ELTE Eötvös Loránd University

Faculty of Science

Restricted feedback arc set and vertex-ordering problems

Math project III. report

Nóra Anna Borsik
Applied Mathematics MSc

Supervisor:
Péter Madarasi

ELTE Institute of Mathematics
Department of Operations Research

Budapest, 2023

1 Introduction

In the first two semesters, we considered the following vertex-ordering problem for directed graphs.

Problem 1 **($(f, g; \sum w)$-FAS problem)** *Let us given a loop-free digraph $D = (V, A)$ with a weight function $w : A \to \mathbb{R}_+$ on the arcs, a lower bound function $f : V \to \mathbb{R}_+$ and an upper bound function $g : V \to \mathbb{R}_+$ on the vertices. Our goal is to decide whether there exists an order of the vertices such that*

$$f(v) \leq \overleftarrow{\delta}^w(v) \leq g(v)$$

*holds for every vertex $v \in D$, where $\overleftarrow{\delta}^w(v)$ denotes the left weighted out-degree of v according to the order. In the unweighted case, the problem is referred to as the (f, g)-*FAS problem*.*

In [10], the authors considered the analogous problem for undirected graphs, called (f, g)-bounded ordering problem.

This semester, we focused on some vertex-ordering problems on undirected graphs. We put special emphasis on the following ordering problems, in which the left degree vector of an optimal ordering is required to be "smooth", "equitable" or "egalitarian".

Problem 2 **(*dec-min* (decreasingly minimal) ordering problem)** *Let us given a graph $G = (V, E)$. Our goal is to find a vertex order of G in which the left degree vector sorted in non-increasing order is lexicographically minimal.*

Problem 3 **(*inc-max* (increasingly maximal) ordering problem)** *Let us given a graph $G = (V, E)$. Our goal is to find a vertex order of G in which the left degree vector sorted in non-decreasing order is lexicographically maximal.*

Problem 4 **($\min \sum_{v \in V} h(\varrho(v))$ ordering problem)** *Let us given a graph $G = (V, E)$ and a discrete strictly convex function $h : \mathbb{Z}_+ \to \mathbb{R}$ (i.e. $h(z + 2) + h(z) > 2h(z + 1)$ holds for any $z \in \mathbb{Z}_+$). Our goal is to find a vertex order of G which minimizes $\sum_{v \in V} h(\overleftarrow{d}(v))$, where $\overleftarrow{d}(v)$ denotes the left degree of $v \in V$.*

The non-acyclic counterparts of these types of problems are called egalitarian orientations in the literature and were recently investigated in [4, 6, 7].

We also considered the following vertex-ordering problem.

Problem 5 **($\max \sum_{v \in V} \overleftarrow{d}(v)\overrightarrow{d}(v)$ ordering problem)** *Let us given a graph $G = (V, E)$. Our goal is to find a vertex order of G which maximizes $\sum_{v \in V} \overleftarrow{d}(v)\overrightarrow{d}(v)$, where $\overleftarrow{d}(v)$ and $\overleftarrow{d}(v)$ denote the left and the right degree of v.*

This problem is significantly different from the previous egalitarian problems, since, instead of only focusing on the left degree vector, the optimal order minimizes the difference between the left degree vector and the right degree vector, in a certain sense.

It is important to note that each vertex-ordering problem above can be equivalently rephrased as an acyclic orientation problem. From this point of view, instead of vertex orders, we search for an acyclic orientation, and replace the left degrees $\overleftarrow{d}$ with the in-degrees ϱ, and the right degrees $\overrightarrow{d}$ with the out-degrees δ in the definitions of the problems above. An order is optimal to the vertex-ordering versions of the problems if and only if the acyclic orientation obtained by orienting each edge from left to right is optimal to the corresponding acyclic orientation problem. To see this, we need to observe that the in-degree (and out-degree) vector of an acyclic orientation is the same as the left degree (and right degree) vector of its topological orderings.

