社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

次数上下界制約付きグラフ向き付けにおけるペナルティ最小化

朝廣　雄一†　Jesper　JANSSON††　宮野　英次†††　小野　廣隆††††

† 九州産業大学情報科学部
†† お茶の水女子大学
††† 九州工業大学大学院情報工学研究院
†††† 九州大学大学院経済学研究院
E-mail: †asahiro@is.kyusan-u.ac.jp, ††Jesper.Jansson@ocha.ac.jp, †††miyano@ces.kyutech.ac.jp,
††††hirotaka@en.kyushu-u.ac.jp

あらまし　無向グラフ $G=(V,E)$ が与えられたとき, ある制約を満たすように各辺に向き付けを与えた有向グラフ $\vec{G}=(V,\Lambda(E))$ を求める問題をグラフ向き付け問題という. $\Lambda(E)$ は辺 $\{u,v\}\in E$ の向き付け割当集合を表す. 本稿では次数制約の下でのグラフ向き付け問題を扱う: 各頂点 v について正整数 a_v と b_v $(a_v \leqq b_v)$ が指定されたとき，できるだけ多くの頂点 v について $a_v \leqq |\{(v,u)\in\Lambda(E)\}| \leqq b_v$ となる G の向き付けを求める. 本稿では，$\sum_{v\in V} c_v$ を最小化するグラフ向き付けを求める問題を考える. ここで，c_v は次数制約を満たさない頂点 v に対するペナルティを表している. 本稿では以下を示す. (i) 任意の凸なペナルティ関数 (線形関数を含む) を考えたときの次数制約の下でのグラフ向き付け問題は $O(m^{1.5}\min\{\log(nC),\Delta^{0.5}\})$ 時間で解くことができる. ここで，$n=|V|$ および $m=|E|$ であり，Δ と C はそれぞれ最大次数とペナルティ関数の最大値である. (ii) 一方，ステップ関数 (すなわち凹関数) の場合には APX 困難である. (iii) 木の場合には，任意のペナルティ関数について $O(n\log\Delta)$ 時間で解を求めることができ，ペナルティ関数が凸の場合には線形時間で解を求めることができる.

Minimizing Penalty on Upper and Lower Degree Constrained Graph Orientation

Yuichi ASAHIRO†, Jesper JANSSON††, Eiji MIYANO†††, and Hirotaka ONO††††

† Department of Information Science, Kyushu Sangyo University
†† Ochanomizu University
††† Department of Systems Design and Informatics, Kyushu Institute of Technology
†††† Department of Economic Engineering, Kyushu University
E-mail: †asahiro@is.kyusan-u.ac.jp, ††Jesper.Jansson@ocha.ac.jp, †††miyano@ces.kyutech.ac.jp,
††††hirotaka@en.kyushu-u.ac.jp

Abstract　Given an undirected graph $G=(V,E)$, a graph orientation problem is to decide a direction of each edge so that the resulting directed graph $\vec{G}=(V,\Lambda(E))$ satisfies a certain condition, where $\Lambda(E)$ is a set of an assignment of a direction to each edge $\{u,v\}\in E$. We consider a degree constrained orientation: Given positive integers a_v and b_v for each v $(a_v \leqq b_v)$, decide an orientation of G so that $a_v \leqq |\{(v,u)\in\Lambda(E)\}| \leqq b_v$ holds for as many vertices v's as possible. In this paper, we consider the problem of finding an orientation that minimizes $\sum_{v\in V} c_v$, where c_v is a penalty incurred for v's violating the degree constraint. We show that: (i) The degree-constrained orientation with any convex (including linear) penalty function can be solved in $O(m^{1.5}\min\{\log(nC),\Delta^{0.5}\})$, where $n=|V|, m=|E|$, Δ and C are the maximum degree and the largest magnitude of a penalty, respectively. (ii) In contrast, it is APX-hard for step (i.e., concave) penalty functions. (iii) For trees, the problem with any penalty functions can be solved in $O(n\log\Delta)$ time, and if the penalty function is convex, it is solvable in linear time.

1. Introduction

1.1 Problem and summary of results

We assume a basic knowledge of graph theory. Let $G = (V, E)$ be an undirected graph, where V and E denote the sets of vertices and edges, respectively. We allow G to have parallel edges; G is possibly a multi-graph. Throughout the paper, let $|V| = n$ and $|E| = m$ for the graph. Two vertices u and v are called *adjacent* to each other if $\{u, v\} \in E$. Let $N(u)$ be the set of adjacent vertices of u, i.e., $N(u) = \{v \mid \{u, v\} \in E\}$, and $d(u) = |N(u)|$ is called *degree* of u. We denote $\max\{d(v) \mid v \in V\}$ by Δ. An *orientation* Λ *of graph* G is a set of an assignment of a direction to each edge $\{u, v\} \in E$, i.e., $\Lambda(\{u, v\}) = (u, v)$ or (v, u). We simply use Λ to represent $\Lambda(E) = \bigcup_{e \in E}\{\Lambda(e)\}$ if no confusion arises. The *outdegree* of u on Λ is $|\{v \mid (u, v) \in \Lambda\}|$, which is denoted by $d^+_\Lambda(u)$.

