Vol.2009-AL-124 No.7
2009/5/11

グラフの最小出次数最大化問題

朝 廣　雄 一[†1]　Jesper Jansson [†2]
宮 野　英 次[†3]　小 野　廣 隆[†4]

MaxMinO という無向グラフの向き付け問題の一種について考える．無向枝重みつきグラフの向き付けを与えることにより，重み付き出次数を定義する（枝重みは正整数であるとする）．この問題は，グラフ中の最小重み付き出次数を最大化する問題である．本研究ではまず，MaxMinO は枝重みを $\{1,2\}$ に限定し，かつ各頂点の（重みなし）次数が高々3，さらにグラフが平面二部であるような場合でも強 NP 困難であり，P= NP でない限り，任意の定数 $\varepsilon > 0$ に対して多項式時間では $2-\varepsilon$ 近似不可能であることを示す．次にすべての枝重みが 1 である場合には，多項式時間で解くことができることを示す．これにより，枝重みが w_{min} よりも大きい枝数が $O(\log n)$ であるときにも，多項式時間で解くことができる．さらに，この手法により MaxMinO に対する単純な w_{max}/w_{min} 近似多項式時間アルゴリズムを得ることができる（w_{max}, w_{min} はそれぞれ枝重みの最大値，最小値）．最後に，入力グラフがカクタスである場合にも MaxMinO は多項式時間で解くことができることを示す．

On Graph Orientation to Maximize the Minimum Weighted Outdegree

Yuichi Asahiro ,[†1] Jesper Jansson ,[†2] Eiji Miyano [†3]
and Hirotaka Ono[†4]

We study a new variant of the graph orientation problem called MaxMinO where the input is an undirected, edge-weighted graph and the objective is to assign a direction to each edge so that the minimum weighted outdegree (taken over all vertices in the resulting directed graph) is maximized. All edge weights are assumed to be positive integers. First, we prove that MaxMinO is strongly NP-hard and cannot be approximated within a ratio of $2-\epsilon$ for any constant $\epsilon > 0$ in polynomial time unless P=NP, even if all edge weights belong to $\{1,2\}$, every vertex has degree at most three, and the input graph is bipartite or planar. Next, we show how to solve MaxMinO exactly in polynomial time for the special case in which all edge weights are equal to 1. This technique gives us a simple polynomial-time $\frac{w_{max}}{w_{min}}$-approximation algorithm for MaxMinO where w_{max} and w_{min} denote the maximum and minimum weights among all the input edges. Furthermore, we also observe that this approach yields an exact algorithm for the general case of MaxMinO whose running time is polynomial whenever the number of edges having weight larger than w_{min} is at most logarithmic in the number of vertices. Finally, we show that MaxMinO is solvable in polynomial time if the input is a cactus graph.

1. Introduction

An orientation of an undirected graph is an assignment of a direction to each of its edges. Graph orientation is a well-studied area of graph theory and combinatorial optimization and thus a large variety of objective functions have been considered so far. The objective function of the present paper is the maximization of the minimum outdegree. It is closely related to the classic job scheduling on parallel machines. In the parallel machine scheduling scenario, our problem can be regarded as the restricted assignment variant of the machine covering problem[18)], where its goal is to assign jobs to parallel machines such that each machine is covered as much as possible. In the following, we first define several terminologies and our objective function, then describe related work, and summarize our results.

Problem definition. Let $G = (V, E, w)$ be a given undirected, edge-weighted graph with vertex set V and edge set E whose weights are numbers specified by a function w. An *orientation* Λ of G is defined to be any function on E of the form $\Lambda : \{u, v\} \mapsto \{(u, v), (v, u)\}$, i.e., an assignment of a direction to each undirected edge $\{u, v\}$ in E. Given an orientation Λ of G, the *weighted outdegree* $d_\Lambda(v)$ of a vertex $v \in V$ is defined

†1 九州産業大学情報科学部情報科学科, Department of Information Science, Kyushu Sangyo University, Higashi-ku, Fukuoka 813-8503, Japan. Email: asahiro@is.kyusan-u.ac.jp
†2 お茶の水大学, Ochanomizu University, Bunkyo-ku, Tokyo 112-8610, Japan. Email: Jesper.Jansson@ocha.ac.jp
†3 九州工業大学大学院 情報工学研究院システム創成情報工学系, Department of Systems Design and Informatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan. Email: miyano@ces.kyutech.ac.jp
†4 九州大学大学院 システム情報科学研究院 情報学部門, Department of Informatics, Kyushu University, Nishi-ku, Fukuoka 819-0395, Japan. Email: ono@csce.kyushu-u.ac.jp

as the total weight of all edges leaving v, i.e., $d_\Lambda(v) = \sum_{\substack{\{u,v\}\in E:\\ \Lambda(\{u,v\})=(v,u)}} w(\{u,v\})$, and the *minimum weighted outdegree* $\delta_\Lambda(G)$ is defined by $\delta_\Lambda(G) = \min_{v\in V}\{d_\Lambda(v)\}$.

