in V'$ is satisfied. In this paper we will only see e-subgraphs satisfying that $f'(v) \geq f(v)$ for every node v.

In the following, we state four propositions 2, 3, 4, and 5. These propositions are utilized in order to develop the polynomial time algorithm for cactus graphs. Proposition 2 relates optimal costs for two graphs only node weight functions of which are different. Propositions 3 and optimal4 are on the optimal costs for en-subgraphs of a graph. Then in Proposition 5, we take a look at the optimal costs for a type of subgraph which is neither e-subgraph nor en-subgraph. Since it is not difficult to show the propositions, we omit the proofs for them.

Proposition 2 *Consider a graph $G = (V, E, f, w)$ and its e-subgraph $G' = (V, E, f', w)$ such that $f(v) \leq f'(v)$ for all $v \in V$. Then $\Delta^*(G) \leq \Delta^*(G')$ holds.* $\square$

Proposition 3 *For a graph G, its en-subgraph $G' = G - e$ for $e \in E(G)$, and an orientation Λ of G (and also of G'), the following two conditions are satisfied: (i) $\Delta_\Lambda(G) \geq \Delta_\Lambda(G')$ and (ii) $\Delta^*(G) \geq \Delta^*(G')$.* $\square$

Proposition 4 *For a graph G, its en-subgraph $G' = G - v$ for $v \in V(G)$, and an orientation Λ of G (and also of G'), the following two conditions are satisfied: (i)$\Delta_\Lambda(G) \geq \Delta_\Lambda(G')$ and (ii)$\Delta^*(G) \geq \Delta^*(G')$.* $\square$

Proposition 5 *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 $\Delta^*(G) \geq \Delta^*(G')$ 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 $\{1, \ldots, k\}$-MMO for cactus graphs.

Problem: (S, T)-MMO(K)
Input: An undirected graph $G = (V, E, f, w)$ and a constant K, where f is a node weight function $f : V \to T$, and w is an edge weight function $w : E \to S$.
Question: Is there an orientation Λ such that $\Delta_\Lambda(G) \leq K$?

Note that $(S, \{0\})$-MMO(K) is equivalent to the decision version of S-MMO.

Before describing the algorithm, we note several properties of the cactus graphs. An undirected graph is 2-*connected* if there exist at least two node-disjoint paths between any two non-adjacent nodes.

- In a cactus graph, each 2-connected component is either an edge or a cycle.
- For any graph, its 2-connected components form a tree structure. All 2-connected components and this tree structure can be found in linear time (cf. pp.242–243 of [21]).
- In a cactus graph, if a 2-connected component which corresponds to a leaf (or root) of the tree structure is a cycle, it has exactly one (or no) gate.

Algorithm `AlgCactus`(G, K)
Input: A cactus graph $G = (V, E, f, w)$ and a constant K.
Output: Yes (and an orientation Λ), or No .
Step 0: Set $\Lambda := \emptyset$ and $G' := G$.
Step 1: Obtain all 2-connected components and their tree structure T.
Step 2: Pick a component C corresponding to a leaf l of T, and then process Steps 2-1 and 2-2 if C is a cycle. Otherwise (C is a path), process Step 2-2 only. Let edges incident to a node u in C be e_u and e'_u (resp., e_u) if C is a cycle (resp., a path). Repeat this step until T is empty, or $f_{G'}(p)$ for a node p becomes larger than K.
Step 2-1 (cycle): This step is divided into three subcases.
Step 2-1a: If there exists a non-gate node u in C such that $f_{G'}(u) + w(e_u) + w(e'_u) \leq K$, then orient e_u and e'_u so as to leave from u. Remove u, e_u, and e'_u from C and G'.
Step 2-1b: Else if there exists a node u such that $f_{G'}(u)+w(e_u) > K$, where $e_u = \{u, v\}$ for a node v. Add (v, u) to Λ and then increase $f_{G'}(v)$ by $w(e_u)$. Remove e_u from C and G'.
Step 2-1c: Else let g denote the gate if it exists in C, or arbitrary node, otherwise. If $w(e_g) \geq w(e'_g)$ for edges $e_g = \{g, x\}$ and $e'_g = \{g, y\}$, add (x, g) and (g, y) to Λ, and then increase $f_{G'}(x)$ and $f_{G'}(g)$ by $w(e_g)$ and $w(e'_g)$, respectively. Remove e_g and e'_g from C and G', and then remove g also from C and G' if g is a non-gate node.
Step 2-2 (path): Let u denote a node in C, whose degree is one, and $e_u = \{u, v\}$. If $f_{G'}(u) + w(e_u) \leq K$, then add (u, v) to Λ, otherwise, add (v, u) to Λ and increase $f_{G'}(v)$ by $w(e_u)$. Remove u and e_u from C. Repeat this step until C has only one node. Remove the remaining node from G', and then remove l from T.
Step 3: If Step 2 is terminated by the condition $f_{G'}(p) > K$ for a node p, then output 'No'. Otherwise, output 'Yes' (and Λ).

Fig. 3. Algorithm `AlgCactus`

Figure 3 shows a detailed description of the whole algorithm `AlgCactus`. The correctness and the time complexity of the algorithm `AlgCactus` are shown in the two lemmas below in this section.