Suppose that a sequence of $2n$ positive integers $a_v, b_v (a_v \leqq b_v)$ for $v \in V$ is given as a degree constraint. For a_v and b_v's, a *degree constrained orientation* is an orientation of G such that $a_v \leqq d^+_\Lambda(v) \leqq b_v$ holds for every $v \in V$. Obviously, G does not always have a degree constrained orientation. In such a case, we would like to find an orientation that "best" fits the degree constraint. A *violation vector* $c(\Lambda)$ of an orientation Λ of G is $(c_1(\Lambda), c_2(\Lambda), \ldots, c_n(\Lambda))$, where for $v \in V$, $c_v(\Lambda) = d^+_\Lambda(v) - b_v$ if $d^+_\Lambda(v) > b_v$, $c_v(\Lambda) = a_v - d^+_\Lambda(v)$ if $d^+_\Lambda(v) < a_v$, $c_v(\Lambda) = 0$ otherwise. A *penalty function* p is a finite, non-negative and non-decreasing function with n variables. By using these, we define the best-fit orientation to the degree constraint by an orientation Λ of G that minimizes $p(c(\Lambda))$. We call this Minimum Penalty Degree Constrained Orientation, MPDCO for short.

For an example, see the undirected graph $G = (V, E)$ in Figure 1-(a). Figures 1-(b) and (c) are two directed graphs obtained by orientations Λ and Λ', respectively. For example, let $a_v = 1$ and $b_v = 2$ for any $v \in V$. We first observe the case when p is a summation of a convex function $g(x) = x^2$ for each vertex, i.e., $p(c(\Lambda)) = \sum g(c_v(\Lambda))$. (b) If the outdegree sequence is $(1, 1, 5, 0, 2, 2, 0, 2)$ in the column major order from the left to the right, then its violation vector $c(\Lambda)$ is $(0, 0, 3, 1, 0, 0, 1, 0)$. Thus, the total penalty function $p(c(\Lambda))$ is $\sum g(c_v(\Lambda)) = 3^2 + 1^2 + 1^2 = 11$. On the other hand, (c) $c(\Lambda') = (0, 0, 1, 1, 1, 1, 1, 0)$ and $p(c_v(\Lambda')) = \sum g(c_v(\Lambda')) = 1^2 \times 5 = 5$.

As another example, consider the following concave function $g'(x)$ instead of $g(x)$:

$$g'(x) = \begin{cases} x & \text{if } x \leqq 1 \\ 1 & \text{if } x > 1. \end{cases}$$

Then, (b) the penalty of Λ is $\sum g'(c_v(\Lambda)) = 1 + 1 + 1 = 3$, and (c) the penalty of Λ' is $\sum g'(c_v(\Lambda')) = 1 \times 5 = 5$. The "balanced" orientation Λ' is worse than the "unbalanced" orientation Λ for g'. Also, note that the penalty highly depends on the values of a_v and b_v.

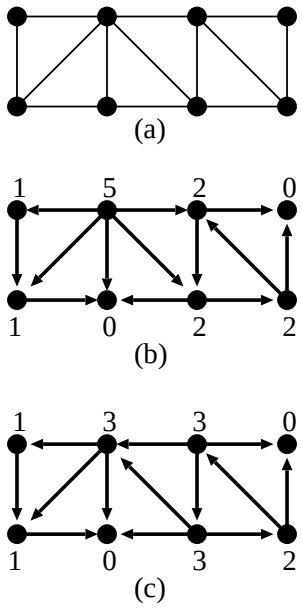

Fig. 1 Example of MPDCO: (a) An undirected graph $G = (V, E)$; (b) an orientation Λ of G with outdegree sequence $(1, 1, 5, 0, 2, 2, 0, 2)$ from the left top to the right bottom; (c) an orientation Λ' of G with outdegree sequence $(1, 1, 3, 0, 3, 3, 0, 2)$.

Degree constrained orientations have been intensively studied for long time, because many graph problems including graph routing, matching, and covering can be formalized as graph orientations. The earliest result about the orientability for given degree constraints is by Landau [19], who proved the necessary and sufficient conditions complete graphs must satisfy. Subsequently, Hakimi generalized Landau's result to general graphs. However, as mentioned above, such an orientation does not always exist. Our problem MPDCO resolves the problem for such a case, by considering degree constraints as soft constraints; it finds an optimal orientation that minimizes penalties charged for violation.

It should be noted that MPDCO is a natural generalization of several optimization problems to control outdegrees of an undirected graph. For example, Minimum Maximum Outdegree Orientation Problem (MinMaxO), i.e., the problem of finding an orientation of minimizing $\max\{d^+_\Lambda(u) \mid u \in V\}$, is formalized as MPDCO with $a_i = b_i = 0, i = 1, 2, \ldots, n$ and n-dimensional ∞ norm α (i.e., $p(x_1, x_2, \ldots, x_n) = (\sum_i^n |x_i|^\alpha)^{1/\alpha}$, $\alpha \to \infty$). Maximum Minimum Outdegree Orientation Problem (MaxMinO), the problem of finding an orientation of maximizing $\min\{d^+_\Lambda(u) \mid u \in V\}$, is also [5, 3, 4].