In this paper we deal with the problem of finding an orientation of the input graph such that the minimum weighted outdegree is maximum. We call this problem Maximum Minimum Weighted Outdegree Graph Orientation Problem (MaxMinO for short): The input is an undirected, edge-weighted graph $G = (V, E, w)$ with $w : E \to \mathbb{Z}^+$, where $\mathbb{Z}^+$ denotes the set of positive integers, and the objective is to find an orientation Λ^* of G which maximizes $\delta_\Lambda(G)$ over all possible orientations Λ of G. Such an orientation is called a *max-min orientation of* G, and the corresponding value $\delta_{\Lambda^*}(G)$ is denoted by $OPT(G)$. The special case of MaxMinO where all edge weights of the input graph are equal to 1 is referred to as *unweighted* MaxMinO.

Throughout the paper, we use the following notations: $n = |V|$, $m = |E|$, and $W = \sum_{e\in E} w(e)$ for the input G. Furthermore, w_{max} and w_{min} denote the maximum and minimum weights, respectively, among all edges in E. For any $v \in V$, the (unoriented) *weighted degree of* v, denoted by $d(v)$, is the sum of all weights of edges incident to v, and $\Delta = \max_{v\in V}\{d(v)\}$ is the maximum (unoriented) weighted degree among all vertices in G. Also for a (fixed) $v \in V$, we call $|\{\{u, v\} \in E\}|$ (i.e., the number of edges incident to v) the (unoriented) *unweighted degree of* v, and denote it by $deg(v)$. We also call $\max_{v\in V} deg(v)$ the (unoriented) *unweighted degree of* G, both of which will be used to focus on the topological structure of the graph.

We say that an algorithm $\mathcal{A}$ is a *σ-approximation algorithm* for MaxMinO or that $\mathcal{A}$'s *approximation ratio* is at most σ, if $OPT(G) \le \sigma\cdot\mathcal{A}(G)$ holds for any input graph G, where $\mathcal{A}(G)$ is the minimum weighted outdegree in the orientation returned by $\mathcal{A}$ on input G.

Related work. MaxMinO studied in the current work is closely related to the restricted assignment variant of the machine covering problem, which is often called the Santa Claus problem[4),5),8),12)]: Santa Claus has m gifts (corresponding to jobs, and to edges in MaxMinO) that he wants to distribute among n kids (corresponding to machines, and to vertices in MaxMinO). Some gift may be worth \$100 but another may be not so expensive, and some kids do not want some of the gifts whatsoever (i.e., its value is 0 for the kids). The goal of Santa Claus is to distribute the gifts in such a way that the least lucky kid is as happy as possible. In addition, MaxMinO has the following restriction (which might be strong and somehow strange in the Santa Claus scenario): Every gift is of great value only to exactly two kids and thus it must be delivered to one of them. For the Santa Claus problem, Golovin[12)] provided an $O(\sqrt{n})$-approximation algorithm for the restricted case where the value of each gift belongs to $\{1, k\}$ for some integer k. Bansal and Sviridenko[4)] considered the general value case and showed that a certain linear programming relaxation can be used to design an $O(\log\log m/\log\log\log m)$-approximation algorithm, while Bezakova and Dani[5)] already showed that the general case is NP-hard to approximate within ratios smaller than 2.

Another objective function studied for the graph orientation problem is that of *minimizing the maximum weighted outdegree* (MinMaxO), also known as *Graph Balancing*[1)–3),7),13),17)]: Given an undirected graph with edge weights, we are asked to assign a direction to each edge so that the maximum outdegree is minimized. It is obvious that MinMaxO is generally NP-hard. Asahiro et al.[2)] showed that it is still weakly NP-hard for outerplanar graphs, and strongly NP-hard for P_4-bipartite graphs. Fortunately, however, they also showed[2)] that MinMaxO is tractable if the input is limited to trees or even to cactus graphs. Note that the class of cactus graphs is a maximal subset of the class of outerplanar graphs and the class of P_4-bipartite graphs, and a minimal superset of the class of trees. Very recently, Ebenlendr et al.[7)] designed a polynomial-time 1.75-approximation algorithm for the general weighted case, and Asahiro et al.[1)] showed that MinMaxO can be approximated within an approximation ratio of 1.5 in polynomial-time if all edge weights belong to $\{1, 2\}$. As for inapproximability, it is known that MinMaxO is NP-hard to approximate within approximation ratios smaller than 1.5 even for this restricted $\{1, 2\}$-case[1),7)].

