# Graph Orientation with Edge Modifications

Yuichi Asahiro[1](✉), Jesper Jansson[2], Eiji Miyano[3], Hirotaka Ono[4], and Sandhya T. P.[2]

[1] Department of Information Science, Kyushu Sangyo University, Fukuoka, Japan
`asahiro@is.kyusan-u.ac.jp`

[2] The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
`jesper.jansson@polyu.edu.hk`, `csstp@comp.polyu.edu.hk`

[3] Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka, Fukuoka, Japan
`miyano@ces.kyutech.ac.jp`

[4] Graduate School of Informatics, Nagoya University, Nagoya, Japan
`ono@i.nagoya-u.ac.jp`

**Abstract.** The goal of an *outdegree-constrained edge-modification problem* is to find a spanning subgraph or supergraph $H$ of an input undirected graph $G$ such that either: *(Type I)* the number of edges in $H$ is minimized or maximized and $H$ can be oriented to satisfy some specified constraints on the vertices' resulting outdegrees; or: *(Type II)* the maximum or minimum outdegree of all vertices under an optimal orientation of $H$ is minimum or maximum among all subgraphs or supergraphs of $G$ that can be constructed by deleting or inserting a fixed number of edges. This paper introduces eight new outdegree-constrained edge-modification problems related to load balancing called *(Type I)* MIN-DEL-MAX, MIN-INS-MIN, MAX-INS-MAX, and MAX-DEL-MIN and *(Type II)* $p$-DEL-MAX, $p$-INS-MIN, $p$-INS-MAX, and $p$-DEL-MIN. We first present a framework that provides algorithms for solving all eight problems in polynomial time on unweighted graphs. Next we investigate the inapproximability of the edge-weighted versions of the problems, and design polynomial-time algorithms for six of the problems on edge-weighted trees.

**Keywords:** Graph orientation · Maximum flow · Computational complexity · Inapproximability · Greedy algorithm · Load balancing

## 1 Introduction

Graph modification problems are fundamental in graph theory and arise in many theoretical and practical settings, including computational biology [15] and machine learning [6]. Given a weighted or unweighted graph $G = (V, E)$ and a graph property $\Pi$, the general objective is to transform the graph $G$ into a graph $G'$ satisfying property $\Pi$ by applying a shortest sequence of graph modification operations. There are two main types of graph modification problems: *vertex-modification problems* and *edge-modification problems.* In the former, one is allowed to add or remove vertices from the graph, while in the latter, the goal is typically to find a spanning subgraph or supergraph of $G$ satisfying property $\Pi$.

A special case of edge-modification problems is when the property $\Pi$ depends on the vertices' degrees. Such *degree-constrained edge-modification* problems are very general and include many natural problems such as the Maximum Weight Perfect Matching, Maximum $r$-Factor, Longest Cycle, and General Factor problems. Indeed, one can regard Maximum Weight Perfect Matching (or Maximum $r$-Factor) as the problem of finding a subgraph $G'$ of $G$ such that: (i) the degree of every vertex in $V(G')$ is one (or $r$.); and (ii) the total weight of the deleted edges is minimized. Similarly, Longest Cycle is equivalent to the problem of finding a subgraph $G'$ such that: (i) the degree of every vertex in $V(G')$ is two; (ii) $G'$ is connected; and (iii) the total weight of the deleted edges is minimized. General Factor asks if it is possible to delete edges from $G$ so that the resulting graph $G'$ is connected and every vertex $v$ in $G'$ has degree equal to a number belonging to a specified set $K(v)$. Many of these problems are NP-hard; e.g., Longest Cycle as well as General Factor are NP-hard even for unweighted graphs. Therefore, it is important to identify special cases of them that can be solved efficiently. One such special case of General Factor is the new problem MIN-DEL-MAX, introduced below.

An *orientation* of an undirected graph is an assignment of a direction to each of its edges. By an *outdegree-constrained edge-modification* problem, we mean an edge-modification problem where the solution is required to admit an orientation in which the vertices' outdegrees satisfy some specified constraints. This paper introduces eight new outdegree-constrained edge-modification problems including MIN-DEL-MAX. Besides the fact that MIN-DEL-MAX is a special case of General Factor as mentioned in the above, MIN-DEL-MAX and the other seven problems are related to load balancing which is also an important research topic. Here we explain about the relation between MIN-DEL-MAX and a load balancing problem as an example: Suppose there is a set of jobs to be completed, each job can be processed by exactly one of two specified machines, assuming that for any pair of machines at most one job is imposed, and we initially want to assign each job to a machine while minimizing the maximum load on the machines. This situation is represented by a graph with vertices interpreted as machines and edges interpreted as jobs. An orientation of such a graph corresponds to an assignment of jobs, where the start vertex (machine) of a directed edge processes the edge (job). Unfortunately, after choosing one assignment, it turns out that the maximum load is too high, so we have to give up trying to complete all of the jobs. Instead, we compute the fewest jobs to abandon in order to decrease the resulting maximum load to within some reasonable amount. This procedure corresponds to finding a smallest set of deleting edges in the above mentioned graph. The other seven problems have similar interpretations.