In this report, we first summarize the results of the previous semesters, then we move on to the topics considered this semester.

2 Results of the previous semesters

First, we summarize the results for the $(f, g; \sum w)$-FAS problem and a more general ordering problem, then we briefly present some applications.

2.1 Complexity of the $(f, g; \sum w)$-FAS problem

The $(f, g; \sum w)$-FAS problem is polynomial-time solvable if only upper bounds are given on the vertices. To see this, consider the algorithm which, in each step, fixes a vertex with out-degree at most $g(v)$ at the last free place, and deletes this vertex from the digraph. If the algorithm finds an order of the vertices, then it is clearly a solution. Otherwise, we proved that no solution exists. This implies that there exists a solution to the $(-\infty, g; \sum w)$-FAS problem if and only if there is no induced subgraph in which the weighted out-degree of v restricted to the subgraph is strictly greater than $g(v)$ for each vertex v. A similar algorithm and characterization can be given for the problem with only lower bounds on each vertex. Furthermore, it is not hard to prove that the $(f, g; \sum w)$-FAS problem remains solvable if there are both lower and upper bounds, but on each vertex either only lower or only upper bound is given.

Because of the simplicity of the algorithm, the question arises whether some slightly modified or generalized versions of the $(-\infty, g; \sum w)$-FAS problem remain solvable. We proved that the $(f, g; \sum w)$-FAS problem is NP-complete when we have only upper bounds except for one vertex for which both lower and upper bounds are given, even for simple graphs. A similar, natural modification is when the arc weights are not necessarily non-negative. We proved that the resulting problem is also NP-complete.

We also considered the (f, g)-FAS problem (without arc weights) in some special bounded cases, for example if the lower bound is equal to the upper bound on each vertex. We proved that this problem is NP-complete, even if the maximum degree in the underlying graph is at most 6, and the bounds $f \equiv g$ are 0 on one vertex and 1 on all other vertices.

The other extreme case is when there is a large difference between the lower and the upper bounds on each vertex. Namely, we are given a first and a last vertex denoted by s and t, respectively, with bounds $f(s) = g(s) = 0$ and $f(t) = g(t) = \delta(t)$, and on each vertex $v \neq \{s, t\}$, let the bounds be $f(v) = a$ and $g(v) = \delta(v) - b$ for some given non-negative integers a and b.

This problem is equivalent to ordering the vertices such that each vertex has at least a outgoing arcs to the preceding vertices and at least b outgoing arcs to the succeeding vertices, except for s and t. If the parameters are $a = b = 1$, then the problem is the so-called s-t numbering problem for directed graphs, which is known to be polynomial-time solvable [5]. If $a = b = 2$, then the problem becomes NP-complete even for symmetric digraphs [10], there proof also implies that it is also NP-complete for any parameters $a \geq 2$ and $b \geq 2$. We proved that the problem is NP-complete in the remaining case with parameters $a = 1$ and $b = 2$.

2.2 Vertex-ordering problems with simultaneous bounds

Motivated by the previous special bounded case, we considered vertex-ordering problems with simultaneous bounds for the left out-degree $\overleftarrow{\delta}$, right out-degree $\overrightarrow{\delta}$, left in-degree $\overleftarrow{\varrho}$ and right in-degree $\overrightarrow{\varrho}$ of each vertex. Notice that a lower/upper bound on the right out-degree is equivalent to an upper/lower bound on the left out-degree and similarly a lower/upper bound on the right in-degree is equivalent to an upper/lower bound to the left in-degree. So it is enough to consider those cases where the bounds are given for the left out-degree and the left in-degree of each vertex. We gave a full complexity analysis in the case when two simultaneous (lower, upper or exact) bounds are given. The results are summarized in Table 1. The diagonal of the table contains the complexities of the (f, g)-FAS problems with only lower bound, only upper bound or exact prescription for the degrees.