Clearly, the nature of MPDCO depends on the penalty function. In this paper, we study the relationship between the computational complexity of MPDCO and penalty functions. We assume here that penalty functions are linearly separable, i.e., it can be written as $p(c(\Lambda)) = \sum_{u \in V} g(c_u(\Lambda))$, where g is a non-negative one variable function with $g(0) = 0$. In this setting, we focus on the following types of functions as typical examples of g: convex (including linear) and concave (including step) functions. Throughout the paper, we assume that g is a fixed function, and evaluated in constant time; for a positive integer x, the value of $g(x)$ is obtained in $O(1)$ time.

The results in this paper are summarized as follows:

(1) If g is a convex function, MPDCO can be solved in $O(m^{1.5} \min\{\log(nC), \Delta^{0.5}\})$ time, where C is the largest magnitude of a penalty. That is, if we adopt a natural polynomial penalty function, such as n^k with a positive integer k, it is $O(m^{1.5} \log n)$. Note that this time complexity is similar to that of MinMaxO for unweighted graphs, a restricted case of MPDCO, in terms of order [5, 18, 25].

(2) MPDCO has no polynomial approximation algorithm whose approximation factor is better than 1.3606 for concave penalty functions, unless P=NP; it is APX-hard. This holds even for step penalty functions. Furthermore, MPDCO is still APX-hard for a strictly increasing concave function, a more restricted class of concave functions.

(3) For trees, the problem with general penalty functions can be solved in $O(n \log \Delta)$ time. This running time is available for any rational valued penalty function if basic arithmetic operations can be done in $O(1)$ time; g is not necessary to be convex or concave. If g is convex, the running time is improved to $O(n)$.

The remainder of the paper is organized as follows. In the next subsection, we introduce related work about graph orientation to control (weighted) outdegrees. In Section 2,we present polynomial time algorithms for MPDCO with convex functions. Section 3 shows the hardness of MPDCO with concave functions, and Section 4 presents $O(n \log \Delta)$ and $O(n)$ time algorithms for trees. Section 5 discusses further research on the problems and concludes the paper.

1. 2 Related work

Graph orientation itself is a fundamental problem in the area of graph theory and combinatorial optimization. In general, graph orientation is a problem of giving an orientation to a given undirected graph to meet some given requirement. There are many types of requirements considered, such as connectivity, reachability and so on [23, 22, 16]. Among several variations of graph orientation, many researchers have been devoted to graph orientation with degree constraints and there are a large literature; e.g., see Sections 61.1 in [24] 7.4.3 in [20] and 2.3 in [11].

For example, Hakimi [15], Frank and Gyárfás [12], Chrobak and Eppstein [9], and Gabow [13] studied the degree-constrained orientation problem, where its goal is to orient as many edges as possible in an undirected graph, subject to the upper and lower bounds on the outdegree of each vertex (or equivalently, the upper bounds on the indegree and outdegree of each vertex). In [15] Hakimi gave the necessary and sufficient conditions of graphs that can be oriented so that every outdegree is at most a given upper bound. These were generalized by Frank and Gyárfás in [12] to a characterization of graphs that can be oriented so that every outdegree is between given upper and lower bounds.

In [9], Chrobak and Eppstein studied the orientation of a planar graph and showed that a 3-bounded outdegree orientation and a 5-bounded acyclic orientation can be obtained in linear time for any planar graph. Furthermore, recently, in [13], Gabow considered the *partial orientation* problem, which formulates the degree-constrained orientation problem as the optimization problem. A partial orientation assigns a unique direction to a *subset* of the edges, leaving the remaining edges unoriented. Then, the goal is to orient as many edges as possible in the input undirected graph without breaking the degree constraints. He proved that the partial orientation problem is MAXSNP-hard and provided an LP-rounding algorithm which achieves approximation ratio 3/4. Remark that our formulation of the degree-constrained orientation can be regarded as the dual problem of the previous one; the whole set of edges has to be assigned a unique direction, but the degree constraints can be broken with penalty.

As mentioned above, MPDCO is a generalization of MinMaxO (resp., MaxMinO), which is an orientation that minimizes (resp., maximizes) the minimum (resp., maximum) outdegree. MinMaxO and MaxMinO also have been studied well by the relation with other combinatorial problems. For example, MinMaxO can be used in efficient dynamic data structures for graphs that support fast vertex adjacency queries under a series of edge operations [7]. Furthermore, edge weighted version of MinMaxO is a special case of *the minimum makespan problem* (e.g., [21]).

2. Polynomial time algorithms for convex penalty

In this section, we present a simple approach to solve MPDCO with convex functions. The time complexity is $O(m^{1.5} \min\{\log(nC), \Delta^{0.5}\})$, where C is the largest magnitude of a penalty. The approach is based on the reduction to the convex cost flow problem, which can be solved in strongly polynomial time.

In general, a network flow problem is the problem of finding some optimal flow on a given network that satisfies capacity constraints on arcs (directed edges) and supply/demand conditions on vertices, and many types of network flow problems are intensively and extensively studied. See [2]. Among them, the convex cost flow problem is an optimization problem on a network in which each arc is associated with a convex function whose variable is flow through the arc. The cost of flow is the total sum of flow costs on the arcs, and the convex cost flow problem is the problem of finding a flow whose cost is minimum. If the convex functions are just linear functions, the problem is simply called the *minimum cost flow problem*, which is a well studied problem. It is known that the convex cost flow problem can be solved in $O(NM\log(N^2/M)\log(nU))$ time, where N, M and U are the number of vertices, the number of edges and the largest magnitude of the lower and upper bounds on capacities in the network, respectively [1]. Fortunately, our problem belongs to a relatively smaller class of the flow problems; the capacity of every arc is integral. For this class, a faster algorithm by [14] can be applied. The running time is $O((\min\{MU\log(MU), N^2\sqrt{M}\})\log(NC) + \min\{\sqrt{MU}, N^{2/3}M^{1/3}, N\}M)$, where M, N, U are defined as above and C is the largest magnitude of a cost (the cost is assumed to be integral), or $O(\sqrt{\tilde{M}}\tilde{M}\log(NC))$, where $\tilde{M}$ is the total capacity of edges.