Our results. In this paper we study the computational complexity and (in)approximability of the machine covering problem from the viewpoint of the graph based problem, i.e., graph orientation. In Section 2, we prove that MaxMinO is strongly NP-hard and cannot be approximated within a ratio of $\min\{2, \frac{w_{max}}{w_{min}}\} - \epsilon$ for any constant

$\epsilon > 0$ in polynomial time unless P=NP, even if all edge weights belong to $\{w_{min}, w_{max}\}$, every vertex has unweighted degree at most three, and the input graph is bipartite and planar. As mentioned above, although MaxMinO imposes a strong restriction on the Santa Claus problem, unfortunately it is still hard.

Section 3 first considers the unweighted MaxMinO problem. We can obtain an optimal orientation algorithm which runs in $O(m^{3/2} \cdot \log m \cdot \log^2 \Delta)$ time for the special case in which all edge weights are equal to 1. Here, it is important to note that Golovin[12] already claimed that the unweighted case of MaxMinO (more precisely, the Santa Claus problem) can be solved in polynomial time, but no proof of this claim has ever appeared as far as the authors know. Our contribution here is to provide the non-trivial, efficient running time with its explicit proof. Then, we observe that our approach yields an exact algorithm for the general case of MaxMinO whose running time is polynomial whenever the number of edges having weight larger than w_{min} is at most logarithmic in the number of vertices. In Section 4, this efficient algorithm for the unweighted MaxMinO also gives us a simple $\frac{w_{max}}{w_{min}}$-approximation algorithm running in the same time for general (weighted) case of MaxMinO, i.e., it always outputs an orientation Λ' of G which satisfies $OPT(G) \le \frac{w_{max}}{w_{min}} \cdot \delta_{\Lambda'}(G)$. This simple approximation algorithm is best possible for the case that the weights of edges belong to $\{w_{min}, w_{max}\}$ with $w_{max} \le 2w_{min}$ since the lower bound of approximation ratios is $\min\{2, \frac{w_{max}}{w_{min}}\}$ described above.

In the field of combinatorial optimization, much work is often devoted to seek a subset of instances that is tractable and as large as possible. For example, if the input graph G is a tree, then $OPT(G)$ is always 0 because the number of vertices is larger than the number of edges, and in any orientation of G, at least one vertex must have no outgoing edges. Also, for the case of cycles, MaxMinO is quite trivial since the clockwise or counterclockwise orientation along the cycle gives us the optimal value of w_{min}. On the other hand, the class of planar graphs is too large to allow a polynomial-time optimal algorithm (under the assumption of P$\neq$NP). Hence, our goal in Section 5 is to find a polynomially solvable subset between trees and planar graphs. Then, we show that MaxMinO remains in P even if we make the set of instances so large that it contains the class of cactus graphs.

2. Hardness results

In this section, we show the MaxMinO problem is strongly NP-hard even if all the edge weights belong to $\{w_{min}, w_{max}\}$ for any integers $w_{min} < w_{max}$ and the input graph is bipartite and planar. The proof is by a reduction from At-Most-3-SAT(2L).

At-Most-3-SAT(2L) is a restriction of 3-SAT where each clause contains at most three literals and each literal (not variable) appears at most twice in a formula. It can be easily proved that At-Most-3-SAT(2L) is NP-hard by using problem [LO1] on p. 259 of[9].

First, we pick any fixed integers for w_{min} and w_{max} such that $w_{min} < w_{max}$. Given a formula ϕ of At-Most-3-SAT(2L) with n variables $\{v_1, \dots, v_n\}$ and m clauses $\{c_1, \dots, c_m\}$, we then construct a graph G_ϕ including gadgets that mimic (a) variables and (b) clauses. To define these, we prepare a gadget consisting of a cycle of 3 vertices and 3 edges (i.e., a triangle) where each edge of the cycle has weight w_{max}. We call this a *triangle gadget*. Apart from these triangle gadgets, we define gadgets for (a) variables and (b) clauses: (a) Each variable gadget corresponding to a variable v_i consists of two vertices labeled by v_i and $\overline{v_i}$ and one edge $\{v_i, \overline{v_i}\}$ between them. The weight of $\{v_i, \overline{v_i}\}$ is w_{max}. By the definition of At-Most-3-SAT(2L), some literals (say v_i for example) do not occur (or may occur only once). In such a case, we attach a triangle gadget to the variable gadget by adding two edges (one edge) of weight w_{min} that connects vertex v_i and two different vertices (one vertex) of the triangle gadget. (b) Each clause gadget consists of one representative vertex labeled by c_j, corresponding to clause c_j of ϕ, and a triangle gadget connected to this c_j-vertex by an edge of weight w_{min}. The representative vertex c_j is also connected to at most three vertices in the literal gadgets that have the same labels as the literals in the clause c_j, by edges of weight w_{min}. For example, if $c_1 = x \vee \overline{y}$ appears in ϕ, then vertex c_1 is connected to vertices x and $\overline{y}$. (See Figure 1.) We have the following lemma, though we omit the proof.