Lemma 6 *The algorithm* `AlgCactus` *outputs correct answers.*

PROOF. Let the final orientation constructed by `AlgCactus` be Λ_f regardless of the outputs of `AlgCactus`, 'Yes' or 'No.' Note that orientations for some edges may not be determined in Λ_f when the algorithm outputs 'No.'

In substeps of Step 2, the algorithm `AlgCactus` determines a part of the orientation Λ_f and constructs a subgraph H' ($= G'$ at the end of the step) from the current graph H ($= G'$ at the beginning of the step) by removing nodes and edges step by step. We prove that such a constructed subgraph H' 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 if the algorithm outputs 'Yes,' and $\Delta^*(H') \leq K$ if $\Delta^*(H) \leq K$. Let Λ' denote a partial orientation obtained at the end of a considered substep in the following.

(**Step 2-1a**) Since this step does not change the weight of any node, H' is an en-subgraph of H. Therefore, $\Delta^*(H) \geq \Delta^*(H')$ holds from Proposition 4(ii) and hence $\Delta^*(H') \leq K$ if $\Delta^*(H) \leq K$. Since $\Lambda_f \supseteq \Lambda'$ and orientations for all the edges incident to the removed node u are already determined in Λ', $d_G^+(\Lambda_f, u) = d_G^+(\Lambda', u) \leq K$.

(**Step 2-1b**) From Proposition 5, we can see that if $\Delta^*(H) \leq K$, then $\Delta^*(H) \geq \Delta^*(H')$, as a result, $\Delta^*(H') \leq K$.

(**Step 2-1c**) Without loss of generality, assume that e_g and e'_g are oriented in clockwise direction along the cycle. If g is not a gate, then g is removed and $d_G^+(\Lambda_f, g) = d^+(\Lambda', g) \leq K$ holds, since C does not meet the condition of Step 2-1b. Now we consider the case that g is the gate. Let H'' denote the connected component in H', which includes g. Since clockwise orientation for e_g and e'_g makes the node weight of g smaller than that under counterclockwise orientation, $\Delta^*(H'') \geq \Delta^*(H')$ is satisfied from Proposition 2. That is, if $\Delta^*(H) \leq K$, then it holds that $\Delta^*(H') \leq K$, because the outdegrees of nodes except g are at most K under clockwise orientation for the whole C. (This entire clockwise orientation for C is actually constructed by Step 2-2.)

(**Step 2-2**) All the nodes removed in this step have outdegree at most K under Λ_f as well. From Propositions 3 and 5, we can see that if $\Delta^*(H) \leq K$, then $\Delta^*(H') \leq \Delta^*(H) \leq K$.

(**Step 3:** Halting criteria) If there exists a node p having $f_{G'}(p) > K$, then it is apparent that $\Delta^*(G') > K$ and so $\Delta^*(G) > K$ since $\Delta^*(G) \geq \Delta^*(G')$ from the above discussions. Otherwise, i.e., if all the nodes are removed in Step 2, then every node has outdegree at most K under Λ_f. Therefore the output of `AlgCactus` is correct. □

Lemma 7 *The running time of algorithm* `AlgCactus` *is* $O(|V|)$.

PROOF. Steps 0, 1, and 3 are done in linear time. We need to maintain the set L of leaves of (current) T for Step 2. The initial construction and

maintenance of L takes linear time in an amortized sense, because every node is added to and removed from L at most once.

In each iteration of Step 2, one component C is picked from L, orientations of its edges are determined, and then C is removed from G', T, and L. As a result of the iteration, another component may be added to L. Each of nodes and edges in C is tested at most constant number of times in every substep of Step 2. Since C is processed at most once, the total number of such tests is linear, i.e., the time needed is also linear in an amortized sense. Since each update for the node weights and Λ can be done in constant time and the total number of such updates is also linear, the total time of Step 2 is linear.

In summary, since the number $|E|$ of edges in a cactus graph is $O(|V|)$, `AlgCactus` runs in linear time $O(|V|)$. $\square$

3.3 Polynomial Time Algorithm

From the result of the previous subsection, we immediately obtain a polynomial time algorithm that solves $\{1, \ldots, k\}$-MMO for cactus graphs; we can find the optimal K by using `AlgCactus` as an engine of the binary search. The running time is $O(|V|(\log |V| + \log k))$ by Lemma 7, since a trivial upper bound of optimal costs is $k|V|$. In this subsection, we improve the running time by proving an upper bound of optimal costs for cactus graphs:

Lemma 8 *For any cactus graph 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. Consider that we run `AlgCactus` for $K = f_{\max} + 2w_{\max}$ with skipping Steps 2-1b and 2-1c.

First we show by induction that at the beginning of each iteration of Step 2, $f_{G'}(u) = f_G(u)$ holds for every node u in G'. For the first iteration of Step 2, the statement is obviously true. Assume that at the beginning of some iteration of Step 2, the statement is true. Step 2-1a does not increase node weight of any remaining node. During Step 2-2, any node u of degree one always satisfies the condition $f_{G'}(u) + w(e_u) \leq f_G(u) + w(e_u) \leq f_{\max} + w_{\max} < K$. Namely, Step 2-2 does not increase node weight of any remaining node, either.