[Theorem 1] MPDCO with convex penalty can be solved in $O(m^{1.5}\min\{\log(nC), \Delta^{0.5}\})$ time, where C is the largest magnitude of a cost.

Proof. From graph $G = (V, E)$, sequence of $2n$ integers $(a_1, a_2, \ldots, a_n), (b_1, b_2, \ldots, b_n)$ and convex penalty function $g(x)$, we construct the following network $\mathcal{N} = (V_{\mathcal{N}}, E_{\mathcal{N}})$:

$$V_{\mathcal{N}} = V \cup E \cup \{s, t\},$$
$$E_{\mathcal{N}} = \bigcup_{e=\{u,v\}\in E} \{(s, e), (e, u), (e, v)\} \cup E_t,$$

where $E_t = \bigcup_{v\in V}\{(v, t)\}$. The capacity of arc (v, t) in E_t is defined by $\mathrm{cap}((v, t)) = d(v)$, and those of the other arcs are 1. The supply of source s and the demand of sink t are set to be m. See Figure 2 as an example of this construction. The dotted edges are in E_t. For this network, a flow of $\mathcal{N}$ is a function $f : E_{\mathcal{N}} \to R^+$, where R^+ is the set of non-negative real number, which satisfies that $f((i, j)) \leqq \mathrm{cap}((i, j))$ for all $(i, j) \in E_{\mathcal{N}}$, $\sum_{(u,i)\in E_{\mathcal{N}}} f((u, i)) = \sum_{(j,u)\in E_{\mathcal{N}}} f((j, u))$ for all $u \in V_{\mathcal{N}} \setminus \{s, t\}$, $\sum_{(s,i)\in E_{\mathcal{N}}} f((s, i)) = m$ and $\sum_{(i,t)\in E_{\mathcal{N}}} f((i, t)) = m$.

We define the cost function of arc $(v, t) \in E_t$ as

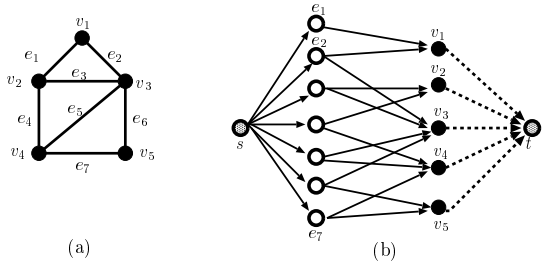

Fig. 2 Network construction in Section 2.: (a) An undirected graph, (b) the network constructed from (a).

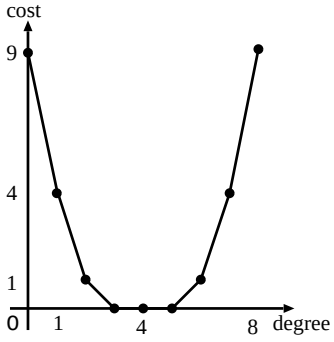

Fig. 3 Example of $\mathrm{cost}_{(v,t)}(x)$: $a_v = 3, b_v = 5$ and $g(x) = x^2$.

$$\mathrm{cost}_{(v,t)}(x) = \begin{cases} g(a_v - x) & \text{if } x < a_v, \\ 0 & \text{if } a_v \leqq x \leqq b_v, \\ g(x - b_v) & \text{if } x > b_v, \end{cases}$$

where x is the amount of flow on this arc. For any other arc (i, j), $\mathrm{cost}_{(i,j)}(x) = 0$. Note that the cost functions are all convex under the assumption g is a convex function with $g(0) = 0$. See Figure 3 for an example of $g(x)$ and $\mathrm{cost}_{(v,t)}(x)$.

Here, we can see that if f is an integral flow, it can be regarded as an orientation of G. In fact, in any feasible integral flow of $\mathcal{N}$, for any $e = \{u, v\} \in E$, exactly one of the following cases holds: $f((e, u)) = 1$ and $f((e, v)) = 0$, or $f((e, u)) = 0$ and $f((e, v)) = 1$, which can be interpreted to an orientation of e. Then, $f((v, t))$ corresponds to the outdegree of v, which implies to $cost_{(v,t)}(f((v, t)))$ is the penalty on v in the orientation, and hence $\sum_{v\in V} cost_{(v,t)}(f((v, t))) = \sum_{(i,j)\in E_{\mathcal{N}}} cost_{(i,j)}(f((i, j)))$ is the penalty of the orientation. Therefore, our problem is to find an integral flow that minimizes $\sum_{(i,j)\in E_{\mathcal{N}}} cost_{(i,j)}(f((i, j)))$, which is a minimum convex cost flow problem with integral constraints.