Lemma1 For the reduced graph G_ϕ, the following holds:
(i) $OPT(G_\phi) \ge \min\{2w_{min}, w_{max}\}$ if ϕ is satisfiable.
(ii) $OPT(G_\phi) \le w_{min}$ if ϕ is not satisfiable.

From Lemma 1, we immediately obtain the following theorem.

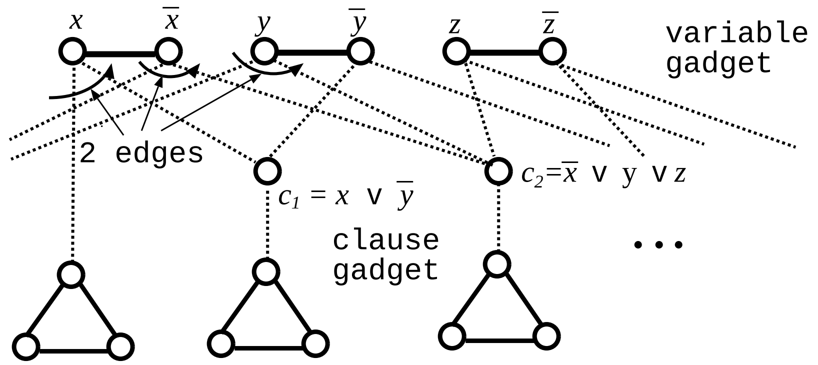

Fig.1 Reduction from At-Most-3-SAT(2L) (Solid and dotted edges have weight w_{max} and w_{min}, respectively.)

Theorem2 MaxMinO is strongly NP-hard even if the edge weights are in $\{w_{min}, w_{max}\}$ ($w_{min} < w_{max}$). □

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

Theorem3 Even if the edge weights are in $\{w_{min}, w_{max}\}$, MaxMinO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $\min\{2, \frac{w_{max}}{w_{min}}\}$, unless P = NP. □

Similarly, we can show the NP-hardness of MaxMinO for planar bipartite graphs by almost the same reduction as the above from Monotone-Planar-One-In-Three-3-SAT(2L), which is a variant of At-Most-3-SAT(2L), having both the planarity[14] and the monotonicity[10].

One-In-Three-3-SAT itself is a variant of 3-SAT problem which asks whether there exists a truth assignment to the variables so that each clause has exactly one true literal (and thus exactly two false literals)[16]. The reason why we use One-In-Three-3-SAT instead of At-Most-3-SAT is to bound the unweighted degrees of the constructed graphs. While the above reduction from At-Most-3-SAT(2L) guarantees that the unweighted degrees of constructed graphs are bounded by four, we can bound the unweighted degrees of constructed graph from One-In-Three-3SAT(2L) by three. In the new reduction, we do not attach triangle gadgets to clause vertices, which makes the unweighted degrees of clause vertices three, and One-In-Three satisfiability guarantees that each clause vertex has two outgoing edges in an optimal MaxMinO solution.

The *planarity* means that the graph constructed from an instance CNF, in which two vertices corresponding to a variable and a clause are connected by an edge if the variable occurs (positively or negatively) in the clause, is planar. The *monotonicity* means that in an input CNF formula each clause contains either only positive literals or only negative literals. Planar-One-In-Three-3-SAT is shown to be NP-complete in[15].

By applying an operation used in[2], we can transform an instance of Planar-One-In-Three-3-SAT into one of Monotone-Planar-One-In-Three-3-SAT. Moreover, by applying another operation used in the same paper[2], we can transform an instance of Monotone-Planar-One-In-Three-3-SAT into Monotone-Planar-One-In-Three-3-SAT(2L). This implies that the constructed graph is planar and bipartite and its unweighted degree is at most three. (To preserve the bipartiteness, we need to use bipartite gadgets, e.g., square gadgets, instead of triangle gadgets.)

Theorem4 MaxMinO is strongly NP-hard even if the edge weights are in $\{w_{min}, w_{max}\}$ for integers $w_{min} < w_{max}$ and the input graph is bipartite and planar in which the unweighted degree is bounded by three. □

Theorem5 Even if the edge weights are in $\{w_{min}, w_{max}\}$ and the input graph is bipartite and planar in which the unweighted degree is bounded by three, MaxMinO has no pseudo-polynomial time algorithm whose approximation ratio is smaller than $\min\{2, \frac{w_{max}}{w_{min}}\}$, unless P=NP. □

This result is tight in a sense, because if the unweighted degree of the input graph is bounded by two (i.e., cycles or trees), obviously MaxMinO can be solved in linear time.

