

Graph orientation algorithms to minimize the maximum outdegree

Asahiro, Yuichi

Department of Social Information Systems, Kyushu Sangyo University

Miyano, Eiji

Department of Systems Innovation and Informatics, Kyushu Institute of Technology

Ono, Hirotaka

Department of Computer Science and Communication Engineering, Kyushu University

Zenmyo, Kouhei

Department of Systems Innovation and Informatics, Kyushu Institute of Technology

<https://hdl.handle.net/2324/14869>

出版情報 : International Journal of Foundations of Computer. 18 (2), pp.197-215, 2007-04. World Scientific

バージョン :

権利関係 : Electronic version of an article published as International Journal of Foundations of Computer , 18, 2, p197-215 10.1142/S0129054107004644 © World Scientific Publishing Company
<http://www.worldscinet.com/ijfcs/ijfcs.shtml>



GRAPH ORIENTATION ALGORITHMS TO MINIMIZE THE MAXIMUM OUTDEGREE

YUICHI ASAHIRO

*Department of Social Information Systems, Kyushu Sangyo University,
Fukuoka 813-8503, Japan. asahiro@is.kyusan-u.ac.jp*

EIJI MIYANO

*Department of Systems Innovation and Informatics, Kyushu Institute of Technology,
Fukuoka 820-8502, Japan. miyano@ces.kyutech.ac.jp*

HIROTAKA ONO

*Department of Computer Science and Communication Engineering, Kyushu University,
Fukuoka 812-8581, Japan. ono@csce.kyushu-u.ac.jp*

and

KOUHEI ZENMYO

*Department of Systems Innovation and Informatics, Kyushu Institute of Technology,
Fukuoka 820-8502, Japan. kouhei@theory.ces.kyutech.ac.jp*

Received (received date)

Revised (revised date)

Communicated by Editor's name

ABSTRACT

This paper studies the problem of orienting all edges of a weighted graph such that the maximum weighted outdegree of vertices is minimized. This problem, which has applications in the guard arrangement for example, can be shown to be \mathcal{NP} -hard generally. In this paper we first give optimal orientation algorithms which run in polynomial time for the following special cases: (i) the input is an unweighted graph, and (ii) the input graph is a tree. Then, by using those algorithms as sub-procedures, we provide a simple, combinatorial, $\min\{\frac{w_{max}}{w_{min}}, (2 - \varepsilon)\}$ -approximation algorithm for the general case, where w_{max} and w_{min} are the maximum and the minimum weights of edges, respectively, and ε is some small positive real number that depends on the input.

Keywords: graph orientation, min-max optimization, \mathcal{NP} -hardness, approximation algorithms.

1. Introduction

1.1. Brief History of Graph Orientation

Let $G = (V, E, w)$ be a simple, undirected, weighted graph with a vertex set V , an edge set E , and a positive integral weight function $w : E \rightarrow \mathbb{Z}^+$, where each edge is a pair $\{u, v\}$ of vertices $u, v \in V$. An *orientation* Λ of the graph G is an assignment of direction to each edge $\{u, v\} \in E$. The graph orientation is a well-studied area in the fields of graph theory and combinatorial optimization, and has a long history. In 1939, Robbins [17] stated a seminal result on the relation between the *connectivity* of a graph and its orientation: A graph has a strongly connected

orientation if and only if it is 2-edge-connected. Thereafter, a variety of classes of questions have been introduced and investigated in the literature, including the characterization of oriented graphs satisfying the specified connectivity, and the problem of finding orientations with topological properties such as the *tightness* (an orientation of G whose diameter is the same as the diameter of G is called tight [12]), the *degree constraint* and the *acyclicity*. For example, as a classical result, Nash-Williams [14] characterized graphs having k -edge-connected orientations. Chung, Garey and Tarjan [3] provided a linear-time algorithm for checking whether a graph has a strongly connected orientation and finding one if it does. In [5] Chvátal and Thomassen introduced the following problem called the *oriented diameter* problem: Given a graph G , find a strongly connected orientation of G with the minimum diameter. They proved that the problem is \mathcal{NP} -hard for general graphs. Then, Fomin, Matamala and Rapaport [7] showed that the problem remains \mathcal{NP} -hard even if the graph is restricted to a subclass of chordal graphs and gave approximability and inapproximability results.

The orientation with the degree constraint is also popular. Chrobak and Eppstein [2] studied the problem of orienting the edges of a planar graph such that the outdegree of each vertex is bounded, and proved that a 3-bounded outdegree orientation and a 5-bounded outdegree *acyclic* orientation can be surely constructed in linear time for every planar graph. Recently, Biedl, Chan, Ganjali, Hajiaghayi, and Wood [1] studied the problem of determining a *balanced acyclic* orientation of unweighted graphs, where balanced means that the difference between the (unweighted) indegree and outdegree of each vertex is minimized, and proved that it is \mathcal{NP} -hard and there is a $\frac{13}{8}$ -approximation algorithm. The \mathcal{NP} -hardness of Biedl et al.'s result is for graphs with maximum degree six. Kára, Kratochvíl, and Wood [11] closed the gap, by proving the \mathcal{NP} -hardness for graphs with maximum degree four, and also showed that it remains \mathcal{NP} -hard for planar graphs with maximum degree six, and so on. The orientation with the degree constraint has several applications in the fields of data structures and graph drawing as mentioned in [2, 1].

1.2. Our Problems and Results

In this paper we propose a new variant of the graph orientation by considering a natural objective function, the *Minimum Maximum Outdegree Orientation* problem (MMO for short):

MMO

Instance: A simple, weighted, undirected graph $G = (V, E, w)$.

Question: Find an orientation Λ of G which minimizes the maximum weighted outdegree of vertices.

MMO is originally motivated by the *capacitated guard arrangement* problem, which is one of the *art gallery* problems [4, 15]: The original art gallery problem for a polygon P is to find a minimum set Q of points in P such that every point of P is visible from some point in Q and to place one guard on each point in Q , $|Q|$

guards in total. If P can be viewed as a graph (a set of line segments such as a mesh) and the guards have to be placed only on its vertices (or intersections of line segments), then the art gallery problem can be straightforwardly formulated by the *vertex cover* problem, i.e., a guard placed on a vertex must watch (cover) all the edges incident with the vertex and the goal is to minimize the number of guards arranged. In the capacitated guard arrangement, guards are positioned on all vertices but they can cover only the specified number of edges, and its goal is to minimize the capacity of each guard, which is represented by MMO.

In this paper we show the following results:

- We prove that, unfortunately, MMO is generally \mathcal{NP} -hard.
- However, fortunately, we can obtain optimal orientation algorithms which run in polynomial time for the following special cases: (i) the input is an unweighted graph, or more generally, a graph with identically weighted edges, and (ii) the input graph is a tree.
- Furthermore, by using those algorithms as sub-procedures, we provide a simple, combinatorial, $\min\{\frac{w_{max}}{w_{min}}, (2 - 1/\lceil L(G) \rceil)\}$ -approximation algorithm for the general case, where w_{max} and w_{min} are the maximum and the minimum weights of edges, respectively, and $L(G)$ is a positive real number defined by the whole graph structure as described in Section 2.

Note that Venkateswaran [19] previously investigated the unweighted version of MMO, for which he also provided an $O(|E|^2)$ -time orientation algorithm. In this paper, we show that the $O(|E|^2)$ bound can be reduced.