The remainder of the proof is to show that every node is removed during Step 2, and has outdegree at most K under the final orientation. If the component C is a cycle, the condition of Step 2-1a is always true, i.e., there is a non-gate node u such that $f_{G'}(u)+w(e_u)+w(e'_u) \leq K$, since C has at most one gate and $f_{G'}(u) = f_G(u)$ from the above discussion. Hence, the node removed

in Step 2-1a has outdegree at most K under the final orientation. By this Step 2-1a, C becomes a path and will be processed in Step 2-2. During Step 2-2, any node u of degree one always satisfies the condition $f_{G'}(u) + w(e_u) \leq K$ as discussed in the above. Hence, Step 2-2 removes all the nodes in C, and the removed nodes have outdegree at most K under the final orientation. $\square$

From Lemma 8, the optimal cost of $(\{1, \ldots, k\}, \{0\})$-MMO is an integer between k and $2k$; the number of candidates of optimal K is at most k. Thus, we can obtain an optimal orientation for $(\{1, \ldots, k\}, \{0\})$-MMO by solving $\log k$ times $(\{1, \ldots, k\}, \{0\})$-MMO$(K)$ in a binary search manner.

Theorem 9 $\{1, \ldots, k\}$-MMO *is solvable in* $O(|V| \log k)$ *time for cactus graphs.*

4 Pseudo-polynomial Time Algorithm for Graphs with Bounded Treewidth

In this section, we present a pseudo-polynomial time algorithm for graphs with bounded treewidth, which is a superclass of series-parallel graphs because the treewidth of the series-parallel graphs is at most 2. Thus the algorithm presented in this section can solve $\{1, 2, \ldots, k\}$-MMO for series-parallel graphs, though a faster running time can be achieved by an algorithm specialized to series-parallel graphs presented in [2], which is an earlier version of this paper.

Intuitively, the treewidth is a measure of tree-likeness of a graph. To explain the notion of treewidth, we define a *tree decomposition*, which was introduced by Robertson and Seymour [19]. A tree decomposition of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where T is a tree with vertex set I and $\mathcal{X} = \{X_i \subseteq V \mid i \in I\}$, with the following properties:

(i) $\bigcup_{i \in I} X_i = V$,
(ii) for every $\{v, w\} \in E$, there exists an $i \in I$ satisfying $\{v, w\} \subseteq X_i$,
(iii) for all $i_1, i_2, i_3 \in I$, if i_2 is on the path between i_1 and i_3 in T, then $X_{i_1} \cap X_{i_3} \subseteq X_{i_2}$.

The width of a tree decomposition $(T = (I, F), \{X_i, i \in I\})$ is defined by $\max_{i \in I}\{|X_i| - 1\}$. The treewidth $\tau(G)$ of G is the minimum width over all tree decompositions of G. Given a graph G and a constant t, it can be determined in linear time whether the treewidth of G is at most t, and also a tree decomposition of G with width t can be obtained in linear time [4].

Here we describe an algorithm that solves $\{1, \ldots, k\}$-MMO for graphs with treewidth τ in $O((k|V|)^{\tau+1}|E|)$ time. Given a graph G with treewidth τ, we can construct a tree decomposition T of G with width τ in linear time. We

can assume T is a binary tree, because otherwise we can easily transform it to a binary tree by introducing dummy nodes without violating the properties of the tree decomposition. The size of T is still bounded by $O(|E|)$.

We consider T is rooted at a node r, and for a node i of T let $T[i]$ denote the subtree of T that consists of descendants of i. Also, we define a partition $\{E_i \mid i \in I\}$ of E according to X_i's, which is always possible due to the property (ii). Now we design a dynamic programming algorithm (DP for short) that runs from leaves of T to the root r. Since each subtree of T corresponds to a subgraph of G, we let $G[i]$ be the edge-induced subgraph defined by $T[i]$, that is, the edge set of $G[i]$ is $\bigcup_{j \in V(T[i])} E_j$ and the vertex set of $G[i]$ is the set of endpoints of edges in $\bigcup_{j \in V(T[i])} E_j$. The DP computes some information about $\{1, 2, \ldots, k\}$-MMO of $G[i]$ from that of $G[i_1]$ and $G[i_2]$, where i_1 and i_2 are two children of i in T. Let $\mathbf{A}(G, U) = \bigotimes_{u \in U} \{0, 1, \ldots, \sum_{v \in N_G(u)} w(u, v)\}$.That is, $\mathbf{A}(G, U)$ is a set of vectors with $|U|$ components, and the upper bound $\sum_{v \in N_G(u)} w(u, v)$ on the u-th component in $\mathbf{A}(G, U)$ represents the assignable weights on u in G. Here, we define a function $WBT(G[i], \mathbf{a})$ by

$$
\begin{aligned}
& WBT(G[i], \mathbf{a}) \\
= & \begin{cases} \min_{\Lambda \in \mathcal{L}(G[i], \mathbf{a})} \Delta_\Lambda(G[i]) & \text{if } \mathcal{L}(G[i], \mathbf{a}) \neq \emptyset, \\ +\infty & \text{otherwise,} \end{cases}
\end{aligned}
$$