3. An exact algorithm for unweighted cases

MaxMinO is closely related to the problem of computing a maximum flow in a flow network with positive edge capacities. Indeed, maximum-flow-based techniques have been used in[3] to solve the analogous problem of computing an edge orientation which *minimizes* the *maximum* outdegree of a given unweighted graph (MinMaxO) in polynomial time. In this section, we extend the results of[3] by showing how a maximum

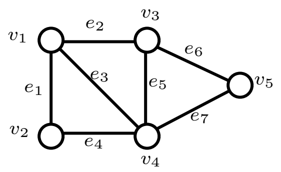

Fig.2 An example of G

flow-algorithm can be used to efficiently solve unweighted MaxMinO.

For any input graph $G = (V, E)$ to unweighted MaxMinO, let $\mathcal{N}_G = (V_G, E_G)$ be the directed graph with vertex set V_G and edge set E_G defined by:

$$V_G = E \cup V \cup \{s,t\},$$
$$E_G = \Big\{(s,e) \mid e \in E\Big\} \cup \Big\{(v,t) \mid v \in V\Big\} \cup \Big\{(e,v_i),(e,v_j) \mid e = \{v_i,v_j\} \in E\Big\},$$

and for any integer $q \in \{0,1,\ldots,\Delta\}$, let $\mathcal{N}_G(q) = (V_G, E_G, cap_q)$ be the flow network obtained by augmenting $\mathcal{N}_G$ with edge capacities cap_q, where:

$$cap_q(a) = \begin{cases} 1, & \text{if } a = (s,e) \text{ with } e \in E; \\ 1, & \text{if } a = (e,v) \text{ with } e \in E,\ v \in V; \\ q, & \text{if } a = (v,t) \text{ with } v \in V. \end{cases}$$

See Figure 2 and Figure 3 for an example of the original graph G and the corresponding network $\mathcal{N}_G$, respectively.

Let $F(q)$ be an integral maximum directed flow[⋆1] from vertex s to vertex t in $\mathcal{N}_G(q)$. Then, for each $e = \{v_i, v_j\} \in E$, either zero or one unit of flow in $F(q)$ passes through the corresponding vertex e in V_G, and thus at most one of the two edges (e, v_i) and (e, v_j) is assigned one unit of flow. This induces an orientation $\Lambda_{F(q)}$ of G based on $F(q)$ as follows: If the flow in $F(q)$ from vertex e to vertex v_i equals 1 then set $\Lambda_{F(q)}(e) := (v_i, v_j)$; else if the flow in $F(q)$ from e to v_j equals 1 then set $\Lambda_{F(q)}(e) := (v_j, v_i)$; else set $\Lambda_{F(q)}(e)$ arbitrarily.

⋆1 Since all edge capacities are integers, we may assume by the integrality theorem (see, e.g.,[6)]) that the flow along each edge in $F(q)$ found by the algorithm in[11)] is an integer.

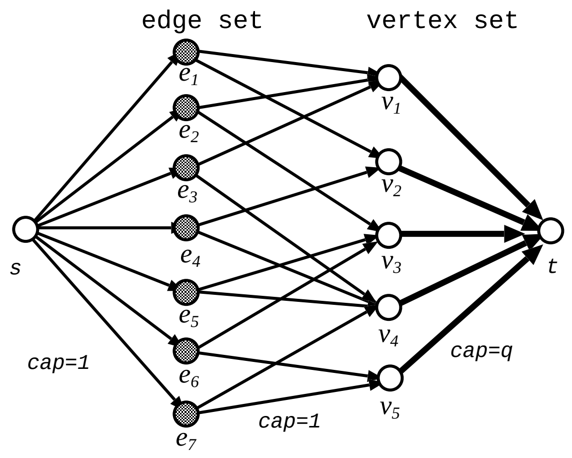

Fig.3 Network $\mathcal{N}_G$ constructed from G of Figure 2

Let $f(q)$ denote the value of a maximum directed flow from vertex s to vertex t in $\mathcal{N}_G(q)$. Then:

Lemma6 For any $q \in \{0,1,\ldots,\Delta\}$, $f(q) \le q \cdot n$.

Proof. The sum of all edge capacities of edges leading into t in $\mathcal{N}_G(q)$ is $q \cdot n$. Clearly, the value of the maximum flow in $\mathcal{N}_G(q)$ cannot be larger than this sum. □

Lemma7 For any $q \in \{0,1,\ldots,\Delta\}$, $f(q) = q \cdot n$ if and only if $OPT(G) \ge q$.

Proof.