	$\overleftarrow{\delta} \leq g_\delta$	$\overleftarrow{\delta} \geq f_\delta$	$\overleftarrow{\delta} = m_\delta$	$\vec{\varrho} \leq g_\varrho$	$\vec{\varrho} \geq f_\varrho$	$\vec{\varrho} = m_\varrho$
$\overleftarrow{\delta} \leq g_\delta$	P	NP-c	NP-c	P	NP-c	NP-c
$\overleftarrow{\delta} \geq f_\delta$		P	NP-c	NP-c	P	NP-c
$\overleftarrow{\delta} = m_\delta$			NP-c	NP-c	NP-c	P
$\vec{\varrho} \leq g_\varrho$				P	NP-c	NP-c
$\vec{\varrho} \geq f_\varrho$					P	NP-c
$\vec{\varrho} = m_\varrho$						NP-c

Table 1: The complexity landscape of the vertex-ordering problems with two simultaneous (lower, upper or exact) bounds, for the left out-degree $\overleftarrow{\delta}$, and left in-degree $\overleftarrow{\varrho}$ of each vertex. The lower bounds, upper bounds and exact prescriptions are always denoted by f, g and m, respectively.

We also considered cases with more than two simultaneous bounds. Surprisingly, the problem with exact prescription for the left out-degree, left in-degree, right out-degree and right in-degree turned out to be polynomial-time solvable.

2.3 Generalization: $(f, g; h)$-bounded ordering problem

We introduced the $(f, g; h)$-ordering problem as a generalization of the $(f, g; \sum w)$-FAS problem.

Problem 6 ($(f, g; h)$-ordering problem) *Let us given a ground set V and, for each element $v \in V$, a set-function $h_v : 2^{V-v} \to \mathbb{R}$ which is non-decreasing (that is, $h_v(A) \leq h_v(B)$ holds for all subsets $A \subseteq B \subseteq V - v$). Assume that these functions can be evaluated in polynomial time. Furthermore, we are given lower and upper bound functions $f : V \to \mathbb{R}$ and $g : V \to \mathbb{R}$. Our goal is to decide whether there exists an order σ of V such that*

$$f(v) \leq h_v(\overleftarrow{\sigma}(v)) \leq g(v)$$

holds for each element $v \in V$, where $\overleftarrow{\sigma}(v)$ denotes the set of the elements preceding v according to the order σ.

Observe that the $(f, g; \sum w)$-FAS problem is a special case of the $(f, g; h)$-ordering problem. To show this, let V be the vertex set of the digraph, and for each vertex v and subset $V' \subseteq V - v$, let

$$h_v(V') = \sum_{e \in \delta(v, V')} w(e),$$

where $\delta(v, V')$ denotes the set of the outgoing arcs from v to V'. By definition, the solutions to the $(f, g; \sum w)$-FAS problem are exactly the $(f, g; h)$-orders.

We gave a polynomial-time algorithm for the $(-\infty, g; h)$-ordering problem by generalizing the algorithm given for the $(-\infty, g; \sum w)$-FAS problem.

2.4 Applications

2.4.1 Partitioning into an in-branching and an acyclic subgraph

As an application of the $(-\infty, g)$-FAS problem, one obtains a polynomial-time algorithm for deciding whether a digraph can be partitioned into an in-branching and an acyclic subgraph. We showed that there exists such a partition if and only if the $(-\infty, g)$-FAS problem with upper bound $g \equiv 1$

is solvable, which means that there is no induced subgraph in which the in-degree of the vertices restricted to the subgraph is at least 2. However, we proved that partitioning into a minimum-size in-branching and an acyclic subgraph is NP-hard.