Since it is known that the convex cost flow problem with integral capacities has an integral optimal flow [14], the integral constraints can be removed. Furthermore, the algorithm presented in the same paper finds such a solution in $O((\min\{MU\log(MU), N^2\sqrt{M}\})\log(NC) + \min\{\sqrt{MU}, N^{2/3}M^{1/3}, N\}M)$ time, where N, M, U and C are the number of vertices, the number of edges and the

largest magnitude of the lower and upper bounds on capacities, the largest magnitude of a cost in the network, respectively, or in $O(\sqrt{\tilde{M}}\tilde{M}\log(NC))$ time, where $\tilde{M}$ is the total capacity of edges. In our reduction, $N = O(m)$, $M = O(m)$, $U = \Delta$, C is the largest cost and $\tilde{M} = O(m)$. Consequently, we can solve MPDCO with convex penalty in $O(m^{1.5}\min\{\log(nC), \Delta^{0.5}\})$ time. □

By this theorem, for natural polynomial penalty functions, such as n^k with a constant k, the problem can be solved in $O(m^{1.5}\log n)$ time. This time complexity is almost same as that of MinMaxO and MaxMinO, $O(m^{1.5}\log\Delta)$.

[Remark 2] This reduction is also available for hypergraph orientation. A *hypergraph* $\mathcal{H} = (V, H)$ is an extension of ordinary graphs, in which each hyperedge $e \in E$ can have more than two vertices. An orientation Λ of a hypergraph is an assignment of a hyperedge e to a vertex in e, which is a generalization of graph orientation [11]. The above reduction works for hypergraphs, and achieves essentially the same time complexity, i.e., $O(m^{1.5}\min\{\tilde{\Delta}^{0.5}, \log(nC)\})$, where $\tilde{\Delta} = \max_{v\in V}\{|\{e \mid v \in e\}|\}$.

3. APX-hardness for step and strictly concave functions

In this section, we show the APX-hardness of MPDCO with concave penalty functions. In Section 3. 1, we show the hardness of MPDCO with step functions, which are also considered concave. In Section 3. 2, we consider strictly concave functions, as another typical classes of concave functions.

In both cases, MPDCO is APX-hard. Especially, MPDCO with step penalty functions is not approximable within 1.3606 unless P = NP, so is MPDCO with convex penalty functions.

3. 1 APX-hardness for step functions

In this section, let $a_u = b_u = 0$ for every $u \in V$, and hence $c_u(\Lambda) = d^+_\Lambda(u)$. Suppose that the penalty function is a step function, defined as

$$g(x) = \begin{cases} c & \text{if } x > 0 \\ 0 & \text{if } x = 0, \end{cases}$$

where c is a positive constant. For simplicity, we call MPDCO in this setting MPDCO_{step}. What we do in the following is to show that MPDCO_{step} is essentially equivalent to Minimum Vertex Cover (MinVC for short). A vertex set $V' \subseteqq V$ of graph $G = (V, E)$ is called a *vertex cover* if for every edge $e \in E$, $e \cap V' \neq \emptyset$ holds. MinVC is the problem of finding the minimum size of a vertex cover of a given graph $G = (V, E)$. We show the following theorem:

[Theorem 3] A graph $G = (V, E)$ has a vertex cover with size k if and only if MPDCO_{step} for G has cost ck.

Proof. Suppose that the graph G has a vertex cover V' of size k. That is, for every edge $\{u, v\}$, $u \in V'$ or $v \in V'$. If $u \in V'$ and $v \notin V'$, then the edge $\{u, v\}$ is oriented from u to v. If both vertices u and v in V', we orient $\{u, v\}$ to the arbitrary direction. One can see that the degree of every vertex in $V \setminus V'$ is 0. Thus, the total cost of the orientation is ck.

If the minimum size of the vertex cover of G is at least $k+1$, then there are at least $k+1$ vertices whose degrees are at least 1 for any orientation. Therefore, the cost is at least $c(k+1)$. □

By the inapproximability of vertex cover problem in [10] and [17], we obtain the following corollaries.

[Corollary 1] There is no polynomial time algorithm for MPDCO_{step} whose approximation factor is better than 1.3606, unless P=NP.

[Corollary 2] There is no polynomial time algorithm for MPDCO_{step} whose approximation factor is better than 2, if the unique game conjecture holds.

3. 2 APX-hardness for strictly concave functions

The goal of this section is to show that the hardness of approximation for MPDCO with a strictly concave function via a gap-preserving reduction from a variant of the *Maximum 4-Dimensional Matching* problem. Here, let $a_u = b_u = 0$ for every $u \in V$ again.

[Definition 4] (Max-4DM) Given the disjoint sets R, S, T and U having the same number n of elements, and a set $Q \subseteqq R \times S \times T \times U$, a *4D-matching* for Q is a subset $Q' \subseteqq Q$ such that no elements in Q' agree in any coordinate. The goal of the Maximum 4-Dimensional Matching (Max-4DM for short) is to find the cardinality of a maximum 4D-matching.

[Definition 5] (Max-4DM-2) If the number of occurrence of any element in R, S, T or U is exactly twice, this restricted version of the Maximum 4-Dimensional Matching is called Max-4DM-2. Note that the instance Q of Max-4DM-2 includes exactly $2n$ quadruplets.

It is known [8] that it is NP-hard to approximate Max-4DM-2 to within $\frac{48}{47}$.

Let $OPT_{4dm}(Q)$ denote the cardinality of a 4D-matching output by an optimal algorithm for the instance Q of Max-4DM-2.