1.3. Related and Previous Work

The difficulty of solving MMO exactly and/or approximately can be closely related to the intractability of the *minimum makespan scheduling*, which is a central problem in the scheduling area, and well studied from the point of view of approximability. In the *scheduling on unrelated parallel machines* ($R||C_{max}$ in the now-standard notation), given a set J of jobs, a set M of machines, and a time period $p_{ij} \in \mathbb{Z}^+$ taken to process a job $j \in J$ on a machine $i \in M$, its goal is to find a job scheduling so as to minimize the makespan, i.e., the maximum processing time of any machine. Lenstra, Shmoys, and Tardos [13] gave a polynomial time 2-approximation algorithm that is based on the LP-formulation for the general version of $R||C_{max}$ and its $\frac{3}{2}$ inapproximability result. Schuurman and Woeginger [18] stated that it is even interesting to improve on the results of [13] in the so-called *restricted assignment* variant of $R||C_{max}$, in which the processing time p_{ij} of job j on machine i is identically fixed p_j , but the job can only be processed on a subset of the machines. In MMO, the processing time p_j of job j corresponds to the weight $w(\{u, v\})$ of an edge $\{u, v\}$ and its assignable machines correspond to two terminals u and v . Hence, an orientation of $\{u, v\}$ is regarded as a job assignment. For the simpler problems of the restricted $R||C_{max}$, an FPTAS [10] or a polynomial time algorithm [16] were provided.

For the general cases, the algorithm of [13] gives a 2-approximation, and Gairing et al., achieves an approximation guarantee of $2 - 1/w_{max}$ [8]. The algorithms are applicable to MMO and their ratios also hold. Since the approximation ratio of our algorithm is at most $2 - 1/\lceil L(G) \rceil$, our ratio is better if $\lceil L(G) \rceil$ is smaller than w_{max} .

1.4. Organization

The rest of this paper is organized as follows. Section 2 introduces some notations. Then we prove the \mathcal{NP} -hardness of the general MMO in Section 3. In Section 4 we consider easy subclasses of MMO and provide two polynomial time algorithms for them. In Section 5 we give a new combinatorial $\min\{\frac{w_{max}}{w_{min}}, (2 - 1/\lceil L(G) \rceil)\}$ -approximation algorithm for the general MMO based on the polynomial time algorithms of Section 4. Finally, we conclude the paper in Section 6.

2. Preliminaries

Let $G = (V, E, w)$ be a simple, undirected, weighted graph, where V , E , and w denote a set of vertices, a set of edges, and an integral weight function, $w : E \rightarrow \mathbb{Z}^+$, respectively. Let w_{max} and w_{min} be the maximum and the minimum weights of edges, respectively. Throughout the paper, let $|V| = n$ and $|E| = m$ for the input graph. By $\{u, v\}$ for $u, v \in V$ we denote the undirected edge with ends in u and v , and by (u, v) the directed arc, directed from u toward v . Let $d(v)$ represent the degree of a vertex v and $D(G)$ the maximum degree of a graph G . An *orientation* Λ of the undirected graph G is an assignment of direction to each edge $\{u, v\} \in E$, i.e., (u, v) or (v, u) . Equivalently, we can regard the orientation Λ as a set of directed arcs such that Λ includes exactly one of the two arcs (u, v) and (v, u) for each $\{u, v\} \in E$. A *directed path* P of length k from a vertex v_0 to a vertex v_k in a directed graph $G = (V, A, w)$ is a set $\{(v_{i-1}, v_i) \mid (v_{i-1}, v_i) \in A, i = 1, 2, \dots, k\}$ of arcs, which is also denoted by a sequence $\langle v_0, v_1, \dots, v_k \rangle$ for simplicity. For the path P , the path of its reverse order is denoted by \overline{P} , i.e., $\overline{P} = \langle v_k, v_{k-1}, \dots, v_0 \rangle$.

We represent by $i \rightarrow j$ for two vertices i and j if an arc (i, j) is in orientation Λ . $\delta_\Lambda^+(v)$ and $\delta_\Lambda^-(v)$ under an orientation Λ denote the total weights of outgoing arcs and that of incoming arcs of a vertex v in the weighted directed graph $G(V, A, w)$, which we call the *weighted outdegree* and the *weighted indegree* of v , respectively. Let $\delta(v) = \delta_\Lambda^+(v) + \delta_\Lambda^-(v)$. (Note that $\delta(v)$ does not change depending on Λ .) Then the *cost* of an orientation Λ for a graph G is defined to be $\Delta_\Lambda(G) = \max_{v \in V} \{\delta_\Lambda^+(v)\}$.

By $G[V']$ we denote a subgraph induced by $V' \subseteq V$ for G , simply represented by $G[V'] \subseteq G$. Let $W(G) = \sum_{\{u, v\} \in E} w(\{u, v\})$ and $\ell(G) = W(G)/|V|$ be the total weight of edges and the average weighted outdegrees of vertices in G , respectively. Also, as for all the induced subgraphs H 's of G , let $L(G) = \max_{H \subseteq G} \{\ell(H)\}^a$.

Every orientation has the following trivial lower bounds caused by the maximum weight of edges, and by the average weighted outdegrees of vertices:

^a Note that $L(G)$ can be obtained by a polynomial time algorithm [9], though $L(G)$ is introduced only for the analysis here.

Proposition 1 For an undirected weighted graph G and any orientation Λ , $\Delta_\Lambda(G) \geq w_{max}$. \square

Proposition 2 For an undirected weighted graph G and any orientation Λ , $\Delta_\Lambda(G) \geq \lceil \ell(G) \rceil$. \square

Let OPT denote an optimal orientation. We say a graph orientation algorithm is a σ -approximation algorithm if $ALG(G)/OPT(G) \leq \sigma$ holds for any undirected graph G , where $ALG(G)$ is the objective value of a solution obtained by the algorithm for G , and $OPT(G)$ is that of an optimal solution.

3. \mathcal{NP} -Hardness

In this section we show the \mathcal{NP} -hardness of MMO for the most general case. Let us consider the following decision version of MMO:

MMO(k)

Instance: A simple undirected graph $G = (V, E, w)$, and an integer k .

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

The proof of its \mathcal{NP} -hardness is by a polynomial time reduction from the *Partition* problem.

PARTITION

Instance: A set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers.

Question: Is there a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = \sum_{s_i \in S - S'} s_i$?

Theorem 1 $MMO(k)$ is \mathcal{NP} -complete.

Proof. Let us consider a restricted set of instances of PARTITION that satisfy the following two conditions (1) $\sum_{s_i \in S} s_i$ is even, and (2) for every j , $s_j < \sum_{s_i \in S} s_i / 2$. Even with these restrictions, PARTITION is still \mathcal{NP} -hard because an instance which does not satisfy either of these conditions can be trivially solved in polynomial time.

From an instance $S = \{s_1, s_2, \dots, s_n\}$ of PARTITION, we construct a weighted undirected graph $G = (V, E, w)$. For example, if $S = \{1, 2, 4, 5, 6\}$, the constructed graph G is as shown in Figure 1. The detailed construction is as follows: The vertex set V of G is divided into three types of vertices: (i) Item vertices v_1, v_2, \dots, v_n associated with n items in S , (ii) subset vertices a and b , and (iii) $n \times 4$ auxiliary vertices $u_{1,1}, u_{1,2}, u_{1,3}, u_{1,4}, \dots, u_{n,1}, u_{n,2}, u_{n,3}, u_{n,4}$. The total number of vertices is $5n + 2$. Let us define $K = \sum_{s_i \in S} s_i / 2$. The edge set E contains the following four types of edges: (i) n edges $\{a, v_i\}$'s with weight s_i , i.e., $w(\{a, v_i\}) = s_i$ for $i = 1, 2, \dots, n$, (ii) n edges $\{b, v_i\}$'s with weight s_i for $i = 1, 2, \dots, n$, (iii) $4n$ edges $\{u_{1,1}, u_{1,2}\}, \{u_{1,2}, u_{1,3}\}, \{u_{1,3}, u_{1,4}\}, \{u_{1,4}, u_{1,1}\}, \dots, \{u_{n,1}, u_{n,2}\}, \{u_{n,2}, u_{n,3}\}, \{u_{n,3}, u_{n,4}\}, \{u_{n,4}, u_{n,1}\}$ with weight K , and (iv) n edges $\{u_{i,1}, v_i\}$'s with weight $K - s_i$ for $i = 1, 2, \dots, n$. The total number of edges is $7n$. Finally, we set $k = K$. This construction of G can be obviously executed in polynomial time.