where $\mathbf{a} \in \mathbf{A}(G[i], X_i)$ and $\mathcal{L}(G[i], \mathbf{a}) = \{\Lambda \mid d^+_{G[i]}(\Lambda, u) = \mathbf{a}_u$ for every $u \in X_i\}$. Intuitively, $WBT(G[i], \mathbf{a})$ represents the optimal cost of minimum maximum outdegree of $G[i]$ under the constraint that the weighted outdegree of every vertex $u \in X_i$ in the resulting orientation is $\mathbf{a}_u$. Thus, $\min_{\mathbf{a} \in \mathbf{A}(G[i], X_i)} WBT(G[i], \mathbf{a})$ is the optimal cost of $\{1, 2, \ldots, k\}$-MMO for $G[i]$, and also $\min_{\mathbf{a} \in \mathbf{A}(G[r], X_r)} WBT(G[r], \mathbf{a})$ is the optimal cost of $\{1, 2, \ldots, k\}$-MMO for G. Here, we show that $WBT(G[i], \mathbf{a})$ can be computed from $WBT(G[i_1], *)$'s and $WBT(G[i_2], *)$'s in polynomial time. For $WBT(G[i_1], \mathbf{b}^{(1)})$ and $WBT(G[i_2], \mathbf{b}^{(2)})$ with some $\mathbf{b}^{(1)} \in \mathbf{A}(G[i_1], X_{i_1})$ and $\mathbf{b}^{(2)} \in \mathbf{A}(G[i_2], X_{i_2})$, the existence of an orientation of $G[i]$ such that the weighted outdegree of every vertex $u \in X_i$ is $\mathbf{a}_u$ and the edge disjoint subgraphs $G[i_1]$ and $G[i_2]$ of $G[i]$ are oriented as solutions of $WBT(G[i_1], \mathbf{b}^{(1)})$ and $WBT(G[i_2], \mathbf{b}^{(2)})$ can be checked by testing all possible orientations of edges in E_i. Since E_i includes a constant number ($O(\tau^2)$) of edges, the total number of such possible orientations is still constant; it can be done in constant time. Let us consider a subgraph $G_i = (X_i, E_i)$ of $G[i]$. If the vector of the assigned weights on X_i is $\mathbf{b}$, what we need to do is just to determine whether there exists an orientation Λ on G_i satisfying $d^+_{G_i}(\Lambda, u) = \mathbf{a}_u - \mathbf{b}_u$ for every $u \in X_i$, which can be done in constant time by checking all the orientations on G_i. Thus, the point is how we get the $\mathbf{b}$ and the orientation cost.

To obtain them, we divide $G[i] \setminus E_i$ into $G[i_1]$ and $G[i_2]$. The vertices in $X_i \setminus (V(G[i_1]) \cup V(G[i_2]))$ newly appear in X_i, and so $\mathbf{b}_u = 0$ for any $u \in X_i \setminus$

$(V(G[i_1]) \cup V(G[i_2]))$. Let $\mathcal{L}'(G_i, \mathbf{a}-\mathbf{b}) = \{\Lambda \mid d^+_{G_i}(\Lambda, u) = (\mathbf{a}-\mathbf{b})_u$ for every $u \in X_i\}$. Since we have $X_i \cap V(G[i_1]) \subseteq X_{i_1}$ and $X_i \cap V(G[i_2]) \subseteq X_{i_2}$ by the decomposition property (iii), all the information about $\mathbf{b}$ can be obtained from $WBT(G[i_1], *)$ and $WBT(G[i_2], *)$. Thus the following equation holds:

$$
\begin{aligned}
& WBT(G[i], \mathbf{a}) \\
= & \begin{cases}
\min_{\substack{\mathbf{b}^{(1)} \in \mathbf{A}(G[i_1], X_{i_1}), \\ \mathbf{b}^{(2)} \in \mathbf{A}(G[i_2], X_{i_2})}} \max \left\{ \begin{array}{l} WBT(G[i_1], \mathbf{b}^{(1)}), \\ WBT(G[i_2], \mathbf{b}^{(2)}), \\ \mathbf{a}_u \text{ for } u \in X_i \end{array} \right\}, & \text{if } \mathcal{L}'(G_i, \mathbf{a}-\mathbf{b}) \neq \emptyset, \\
+\infty & \text{otherwise,}
\end{cases}
\end{aligned}
$$

where

$$
\mathbf{b}_u = \begin{cases}
\mathbf{b^{(1)}}_u + \mathbf{b^{(2)}}_u & \text{if } u \in X_{i_1} \cap X_{i_2}, \\
\mathbf{b^{(1)}}_u & \text{if } u \in (X_i \cap X_{i_1}) \setminus (X_{i_1} \cap X_{i_2}) \\
\mathbf{b^{(2)}}_u & \text{if } u \in (X_i \cap X_{i_2}) \setminus (X_{i_1} \cap X_{i_2}) \\
0 & \text{if } u \in X_i \setminus (X_{i_1} \cup X_{i_2}).
\end{cases}
$$

Therefore, the DP can solve $\{1, 2, \ldots, k\}$-MMO based on a tree decomposition. Since the running time on each node of T is bounded by $|\mathbf{A}(G[i], X_i)| = O((k|V|)^{\tau+1})$, the DP runs in $O((k|V|)^{\tau+1}|E|)$ time in total.