We first provide algorithms for solving all of the eight new problems in polynomial time on unweighted graphs. We then prove that their generalizations to edge-weighted graphs cannot be approximated within a ratio of $\rho(n)$, for instance, in polynomial time unless P = NP, where $n$ is the number of vertices in the input graph and $\rho(n) \geq 1$ is any polynomial-time computable function. This inapproximability holds even for planar bipartite graphs. Finally, as a tractable subclass of the planar bipartite graphs, we consider the problems on edge-weighted trees.

### 1.1 Problem Definitions

Let $G = (V, E)$ be a simple, undirected graph, where $V$ and $E$ are the set of vertices and the set of edges, respectively. For any $u, v \in V$, the undirected edge with endpoints in $u$ and $v$ is denoted by $\{u, v\}$, and the directed edge from $u$ to $v$ is denoted by $(u, v)$. For $G$, its complement $(V, \overline{E})$ is denoted by $G^c$, where $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$. An *orientation* $\Lambda$ of $G$ is an assignment of a direction to each edge in $E$, i.e., every $\{u, v\} \in E$ is set to either $(u, v)$ or $(v, u)$. (Equivalently, $\Lambda$ is a set of directed edges that consists of exactly one of the two directed edges $(u, v)$ and $(v, u)$ for every $\{u, v\} \in E$.) Let $\Lambda(G)$ denote the directed graph $(V, \Lambda)$. For any $v \in V$ and a fixed orientation $\Lambda$ of $G$, define $d^+(v)$ as the outdegree of $v$ under $\Lambda$. Finally, let $\Gamma(G)$ be the set of all orientations of $G$.

We now define the first four new graph orientation problems. Each of them takes as input a simple, undirected graph $G = (V, E)$ and a positive integer $k'$.

- MIN-DEL-MAX: (Assumes w.l.o.g. that $k' \leq \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$.) Find the minimum number of edges whose deletion results in a graph $G'$ with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.
- MIN-INS-MIN: (Assumes w.l.o.g. that $k' \geq \max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u)$.) Find the minimum number of edges whose addition results in a simple graph $G'$ with $\max_{\Lambda \in \Gamma(G')} \min_{u \in V} d^+(u) \geq k'$.
- MAX-INS-MAX: (Assumes w.l.o.g. that $k' \geq \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$.) Find the maximum number of edges whose addition results in a simple graph $G'$ with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.
- MAX-DEL-MIN: (Assumes w.l.o.g. that $k' \leq \max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u)$.) Find the maximum number of edges whose deletion results in a graph $G'$ with $\max_{\Lambda \in \Gamma(G')} \min_{u \in V} d^+(u) \geq k'$.

Observe that in the problems MIN-INS-MIN and MAX-INS-MAX, the resulting graphs must be simple. The above four problems optimize the number of edges to delete or insert; we also define four related problems that take as input a simple, undirected graph $G = (V, E)$ and whose objectives are to optimize the maximum/minimum outdegree for a fixed integer $p$ representing the number of deleted/inserted edges:

- $p$-DEL-MAX: Find the smallest possible value of $\min\limits_{\Lambda \in \Gamma(G(V,E \setminus E'))} \max\limits_{u \in V} d^+(u)$ taken over all $E' \subseteq E$ with $|E'| = p$.
- $p$-INS-MAX: Find the smallest possible value of $\min\limits_{\Lambda \in \Gamma(G(V,E \cup E'))} \max\limits_{u \in V} d^+(u)$ taken over all $E' \subseteq E(G^c)$ with $|E'| = p$.
- $p$-INS-MIN: Find the largest possible value of $\max\limits_{\Lambda \in \Gamma(G(V,E \cup E'))} \min\limits_{u \in V} d^+(u)$ taken over all $E' \subseteq E(G^c)$ with $|E'| = p$.
- $p$-DEL-MIN: Find the largest possible value of $\max\limits_{\Lambda \in \Gamma(G(V,E \setminus E'))} \min\limits_{u \in V} d^+(u)$ taken over all $E' \subseteq E$ with $|E'| = p$.