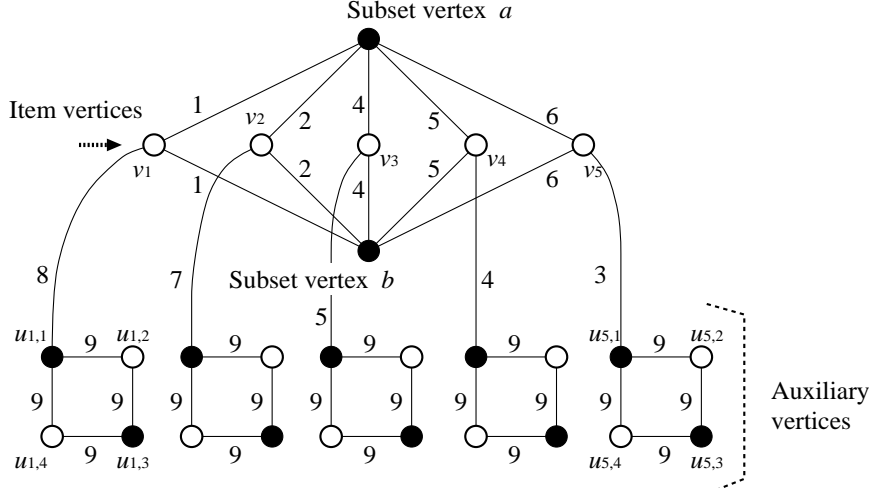


Fig. 1. Reduction from an instance $S = \{1, 2, 4, 5, 6\}$ of PARTITION.

Since clearly $\text{MMO}(k)$ is in \mathcal{NP} , we only show in the next its \mathcal{NP} -hardness: We prove that there is $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = K$ if and only if there is an orientation Λ of G such that $\Delta_\Lambda(G) = K$.

Lemma 1 *If there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = K$, then there exists an orientation Λ of G such that $\Delta_\Lambda(G) = K$.*

Proof. Suppose that there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = K$. Consider the following orientation Λ according to S' : (i) $u_{i,1} \rightarrow u_{i,2}$, $u_{i,2} \rightarrow u_{i,3}$, $u_{i,3} \rightarrow u_{i,4}$, and $u_{i,4} \rightarrow u_{i,1}$ for $i = 1, 2, \dots, n$ (ii) $v_i \rightarrow u_{i,1}$ for $i = 1, 2, \dots, n$. (iii) If $s_i \in S'$, $v_i \rightarrow a$ and $b \rightarrow v_i$; otherwise $a \rightarrow v_i$ and $v_i \rightarrow b$, for $i = 1, 2, \dots, n$.

In the following, we show that $\delta^+(v) = K$ for every vertex v and thus $\Delta_\Lambda(G) = K$. (i) For the auxiliary vertices, one can verify that $\delta_\Lambda^+(u_{i,1}) = \delta_\Lambda^+(u_{i,2}) = \delta_\Lambda^+(u_{i,3}) = \delta_\Lambda^+(u_{i,4}) = K$ holds for every i . (ii) As for each item vertex v_i , since $v_i \rightarrow u_{i,1}$ and either $v_i \rightarrow a$ or $v_i \rightarrow b$ holds, $\delta_\Lambda^+(v_i) = (K - s_i) + s_i = K$. (iii) For each $s_i \in S'$, $b \rightarrow v_i$ holds, and for each $s_i \in S - S'$, $v_i \rightarrow b$ also holds. Therefore $\delta_\Lambda^+(b) = \sum_{s_i \in S'} w(v_i, b) = \sum_{s_i \in S'} s_i = K$. As for the vertex a , we can show $\delta_\Lambda^+(a) = K$ by a similar discussion. \square

Lemma 2 *If there exist an orientation Λ of G such that $\Delta_\Lambda(G) \leq K$, there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = K$.*

Proof. Suppose that Λ is an orientation of G satisfying $\Delta_\Lambda(G) \leq K$. In Λ , the auxiliary vertices form directed cycles. (Otherwise, $\Delta_\Lambda(G) > K$.) Then, the directions of edges between item vertices and auxiliary vertices must be $v_i \rightarrow u_{i,1}$ for each i . Consider the directions of edges $\{v_i, a\}$ and $\{v_i, b\}$ for each i . Here, we define two sets of indices, $A = \{i \mid a \rightarrow v_i \text{ under } \Lambda\}$ and $B = \{i \mid b \rightarrow v_i \text{ under } \Lambda\}$. Since $w(\{v_i, u_{i,1}\}) = K - w(\{v_i, a\}) = K - w(\{v_i, b\})$ and $\delta^+(v_i) \leq K$, there is no i satisfying both $v_i \rightarrow a$ and $v_i \rightarrow b$. Thus, $A \cup B = S$ holds. Note that $\sum_{i \in A \cup B} s_i = \sum_{i \in S} s_i = 2K$, $\delta^+(a) = \sum_{i \in A} s_i \leq K$ and $\delta^+(b) = \sum_{i \in B} s_i \leq K$.

It follows that $2K = \sum_{i \in A \cup B} s_i \leq \sum_{i \in A} s_i + \sum_{i \in B} s_i \leq 2K$. We then obtain $\sum_{i \in A} s_i = \sum_{i \in B} s_i = K$, which shows that there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} s_i = K$. \square

From the above two lemmas, the \mathcal{NP} -hardness of $\text{MMO}(k)$ is shown, which concludes Theorem 1. \square

Remark. See the reduced graph in Figure 1 again. One can verify that it is a planar bipartite graph, since all the “white” vertices are connected only with “black” vertices and vice versa, and each cycle constructed by four auxiliary vertices $u_{i,1}$ through $u_{i,4}$ can be placed between two item vertices v_i and v_{i+1} without edge crossing, which gives us a planar drawing of the graph in Figure 1. Hence, Theorem 1 gives the following intractable subclasses: $\text{MMO}(k)$ is \mathcal{NP} -complete even for planar bipartite graphs.

4. Optimal Algorithms for Special Cases

In this section we present two polynomial time algorithms when an instance is (1) a weighted tree, and (2) an unweighted graph, or more generally, a graph such that all the weights of their edges are identical. The basic ideas of those algorithms are simple but they will play important roles in our approximation algorithms for the most general case.

4.1. Trees

Recall that the maximum weighted outdegree of a graph G under every orientation is at least the maximum weight w_{\max} of its edges as mentioned in Proposition 1. We can efficiently find an orientation Λ such that $\Delta_\Lambda(G) = w_{\max}$ if G is a tree:

Theorem 2 *For trees, an optimal solution can be obtained in $O(n)$ time.*

Proof. All we have to do is to orient all edges toward a root chosen arbitrarily, which can be obviously achieved in linear time. (This linear time algorithm will be referred to as **Convergence** later.) \square

4.2. Identical Weights

Here, in order to make our basic idea clear, we give our elementary algorithm that is optimal if all the weights of edges are identical. A similar optimal algorithm has been independently shown in [19], but our proof of the optimality is much simpler. Note that now we consider connected graphs as input, i.e., $n - 1 \leq m$.

Theorem 3 [19] *If all the weights of edges are identical, an optimal solution can be obtained in $O(m^2)$ time.*

Proof. We can consider the weight of the edges is one without loss of generality, and hence the weighted outdegree in this case is the same as the outdegree of the unweighted graph. The following **Reverse** is an algorithm to solve this case optimally, whose basic strategy is quite straightforward: Observe an unweighted graph and an orientation illustrated in Figure 2-(a). One can see that the out-degrees of the four vertices u_0 , u_1 , u_2 and u_3 are $(5, 3, 3, 1)$, respectively, and the

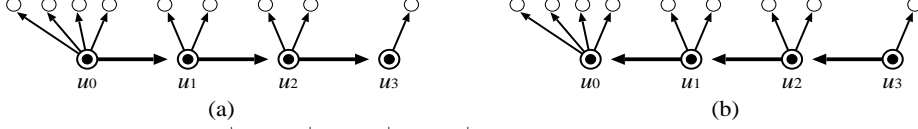


Fig. 2. (a) $(\delta^+(u_0), \delta^+(u_1), \delta^+(u_2), \delta^+(u_3)) = (5, 3, 3, 1)$ and the maximum outdegree is five, but (b) $(\delta^+(u_0), \delta^+(u_1), \delta^+(u_2), \delta^+(u_3)) = (4, 3, 3, 2)$ and the maximum outdegree decreases to four.

maximum outdegree is five. However, if we reverse the orientation of the directed path $\langle u_0, u_1, u_2, u_3 \rangle$ as shown in Figure 2-(b), the outdegrees become $(4, 3, 3, 2)$ and the maximum outdegree decreases to four without increasing the outdegrees of intermediate vertices u_1 and u_2 . **Reverse** repeatedly finds such a directed path and reduces its maximum outdegree by reversing its direction:

Algorithm Reverse:

Input: An unweighted graph $G = (V, E)$.