Theorem 10 *$\{1, \ldots, k\}$-MMO is solvable in $O((k|V|)^{\tau+1}|E|)$ time for graphs with treewidth τ.*

5 $\mathcal{NP}$-hardness

In this section, we show the $\mathcal{NP}$-hardness of $\{1, \ldots, k\}$-MMO for restricted graph classes, outerplanar, series-parallel, planar bipartite, P_4-bipartite, diamond-free, and house-free graphs. Note that outerplanar, P_4-bipartite, diamond-free, and house-free graphs are minimal superclasses of cactus graphs in the comprehensive list shown in [5]. The following theorem shows the weak $\mathcal{NP}$-hardness of $\{1, \ldots, k\}$-MMO for (multi) outerplanar graphs, but its proof is quite easy.

Theorem 11 $\{1, \ldots, k\}$-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 ([SP12] on p.223 of [9]): *Given a set $S = \{s_1, s_2, \ldots, s_n\}$ of n positive integers, determine if there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = \sum_{s_i \in S \setminus S'} s_i$.*

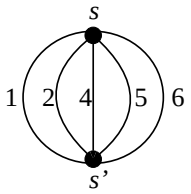

Fig. 4. Proof of Theorem 11.

We construct an edge weighted graph $G = (V, E, w)$ from an instance of PARTITION. Let the instance of PARTITION be $S = \{s_1, s_2, \ldots, s_n\}$. The node set V consists of two nodes, $V = \{s, s'\}$. The edge set E contains n *multiple* edges $e_1, e_2, \ldots, e_n$ 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. 4 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 $\{1, \ldots, k\}$-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 12 $\{1, \ldots, k\}$-MMO *is weakly $\mathcal{NP}$-hard for series-parallel graphs.*

PROOF. From an instance $S = \{s_1, s_2, \ldots, s_n\}$ 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 $1 \leq i \leq n$. It is easy to see that K_4 is not a minor of G, and so G is a series-parallel graph, and moreover, this reduction is done in polynomial time. See Fig. 5 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.

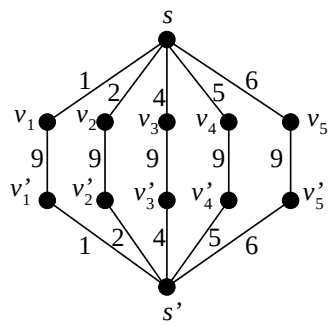

Fig. 5. Proof of Theorem 12.

(If part) Suppose that there exists a subset S' satisfying $\sum_{s_i \in S'} s_i = W$. 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 $d^+(\Lambda, v_i) = s_i$ and $d^+(\Lambda, v_i') = W$ hold. Therefore, since all the nodes have outdegree at most W, $\Delta_\Lambda(G) \leq W$ 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 proof. We show that $\{1, k\}$-MMO for planar bipartite graphs is strongly $\mathcal{NP}$-hard. The proof is based on a polynomial time reduction from a variant of 3SAT problem: *Given a set $U = \{x_1, \ldots, x_n\}$ of Boolean variables and a 3CNF formula $\phi = \bigwedge_{c_i \in C} c_i$, where C is a set of clauses over U and each clause c_i includes at most three literals, determine if there exists a truth assignment for ϕ.* We call a 3CNF formula *planar* if its associated bipartite graph $G(\phi) = (V, E)$ is planar, where $V = U \cup C$ and E contains exactly edges $\{x, c\}$ such that either x or $\overline{x}$ belongs to the clause c for $x \in U$ and $c \in C$. It is known that Planar 3SAT, where an input 3CNF is restricted to be planar, remains strongly $\mathcal{NP}$-complete [16].

Theorem 13 *For any integer $k \geq 2$, $\{1, k\}$-MMO is strongly $\mathcal{NP}$-hard for planar bipartite graphs.*

PROOF. We give a polynomial time reduction from Planar 3SAT. Suppose that a 3CNF formula ϕ of Planar 3SAT with n variables $\{x_1, \ldots, x_n\}$ and m clauses $\{c_1, \ldots, c_m\}$ and its plane embedding are given. For the instance ϕ, our reduction constructs a graph G_ϕ including gadgets associated with (a) variables and (b) clauses, and (c) special gadgets, each of which is explained below.

(a) Let $|x_i|$ be the number of appearances of a variable x_i in ϕ, for each $i = 1, 2, \ldots, n$. Let c_1, c_2, $\ldots$, $c_{|x_i|}$ be the neighboring clauses of x_i, in the counterclockwise order around x_i in the given plane embedding (Fig. 6, top). For a variable x_i of ϕ, our reduction constructs a variable gadget X_i, which consists of $4|x_i|$ nodes and $4|x_i|$ edges, and forms a simple cycle. Then, those $4|x_i|$ nodes are labeled by $x_i^{(1)}$, $a_i^{(1)}$, $\overline{x_i^{(1)}}$, $b_i^{(1)}$, $x_i^{(2)}$, $a_i^{(2)}$, $\overline{x_i^{(2)}}$, $b_i^{(2)}$, $\ldots$, $x_i^{(j)}$, $a_i^{(j)}$, $\overline{x_i^{(j)}}$, $b_i^{(j)}$, $\ldots$, $x_i^{(|x_i|)}$, $a_i^{(|x_i|)}$, $\overline{x_i^{(|x_i|)}}$, $b_i^{(|x_i|)}$ in the counterclockwise order (Fig. 6, bottom). This labeling corresponds to the ordering of c_j's: For example, $x_i^{(1)}$ and $x_i^{(2)}$ are prepared for c_1 and c_2, respectively, and so on. We put edges $\{x_i^{(j)}, a_i^{(j)}\}$ with weight k, $\{a_i^{(j)}, \overline{x_i^{(j)}}\}$ with weight k, $\{\overline{x_i^{(j)}}, b_i^{(j)}\}$ with weight 1 and $\{b_i^{(j)}, x_i^{(j+1)}\}$ with weight 1, for $j = 1, 2, \ldots, |x_i|$ ($|x_i| + 1 \equiv 1$). In Fig. 6, bold solid edges are of weight k and dashed edges are of weight 1. Note that a variable gadget itself is planar.