[Lemma 1] Suppose that $|R| = |S| = |T| = |U| = n$ and thus $|Q| = 2n$. Then, the following inequality holds:

$$\frac{2n}{5} \leqq OPT_{4dm}(Q) \leqq n.$$

Proof. If we select one quadruplet, say, (r, s, t, u), and put it into a 4D-matching Q', then there are at most four quadruplets which contains r, s, t, or u in the worse case. In other words, we can always obtain at least $2n/5$ disjoint quadruplets. By the assumption $|R| = |S| = |T| = |U| = n$,

$OPT_{4dm}(Q) \leqq n$ holds. □

[Theorem 6] There is no polynomial time algorithm for $\mathsf{MPDCO}_{concave}$ whose approximation factor is better than $\frac{433}{432}$, unless P=NP.

Proof. We can give a gap-preserving reduction from Max-4DM-2 to $\mathsf{MPDCO}_{concave}$ that transforms an instance Q of Max-4DM-2 to an undirected graph $G = (V, E)$ and its concave functions on vertices such that (1) if $OPT_{4dm}(Q) = \mathsf{max}$, then $OPT_{mpdco}(G) = 4n - \mathsf{max}$, and (2) if $OPT_{4dm}(Q) \leqq \frac{47}{48} \cdot \mathsf{max}$, then $OPT_{mpdco}(G) \geqq 4n - \frac{47}{48} \cdot \mathsf{max}$. We can set $\mathsf{max} = c \cdot n$ for $c \geqq \frac{2}{5}$ by Lemma 1. Then, the gap between (1) and (2) is calculated as follows:

$$\begin{aligned} \frac{4 - \frac{47}{48}c}{4 - c} &= 1 + \frac{c}{192 - 48c} \\ &\geqq 1 + \frac{1}{432}. \end{aligned}$$

This means that the inapproximability bound is 433/432. Details are omitted. □

4. Faster time algorithm for trees

In this section, we propose an $O(n \log \Delta)$ time algorithm that solves MPDCO with any penalty functions for trees. In contrast, MPDCO with concave penalty is APX-hard in general. We also show that the algorithm can be modified to run in linear time for convex penalty.

4.1 $O(n \log \Delta)$ time algorithm for general case

Our algorithm is based on dynamic programming. To explain the idea, we first introduce some new notation.

We denote the optimal value of MPDCO of a graph G by $Opt(G)$ in the following. For a tree T rooted at some r, we consider a tree $T + s$ in which a vertex s is attached with r, that is, $T + s = (V(T) \cup \{s\}, E(T) \cup \{\{r, s\}\})$ rooted at s. In the context of MPDCO, s is a virtual vertex for which no penalty is charged in any orientation (or, we assume $a_s = 0$ and $b_s = \infty$). For vertex v with degree k, we denote its violation by $\theta_v(k)$, that is, $\theta_v(k) = k - b_v$ if $k > b_v$, $\theta_v(k) = a_v - k$ if $k < a_v$, $\theta_v(k) = 0$ otherwise. Thus, for an orientation Λ, $c_v(\Lambda) = \theta_v(d^+_\Lambda(v))$.

In this setting, we consider two "optimal" orientations of $T+s$; one is an optimal orientation under the constraint that $\{r, s\}$ is oriented as (s, r), and the other is one under $\{r, s\}$ is oriented as (r, s). We denote the values of such orientations by $q^-(T)$ and $q^+(T)$, respectively. That is, these can be represented by

$$q^-(T) = \min_\Lambda \left\{ \sum_{v \in V(T)} g(c_v(\Lambda)) \right\}$$

and

$$q^+(T) = \min_\Lambda \left\{ g(\theta_r(d^+_\Lambda(r) + 1)) + \sum_{v \in V(T) \setminus \{r\}} g(c_v(\Lambda)) \right\}.$$

Note that $q^-(T) = Opt(T)$. Clearly, $Opt(T + s) = \min\{q^-(T), q^+(T)\}$.

Now we show a "principle of optimality" equation. Let

$$\mathcal{L}(v, k) = \{\Lambda \mid d^+_\Lambda(v) = k\},$$

and T_w be the subtree of T rooted at w. Then, we have

$$q^-(T) = \min_{k=0,\ldots,d(r)} \min_{\Lambda \in \mathcal{L}(r,k)} \{g(\theta_r(k)) + q(T, k)\},$$

and

$$q^+(T) = \min_{k=0,\ldots,d(r)} \min_{\Lambda \in \mathcal{L}(r,k)} \{g(\theta_r(k+1)) + q(T, k)\},$$

where

$$q(T, k) = \min_{\substack{N' \subseteq N(r) \\ |N'| = k}} \left\{ \sum_{w \in N'} q^-(T_w) + \sum_{w \in N(r) \setminus N'} q^+(T_w) \right\}.$$

Here, $q(T, k)$ is the minimum orientation value without the cost on r under the constraint that $d^+_\Lambda(r) = k$. Since $g(\theta_r(k))$ and $g(\theta_r(k + 1))$ do not depend on orientations and $q(T, k)$ is transformed to,

$$q(T, k) = \sum_{w \in N(r)} q^+(T_w) + \min_{\substack{N' \subseteq N(r) \\ |N'| = k}} \left\{ \sum_{w \in N'} h(T_w) \right\},$$

where $h(T_w) = q^-(T_w) - q^+(T_w)$, $q^-(T)$ and $q^+(T)$ can be essentially computed by k-times evaluation of

$$\min_{\substack{N' \subseteq N(r) \\ |N'| = k}} \left\{ \sum_{w \in N'} h(T_w) \right\}. \tag{1}$$