Output: An arc set Λ which determines directions of edges in E .

Step 0: Set $\Lambda = \emptyset$.

Step 1: Find arbitrary orientation of the graph G and update Λ .

Step 2: Compute the (weighted) outdegree $\delta_\Lambda^+(v)$ for each vertex v .
Let u be a vertex having maximum outdegree among all vertices
(in case of ties, select one vertex arbitrarily).

Step 3: Find a directed path $P = \langle u, v_1, \dots, v_k \rangle$ of length k ($k \geq 1$)
that satisfies

- $\delta_\Lambda^+(v_i) \leq \delta_\Lambda^+(u)$ for $1 \leq i \leq k-1$, and
- $\delta_\Lambda^+(v_k) \leq \delta_\Lambda^+(u) - 2$.

If such a path P exists, then set $\Lambda = (\Lambda \setminus P) \cup \overline{P}$ (i.e., orient the path P in reverse order) and goto Step 2. Otherwise, output Λ and halt.

First of all, we estimate the running time. Step 0 is done in $O(1)$ time. Both of Steps 1 and 2 require $O(m)$ time. Step 3 can be implemented by using a breadth first search and it also takes $O(m)$ time. Therefore, the number of iterations of Steps 2 and 3 determines the total amount of time. Consider a subset of vertices $M = \{v \mid v \in V, \delta_\Lambda^+(v) \text{ is the maximum, or it is (the maximum } - 1)\}$. (This set does not appear in the description of the algorithm and does only for this analysis.)

In Step 3, a vertex u in M is selected as the starting vertex of a directed path. The modification of the orientation Λ in Step 3 (i) decreases the outdegree of the vertex u by one and then u still belongs to M at the next iteration, and (ii) increases the outdegree of some vertex $v_k \in V - M$ and then v_k possibly comes to be in M at the next iteration. Therefore, the outdegree of a vertex in M monotonically decreases after it belongs to M . As mentioned before, the outdegrees of intermediate vertices v_1, \dots, v_{k-1} in the path remain unchanged in Step 3.

From Proposition 2 and the above observation, each vertex v can be the starting vertex of such paths by the limited number of times $d(v) - m/n$, since now we consider the weight of each edge is one. Summing up $d(v) - m/n$ over all the vertices gives an upper bound on the number of iterations of Steps 2 and 3, that is $\sum_{v \in V} d(v) - m = 2m - m = m$. Therefore, since Steps 2 and 3 takes $O(m)$ time, the total running time of the algorithm is $O(m^2)$.

The rest of the proof is to show optimality of the algorithm. Since an orientation for a graph gives an orientation for any of its subgraphs, we have:

Proposition 3 *For a graph $G(V, E, w)$ and its subgraph $G'(V', E', w)$, $\Delta_{OPT}(G) \geq \Delta_{OPT'}(G')$ holds, where OPT and OPT' are optimal solutions for G and G' , respectively.* \square

Let v_p be a vertex having the maximum outdegree $p = \delta_{\Lambda}^+(v_p)$ under an orientation Λ obtained by the algorithm. All the vertices reachable from v_p by following directed paths have outdegree at least $p - 1$ from the halting criteria in Step 3. Let the set of those vertices and v_p be V' and consider the induced subgraph $G[V']$.

Lemma 3 $\Delta_{OPT}(G[V']) = \Delta_{\Lambda}(G[V']) = p$, that is, $\Delta_{\Lambda}(G[V'])$ is optimal.

Proof. The number of edges in $G[V']$ is at least $p + (p - 1)(|V'| - 1) = (p - 1)|V'| + 1$. From Proposition 2, $\Delta_{OPT}(G[V']) \geq \lceil ((p - 1)|V'| + 1)/|V'| \rceil = \lceil p - 1 + 1/|V'| \rceil = p = \Delta_{\Lambda}(G[V'])$. \square

Since p is the maximum outdegree under Λ and $\Delta_{OPT}(G) \geq \Delta_{OPT}(G[V']) = p$ for any such $G[V']$ from Proposition 3, the output Λ of the algorithm is an optimal orientation for G . This ends the proof of Theorem 3. \square

A Faster Algorithm. Algorithm **Reverse** can find an optimal orientation for unweighted graphs in polynomial time, however, there might be a possibility of improvements on the running time: In Step 3 of **Reverse**, the algorithm finds just one simple directed path in $O(m)$ time and iterates that m times, but if we can find several edge-disjoint directed paths at one blow, it might reduce the number of iterations. To this end, we consider the following network for a given orientation Λ and a parameter k : Let A denote an arc set of weighted directed graph $G' = (V, A)$ obtained by applying an orientation Λ to the input graph $G = (V, E)$. For G' , we consider two subsets $V_k^+ = \{v \mid \delta_{\Lambda}^+(v) > k\}$ and $V_k^- = \{v \mid \delta_{\Lambda}^+(v) < k\}$ of V . Note that $V = V_k^+ \cup V_k^- \cup \{v \mid \delta_{\Lambda}^+(v) = k\}$. Then the network $\mathcal{N}_k(\Lambda)$ we construct is defined as $\mathcal{N}_k(\Lambda) = (\tilde{V}, \tilde{E}) = (\{s\} \cup \{t\} \cup V, A \cup A_k^+ \cup A_k^-)$, where

$$\begin{aligned} A_k^+ &= \bigcup_{v \in V_k^+} \{e_{(v,i)}^+ = (s, v) \mid i = k + 1, \dots, \delta_{\Lambda}^+(v)\}, \\ A_k^- &= \bigcup_{v \in V_k^-} \{e_{(v,i)}^- = (v, t) \mid i = \delta_{\Lambda}^+(v) + 1, \dots, k\}, \end{aligned}$$

and the capacity of all the edges in \tilde{E} is one. See Figure 3. The above two sets of arcs may contain parallel arcs for each vertex in V_k^+ and V_k^- . The number of vertices is $|V| + 2$ and that of arcs is at most $3|E|$ in $\mathcal{N}_k(\Lambda)$ (although the exact number of arcs depends on k and Λ).

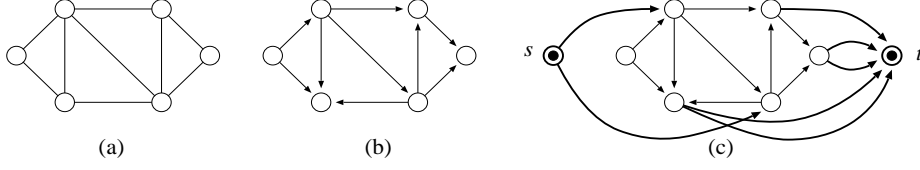


Fig. 3. (a) Input graph G (b) An orientation Λ (c) The network $\mathcal{N}_2(\Lambda)$

Lemma 4 *The size of the maximum flow for a network $\mathcal{N}_k(\Lambda)$ is $f_k = \sum_{u \in V_k^+} (\delta_\Lambda^+(u) - k)$ if and only if the answer of $\text{MMO}(k)$ is “yes”. (The proof will be given later.)*

Since the network \mathcal{N}_k is a unit capacity flow network, this lemma says that the following are equivalent for an undirected graph G : (1) G under an orientation includes f_k edge-disjoint directed paths between V_k^+ and V_k^- . (2) G has an orientation with the maximum outdegree bounded by k . The maximum flow problem for a unit capacity flow network can be solved in $O(|E_{\mathcal{N}}|^{3/2})$ time, where $|E_{\mathcal{N}}|$ is the number of edges of the flow network [6]. Note that this algorithm can work for networks with parallel edges in the same upper bound. Since \mathcal{N}_k has at most $3m$ edges, we immediately obtain the following theorem.