Partitioning into an in-branching and an acyclic subgraph is also polynomial-time solvable in a more general case, when some vertices are required to be roots in the in-branching (it is possible that some other vertices will be also roots). If we require that the in-branching has exactly one root, that is, we want to partition into an in-arborescence and acyclic subgraph, then our approach does not work, and the complexity remains open. Note that partitioning into an in-arborescence and a spanning acyclic subgraph is known to be NP-complete [1]. We showed that partitioning into a minimum cost in-arborescence and an acyclic subgraph is NP-hard, even with 0-1 edge costs.

Similar arc-partitioning problems A recent paper investigates similar arc-partitioning problems [1]. For example, it was proven that deciding whether a digraph can be partitioned into a directed cycle and an acyclic subgraph, or into a directed 2-factor and an acyclic subgraph are NP-complete problems. We considered the partitioning problem into a matching and an acyclic subgraph, and we proved that this problem is NP-complete, even in case of bipartite graphs. The same result holds for perfect matchings.

2.4.2 A rank-aggregation problem

Consider a competition, where different judges give complete rankings (orders) of the candidates and our goal is to find a common ranking which is a “fair” consensus between the judges. In the Kemény rank-aggregation problem, the distance of two rankings is defined as the number of those pairs of candidates whose order is reversed in the two rankings [9]. The goal is to find a common ranking minimizing the sum of the distances from the rankings given by the judges. In another rank-aggregation problem, we want to find a ranking which is closest to the farthest ranking, that is, we want to minimize the maximum distance. It is known that both problems are NP-hard [2]. We introduced a similar problem, where we defined the distance from the viewpoint of the candidates instead of the judges: For a candidate v, let $\mathrm{dist}(v)$ denote the number of the candidates whom v precedes by the majority of the judges, but not in the common ranking. This measures in a natural way how unfair the common ranking seems to the candidate v. Our goal is to find a common ranking minimizing the largest distance $\mathrm{dist}(v)$ over all candidates. This problem can be easily rephrased as ordering the vertices of a directed graph such that the maximum left out-degree is minimized, which is solvable by finding the smallest positive integer c for which the $(-\infty, g)$-FAS problem with upper bound $g \equiv c$ has a feasible solution.

2.4.3 Scheduling problems

$1|g$-prec$|L_{\max}$ problem Consider the following single-machine scheduling problem: Let us given a set V of jobs and some precedence constrains between the jobs. These precedence constrains form a digraph D, which is not required to be acyclic. Each job v has a processing time $p(v)$, a deadline $d(v)$ and an upper bound $g(v)$ for the number of the violated precedence constraints for v. A schedule is feasible if, for each job v, there are at most $g(v)$ outgoing arcs from v to the set V' of the jobs preceding v. Our goal is to find a feasible schedule which minimizes the maximum lateness of the jobs. (Where the lateness of a job is the difference of its finishing time and its deadline if it is positive and zero otherwise.)

First, consider the problem of deciding whether there exists a feasible schedule with maximum lateness at most K. This question is only interesting for $K < \sum_{v\in V} p(v)$, because otherwise each feasible schedule without idle times has maximum lateness at most K.

Let
$$h_v(V') = M\Big(\delta(v, V') - g(v)\Big)^+ + \Big(\sum_{u\in V'} p(u) + p(v) - d(v)\Big)^+,$$

where M is sufficiently large constant, for example $K+1$. It can be seen that these h_v set-functions are non-decreasing.

We showed that there exists a feasible schedule with maximum lateness at most K if and only if there exists a $(f, g; h)$-order of the jobs for $g \equiv K$ and the set-functions defined above.

Therefore, minimizing the maximum lateness is equivalent to finding an $(f, g; h)$-order σ which minimizes the maximum $h_v(\overleftarrow{\sigma}(v))$ for all $v \in V$. This can be minimized by finding the lowest K using binary search such that there exists a $(-\infty, g; h)$-order with upper bound $g \equiv K$. This approach calls the algorithm for the $(-\infty, g; h)$-ordering problem $O(\log M)$ times and solves the scheduling problem in polynomial time, but not in strongly polynomial time. We showed that we can minimize the maximum $h_v(\overleftarrow{\sigma}(v))$ for all $v \in V$ by calling a slightly modified version of the algorithm, which solves the scheduling problem in strongly polynomial time.