Throughout the paper, let $n = |V|$ and $m = |E|$ for any given instance of the above eight problems. $\Delta$ is the (unweighted) maximum degree taken over all vertices in the input $G$. Any algorithm ALG is called a $\sigma$-approximation algorithm if the following inequality holds for every input graph $G$: $\max\left\{\frac{\#ALG(G)}{\#OPT(G)}, \frac{\#OPT(G)}{\#ALG(G)}\right\} \leq \sigma$, where $\#ALG(G)$ and $\#OPT(G)$ are the number of deleted (or inserted) edges by ALG and an optimal algorithm, respectively.

## 1.2 Related Work

To compute $\min\limits_{\Lambda \in \Gamma(G)} \max\limits_{u \in V} d^+(u)$ for an input undirected, unweighted graph $G$ is the Minimum Maximum Outdegree Problem (MinMaxO), previously studied in [4,7,8,10,16]. MinMaxO can be solved in linear time for planar graphs [10] and in polynomial time for general graphs [16]. The problem of computing $\max\limits_{\Lambda \in \Gamma(G)} \min\limits_{u \in V} d^+(u)$, referred to as MaxMinO in [2], can also be solved in polynomial time (Theorem 8 in [2]). (This is why the input $k'$ to MIN-DEL-MAX, MIN-INS-MIN, MAX-INS-MAX, and MAX-DEL-MIN can be assumed w.l.o.g. to satisfy $k' \leq \min\limits_{\Lambda \in \Gamma(G)} \max\limits_{u \in V} d^+(u)$ or $k' \geq \max\limits_{\Lambda \in \Gamma(G)} \min\limits_{u \in V} d^+(u)$.) When generalized to edge-weighted graphs, both MinMaxO and MaxMinO become NP-hard [2,4].

Theorem 8 in [2] states that MaxMinO is solvable in $O(m^{3/2} \log m \log^2 \Delta)$ time for unweighted graphs, which directly gives:

**Theorem 1.** MAX-DEL-MIN *can be solved in* $O(nm^{3/2} \log m \log^2 \Delta)$ *time.*

*Proof.* First compute an orientation by which the minimum outdegree has value $\max\limits_{\Lambda \in \Gamma(G)} \min\limits_{u \in V} d^+(u) (\geq k')$ using the algorithm for MaxMinO from [2], and obtain a directed graph. Then delete arbitrary $d^+(v) - k'$ outgoing edges for each vertex $v$ in the directed graph to get a directed graph $G'$ with $d^+(v) = k'$ for every $v \in V$. This deletion of edges only needs linear time since we can delete arbitrary set of outgoing edges for each vertex. The number of deleted edges is the maximum

**Table 1.** The computational complexity of the algorithms in Sect. 2 for unweighted graphs. For edge-weighted graphs all these problems are intractable, shown in Sect. 3.

| Problem | Time complexity | Reference |
|---|---|---|
| MIN-DEL-MAX | $O(m^2 \log n)$ | Theorem 3 |
| MIN-INS-MIN | $O(n^4 \log n)$ | Theorem 4 |
| MAX-INS-MAX | $O(n^4 \log n)$ | Theorem 4 |
| MAX-DEL-MIN | $\min\{O(nm^{3/2} \log m \log^2 \Delta)$, $O(m^2 \log n)\}$ | Theorems 1 and 3 |
| $p$-DEL-MAX | $O(m^2 \log n)$ | Theorem 3 |
| $p$-INS-MIN | $O(n^4 \log n)$ | Theorem 4 |
| $p$-INS-MAX | $O(n^4 \log n)$ | Theorem 4 |
| $p$-DEL-MIN | $O(m^2 \log n)$ | Theorem 3 |

possible: Since every vertex has outdegree $k'$, the number of the directed edges is $nk'$ in the graph, and so deleting any more edges would result in some vertex having outdegree strictly less than $k'$ by the pigeonhole principle. Thus MAX-DEL-MIN can be solved in $O(nm^{3/2} \log m \log^2 \Delta)$ time. □

A variant of MinMaxO in which one may perform $p$ *split operations* on the vertices (corresponding to adding $p$ extra machines in the load balancing setting described above) before orienting the edges was studied in [1]. That problem seems harder than the eight problems studied here, as it is NP-hard even for unweighted graphs when $p$ is unbounded [1].

### 1.3 Our Contributions and Organization of the Paper

Section 2 presents polynomial-time algorithms for the new problems on unweighted graphs. See Table 1 for a summary of their computational complexity. In Sect. 3 we develop polynomial-time algorithms for six of the eight problems on edge-weighted trees, and prove the polynomial-time inapproximability for planar bipartite edge-weighted graphs for all eight problems. Finally, we conclude the paper in Sect. 4. Due to space constraints, some proofs are deferred to the full version of this paper.