$\Longrightarrow$) Suppose that $f(q) = q \cdot n$ and consider the maximum flow $F(q)$ defined above. For each $v \in V$, exactly q units of flow leave the corresponding vertex v in V_G because the edge capacity of (v,t) is q and there are n such vertices. This implies that q units of flow enter v, which is only possible if there are q edges of the form (e,v) in E_G that have been assigned one unit of flow each. Therefore, the induced orientation $\Lambda_{F(q)}$ ensures that $d_{\Lambda_{F(q)}}(v) \ge q$ for every $v \in V$, which yields $OPT(G) \ge q$.

$\Longleftarrow$) Suppose that $OPT(G) \ge q$ and let Λ be a max-min orientation of G. Let F' be the following directed flow from s to t in $\mathcal{N}_G(\Delta)$:

Vol.2009-AL-124 No.7
2009/5/11

$$F'(a)=\begin{cases}1, & \text{if } a=(s,e) \text{ with } e\in E;\\ 1, & \text{if } a=(e,v_i) \text{ with } e=\{v_i,v_j\}\in E \text{ and } \Lambda(e)=(v_i,v_j);\\ 0, & \text{if } a=(e,v_i) \text{ with } e=\{v_i,v_j\}\in E \text{ and } \Lambda(e)=(v_j,v_i);\\ d_\Lambda(v), & \text{if } a=(v,t) \text{ with } v\in V.\end{cases}$$

For every $v \in V$, the flow in F' along the edge (v,t) in $\mathcal{N}_G(\Delta)$ is $d_\Lambda(v) \geq OPT(G) \geq q$. By reducing each such edge flow to q, one obtains a directed flow which obeys the (stricter) edge capacity constraints of the flow network $\mathcal{N}_G(q)$ and has flow value $n \cdot q$. Thus, there exists a maximum directed flow from s to t in $\mathcal{N}_G(q)$ with value $q \cdot n$, so $f(q) \geq q \cdot n$. It follows from Lemma 6 that $f(q) = q \cdot n$. □

Lemmas 6 and 7 suggest the algorithm for unweighted MaxMinO named Algorithm Exact-1-MaxMinO.

```
1: Construct $\mathcal{N}_G$.
2: Use binary search on $q$ in the interval $\{0,1,\ldots,\Delta\}$ to find the integer $q$ such that
   $f(q) = q \cdot n$ and $f(q+1) < (q+1) \cdot n$.
3: Compute $F(q)$ as a maximum directed flow from $s$ to $t$ in $\mathcal{N}_G(q)$.
4: Return $\Lambda_{F(q)}$.
```

Fig.4 Algorithm Exact-1-MaxMinO

Theorem8 Exact-1-MaxMinO solves unweighted MaxMinO in $O(m^{3/2} \cdot \log m \cdot \log^2 \Delta)$ time. □

Proof. The correctness of Exact-1-MaxMinO is guaranteed by Lemmas 6 and 7. For any $q \in \{0,1,\ldots,\Delta\}$, to compute a maximum flow in the flow network $\mathcal{N}_G(q)$ takes $O(m^{3/2} \cdot \log m \cdot \log \Delta)$ time with the algorithm of Goldberg and Rao[11] because $\mathcal{N}_G(q)$ contains $m+n+2 = O(m)$ vertices and $3m+n = O(m)$ edges and the capacity of each edge in $\mathcal{N}_G(q)$ is upper-bounded by Δ. Algorithm Exact-1-MaxMinO can therefore be implemented to run in $O(m^{3/2} \cdot \log m \cdot \log^2 \Delta)$ time. □

Finally, we outline how Exact-1-MaxMinO can be applied to solve weighted MaxMinO. Let X be the set of all edges in E with weight larger than w_{min}. First modify the flow network $\mathcal{N}_G(q)$ to set $cap_q(a) = \lceil w(e)/w_{min} \rceil$. for every edge $a \in E_G$ of the form $a = (s,e)$. Then, run Exact-1-MaxMinO a total of $2^{|X|}$ times while testing all possible ways of setting the capacity of exactly one of (e,v_i) and (e,v_j) in $\mathcal{N}_G(q)$ to $w(e)$ and the other to 0 for each $e \in X$, using binary search on q in the interval $\{0,1,\ldots,\lceil W/n \rceil\}$, and select the best resulting orientation. The asymptotic running time becomes the same as that of Exact-1-MaxMinO multiplied by $2^{|X|}$ and with an increase due to the larger interval for the binary search on q and the edge capacities being upper-bounded by $\max\{w_{max}, W/n\}$ instead of Δ.

Theorem9 Weighted MaxMinO can be solved in $O(m^{3/2} \cdot \log m \cdot \log(w_{max} + W/n) \cdot \log(W/n) \cdot 2^{|X|})$ time, where $X = \{e \in E \mid w(e) > w_{min}\}$.