(b) For a clause c_j of ϕ, our reduction constructs a clause gadget, which is one node labeled by c_j. We connect clause gadgets to nodes of variable gadgets as follows: If x_i (resp., $\overline{x_i}$) appears in c_j, then put edge $\{c_j, x_i^{(j)}\}$ (resp., $\{c_j, \overline{x_i^{(j)}}\}$) with weight 1 (Fig. 6, bottom). Since ϕ is 3CNF, c_j is connected to at most three nodes in variable gadgets, say, X_i, $X_{i'}$, and $X_{i''}$.

(c) Finally, our reduction constructs a special gadget, which forms a cycle of $2k$ nodes, say, s_1, s_2, $\ldots$, s_{2k}, and $2k$ edges where each edge of the cycle has weight k. We prepare m duplicates of the special gadget for m clause gadgets, and $|x_i|n$ duplicates for $|x_i|n$ nodes, $b_i^{(j)}$'s in variable gadgets X_i's for $i = 1, \ldots, n$. If a clause consists of one (two or three, resp.,) variable(s), then its associated clause node is connected to k ($k-1$ or $k-2$, resp.,) nodes of "even" subscripts, s_2, s_4, through s_{2k} (s_{2k-2} or s_{2k-4}, resp.,), in its special gadget by edges of weight 1. For each node $b_i^{(j)}$, it is connected to $k-1$ nodes of "even" subscripts in its own special gadget, s_2, s_4 through s_{2k-2}, by edges of weight 1. Hence, the degree of every clause node or every node $b_i^{(j)}$ is exactly $k+1$. This completes the construction of G_ϕ.

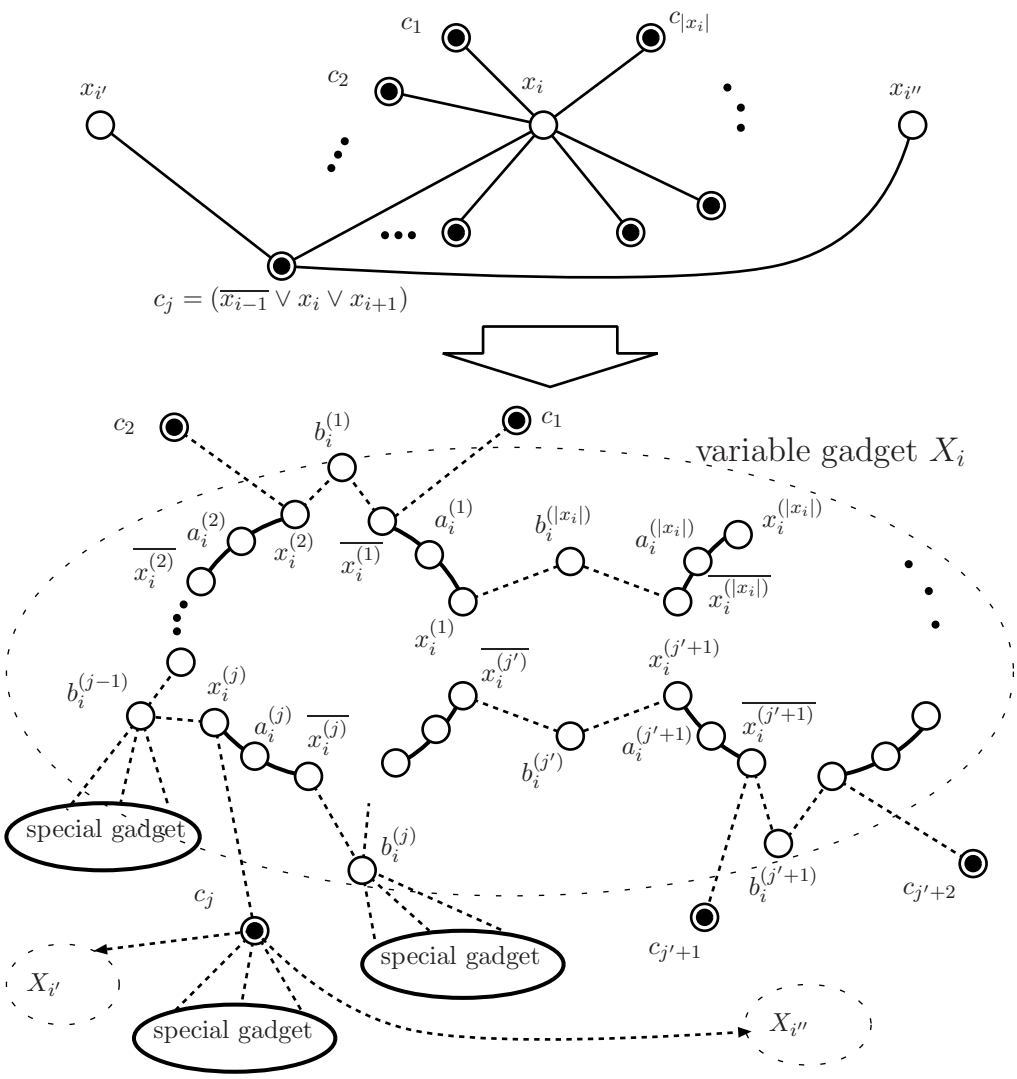

Fig. 6. Proof of Theorem 13.