## 2 Unweighted Graphs

In this section, we present polynomial-time algorithms for the unweighted versions of the new problems. Rather than developing a separate algorithm for each problem, our strategy is to give a unified framework from which each of the eight algorithms follows as a special case. We take advantage of the problems' structural similarities by encoding the input graph $G$ in all eight cases as a directed graph $N_G$, augmenting $N_G$ with edge capacities as defined below to obtain a

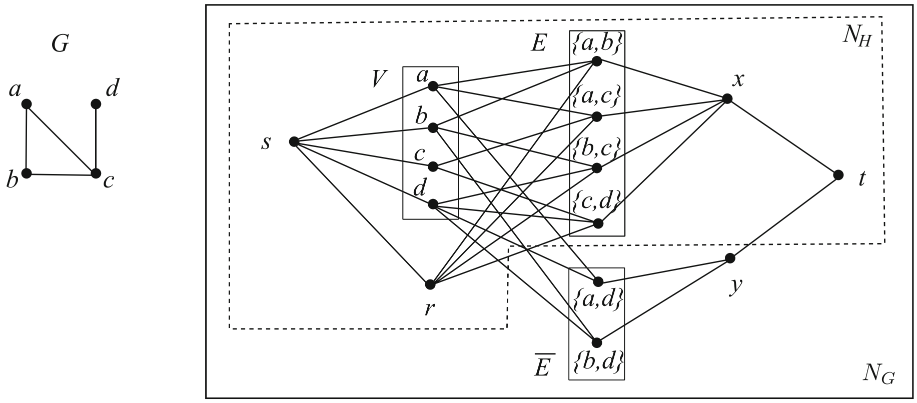

**Fig. 1.** A graph $G$ and the directed graph $N_G$ constructed from $G$. All edges are directed from left to right in $N_G$. The part surrounded by the dashed lines represents $N_H$.

flow network, and then using binary search together with a fast algorithm for computing maximum flows. $N_G$ is the same for all problems; only the edge capacities in the flow network depend on which of the problems is being solved. The encoding used here is an extension of the one in [4]; to be precise, the definition of $N_G$ follows the basic construction in [4] and then adds auxiliary vertices and directed edges that can capture the deletion and insertion of edges in the input graph.

The formal definition of $N_G$ is as follows. For any undirected graph $G(V, E)$, construct the directed graph $N_G = (V_G, E_G)$ with vertex set $V_G$ and edge set $E_G$ by defining: $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$, $V_G = V \cup E \cup \overline{E} \cup \{x, y, r, s, t\}$, $E_G = \{(s, v) \mid v \in V\} \cup \{(v_i, e), (v_j, e) \mid e = \{v_i, v_j\} \in E \cup \overline{E}\} \cup \{(r, e) \mid e \in E\} \cup \{(e, x) \mid e \in E\} \cup \{(e, y) \mid e \in \overline{E}\} \cup \{(s, r), (x, t), (y, t)\}$. Note that if $e = \{u, v\} \in E$ (or $\overline{E}$), then $N_G$ contains two directed edges $(u, e)$ and $(v, e)$ for $e$ in the vertex subset $E$ (or $\overline{E}$) of $N_G$. Note that the vertex $r$ and the set of vertices in $\overline{E}$ capture deletion and insertion of edges respectively, mentioned in the previous paragraph. See Fig. 1 for an illustration. We remark that maximum flows in suitably defined flow networks were previously used to solve some other graph orientation problems in [1–4]. The definition of $N_G$ that we present here is more general.

Section 2.1 below explains how to use $N_G$ to solve the problems MIN-DEL-MAX and $p$-DEL-MAX. Due to the space limitation, discussions for the other six problems are omitted. Then, Sect. 2.2 analyzes the time complexity of the obtained algorithms. To solve MIN-INS-MIN, $p$-INS-MIN, MAX-INS-MAX, and $p$-INS-MAX, we need to explicitly construct the entire directed graph $N_G$. However, for MIN-DEL-MAX, $p$-DEL-MAX, MAX-DEL-MIN, and $p$-DEL-MIN, the algorithms will only need the induced subgraph $N_H = N_G[V \cup E \cup \{s, r, x, t\}] = N_G \setminus (\overline{E} \cup \{y\})$ of $N_G$, and so these algorithms' running times will be lower when $G$ is sparse.

### 2.1 Using $N_G$ to Solve the Problems

First we focus on the problem MIN-DEL-MAX. For an undirected graph $G(V, E)$ with $k = \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$ and a positive integer $k' \leq k$, construct the directed graph $N_H = N_G[V \cup E \cup \{s, r, x, t\}]$. For any positive integer $p$, let $N_H(p) = (V(N_H), E(N_H), cap)$ be the flow network obtained by augmenting $N_H$ with edge capacities $cap$, where:

$$cap(a) = \begin{cases} k', \text{ if } a = (s, v) \text{ for } v \in V \\ p, \text{ if } a = (s, r) \\ |E|, \text{ if } a = (x, t) \\ 1, \text{ otherwise} \end{cases}$$

The following lemma relates the size of the maximum flow in $N_H(p)$ to the number of deleted edges from $G$.