Theorem 4 *$\text{MMO}(k)$ can be solved in $O(m^{3/2})$ time, if all the edge weights are identical.* \square

We can find the optimal k by doing the binary search of Theorem 4 as its engine in $O(m^{3/2} \cdot \log \Delta_{OPT}(G))$. Since the obtained network flow solution for the optimal k gives edge-disjoint paths, we apply Step 3 of **Reverse** to them, which can be done in $O(m)$ in total and provides an optimal orientation.

Corollary 1 *MMO can be solved in $O(m^{3/2} \cdot \log \Delta_{OPT}(G))$ time, if all the edge weights are identical.* \square

Proof of Lemma 4. (Only-if part) Suppose that the size of the maximum flow is f_k . Since the network $\mathcal{N}_k(\Lambda)$ has only edges with capacity one, there exist f_k edge-disjoint paths from s to t . Therefore, by the construction of the network, each $u \in V_k^+$ has $(\delta_\Lambda^+(u) - k)$ directed paths to some vertices in V_k^- , which are edge-disjoint to each other. By applying the procedure of reversing in Step 3 of **Reverse** to those paths in arbitrary order, we can reduce the outdegree of the vertex u to k . Since the outdegree of any $v \in V_k^-$ does not exceed k by this operation, the maximum outdegree of the resulting orientation is k .

(If part) In this case, if we apply the algorithm **Reverse** to the input graph G with Λ as the initial orientation instead of a random one, we can find an orientation OPT such that $\Delta_{OPT}(G) \leq k$.

Consider the collection of directed paths processed in Step 3 of **Reverse** to obtain OPT , which are represented by P_1, P_2, \dots, P_h and supposed to be obtained in this order. Here it is important to note that some edge $\{u, v\}$ may appear several times in those paths but its directions differ; (u, v) may be included in some paths though (v, u) is also in others. Also we assume that the sequence P_1, P_2, \dots, P_h is minimal in a sense that for $1 \leq i \leq h-1$, even after processing P_i the maximum outdegree is still greater than k but just after processing P_h the maximum outdegree decreases

to k . Although **Reverse** may process a sequence of paths that is not minimal, in such a case it is sufficient to consider only its minimal subset. Because of the optimality of **Reverse**, $h = f_k$. In the following, we show that the sequence of the paths P_1, P_2, \dots, P_h can be transformed into a sequence of paths in which every path is edge-disjoint to others. Moreover the algorithm **Reverse** runs correctly for the modified sequence.

After reversing P_i to $\overline{P_i}$ in Step 3, we suppose to obtain an orientation Λ_i , i.e., at the beginning of the i -th execution of Step 3, we have Λ_{i-1} (Λ in case $i = 0$) and update it to Λ_i in Step 3, and eventually we obtain $OPT = \Lambda_h$. We divide the sequence of paths P_1, P_2, \dots, P_h into $\Delta_\Lambda(G) - k \stackrel{\text{def}}{=} z$ groups based on the decrease of the maximum outdegree of vertices, $P_{h_0}(=P_1), \dots, P_{h_1-1}, P_{h_1}, \dots, P_{h_2-1}, P_{h_2}, \dots, P_{h_z-1}(=P_h)$ such that $\Delta_{\Lambda_i}(G) = \Delta_{\Lambda_{i+1}}(G)$ for $h_j \leq i \leq h_{j+1} - 2$ and $\Delta_{\Lambda_{i-1}}(G) - 1 = \Delta_{\Lambda_i}(G)$ for $i = h_j$, where $0 \leq j \leq z - 1$.

Let us consider the first group of the paths, $P_{h_0}(=P_1), \dots, P_{h_1-1}$. We show that even if they are not edge-disjoint, we can transform them to edge-disjoint ones. Suppose that they are not edge-disjoint. Let $2 \leq q \leq h_1 - 1$ be the (smallest) index of a path such that P_1, \dots, P_{q-1} are edge-disjoint but P_r and P_q are not for some $1 \leq r \leq q - 1$. We focus on the subsequence P_1, \dots, P_q of paths.

From the rule of the grouping of the paths, each path P_i in the first group starts from a vertex u with $\delta_{\Lambda_{i-1}}^+(u) = \Delta_\Lambda(G)$. Therefore, each vertex can be the starting vertex only once in the group. Otherwise, its outdegree decreases by at least two that contradicts the path starting from it is processed in the first group.

Step 3 of **Reverse** only changes the outdegrees of the first and the last vertices of a path. Especially, as for the last vertices, their outdegrees increase but never reaches to $\Delta_\Lambda(G)$, namely, such vertices can not be a start vertex of such a path in the first group of paths. Therefore, another sequence $P_1, \dots, P_{r-1}, P_{r+1}, \dots, P_{q-1}, P_r, P_q$ is also valid in terms of that **Reverse** possibly runs following this sequence and the final result is the same as that of the original sequence P_1, \dots, P_q . Hence we can assume that $r = q - 1$, namely, P_{q-1} and P_q are not edge-disjoint and it is the first occurrence in the group, without loss of generality.

We assume that P_{q-1} and P_q share only one edge $\{x, y\}$. The case more than one edge is shared by those paths can be discussed similarly and is omitted. Let the two paths P_{q-1} and P_q be $\langle u_{q-1}, \dots, x, y, \dots, v_{q-1} \rangle$ and $\langle u_q, \dots, y, x, \dots, v_q \rangle$, respectively. Note that the direction of the edge $\{x, y\}$ differs in those paths because of the reversing procedure. See Figure 4. The vertices $u_{q-1}, v_{q-1}, u_q, v_q, x$, and y are distinct from the observations that if two of them are identical, either P_{q-1} or P_q is not such a path processed in the first group; for example, if v_{q-1} and u_q are identical, then $\delta_{\Lambda_{q-1}}^+(u_q) = \delta_{\Lambda_{q-2}}^+(u_q) + 1 < \delta_{\Lambda_{q-2}}^+(u_{q-1})$ that implies the path P_q is not processed in the first group.

From the rule of grouping the paths, it holds that

$$\delta_{\Lambda_{q-2}}^+(u_{q-1}) = \delta_{\Lambda_{q-2}}^+(u_q) = \delta_{\Lambda_{q-1}}^+(u_q). \quad (1)$$

Then, since the path P_{q-1} is reversed at the $(q - 1)$ -th execution of Step 3 of

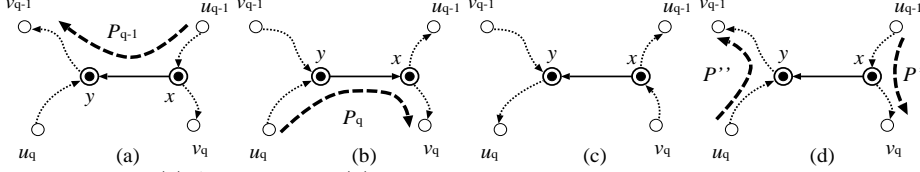


Fig. 4. (a) A path P_{q-1} (b) Reversing P_{q-1} and the next candidate path P_q
(c) The obtained orientation Λ_q (d) Replacing P_{q-1} and P_q with P' and P''

Reverse, the following also holds

$$\delta_{\Lambda_{q-2}}^+(u_{q-1}) \geq \delta_{\Lambda_{q-2}}^+(v_{q-1}) + 2. \quad (2)$$