The above reduction clearly can be performed in time polynomial in the input length of the 3CNF formula ϕ. One can easily verify that our construction of the graph G_ϕ holds the planarity of the instance ϕ, and G_ϕ is bipartite since one can partition the nodes into two independent sets as follows: $\{a_i^{(j)}, b_i^{(j)} \mid i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, |x_i|\} \cup \{c_j \mid j = 1, 2, \cdots, m\} \cup \{s_{2i-1} \mid i = 1, 2, \cdots, k\}$ and $\{x_i^{(j)}, \overline{x_i^{(j)}} \mid i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, |x_i|\} \cup \{s_{2i} \mid i = 1, 2, \cdots, k\}$.

For the planar bipartite G_ϕ, we can show that the following holds: (i) If ϕ is satisfiable, then $\Delta^*(G_\phi) \leq k$. (ii) If ϕ is not satisfiable, then $\Delta^*(G_\phi) \geq k + 1$. First of all, note that, for each special gadget, if we orient the $2k$ edges in one direction along the cycle and edges incident to the gadget inward to it, then the outdegree of every node of the special gadget is exactly k; otherwise, by applying any other orientation the maximum outdegree increases up to at least

$k+1$. This also ensures that all the edges in each variable gadget are oriented in clockwise or counterclockwise direction if the outdegree of the nodes in the variable gadget is at most k. The reason is that if both of two incident edges of some node in variable gadgets are oriented outward, its outdegree turns to be greater than k.

We prove (i). Suppose that there is a satisfying truth assignment for the formula ϕ. From the assignment, we construct an orientation Λ with $\Delta_\Lambda(G_\phi) \leq k$. If $x_i = \mathit{true}$ in the assignment, all the edges in the variable gadget X_i are oriented clockwise, $(\overline{x_i^{(|x_i|)}}, a_i^{(|x_i|)})$, $(a_i^{(|x_i|)}, x_i^{(|x_i|)})$, $(x_i^{(|x_i|)}, b_i^{(|x_i|-1)})$, $(b_i^{(|x_i|-1)}, \overline{x_i^{(|x_i|-1)}})$, $\ldots$, $(\overline{x_i^1}, a_i^{(1)})$, $(a_i^{(1)}, x_i^{(1)})$, $(x_i^{(1)}, b_i^{(|x_i|)})$, $(b_i^{(|x_i|)}, \overline{x_i^{(|x_i|)}})$; otherwise, all the edges in X_i are oriented counterclockwise, $(x_i^{(1)}, a_i^{(1)})$, $(a_i^{(1)}, \overline{x_i^{(1)}})$, $(\overline{x_i^{(1)}}, b_i^{(1)})$, $(b_i^{(1)}, x_i^{(2)})$, $\ldots$, $(b_i^{(|x_i|)}, x_i^{(1)})$. Namely, at this moment, the outdegrees of nodes associated with the literals of *true* and *false* assignments are 1 and k, respectively. We call the nodes associated with literals of *true* (resp., *false*) assignments *true* (resp., *false*) nodes. For example, in Fig. 6, if the variable $x_i = \mathit{false}$ in a truth assignment, then $x_i^{(j)}$'s are called false nodes and $\overline{x_i^{(i)}}$'s are called true ones. In every clause gadget, we arbitrary select one edge connected to a true node and orient it inward to the clause gadget, but the remaining exactly k edges outward to the variable and special gadgets. This orientation of the edges does not increase the outdegree of false nodes or any extra true node; it is still at most k. Since the outdegree of true nodes in the variable gadgets is at most two, the maximum outdegree of G_ϕ is at most k.

Next, we prove (ii) by showing that if the graph G_ϕ has an orientation whose cost is at most k, then ϕ is satisfiable by constructing a satisfying truth assignment. If all the edges in the ith variable gadget X_i are oriented in counterclockwise, i.e., from $x_i^{(j)}$ to $a_i^{(j)}$ and from $a_i^{(j)}$ to $\overline{x_i^{(j)}}$, then we assign $x_i = \mathit{false}$; otherwise, $x_i = \mathit{true}$. Every node in a clause gadget is connected to the variable and special gadgets by $k+1$ edges of weight 1. All the edges connected to the special gadget have to go out of the node in the clause gadget. If not, the maximum outdegree of the special gadget is at least $k+1$. It follows that at least one of the edges connecting the clause gadget to the variable gadgets must enter into the clause node, which surely starts from a literal node assigned *true* in the variable gadgets. This means that the above truth assignment satisfies all the clauses in ϕ. It follows that Planar 3SAT is reduced to $\{1, k\}$-MMO for planar bipartite graphs, which completes the proof. $\square$