**Lemma 2.** *For an input graph $G(V, E)$ to* MIN-DEL-MAX*, there exists a flow in $N_H(p)$ with value $|E|$ if and only if there are exactly $p$ edges in $G$ whose deletion leaves a graph $G'$ with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.*

*Proof.* Suppose there exists a flow for the network $N_H(p)$ with value $|E|$. Since the edge capacities are integers, we can assume that the flow has nonnegative integral flows in each edge by the integrality theorem (see, e.g., [11]). Let $j$ be the size of the flow through the vertex $r$ and $j \leq p$. We consider those $j$ edges of the form $(r, e)$, incident to $r$, through which the flow passes by. Let $G'$ be the graph obtained by deleting those edges from $G$ (If $j < p$, then delete $p - j$ arbitrary edges from $G$ in order to ensure that exactly $p$ edges are deleted from $G$). Then, construct an orientation of $G'$ as follows. For every edge $f = (v, e)$ in $N_H(p)$ that contributes a unit of flow, where $v \in V$ and $e \in E$, orient the edge $e$ away from $v$. Since every vertex $v \in V$ can have at most $k'$ flow, $\max_{u \in V} d^+(u) \leq k'$ and hence $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.

Conversely, if there are $p$ edges whose deletion leaves a graph $G'$ with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$, then construct a flow with value $|E|$ as follows. Let $S = E \backslash E(G')$ and let $\Lambda$ be a fixed orientation of $G'$ that minimizes the maximum outdegree. For $e \in S$, send one unit of flow from $s$ to $r$, $r$ to $e$, $e$ to $x$, and then $x$ to $t$, and the total flow is $p$. If $e \notin S$, then send one unit of flow from $s$ to $v$, $v$ to $e$, $e$ to $x$, and then $x$ to $t$, where $e$ is oriented away from $v$ in $\Lambda$. For $\Lambda$, $\max_{u \in V} d^+(u) \leq k'$ and thus $d^+(u) \leq k'$, for every $u \in V$. Hence, through every vertex $v \in N_H(p)$ that corresponds to $v \in V$, at most $k'$ units of flow pass, which is the capacity of the vertex $v$. In that way, every edge in $G$ contributes one unit of flow and hence this particular flow of $N_H(p)$ has value $|E|$. □

By Lemma 2, the minimum number of edges that can be deleted to get a graph $G'$ with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$ is the same as the smallest value of $p$

```
1  $p_{\min} \leftarrow 0$, $p_{\max} \leftarrow |E|$;
2  repeat
3      $p \leftarrow \lfloor (p_{\min} + p_{\max})/2 \rfloor$;
4      Construct $N_H(p)$ and let $f$ be the value of the maximum flow of $N_H(p)$ ;
5      if $f = |E|$ then
6          $p_{\max} \leftarrow p$;
7      else
8          $p_{\min} \leftarrow p$;
9      end
10 until $p_{\min} \geq p_{\max}$;
11 Output $p$ and halt;
```

**Fig. 2.** The algorithm for MIN-DEL-MAX on unweighted graphs

such that there exists a flow in $N_H(p)$ of value $|E|$. A binary search on $p$ will give the minimum value of $p$ for which there exists a flow of size $|E|$ in $N_H(p)$. The algorithm's pseudocode is listed in Fig. 2.

As for $p$-DEL-MAX, given an undirected graph $G(V, E)$ with $k = \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$, build the directed graph $N_H = N_G[V \cup E \cup \{s, r, x, t\}]$. For any positive integer $\ell$, let $N_H(\ell) = (V(N_H), E(N_H), cap)$ be the flow network obtained by augmenting $N_H$ with edge capacities $cap$, where:

$$cap(a) = \begin{cases} \ell, & \text{if } a = (s, v) \text{ for } v \in V \\ p, & \text{if } a = (s, r) \\ |E|, & \text{if } a = (x, t) \\ 1, & \text{otherwise} \end{cases}$$

By Lemma 2, the minimum value of the maximum outdegree is the same as the minimum value of $\ell$ such that there exists a flow of value $|E|$ in $N_H(\ell)$. Hence, by an algorithm similar to the one for MIN-DEL-MAX, for any $\ell$, $0 \leq \ell \leq k$, we can check whether there is a flow of value $|E|$ and locate the smallest integer $\ell$ with this property by a binary search on $\ell$.