By reversing P_{q-1} to $\overline{P_{q-1}}$, we obtain (by an orientation Λ_{q-1})

$$\delta_{\Lambda_{q-1}}^+(u_{q-1}) = \delta_{\Lambda_{q-2}}^+(u_{q-1}) - 1, \quad (3)$$

$$\delta_{\Lambda_{q-1}}^+(v_{q-1}) = \delta_{\Lambda_{q-2}}^+(v_{q-1}) + 1, \quad \text{and} \quad (4)$$

$$\delta_{\Lambda_{q-1}}^+(v_q) = \delta_{\Lambda_{q-2}}^+(v_q). \quad (5)$$

Also, the path P_q is reversed at the q -th iteration of Step 3 of **Reverse**, which implies

$$\delta_{\Lambda_{q-1}}^+(u_q) \geq \delta_{\Lambda_{q-1}}^+(v_q) + 2. \quad (6)$$

Let us decompose the two paths P_{q-1} and P_q to the following three portions, respectively: $P_{q-1} = (P_{q-1}^{(1)}, P_{q-1}^{(2)}, P_{q-1}^{(3)})$, where $P_{q-1}^{(1)} = \langle u_{q-1}, \dots, x \rangle$, $P_{q-1}^{(2)} = \langle x, y \rangle$, and $P_{q-1}^{(3)} = \langle y, \dots, v_{q-1} \rangle$, and $P_q = (P_q^{(1)}, P_q^{(2)}, P_q^{(3)})$, where $P_q^{(1)} = \langle u_q, \dots, y \rangle$, $P_q^{(2)} = \langle y, x \rangle$, and $P_q^{(3)} = \langle x, \dots, v_q \rangle$.

Consider alternating paths $P' = (P_{q-1}^{(1)}, P_q^{(3)}) = \langle u_{q-1}, \dots, x, \dots, v_q \rangle$ and $P'' = (P_q^{(1)}, P_{q-1}^{(3)}) = \langle u_q, \dots, y, \dots, v_{q-1} \rangle$. At the $(q-1)$ -th iteration of Step 3 of **Reverse**, P' is also a candidate of a path processed, because $\delta_{\Lambda_{q-2}}^+(u_{q-1}) = \delta_{\Lambda_{q-1}}^+(u_q) \geq \delta_{\Lambda_{q-1}}^+(v_q) + 2 = \delta_{\Lambda_{q-2}}^+(v_q) + 2$ by the above conditions (1), (6), and (5).

Consider processing P_1, \dots, P_{q-2}, P' in this order instead of $P_1, \dots, P_{q-2}, P_{q-1}$ in Step 3 of **Reverse**. Let the resulting orientation be Λ' . We want to show that the path P'' is also a candidate of path processed at the q -th execution of Step 3 of **Reverse**, i.e., it holds that $\delta_{\Lambda'}^+(u_q) \geq \delta_{\Lambda'}^+(v_{q-1}) + 2$. By reversing P' to $\overline{P'}$ at the $(q-1)$ -th execution of the step, the weighted outdegrees of the vertices are

$$\begin{aligned} \delta_{\Lambda'}^+(u_{q-1}) &= \delta_{\Lambda_{q-2}}^+(u_{q-1}) - 1, \\ \delta_{\Lambda'}^+(v_q) &= \delta_{\Lambda_{q-2}}^+(v_q) + 1, \\ \delta_{\Lambda'}^+(v_{q-1}) &= \delta_{\Lambda_{q-2}}^+(v_{q-1}), \quad \text{and} \\ \delta_{\Lambda'}^+(u_q) &= \delta_{\Lambda_{q-2}}^+(u_q). \end{aligned}$$

From these and the above conditions (1) and (2), it holds that $\delta_{\Lambda'}^+(u_q) = \delta_{\Lambda_{q-2}}^+(u_q) = \delta_{\Lambda_{q-2}}^+(u_{q-1}) \geq \delta_{\Lambda_{q-2}}^+(v_{q-1}) + 2 = \delta_{\Lambda'}^+(v_{q-1}) + 2$. Therefore P'' is

an alternate candidate for the reverse operation at the q -th iteration of Step 3 of **Reverse**, that is, $P_1, \dots, P_{q-2}, P', P''$ is also a valid sequence of paths processed by **Reverse**. In addition to that the resulting orientation Λ'' is the same as the original orientation Λ_q at the end of the sequence.

By the above procedure, although the set of paths $P_1, \dots, P_{q-2}, P', P''$ may not be yet edge-disjoint, the number of shared edges decreased. Therefore, by repeatedly applying the above procedure, we can transform the first (original) group of paths to disjoint one without changing the temporal orientation(solution) Λ_{h_1-1} at the end of the first group.

The above discussion can be applied to the case across two groups, say, considering P_{h_1-1} and P_{h_1} with distinct four vertices $u_{h_1-1}, v_{h_1-1}, u_{h_1}$, and v_{h_1} at the ends of the paths. In such a case, similar to the above condition (1), it holds that $\delta_{\Lambda_{h_1-2}}^+(u_{h_1-1}) > \delta_{\Lambda_{h_1-2}}^+(u_{h_1})$ because of the grouping scheme for paths. The other conditions in this case are similar to the above (2) to (6). Then a similar discussion derives $\delta_{\Lambda_{h_1-2}}^+(u_{h_1-1}) \geq \delta_{\Lambda_{h_1-2}}^+(v_{h_1}) + 2$ and $\delta_{\Lambda_{h_1}}^+(u_{h_1}) \geq \delta_{\Lambda_{h_1}}^+(v_{h_1-1}) + 2$, which makes us possibly construct a pair of alternating paths similar to P' and P'' in the above; one is from u_{h_1-1} to v_{h_1} and the other is from u_{h_1} to v_{h_1-1} . In this case, two of the vertices $u_{h_1-1}, v_{h_1-1}, u_{h_1}$, and v_{h_1} may be identical, but a similar discussion can be done.

As a result, if we apply the above procedure repeatedly, we can obtain an edge-disjoint sequence of paths from the original sequence P_1, \dots, P_h . Then, the edge-disjoint sequence is possible to be produced by an execution of **Reverse** for the input graph and the initial orientation Λ . Since **Reverse** outputs an optimal solution and now we consider the minimality of the sequence, the number of such paths is equal to $f_k (= h)$. Therefore, by construction of the network $\mathcal{N}_k(\Lambda)$, there exist f_k edge-disjoint paths from vertices in V_k^+ to vertices in V_k^- . That is, the size of the maximum flow is equal to f_k . \square

Remark. We show Theorem 4 by the extension of **Reverse** algorithm. However, it is possible to reduce MMO to a network flow problem directly. The algorithm based on the reduction also fulfills the same time complexity, $O(m^{3/2} \cdot \log \Delta_{OPT}(G))$.

5. Approximation Algorithms

In this section we present two approximation algorithms for the general case of MMO using the algorithms presented in the previous section as sub-procedures. One can notice that algorithm **Reverse** can be applied to a general weighted graph if we ignore its weights of edges. This simple idea achieves the following approximation guarantee:

Theorem 5 *Algorithm Reverse is a w_{max}/w_{min} -approximation algorithm for general input graphs.*

Proof. Let an input graph be G and an optimal orientation for G be OPT . Consider two weighted graphs G_{min} and G_{max} that are obtained by replacing all the edge weights to w_{min} and w_{max} , respectively. It is important to note that OPT is not always optimal for G_{min} or G_{max} .