Here let $N(r, k)$ be a set of vertices in $N(r)$ that have the k smallest $h(T_w)$. Then equation (1) is simply represented by

$$\tilde{h}(T, k) \stackrel{\text{def}}{=} \sum_{w \in N(r,k)} h(T_w). \tag{2}$$

To obtain $N(r, k)$ quickly, it is sufficient to sort $h(T_w)$ of $w \in N(r)$; by sorting $h(T_w)$'s, we obtain $h_1(T), h_2(T), \ldots, h_{d(r)}(T)$, where $h_1(T) \leqq h_2(T) \leqq \ldots \leqq h_{d(r)}(T)$. Then, $\tilde{h}(T, k) = \sum_{i=1}^{k} h_i(T)$.

Based on these ideas, we can compute $q^-(T)$ and $q^+(T)$ of T rooted at r from the values of $q^-(T_w)$ and $q^+(T_w)$ of $w \in N(r)$, children of r as follows: compute $h(T_w)$ for $w \in N(r)$ in $O(d(r))$ time, $\sum_{w \in N(r)} q^+(T_w)$ in $O(d(r))$ time, and sort $h(T_w)$'s in $(d(r) \log d(r))$ time. Then we can compute $g(\theta_r(k)) + \sum_{w \in N(r)} q^+(T_w) + \tilde{h}(T, k)$ (resp., $g(\theta_r(k+1)) + \sum_{w \in N(r)} q^+(T_w) + \tilde{h}(T, k)$) incrementally, where each increment takes $O(1)$ time, and the minimum of them is also found. That is, $q^-(T)$ and $q^+(T)$ are computed in $O(d(r) \log d(r))$ time. Thus, $q^+(T)$ and $q^-(T) (= Opt(T))$ are obtained by the bottom up manner, and the total running time is

$$\sum_{v\in V(T)} O(d(v)\log d(v)) = \sum_{v\in V(T)} O(d(v)\log\Delta) = O(n\log\Delta).$$

[Theorem 7] For any penalty function, MPDCO can be solved in $O(n\log\Delta)$ time, when G is a tree.

[Remark 8] This $O(n\log\Delta)$ algorithm works for any rational valued penalty function if basic arithmetic operations can be done in $O(1)$ time; it is not necessary to be monotone or $g(0)=0$.

4.2 $O(n)$ time algorithm for convex penalty

In the above algorithm, the $O(\log\Delta)$-factor of the running time is due to sorting. If the penalty function is convex, we can avoid the sorting by utilizing the monotonicity, by which the optimal solution is found by checking $g(\theta_r(k))$ and equation (2) only for restricted numbers of k.

As seen in above, to obtain $q^-(T)$ and $q^+(T)$ it is enough to compute

$$\min_{k=0,\ldots,d(r)}\{g(\theta_r(k))+\tilde{h}(T,k)\}$$

and

$$\min_{k=0,\ldots,d(r)}\{g(\theta_r(k+1))+\tilde{h}(T,k)\},$$

respectively. Thus we focus on the behavior of

$$\tilde{q}^-(T,k) \stackrel{\text{def}}{=} g(\theta_r(k))+\tilde{h}(T,k)$$

and

$$\tilde{q}^+(T,k) \stackrel{\text{def}}{=} g(\theta_r(k+1))+\tilde{h}(T,k).$$

In the following, we only argue about $\tilde{q}^-(T,k)$ for simplicity, because almost the same argument holds for $\tilde{q}^+(T,k)$.

A function on the integer domain is convex if and only if the differences are monotonically non-decreasing. In this section, we assume that g is convex, so $g_i \stackrel{\text{def}}{=} g(i)-g(i-1)$ is monotonically non-decreasing. Since $\tilde{h}(T,k)-\tilde{h}(T,k-1)=h_k(T)$, which is monotonically non-decreasing, $\tilde{h}(T,k)$ is also convex. Therefore, $\tilde{q}^-(T,k)-\tilde{q}^-(T,k-1)=g_{\theta_r(k)}+h_k(T)$, is also monotonically non-decreasing, and thus $\tilde{q}^-(T,k)$ is convex.

Hence, $\tilde{q}^-(T,k)$ takes its minimum at $k=0$, $k=d(r)$, or k^* such that

$$g_{\theta_r(k^*)}+h_{k^*}(T)\leqq 0 \text{ and } g_{\theta_r(k^*+1)}+h_{k^*+1}(T)\geqq 0.$$

Since for $k=0$ and $d(r)$, $\tilde{q}^-(T,k)$ can be easily computed, the crucial point is how we know k^*. This is not trivial, because $\tilde{h}(T,k)$ and $h_k(T)$'s are not explicitly given, and we just have raw values of $h(T_w)$ (and we do not want to sort them, because it takes $O(d(r)\log d(r))$ time).

Here, we use a SELECTION algorithm by the binary search manner to obtain k^*. SELECTION(S,k) is the problem of finding the k-th smallest number in a given list S, and it can be solved in $O(|S|)$ time [6].