## 2.2 Time Complexity of the Algorithms

Let $n = |V|$ and $m = |E|$ for a given graph $G = (V, E)$, and let $N = |V(N_H)|$ and $M = |E(N_H)|$. We note that for the directed graph $N_H$, $N = n + m + 4$ and $M = n + 4m + 2$. For MIN-DEL-MAX, the search for $p = O(m)$ can be carried out using binary search, which takes $O(\log p) = O(\log m) = O(\log n)$ time since $m = O(n^2)$. The maximum flow problem on $N_H(p)$ can be solved in $O(MN)$ time [14], so MIN-DEL-MAX can also be solved in $O(m^2 \log n)$ time. Since $\ell = O(n)$, in a similar way, we can analyze the time complexity of our algorithms for $p$-DEL-MAX, MAX-DEL-MIN, and $p$-DEL-MIN. Thus we have:

**Theorem 3.** *For unweighted graphs,* MIN-DEL-MAX*,* $p$-DEL-MAX*,* MAX-DEL-MIN*, and* $p$-DEL-MIN *can be solved in* $O(m^2 \log n)$ *time.*

As shown in Theorem 1, MAX-DEL-MIN can be solved in $O(nm^{3/2} \log m \log^2 \Delta)$ time. The above algorithm is faster than it when $m = \Omega(n^2)$ and so $\Delta = \Omega(n)$, that is, the input graph is very dense.

On a similar note, for the directed graph $N_G$, $N = |V_G| = \frac{n(n+1)}{2} + 5$ and $M = |E_G| = \frac{n(3n-1)}{2} + m + 3$. For MIN-INS-MIN, the search for $p = O(m)$ can be carried out using binary search and therefore takes $O(\log p) = O(\log m) = O(\log n)$ time. The maximum flow problem on $N_G(p)$ can be solved in $O(MN)$ time [14], so MIN-INS-MIN can also be solved in $O(n^4 \log n)$ time. Using $\ell = O(m)$, we can bound the time complexity of our algorithms for $p$-INS-MIN, MAX-INS-MAX, and $p$-INS-MAX in the same way. We obtain:

**Theorem 4.** *For unweighted graphs,* MIN-INS-MIN*,* $p$-INS-MIN*,* MAX-INS-MAX*, and* $p$-INS-MAX *can be solved in* $O(n^4 \log n)$ *time.*

## 3 Edge-Weighted Graphs

For the problems in this paper, we can define corresponding weighted versions. For the weighted versions of the problems, $d^+(v)$ for every vertex $v$ and a fixed orientation represents the total weight of outgoing edges of $v$. Each of MIN-DEL-MAX, MAX-DEL-MIN, MIN-INS-MIN, and MAX-INS-MAX takes as input a simple undirected edge-weighted graph $G = (V, E, w)$ and the target maximum/minimum outdegree $k'$, where the function $w$ assigns a positive integer (weight) to each edge. Then the objective of these problems is still to optimize the number of edges to delete/insert. (We do not optimize the total weight of deleted/inserted edges.) $p$-DEL-MAX, $p$-DEL-MIN, $p$-INS-MIN, and $p$-INS-MAX are similarly defined on edge-weighted graphs, where we need to delete/insert $p$ edges for the problems.

### 3.1 Polynomial-Time Algorithms for Edge-Weighted Trees

Assuming $P \neq NP$, inapproximability of the weighted versions of the problems on planar bipartite graphs will be shown in Sect. 3.2. In this section, we design exact polynomial-time algorithms for some of the problems on weighted trees which is a representative subclass of planar bipartite graphs.

The important thing here is that we know the optimal costs for MinMaxO and MaxMinO on edge-weighted trees: For MinMaxO, the maximum outdegree of a vertex under any orientation is at least the maximum weight of the edges. Then, for MaxMinO, the minimum outdegree of a vertex under any orientation is 0, since there exist only $n - 1$ edges so that at least one of $n$ vertices cannot have outgoing edges under any orientation. From this observation, an optimal orientation for edge-weighted trees is to orient all the edges towards an arbitrarily selected root vertex. Based on these observations, straightforward discussion gives the following lemma (the proof has been omitted here due to space limitations):

**Lemma 5.** *For edge-weighted trees,* MAX-DEL-MIN *and* $p$-DEL-MIN *can be solved in* $O(n)$ *time.*

In conjunction with the above lemma, we can show the next theorem. Although the above lemma is obtained as a direct consequence of the known results for MaxMinO, we need to design more complicated algorithms for MAX-DEL-MIN, $p$-DEL-MIN, MIN-INS-MIN, and $p$-INS-MIN (the proof has been omitted here due to space limitations).