Corollary1 If $|X| = O(\log n)$ then weighted MaxMinO can be solved in polynomial time. □

4. A simple approximation algorithm for general cases

Here, we prove that ignoring the edge weights of the input graph and applying Exact-1-MaxMinO on the resulting unweighted graph immediately yields a $\frac{w_{max}}{w_{min}}$-approximation algorithm for the general case of the problem. The algorithm is named Approximate-MaxMinO and is listed in Figure 5.

```
1: Let $G'$ be the undirected graph obtained from $G$ by replacing the weight of every
   edge by 1.
2: Apply Algorithm Exact-1-MaxMinO on $G'$ and let $\Lambda'$ be the obtained orientation.
3: Return $\Lambda'$.
```

Fig.5 Algorithm Approximate-MaxMinO

Theorem10 Approximate-MaxMinO runs in $O(m^{3/2} \cdot \log m \cdot \log^2 \Delta)$ time and is a $\frac{w_{max}}{w_{min}}$-approximation algorithm for MaxMinO.

Proof. The asymptotic running time of Algorithm Approximate-MaxMinO is the same as that of Exact-1-MaxMinO.

To analyze the approximation ratio, observe that $\delta_\Lambda(G) \geq w_{min} \cdot \delta_\Lambda(G')$ for any orientation Λ of G because the weight of any edge in G is at least w_{min} times larger than its weight in G'. Similarly, $w_{max} \cdot \delta_\Lambda(G') \geq \delta_\Lambda(G)$ for any orientation Λ of G. Now, let Λ' be the optimal orientation for G' returned by Approximate-MaxMinO and let Λ^* be an optimal orientation for G. Note that $\delta_{\Lambda'}(G') \geq \delta_{\Lambda^*}(G')$. Thus, $\delta_{\Lambda'}(G) \geq w_{min} \cdot \delta_{\Lambda'}(G') \geq w_{min} \cdot \delta_{\Lambda^*}(G') \geq \frac{w_{min}}{w_{max}} \cdot \delta_{\Lambda^*}(G) = \frac{w_{min}}{w_{max}} \cdot OPT(G)$. □

5. An exact algorithm for cactus graphs

In this section, we present a polynomial time algorithm which obtains optimal orientations for cactus graphs. A graph is a *cactus* if every edge is part of at most one cycle. To this end, we introduce vertex weight $\alpha_G(v)$ for each vertex v in a graph G which is considered as 0 in the input graph (we omit the subscript G of $\alpha_G(v)$ if it is apparent). Here we define the notion of weighted outdegree for a vertex in a vertex and edge weighted graph. The *weighted outdegree* $d_\Lambda(v)$ of a vertex v is defined as the weight of v itself plus the total weight of outgoing arcs of v, i.e.,

$$d_\Lambda(v) = \alpha(v) + \sum_{\substack{\{u,v\}\in E:\\ \Lambda(\{u,v\})=(v,u)}} w(\{u,v\}).$$

In a cactus graph, a vertex in a cycle is a *gate* if it is adjacent to any vertex that does not belong to the cycle. Note that the unweighted degree of a gate is at least three. As for the number of gates in a cycle, the following is known:

Proposition11 (Proposition 2 in[2]) In a cactus graph G in which $deg(v) \geq 2$ for every vertex v, there always exists a cycle with at most one gate.

The main part of the proposed algorithm Exact-Cactus-MaxMinO is shown in Figures 6 and 7, which solves the decision version of the problem MaxMinO: Given a number K, this problem asks whether there exists an orientation whose value is at least K. We can develop an algorithm for the original problem MaxMinO by using this algorithm $O(\log \Delta)$ times in a binary search manner on optimal value, which is upper-bounded by Δ.

The correctness of Exact-Cactus-MaxMinO is based on the following property on optimal orientations for two graphs.

Proposition12 Consider two graphs G and G' that differ only on their vertex weights. If $\alpha_G(v) \leq \alpha_{G'}(v)$ for every vertex v, then $OPT(G) \leq OPT(G')$ holds. □

Theorem13 Given a cactus graph G and a target K, Exact-Cactus-MaxMinO outputs an orientation Λ such that $\delta_\Lambda(G) \geq K$ if such an orientation exists, in polynomial time. □

From Theorem 13, we can solve MaxMinO for cactus graphs in polynomial time by using Exact-Cactus-MaxMinO as an engine of the binary search.