Corollary 14 *Even for planar 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 house graph nor diamond graph 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 15 $\{1, k\}$-MMO *is strongly $\mathcal{NP}$-hard for P_4-bipartite, house-free, and diamond-free. Moreover, for these graph classes, $\{1, k\}$-MMO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $1 + 1/k$ unless $\mathcal{P} = \mathcal{NP}$.* $\square$

6 Conclusion

We have discussed about the complexity of MMO for several graph classes. The results are shown in Figure 2. In Section 3, an exact algorithm for cactus graphs is presented, which runs in $O(|V| \log k)$ time. An open question here is that whether an $O(|V|)$-time algorithm can be developed or not. In Section 4, we present an $O((k|V|)^\tau |E|)$ time algorithm for graphs with treewidth τ. This is a pseudo-polynomial time algorithm if τ is a constant, but the pseudo-polynomial solvability itself can be obtained from results of the monadic second order logic [23]. Also, the hardness result is generalized as $W[1]$-hard [22].

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.

Acknowledgements

The authors would like to thank anonymous referees for their helpful comments, which improved both the presentation and the results of the paper.

References

[1] Asahiro, Y., Jansson, J., Miyano, E., Ono, H., and 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,* LNCS 4508, pp. 167–177.

[2] Asahiro, Y., Miyano, E., and Ono, H. [2008], 'Graph Classes and the Complexity of the Graph Orientation Minimizing the Maximum Weighted

Outdegree', in *'Proc. 14th Computing: The Australasian Theory Symposium (CATS 2008)'*, CRPIT, 77, pp. 97–106.

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

[4] Bodlaender, H. L. [1996], 'A linear time algorithm for finding tree-decompositions of small treewidth', *SIAM Journal on Computing*, **25**, pp. 1305–1317.

[5] Brandstädt, A., BangLe, V., and Spinrad, J. P. [1987], *Graph Classes: A Survey*, SIAM.

[6] Brodal, G. S., and Fagerberg, R. [1999], 'Dynamic representations of sparse graphs', in *Proc. 6th Workshop on Algorithms and Data Structures,* LNCS 1663, pp. 342–351.

[7] Chvátal, V. [1975], 'A combinatorial theorem in plane geometry', *J. Combinatorial Theory, series B*, **18**, pp. 39–41.

[8] Gairing, M., Lücking, T., Mavronicolas, M., and Monien, B. [2004], 'Computing Nash equilibria for scheduling on restricted parallel links,' in *Proc. 36th ACM Symposium on Theory of Computing*, pp. 613–622.

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

[10] Gross, J. L., and Yellen, J.(eds) [2004], *Handbook of Graph Theory*, CRC Press.

[11] Hoàng, C.T., and Le, V.B. [2001], 'P_4-free colorings and P_4-bipartite graphs', *Discrete Mathematics and Theoretical Computer Science*, **4**, pp. 109–122.

[12] Hopcroft, J.E., and Tarjan, R.E. [1974], 'Efficient planarity testing', *J. ACM*, **21**, pp. 549–568.

[13] Kowalik, L. [2006], 'Approximation scheme for lowest outdegree orientation and graph density measures', in *Proc. 17th International Symposium on Algorithms and Computation,* LNCS 4288, pp. 557–566.

[14] Kloks, T., Kratsch, D., and Müller, H. [2000], 'Finding and counting small induced subgraphs efficiently,' *Information Processing Letters*, **74**(3-4), pp.115-121

[15] Lenstra, J. K., Shmoys, D. B., and Tardos., É. [1990], 'Approximation algorithms for scheduling unrelated parallel machines', *Mathematical Programming*, **46**(3), 259–271, 1990.

[16] Lichtenstein, D. [1982], 'Planar formulae and their uses', *SIAM Journal on Computing*, **11**(2), pp. 329–343.

[17] Mitchell, S.L. [1979], 'Linear algorithms to recognize outerplanar and maximal outerplanar graphs', *Information Processing Letters*, **9**, pp. 229–232.

[18] O'Rourke, J. [1987], *Art Gallery Theorems and Algorithms*, Oxford University Press.

[19] Robertson, N., and Seymour, P. D. [1984], 'Graph Minors. III. Planar Tree-Width,' *J. Combinatorial Theory. B*, **36**, pp. 49–64.

[20] Schuurman, P., and Woeginger, G. J. [1999], 'Polynomial time approximation algorithms for machine scheduling: Ten open problems,' *J. Scheduling*, **2**, pp. 203–213.

[21] Schrijver, A. [2003], *Combinatorial Optimization*, Springer.

[22] Szeider, S. [2008], 'Not So Easy Problems For Tree Decomposable Graphs', in *Proc. International Conference on Discrete Mathematics*, pp. 161–171.

[23] Szeider, S. [2008], 'Monadic Second Order Logic on Graphs with Local Cardinality Constraints', in *Proc. 33rd International Symposium on Mathematical Foundations of Computer Science,* LNCS 5162, pp. 601 – 612.

[24] Venkateswaran, V. [2004], Minimizing maximum indegree, *Discrete Applied Mathematics*, **143**(1-3), pp. 374–378.

[25] Valdes, J., Tarjan, R.E., and Lawler, E.L. [1982], 'The recognition of series-parallel digraphs' *SIAM J. Computing*, **11**, pp. 298–313.