**Theorem 6.** *For edge-weighted trees,* MIN-DEL-MAX*,* $p$-DEL-MAX*,* MAX-DEL-MIN*,* $p$-DEL-MIN*, and* MIN-INS-MIN *are solvable in* $O(n)$ *time. Also,* $p$-INS-MIN *is solvable in* $O(n\log(w_{\max}\Delta))$ *time, where* $w_{\max}$ *is the maximum weight of edges and* $\Delta$ *is the maximum (unweighted) degree of vertices.*

## 3.2 Inapproximability for Edge-Weighted Planar Bipartite Graphs

MinMaxO is known to be NP-hard for edge-weighted planar bipartite graphs [5]. This implies the following inapproximability:

**Theorem 7.** *There is no polynomial-time* $\rho(n)$*-approximation algorithm for* MIN-DEL-MAX *on edge-weighted planar bipartite graphs unless P = NP, where* $\rho(n) \geq 1$ *is any polynomial-time computable function.*

*Proof.* Suppose for the sake of obtaining a contradiction that there exists a polynomial-time $\rho(n)$-approximation algorithm `ALG` for some polynomial-time computable function $\rho(n) > 1$ for MIN-DEL-MAX on edge-weighted planar bipartite graphs. Then, `ALG` can find an orientation in a given edge-weighted graph $G$ in polynomial time such that the objective value $ALG(G)$ satisfies $OPT(G) \leq ALG(G) \leq \rho(n) \cdot OPT(G)$. Therefore, one can distinguish either $OPT(G) > 0$ or $OPT(G) = 0$ in polynomial time using `ALG` which admits the approximation ratio of $\rho(n)$, based on the observation that $ALG(G) > 0$ if and only if $OPT(G) > 0$. If $OPT(G) = 0$, then there is no need to remove any edge to make the outdegree of every vertex at most $k$, whereas if $OPT(G) \geq 1$, we need to remove at least one edge. This means that a decision version of MinMaxO with target value $k$ can be solved in polynomial time, which contradicts the NP-hardness of MinMaxO on edge-weighted planar bipartite graphs. □

The inapproximability bound 1.5 of MinMaxO for edge-weighted planar bipartite graphs gives the next theorem.

**Theorem 8.** *There is no polynomial-time 1.5-approximation algorithm for* $p$-DEL-MAX *on edge-weighted planar bipartite graphs unless P = NP.*

*Proof.* Consider an input planar bipartite graph $G$ of MinMaxO. Add one new vertex $u$ and one edge $\{u, v\}$ of weight $n \cdot w_{\max}$ to $G$ for arbitrary $v \in V(G)$, where $w_{\max}$ is the maximum weight of edges in $G$. Let this new graph be $G'$, where $G'$ is also a planar bipartite graph. Observe that for 1-DEL-MAX, this new edge should be deleted since otherwise the maximum outdegree is at least $nw_{\max}$, which is larger than the total weight of edges in $G$, and then we need to orient the edges in $G$ optimally. Namely, if we can approximate 1-DEL-MAX within a ratio 1.5 for $G'$ in polynomial-time, it also gives the answer to MinMaxO for $G$. This contradicts the inapproximability of MinMaxO, so that 1-DEL-MAX cannot be approximated within a ratio of 1.5 in polynomial-time. This discussion can be extended to the case $p \geq 2$ by adding $p$ new vertices and $p$ new edges of weight $n \cdot w_{\max}$ to $G$. The theorem follows. $\square$

The NP-hardness and the inapproximability bounds of MaxMinO and MinMaxO for edge-weighted planar bipartite graphs [2,4] can be applied in the same way as Theorems 7 and 8 to obtain the following theorem (the proof has been omitted here due to space limitations):

**Theorem 9.** *There is no polynomial-time $\rho(n)$($\rho(n)$, $\rho(n)$, 2, 1.5, 2, resp.)-approximation algorithm for* MIN-INS-MIN*(*MAX-INS-MAX*,* MAX-DEL-MIN*,* $p$-INS-MIN*,* $p$-INS-MAX*,* $p$-DEL-MIN*, resp.) on edge-weighted planar bipartite graphs unless $P = NP$, where $\rho(n) \geq 1$ is any polynomial-time computable function.*

## 4 Concluding Remarks

We studied eight new graph orientation problems whose objective is to minimize/maximize the outdegree of the vertices after inserting/deleting edges, and presented polynomial-time algorithms for these problems on unweighted graphs. Also we showed the polynomial-time inapproximability for those problems on edge-weighted graphs, and polynomial-time algorithms for six of the problems were designed on edge-weighted trees. One of the further research topics is to study the complexity of MAX-INS-MAX and $p$-INS-MAX on edge-weighted trees. A natural generalization of MIN-DEL-MAX can be defined as follows:

**Input:** An unweighted graph $G = (V, E)$ and a mapping $\rho$ that assigns to each vertex $v \in V$ an integer from $\{0, 1, \ldots, \deg(v)\}$.
**Goal:** To find the minimum number of edges to delete to get a spanning subgraph $H$ of $G$ such that $d^+(u) \leq \rho(u)$ for every $u \in V$.