Suppose that algorithm **Reverse** outputs an orientation Λ for the input graph G . Then, from the optimality of Λ for both G_{min} and G_{max} ,

$$\Delta_{\Lambda}(G_{min}) \leq \Delta_{OPT}(G_{min}) \leq \Delta_{OPT}(G) \leq \Delta_{\Lambda}(G) \leq \Delta_{\Lambda}(G_{max})$$

holds, and hence

$$\frac{\Delta_{\Lambda}(G)}{\Delta_{OPT}(G)} \leq \frac{\Delta_{\Lambda}(G_{max})}{\Delta_{\Lambda}(G_{min})}.$$

The oriented graphs of G_{max} and G_{min} have the same structure except for their edge weights, and therefore

$$\frac{\Delta_{\Lambda}(G)}{\Delta_{OPT}(G)} \leq \frac{\Delta_{\Lambda}(G_{max})}{\Delta_{\Lambda}(G_{min})} = \frac{w_{max}}{w_{min}}.$$

□

Since **Reverse** does not work well when $w_{max} \gg w_{min}$ and its performance is heavily dependent on the edge weights of the input graph, we would like to design another approximation algorithm with a ‘stable’ worst case ratio. Indeed a quite simple strategy can achieve an approximation ratio of 2: For ease of exposition, observe a weighted graph G_a illustrated in Figure 5-(a), which consists of four vertices, v_1, v_2, v_3 , and v_4 , and six edges whose weights are $w(\{v_1, v_2\}) = 1$, $w(\{v_1, v_3\}) = 1$, $w(\{v_1, v_4\}) = 1$, $w(\{v_2, v_3\}) = 1$, $w(\{v_2, v_4\}) = 2$, and $w(\{v_3, v_4\}) = 3$. The average weight of the edges is $\ell(G_a) = 9/4$, which is a trivial lower bound of MMO, as shown in Proposition 2. The outline of the 2-approximation algorithm is as follows: (i) First we choose a vertex v whose weighted degree $\delta(v)$ is at most $2\ell(G_a) = 9/2$ since such a vertex surely exists in G_a . In this case we choose vertex v_1 . (ii) All the edges incident with v_1 are oriented outwards from v_1 to its neighbors, and v_1 is removed from G_a . (iii) We recalculate the average weight of the remaining graph $G_a - \{v_1\}$, and iterate those stages while edges not oriented are remaining. As a result, we select v_1, v_2 , and v_3 in this order, and the oriented graph is shown in Figure 5-(b).

The above simple procedure guarantees that the maximum weighted outdegree is at most $2\ell(H)$ for each induced subgraph $H \subseteq G_a$, which implies that the approximation ratio is 2. Furthermore, from this observation, we might reduce the approximation ratio to $2 - \varepsilon$ for some positive ε if it is possible to select at least one vertex whose degree is less than twice the average weights of induced subgraphs in each iteration; however, it is impossible. There is an apparent counterexample as illustrated in Figure 5-(c). Its average weight is 2 and the weighted degree of each vertex is 4, double the former. In order to improve the approximation ratio we require further ideas.

Throughout the following, by $\delta_G(u)$ we denote the total weight of edges that connect to a vertex u in a graph G . Here we provide our approximation algorithm, **ALGMMO**, which can improve the approximation ratio of 2:

Algorithm ALGMMO:

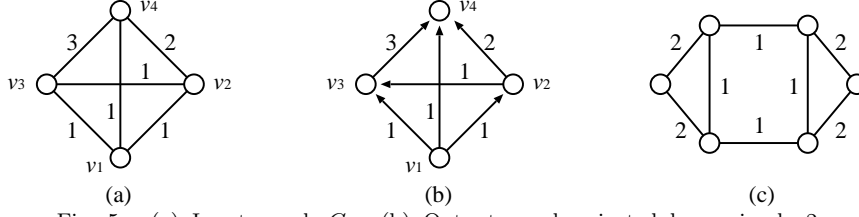


Fig. 5. (a) Input graph G_a (b) Output graph oriented by a simple 2-approximation algorithm (c) Worst case example

Input: A weighted graph $G = (V, E, w)$.

Output: An arc set Λ which determines directions of edges in E .

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

Step 1: Repeat the following while there exists a vertex u in G' such that $\delta_{G'}(u) \leq \lceil 2\ell \rceil - 1$,

- For each edge $\{u, v\}$ for some v in G' , add (u, v) to Λ . Then, remove the vertex u and all edges incident to u from G' .

Step 2: There are three cases:

- (Case 2-1): No vertex is in G' , i.e., all the vertices are removed in Step 1. In this case, output Λ and halt.
- (Case 2-2): For every vertex v in G' , $\delta_{G'}(v) = \lceil 2\ell \rceil$. In this case, proceed to Step 3.
- (Case 2-3): There is at least one vertex v such that $\delta_{G'}(v) \geq \lceil 2\ell \rceil + 1$. In this case, update $\ell = \ell(G')$ and goto Step 1.

Step 3: Find a cycle $\langle v_0, v_1, \dots, v_k, v_0 \rangle$ in G' . If such a cycle does not exist, goto Step 4. Add $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k), (v_k, v_0)$ to Λ , and remove edges $\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}, \{v_k, v_0\}$ from G' and repeat this step.

Step 4: (At the beginning of this step G' is a forest.) Apply algorithm **Convergence** to each connected component of G' . Let an orientation obtained for G' by **Convergence** be Λ' . Output $\Lambda \cup \Lambda'$ and halt.

Theorem 6 *Algorithm ALGMMO runs in $O(m^2)$ time and is a $(2 - 1/\lceil L(G) \rceil)$ -approximation algorithm.*

Proof. Running Time: Steps 0, 1, and 2 require $O(m)$ time, for each. The number of iterations of Steps 1 and 2 is at most $O(n)$ because one vertex is removed from the graph in single iteration or the algorithm proceeds to Step 3. In Step 3, finding a cycle is done by a breadth-first search which takes $O(m)$ time, and this step is repeated at most $O(m)$ times. In Step 4, we need $O(n)$ time because algorithm **Convergence** runs in $O(c)$ time for a connected component having c vertices, and the total number of vertices in the forest is at most n . In summary, $O(nm)$ time for

Steps 0 through 2, $O(m^2)$ time for Step 3, and $O(n)$ time for Step 4 are required for each, and hence the running time in total is $O(m^2)$.

Approximation Ratio: Let G_i be a graph at the beginning of the i -th iteration of Steps 1 and 2, which is represented in **ALGMM0** by G' , e.g., $G_1 = G$. Suppose that the number of iterations of Steps 1 and 2 is j (≥ 1). At the i -th iteration ($i \geq 2$) of Steps 1 and 2, for every vertex u , $\delta_{G_i}(u) \geq \lceil 2\ell(G_{i-1}) \rceil$, and some vertex v has $\delta_{G_i}(v) \geq \lceil 2\ell(G_{i-1}) \rceil + 1$. This means that, for all $i \leq j - 1$, $\lceil \ell(G_i) \rceil \leq \lceil \ell(G_{i+1}) \rceil$. Then, since G_j is a subgraph of the input graph $G(= G_1)$, from Propositions 2 and 3, $\Delta_{OPT}(G) \geq \Delta_{OPT_j}(G_j) \geq \lceil \ell(G_j) \rceil$, where OPT_j is an optimal solution for G_j .

The proof is based on the lemma below, by which we can conclude

$$\begin{aligned} \Delta_\Lambda(G) &= \max_{u \in V} \{\delta_\Lambda^+(u)\} \\ &\leq \lceil 2\ell(G_j) \rceil - 1 \\ &\leq 2\lceil \ell(G_j) \rceil - 1 \\ &\leq \left(2 - \frac{1}{\lceil \ell(G_j) \rceil}\right) \lceil \ell(G_j) \rceil \\ &\leq \left(2 - \frac{1}{\lceil L(G) \rceil}\right) \Delta_{OPT}(G). \end{aligned}$$

□

Lemma 5 *For every vertex u , $\delta_\Lambda^+(u) \leq \lceil 2\ell(G_j) \rceil - 1$, or Λ is optimal.*

Proof. (Step 1) For $1 \leq i \leq j$, a vertex u removed from the graph at the i -th iteration of Steps 1 and 2 has $\delta_\Lambda^+(u) \leq \lceil 2\ell(G_i) \rceil - 1 \leq \lceil 2\ell(G_j) \rceil - 1$. (Note that $\delta_\Lambda^+(u) \leq \lceil 2\ell(G_j) \rceil - 1$ does not imply that $\delta_G(u) \leq \lceil 2\ell(G) \rceil - 1$.)