A description of the whole algorithm to obtain $q^-(T)$ is given below (Algorithm 1). Since the $g(\theta_r(i))$ part can be obtained in $O(1)$ time by the assumption, the running time of the algorithm is dominated by the running time of SELECTION's and updating S, which takes $O(|S|)$ time. Since the size of S becomes half in every execution of Step 2, the total running time is $O(|N(r)|+|N(r)|/2+|N(r)|/4+\cdots)=O(|N(r)|)=O(d(r))$. Therefore, we can compute $q^-(T)$ and $q^+(T)$ in $\sum_{v\in V}O(d(v))=O(n)$ time.

[Theorem 9] For convex penalty functions, MPDCO can be solved in $O(n)$ time, when G is a tree.

Algorithm 1 Algorithm Compute $q^-(T)$

1: Let $l:=0$ and $u:=d(r)$, and $S:=\{h(T_w)\mid w\in N(r)\}$.
2: Let $i := \lfloor (l+u)/2\rfloor$. Compute $g_{\theta_r(i)}$ and $h_i(T)$ by SELECTION(S,i).
3: If $g_{\theta_r(i)}+h_i(T)>0$ holds, then update $S:=S\setminus\{h(T_w)\in S \mid h(T_w)\geqq h_i(T)\}$, $u=i$, and goto Step 2.
4: If $g_{\theta_r(i)}+h_i(T)<0$ holds, then update $S:=S\setminus\{h(T_w)\in S \mid h(T_w)\leqq h_i(T)\}$ and $l:=i$, and goto Step 2.
5: If $g_{\theta_r(i)}+h_i(T)=0$ or $u-l\leqq 1$ holds, then let $k^*=i+l$ and compute $q^-(T)=\min\{\tilde{q}^-(T,0),\tilde{q}^-(T,k^*),\tilde{q}^-(T,d(r))\}$. Then output it. halt.

5. Concluding remarks

In this paper, we studied a type of degree constrained orientation of undirected graphs. The problem is formulated as an optimization problem that minimizes penalties for violation. There are many possible extensions. One such extension is to further consider the hypergraph setting more, as seen in Section 2.

Another extension is to consider the weighted case. MinMaxO and MaxMinO, variants of MPDCO, have also been studied in the weighted setting. This setting is considered interesting, because MinMaxO is regarded as a special case of *minimum makespan* or *scheduling on unrelated parallel machines* ($R||C_{max}$ in the now-standard notation). In passing, a polynomial time 2-approximation algorithm for the general $R||C_{max}$ and its 3/2 inapproximability are shown in [21], and the inapproximability bound holds even for the restricted case, MinMaxO [4].

Acknowledgement

This study is partially supported by KAKENHI grant number 21680001, 22650004, 22700019, 23500020 and 23700011.

References

[1] R. K. Ahuja, D. S. Hochbaum, and J. B. Orlin. Solv-

ing the convex cost integer dual network flow problem. *Management Science*, 49(7):950–964, 2003.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[3] Y. Asahiro, J. Jansson, E. Miyano, and H. Ono. Graph orientation to maximize the minimum weighted outdegree. *Int. J. Found. Comput. Sci.*, 22(3):583–601, 2011.

[4] Y. Asahiro, J. Jansson, E. Miyano, H. Ono, and K. Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *J. Comb. Optim.*, 22(1):78–96, 2011.

[5] Y. Asahiro, E. Miyano, H. Ono, and K. Zenmyo. Graph orientation algorithms to minimize the maximum outdegree. *Int. J. Found. Comput. Sci.*, 18(2):197–215, 2007.

[6] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

[7] G. S. Brodal and R. Fagerberg. Dynamic representation of sparse graphs. In *WADS*, pages 342–351, 1999.

[8] M. Chlebík and J. Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354(3):320–338, 2006.

[9] M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theor. Comput. Sci.*, 86(2):243–266, 1991.

[10] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, pages 439–485, 2005.

[11] A. Frank. *Connections in Combinatorial Optimization*. Oxford University Press, USA, 2011.

[12] A. Frank and A. Gyárfás. How to orient the edges of a graph? *Combinatorics*, I:353–364, 1978.

[13] H. N. Gabow. Upper degree-constrained partial orientations. In *SODA*, pages 554–563, 2006.

[14] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989.

[15] S. L. Hakimi. On the degree of the vertices of a directed graph. *J. Franklin Institute*, 279:290–308, 1965.

[16] T. Ito, Y. Miyamoto, H. Ono, H. Tamaki, and R. Uehara. Route-enabling graph orientation problems. In *ISAAC*, pages 403–412, 2009.

[17] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. *J. Comput. Syst. Sci.*, 74:335–349, 2008.

[18] L. Kowalik. Approximation scheme for lowest outdegree orientation and graph density measures. In *ISAAC*, pages 557–566, 2006.

[19] H. Landau. On dominance relations and the structure of animal societies: Iii the condition for a score structure. *Bulletin of Mathematical Biology*, 15(2):143–148, 1953.

[20] L. Lau, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization (Cambridge Texts in Applied Mathematics)*. Cambridge Univ. Press, 2011.

[21] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

[22] C. Nash-Williams. On orientations, connectivity and odd vertex pairings in finite graphs. *Canad. J. Math*, 12:555–567, 1960.

[23] H. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939.

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

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