To solve this problem, we can first construct a directed graph $N_G$ in the same way as in Sect. 2. Next, we augment edge capacities and costs to $N_G$ to get a flow network by keeping the capacities and costs same as that of $N_G$ for $k'$ except the capacities of the directed edges of the form $\{(s, u) \mid u \in V\}$. The capacity of the directed edge $(s, u)$ is defined to be $\rho(u)$ instead of $k'$, for every $u \in V$. Then we see that there exists a flow in $N_G$ with value $|E|$ if and only if there are $p$ edges in $G$ whose deletion leaves a graph $d^+(u) \leq \rho(u)$, for every vertex $u \in V(G)$. In other words, the modified problem can be solved in polynomial time. In fact, both the modified problem and the original MIN-DEL-MAX have the same time complexity since the directed graphs that we construct in both cases are the same. We can generalize MIN-INS-MIN, MAX-INS-MAX, MAX-DEL-MIN in the same way and solve them in polynomial time as well.

The above problem is a special case of General Factor introduced by Lovász [12,13], which is defined as

**Input:** An unweighted graph $G = (V, E)$ and a mapping $K$ that assigns to each vertex $v \in V$ a set $K(v) \subseteq \{0, 1, \ldots, \deg(v)\}$ of integers.
**Goal:** To check if there is a subgraph $H$ of $G$ s.t. $d_H(v) \in K(v)$ for every $v \in V$.

General Factor is a generalization of the factor problem, and NP-hard even for unweighted graphs [9]. Here the extension to the mapping from an integer to a set of integers makes the problem harder. We conjecture that the analogous generalizations to the problems in this paper are NP-hard as well.

**Acknowledgments.** This work was partially supported by JSPS KAKENHI Grant Numbers JP17K00016 and JP17K00024, and JST CREST JPMJR1402.

## References

1. Asahiro, Y., Jansson, J., Miyano, E., Nikpey, H., Ono, H.: Graph orientation with splits. In: Lee, J., Rinaldi, G., Mahjoub, A.R. (eds.) ISCO 2018. LNCS, vol. 10856, pp. 52–63. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96151-4_5
2. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Graph orientation to maximize the minimum weighted outdegree. Int. J. Found. Comput. Sci. **22**(3), 583–601 (2011)
3. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Graph orientations optimizing the number of light or heavy vertices. J. Graph Algorithms Appl. **19**(1), 441–465 (2015)
4. Asahiro, Y., Jansson, J., Miyano, E., Ono, H., Zenmyo, K.: Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. J. Comb. Optim. **22**(1), 78–96 (2011)
5. Asahiro, Y., Miyano, E., Ono, H.: Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. Discrete Appl. Math. **159**, 498–508 (2011)
6. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Mach. Learn. **56**, 89–113 (2004)
7. Borradaile, G., Iglesias, J., Migler, T., Ochoa, A., Wilfong, G., Zhang, L.: Egalitarian graph orientations. J. Graph Algorithms Appl. **21**(4), 687–708 (2017)
8. Brodal, G.S., Fagerberg, R.: Dynamic representations of sparse graphs. In: Dehne, F., Sack, J.-R., Gupta, A., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 342–351. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48447-7_34
9. Cornuéjols, G.: General factors of graphs. J. Comb. Theory Ser. B **45**, 185–198 (1988)
10. Chrobak, M., Eppstein, D.: Planar orientations with low out-degree and compaction of adjacency matrices. Theor. Comput. Sci. **86**(2), 243–266 (1991)
11. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press, Cambridge (2009)
12. Lovász, L.: The factorization of graphs. In: Combinatorial Structures and Their Applications, pp. 243–246 (1970)
13. Lovász, L.: The factorization of graphs II. Acta Math. Acad. Sci. Hung. **23**, 223–246 (1972)

14. Orlin, J.B.: Max flows in $O(nm)$ time, or better. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013), pp. 765–774. Association for Computing Machinery (ACM) (2013)
15. Sharan, R.: Graph modification problems and their applications to genomic research. Ph.D. thesis, School of Computer Science, Tel-Aviv University (2002)
16. Venkateswaran, V.: Minimizing maximum indegree. Discrete Appl. Math. **143**, 374–378 (2004)