(Step 2) If the algorithm terminates (Case 2-1), every vertex u satisfies the condition $\delta_\Lambda^+(u) \leq \lceil 2\ell(G_j) \rceil - 1$ based on the analysis of Step 1 above.

(Step 3) At the beginning of this step, for any edge $\{u, v\}$ in the original graph G that connects to a vertex u in G' , either of the following conditions holds: (a) neither (u, v) nor (v, u) is in Λ , and (b) (v, u) is in Λ , i.e., “current” weighted outdegree of u is zero. Also $\delta_{G'}(u) = \lceil 2\ell(G_j) \rceil$ because of the condition in (Case 2-2) of Step 2.

Consider a vertex v_i that is contained in some cycle $\langle v_0, \dots, v_k, v_0 \rangle$ whose orientation is determined in Step 3. Since the orientation Λ contains (v_{i-1}, v_i) (or (v_k, v_0) when $i = 0$), $w(\{v_{i-1}, v_i\}) \geq w_{min}$, and $\delta_{G'}(v_i) = \lceil 2\ell(G_j) \rceil$, it holds that $\delta_\Lambda^+(v_i) \leq \lceil 2\ell(G_j) \rceil - w_{min} \leq \lceil 2\ell(G_j) \rceil - 1$.

(Step 4 and the overall performance) There are two cases on the vertex set V' of G' at the beginning of the Step 4: (i) All the vertices in V' are included in at least one cycle whose orientation is determined in Step 3, and (ii) otherwise, i.e., some vertex x is not included in such cycles.

In the case (i), every vertex u in V' has $\delta_\Lambda^+(u) \leq \lceil 2\ell(G_j) \rceil - 1$ based on the analysis of Step 3 without regard to the orientation determined in Step 4. Then, also from the analysis for Steps 1 through 3 in the above, we can see that the weighted outdegree of all the vertices in G_j under Λ is at most $\lceil 2\ell(G_j) \rceil - 1$.

In the case (ii), If $\delta_\Lambda^+(x) \leq \lceil 2\ell(G_j) \rceil - 1$ for every such vertex x , a similar discussion as for the case (i) can be done and we can conclude that the weighted outdegree of all the vertices in G_j under Λ is at most $\lceil 2\ell(G_j) \rceil - 1$.

Let us assume that $\delta_\Lambda^+(x) = \lceil 2\ell(G_j) \rceil$. We observe that x is a vertex in a tree and $\delta_{G'}(x) = \lceil 2\ell(G_j) \rceil$ at the beginning of Step 4. Based on algorithm **Convergence**, x must be a leaf vertex in that tree to have weighted outdegree $\lceil 2\ell(G_j) \rceil$, because if x is an internal vertex (including root), then at least one edge $\{x, y\}$ for some y is directed as (y, x) in Λ so that $\delta_\Lambda^+(x) \leq \lceil 2\ell(G_j) \rceil - w(y, x) \leq \lceil 2\ell(G_j) \rceil - w_{\min}$. This implies that there is an edge $\{x, z\}$ for some z such that $w(\{x, z\}) = \lceil 2\ell(G_j) \rceil$. Therefore since such an edge exists in the input graph, from Proposition 1, $\Delta_{OPT}(G) \geq w_{\max} \geq \lceil 2\ell(G_j) \rceil = \Delta_\Lambda(G)$, that is, Λ is optimal. \square

The proof of Lemma 5 is for general input graphs. If all the $\delta_G(v)$'s are equal, (i.e., regular in a sense of weights), a better ratio can be obtained.

Corollary 2 *If all the $\delta_G(v)$'s are equal for the input graph, algorithm **ALGMMO** is a $(2 - w_{\min}/\lceil L(G) \rceil)$ -approximation algorithm.*

Proof. The proof is very similar to that of Lemma 5. If all the $\delta_G(v)$'s are equal, the algorithm skips Step 1 and the output (an orientation) of the algorithm is determined in only Steps 3 and 4.

In the above proof of Theorem 6, we showed that either (1) for every vertex v processed in Steps 3 and 4, $\delta_\Lambda^+(v) \leq \lceil 2\ell(G_j) \rceil - w_{\min}$, or (2) Λ is optimal. Therefore,

$$\begin{aligned} \Delta_\Lambda(G) &= \max_{u \in V} \{\delta_\Lambda^+(u)\} \\ &\leq \lceil 2\ell(G_j) \rceil - w_{\min} \\ &\leq (2 - \frac{w_{\min}}{\lceil \ell(G_j) \rceil}) \lceil \ell(G_j) \rceil \\ &\leq (2 - \frac{w_{\min}}{\lceil L(G) \rceil}) \Delta_{OPT}(G). \end{aligned}$$

\square

Remark. There is an example showing that the analysis is tight. That is, when run on the instance, **ALGMMO** outputs an orientation whose maximum weighted outdegree is at least $(2 - 1/L(G))\Delta_{OPT}(G)$.

Consider a weighted graph G with n vertices v_0, v_1, \dots, v_{n-1} illustrated in Figure 6. Edges of G are $\{v_i, v_{i+1}\}$ with weight f for $1 \leq i \leq n-2$, $\{v_{n-1}, v_1\}$ with weight f , and $\{v_0, v_i\}$ with weight 1 for $1 \leq i \leq n-1$. Here f has to meet the condition $2f + 1 < n - 1$. For this graph, $\ell(G) = L(G) = (f + 1)(n - 1)/n$ that derives $L(G) < f + 1$.

Since $\lceil 2\ell(G) \rceil - 1 \geq 2f + 1$, the algorithm first determines directions of three edges, say, $\{v_1, v_2\}$, $\{v_2, v_3\}$, and $\{v_0, v_2\}$ for a vertex v_2 as (v_2, v_1) , (v_2, v_3) , and (v_2, v_0) . Therefore, in the final orientation Λ obtained, $\delta_\Lambda^+(v_2) = 2f + 1$ and hence $\Delta_\Lambda(G) \geq 2f + 1$.

Consider an orientation $\Gamma = \{(v_i, v_{i+1}) \mid 1 \leq i \leq n-2\} \cup \{(v_{n-1}, v_1)\} \cup \{(v_i, v_0) \mid 1 \leq i \leq n-1\}$. We can easily observe that $\Delta_\Gamma(G) = f + 1$. Therefore,

$$\frac{\Delta_\Lambda(G)}{\Delta_{OPT}(G)} \geq \frac{\Delta_\Lambda(G)}{\Delta_\Gamma(G)} \geq 2 - \frac{1}{f + 1} > 2 - \frac{1}{L(G)}.$$

6. Conclusion

9. G. Gallo, M.D. Grigoriadis, and R.E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM J. Computing* **18** (1989) 30–55.
10. E. Horowitz and S. Sahni, "Exact and approximate algorithms for scheduling non-identical processors," *J. ACM* **23** (1976) 317–327.
11. J. Kára, J. Kratochvíl, and D.R. Wood, "On the complexity of the balanced vertex ordering problem," in *Proc. COCOON2005, LNCS 3595* (2005) 849–858.
12. J.-C. König, D.W. Krumme, and E. Lazard, "Diameter-preserving orientations of the torus," *Networks* **32** (1998) 1–11.
13. J.K. Lenstra, D.B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Mathematical Programming* **46** (1990) 259–271.
14. C.St.J.A. Nash-Williams, "On orientations, connectivity and odd vertex pairings in finite graphs," *Canadian J. Math.* **12** (1960) 555–567.
15. J. O'Rourke, *Art Gallery Theorems and Algorithms*, (Oxford University Press, 1987)
16. M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2nd Ed., (Prentice Hall, 2002)
17. H.E. Robbins, "A theorem on graphs with an application to a problem of traffic control," *American Math. Monthly* **46** (1939) 281–283.
18. P. Schuurman and G.J. Woeginger, "Polynomial time approximation algorithms for machine scheduling: Ten open problems," *J. Scheduling* **2** (1999) 203–213.
19. V. Venkateswaran, "Minimizing maximum indegree," *Discrete Applied Mathematics* **143** (2004) 374–378.