$1|g$-prec$|f_{\max}$ This scheduling problem is a generalization of the $1|g$-prec$|L_{\max}$ scheduling problem. In this more general problem, there is given a non-decreasing function $f_v : \mathbb{R}_+ \to \mathbb{R}_+$ for each job $v \in V$ instead of the deadline $d(v)$. Our goal is to find a feasible schedule minimizing $\max\{f_v(t(v))\}$, where $t(v)$ denotes the finishing time of v in the schedule.

This problem is also polynomial-time solvable similarly to the $1|g$-prec$|L_{\max}$ problem. The only difference is that the h_v set-functions are defined as follows:

$$h_v(V') = M\Big(\delta(v, V') - g(v)\Big)^+ + f_v\Big(\sum_{u \in V'} p(u) + p(v)\Big).$$

3 This semester: Egalitarian vertex-ordering problems

From now on, we describe the results achieved in this semester about egalitarian vertex-ordering problems. We focus on Problems 2-5, defined in Section 1. We emphasize that the inputs of these problems are undirected graphs, unlike the input of the $(f, g; \sum w)$-FAS problem.

3.1 Lexicographically optimal vertex orderings

First, we recall the definitions of the dec-min and inc-max problems. Let us given a graph $G = (V, E)$. A vertex order is optimal for the dec-min (decreasingly minimal) problem if the sequence of the left degrees sorted in non-increasing order is lexicographically minimal. Similarly, a vertex order is optimal for the inc-max (increasingly maximal) problem if the sequence of the left degrees sorted in non-decreasing order is lexicographically maximal.

The dec-min or inc-max ordering problems are equivalent to finding a dec-min or inc-max acyclic orientation of G, respectively. It is known that without acyclicity the dec-min and inc-max orientation problems are equivalent [4, 6, 7]. So the question arises whether these problems are also

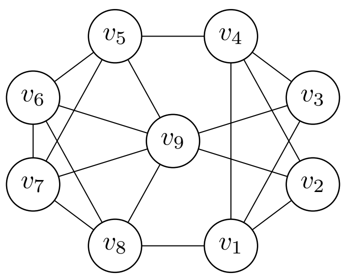

Figure 1: The smallest simple graph for which the dec-min and inc-max problems are not the same.

equivalent in the acyclic case. With a program, we found that the example shown in Figure 1 is the smallest simple graph for which the dec-min and inc-max ordering problems are different.

We considered the complexities of the dec-min and inc-max ordering problems in case of k-bounded orders. An order is called k-bounded if the left degree of each vertex is at most k.

The NP-hardness of finding a dec-min k-bounded order was already proven for all odd $k \geq 5$ and in the case when there is no bound on the left degrees, even for simple graphs [4]. We proved that the dec-min k-bounded ordering problem is NP-hard for all $k \geq 3$, even in case of simple graphs. Moreover, we proved the analogous result for the inc-max ordering problem. Note that, unlike for dec-min orders, the complexity of the k-bounded case does not imply the complexity of the inc-max problem without bounds on the left degrees, because it can happen that a graph has a k-bounded order, but the optimal order for the inc-max problem is not k-bounded, see the graph shown in Figure 1 for an example. We proved that finding an inc-max problem without bounds on the left degrees is NP-hard as well, even for simple graphs.

We also considered the problem for smaller k parameters. In the case of $k = 1$, the problems are polynomial-time solvable, and in the case of $k = 2$, we showed that the dec-min k-bounded and inc-max k-bounded orders are the same, but the complexity of these problems remains open.

One can also define two natural counterparts, the inc-min and dec-max problems: A vertex order is optimal for the inc-min (increasingly minimal) problem if the sequence of the left degrees sorted in non-decreasing order is lexicographically minimal. Similarly, a vertex order is optimal for the dec-max (decreasingly maximal) problem if the sequence of the left degrees sorted in non-increasing order is lexicographically maximal.