Acknowledgments

We thank Tetsuo Shibuya for some inspiring discussions. This work is partially sup-

```
1: repeat
2:   For a vertex $v$,
3:   if $\alpha(v) + d(v) < K$ then
4:     output No and halt.
5:   else if $deg(v) = 1$ then
6:     (let its connecting edge be $e = \{v, u\}$)
7:     if $\alpha(v) < K$ then
8:       $\Lambda(e) := (v, u)$
9:     else
10:      $\Lambda(e) := (u, v)$ and increase $\alpha(u)$ by $w(e)$
11:    end if
12:    Remove $v$ and $e$.
13:  else if $deg(v) = 2$ then
14:    (let $e_1 = \{p, v\}$ and $e_2 = \{v, q\}$)
15:    if $\alpha(v) + w(e_1) < K$ and $\alpha(v) + w(e_2) < K$ then
16:      $\Lambda(e_1) := (v, p)$ and $\Lambda(e_2) := (v, q)$. Remove $v$, $e_1$, and $e_2$.
17:    else if $\alpha(v) + w(e_1) < K$ and $\alpha(v) + w(e_2) \geq K$ then
18:      $\Lambda(e_1) := (p, v)$ and $\Lambda(e_2) := (v, q)$ and also increase $\alpha(p)$ by $w(e_1)$. Remove
         $v$, $e_1$, and $e_2$.
19:    end if
20:  end if
21: until No vertex $v$ satisfies either one of the above conditions
```

Fig.6 Algorithm Exact-Cactus-MaxMinO

```
22: for all $C := \langle v_0, v_1, \cdots, v_\ell = v_0 \rangle$ that has at most one gate do
23:   if $C$ does not have a gate then
24:     $\Lambda(\{v_i, v_{i+1}\}) := (v_i, v_{i+1})$ for $0 \le i \le \ell - 1$. Remove $C$.
25:   else
26:     Let $v_0$ be the gate.
27:     if there exists a vertex $v_j$, $j \ne 0$ satisfying $\alpha(v_j) \ge K$ in $C$ then
28:       Assign $\Lambda(\{v_i, v_{i+1}\}) := (v_i, v_{i+1})$ for $0 \le i \le j - 1$ and $\Lambda(\{v_i, v_{i+1}\}) := (v_{i+1}, v_i)$ for $j \le i \le \ell - 1$. Increase $\alpha(v_0)$ by $w(\{v_0, v_1\}) + w(\{v_0, v_{\ell-1}\})$.
29:     else
30:       If $w(\{v_0, v_1\}) > w(\{v_0, v_{\ell-1}\})$ then assign $\Lambda(\{v_i, v_{i+1}\}) := (v_i, v_{i+1})$ for $0 \le i \le \ell - 1$ and increase $\alpha(v_0)$ by $w(\{v_0, v_1\})$, otherwise $\Lambda(\{v_i, v_{i+1}\}) := (v_{i+1}, v_i)$ for $0 \le i \le \ell - 1$ and increase $\alpha(v_0)$ by $w(\{v_0, v_{\ell-1}\})$.
31:     end if
32:     Remove $C$ except the gate $v_0$.
33:   end if
34: end for
35: if the graph is empty then
36:   output $\Lambda$ and halt.
37: else
38:   go back to line 1.
39: end if
```

Fig.7 Algorithm Exact-Cactus-MaxMinO(cont.)

ported by KAKENHI (No. 18700014, 18700015, 20500017 and 21680001).

参 考 文 献

1) Y. Asahiro, J. Jansson, E. Miyano, H. Ono, and K. Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. In *Proc. of AAIM2007*, pp.167–177, 2007.
2) Y. Asahiro, E. Miyano, and H. Ono. Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. In *Proc. of CATS2008*, pp.97–106, 2008.
3) Y. Asahiro, E. Miyano, H. Ono, and K. Zenmyo. Graph orientation algorithms to minimize the maximum outdegree. *IJFCS*, 18(2), pp.197–215, 2007.
4) N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proc. of STOC2006*, pp.31–40, 2006.
5) I. Bezáková and V. Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3), pp.11–18, 2005.
6) T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
7) T. Ebenlendr, M. Krčál, and J. Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *Proc. of SODA2008*, pp.483–490, 2008.
8) U. Feige. On allocations that maximize fairness. In *Proc. of SODA2008*, pp.287–293, 2008.
9) M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
10) E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3), pp.302–320, 1978.
11) A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *JACM*, 45(5), pp.783–797, 1998.
12) D. Golovin. Max-min fair allocation of indivisible goods. *Tech. Report*, 2005.
13) L. Kowalik. Approximation scheme for lowest outdegree orientation and graph density measures. In *Proc. of ISAAC2006*, pp.557–566, 2006.
14) D. Lichtenstein. Planar formulae and their uses. *SIAM J. Computing*, 11(2), pp.329–343, 1982.
15) W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. In *Proc. of SoCG*, pp.1–10, 2006.
16) T. J. Schaefer. The complexity of satisfiability problems. In *Proc. of STOC1978*, pp.216–226, 1978.
17) V. Venkateswaran. Minimizing maximum indegree. *Disc. Appl. Math.*, 143(1–3), pp.374–378, 2004.
18) G. J. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Opp. Res. Lett.*, 20, pp.149–154, 1997.