We proved that the inc-min ordering (i.e. acyclic orientation) problem is also NP-hard, regardless of whether we require that the orientation is acyclic. We showed this even in case of 3-regular graphs. In the case of 3-regular graphs, an orientation is an inc-min acyclic orientation if and only if the reverse orientation is a dec-max acyclic orientation. To see this, observe that reversing an orientation preserves the acyclicity, and if the in-degree of v is $\varrho(v)$ in an orientation, then the in-degree of v is $3 - \varrho(v)$ in the reversed orientation. The same argument holds for the analogous problems without acyclicity. So the NP-hardness of the inc-min ordering and orientation problems immediately implies the NP-hardness of the dec-max ordering and orientation problems.

Table 2 summarizes the complexities of the different orientation and acyclic orientation (i.e. ordering) problems:

	dec-min	inc-max	inc-min	dec-max
orientation	P [4]	P [4, 7]	NP-H	NP-H
acyclic orientation	NP-H [4]	NP-H	NP-H	NP-H

Table 2: The complexities of the lexicographical orientation problems without in-degree bound.

3.2 Minimizing $\sum_{v \in V} h(\overleftarrow{d}(v))$

We also studied the problem of finding a vertex order for a given graph $G = (V, E)$ minimizing $\sum_{v \in V} h(\overleftarrow{d}(v))$ for some discrete strictly convex function $h : \mathbb{Z}_+ \to \mathbb{R}$ (i.e. $h(z+2)+h(z) > 2h(z+1)$ holds for any $z \in \mathbb{Z}_+$). In other words, this is equivalent to finding an acyclic orientation of G minimizing $\sum_{v \in V} h(\varrho(v))$. András Frank and Kazuo Murota investigated the analogous question without acyclicity, that is, finding an orientation of G that minimizes $\sum_{v \in V} h(\varrho(v))$. They showed that the exact same orientations are optimal for any discrete strictly convex function h, furthermore, these are also optimal for the (non-acyclic) dec-min and inc-max orientation problems [7]. Moreover, the optimal orientations can be found in strongly polynomial time [4, 6].

3.2.1 Hardness

Under the acyclicity constraint, however, the optimal solutions to different discrete strictly convex functions do not coincide anymore. To see this, it is enough to notice that the dec-min and inc-max ordering (i.e. acyclic orientation) problems are special cases of the $\sum_{v\in V} h(\overleftarrow{d}(v))$ ordering (i.e. acyclic orientation) problem. It is not difficult to see that we obtain the dec-min ordering problem for $h(z) = |V|^z$, and the inc-max ordering problem for $h(z) = |V|^{-z}$. We already gave a simple graph in Figure 1 for which these two problems do not coincide.

The complexity of the $\sum_{v\in V} h(\overleftarrow{d}(v))$ ordering problem for different strictly convex functions is a natural question. We proved that the $\sum_{v\in V} h(\overleftarrow{d}(v))$ ordering (i.e. acyclic orientation) problem becomes NP-hard for *any* discrete strictly convex function h — provided that parallel edges are also allowed. We proved this by first proving the hardness for a special class of multigraphs containing loops, which have a vertex order in which each left degree is d, $d+1$, or $d+2$. We proved that the optimal orders for different strictly convex functions coincide for such multigraphs. Moreover, the only optimal orders are the ones in which each left degree is d, $d+1$ or $d+2$. After proving the NP-hardness of the problem in this special case, we managed to eliminate the loops using parallel arcs. However, the complexity remains open in case of simple graphs. In the special cases of the dec-min, inc-max and $h(z) = z^2$, we also proved the hardness in case of simple graphs.

3.2.2 Generalization: Minimizing $\sum_{v\in V} h_v(\overleftarrow{d}(v))$

We also considered the following more general problem in which, instead of a single function h, there is given a function h_v for each $v \in V$. Let us given a multigraph $G = (V, E)$ and a discrete (not necessarily strictly convex) function $h_v : \mathbb{Z}_+ \to \mathbb{R}$ for each $v \in V$. Assume that the function h_v can be evaluated in polynomial time for each $v \in V$. Our goal is to find an order minimizing $\sum_{v\in V} h_v(\overleftarrow{d}(v))$.

We gave an exact dynamic programming algorithm for this more general problem.

Dynamic programming algorithm The natural approach to solve this problem is to try all $|V|!$ permutations of the vertices, and choose one minimizing the objective value. We gave a dynamic programming algorithm for finding an order minimizing $\sum_{v\in V} h_v(\overleftarrow{d}(v))$, which takes $O(2^{|V|}\text{poly}(|V|, |E|))$ steps. Let $f(\emptyset) = 0$. For each $\emptyset \neq V' \subseteq V$, in non-decreasing order by $|V'|$, compute the values

$$f(V') = \min_{v\in V'}\{f(V'-v) + h_v(d(v, V'))\},$$

$$g(V') = \arg\min_{v\in V'}\{f(V'-v) + h_v(d(v, V'))\}.$$

After that, construct the optimal order by repeating the following step until no vertex remains: Put $g(V)$ at the last free place of the order, and delete v from the graph. The running time of this algorithm is clearly $O(2^{|V|}\text{poly}(|V|, |E|))$, because we assumed that the h_v functions can be evaluated in polynomial time, so the computation of $f(V')$ and $g(V')$ takes $\text{poly}(|V|, |E|)$ time for each subset.

Linear functions We showed that the $\sum_{v\in V} h_v(\overleftarrow{d}(v))$ problem is polynomial-time solvable if h_v is a linear function for each $v \in V$. In this case, ordering the vertices in non-increasing order by the slope of h_v minimizes $\sum_{v\in V} h_v(\overleftarrow{d}(v))$. This implies that the strict convexity of the function h is a necessary condition in the NP-hardness theorem in Section 3.2.1, because the $\sum_{v\in V} h(\overleftarrow{d}(v))$ problem is polynomial-time solvable if h is a linear function. Moreover, any vertex order is optimal in this case.

Without acyclicity We have already mentioned that finding a (not necessarily acyclic) orientation of G minimizing $\sum_{v\in V} h(\varrho(v))$ is polynomial-time solvable [6, 7]. We gave a polynomial-time method for a more general orientation problem, in which each vertex v has its own discrete strictly convex function h_v. Given a multigraph $G = (V, E)$ along with a discrete strictly convex function $h_v : \mathbb{Z}_+ \to \mathbb{R}$ for each $v \in V$, find an orientation of G that minimizes $\sum_{v \in V} h_v(\varrho(v))$. We get back the previous problem when we set $h_v = h$ for every $v \in V$. It is easy to see that if the h_v functions differ from each other, then the optimal solutions are not necessarily the dec-min orders anymore. Assuming that the function h_v can be evaluated in polynomial time, we reduced this problem to the minimum cost flow problem, which can be solved in strongly polynomial time [13].

3.3 A special case: Minimizing $\sum_{v \in V} \overleftarrow{d}^2(v)$

We paid special attention to the problem of minimizing $\sum_{v \in V} \overleftarrow{d}^2(v)$. This problem is one of the most natural special cases of minimizing $\sum_{v \in V} h(\overleftarrow{d}(v))$, which we obtain by setting $h(z) = z^2$. We analyzed the approximation ratio of the following greedy algorithm for the problem of finding an order minimizing the square-sum of the left degrees.

The algorithm repeats the following, until no vertex remains: It fixes a vertex with minimum degree at the last free position and deletes it from the graph. Note that there can be multiple vertices with minimum degree, so the output of the algorithm depends on which minimum-degree vertex we choose.

We proved that this algorithm gives a $\min\{4H_n, k_{\min}\}$-approximating order, where H_n is the n^{th} harmonic number and $k_{\min}$ is the minimum integer k for which the graph has a k-bounded order.

It remains open whether the greedy algorithm approximates the problem within a constant factor. In case of simple graphs, the best lower bound that we found for the approximation ratio is $\frac{9}{7}$, which is the limit of the worst-case approximation ratios of the greedy algorithm for the sequence of graphs shown in Figure 2.

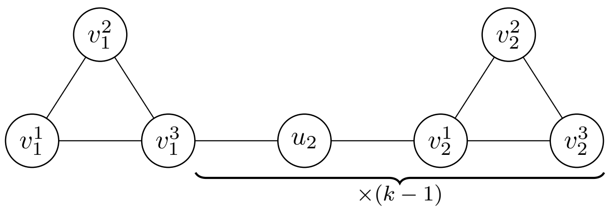

Figure 2: The graph sequence G_k for which the approximation ratio of the greedy algorithm tends to $\frac{9}{7}$. Here k denotes the number of the triangles, and v_i^j denotes the i^{th} vertex of the j^{th} triangle, for each $i \in \{1, \dots, k\}$ and $j \in \{1 \dots, 3\}$, and u_i denotes the vertex connecting v_{i-1}^3 and v_i^1 for each $i \in \{2, \dots, k\}$.

We tested the worst-case approximation ratio of the greedy algorithm for every simple graph on at most 12 vertices (using the dynamic programming algorithm from Section 3.2.2), and found that the approximation ratio for the graph G_k, shown in Figure 2, is an upper bound for the approximation ratio for every simple graph on at most $4k - 1$ vertices. It is an open question whether the approximation ratio of the greedy algorithm is $\frac{9}{7}$ for simple graphs.

3.3.1 Application: Routing protocol

In routing problems, there are given an undirected graph called network and some packets with start and end destinations. The vertices of the network represent switches with input buffers and the edges represent links. Our goal is to determine a route for every packet. In a general step, every switch forwards the current packet from their input buffer to the next switch in the packets route provided that the input buffer of that switch is not full. Such networks can suffer from deadlock, when there are several packets waiting along a cycle.

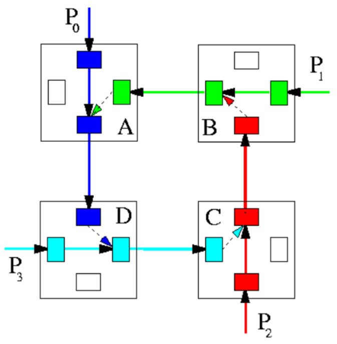

Figure 3: A deadlock with four packets.

The goal is to determine a route for each packet without causing a deadlock. It is known that deadlocks can be prevented by the following method called Up/Down routing [11, 12, 14]: In an arbitrary order of the switches, orient every edge from left-to right. Forbid the packets to go through the sub-paths formed by any two edges directed into the same vertex.

Ordering the switches in a dec-min order minimizes the maximum number of forbidden sub-paths at a switch. Similarly, ordering the switches in a min $\sum_{v\in V} \overleftarrow{d}(v)$ order minimizes the number of forbidden sub-paths. Moreover, if we only search for such orders for which the resulting directed graph is rooted connected, then there exists a feasible route between every two switches.

3.4 Maximizing $\sum_{v\in V} \overleftarrow{d}(v)\vec{d}(v)$

We also considered another notion of "equitable" orientations. Our goal is to find an order of the vertices of a graph $G = (V, E)$ maximizing $\sum_{v\in V} \overleftarrow{d}(v)\vec{d}(v)$. The optimal orderings are equitable in the sense that the product $\overleftarrow{d}(v)\vec{d}(v)$ is maximal for a single vertex v if the neighbors of v are evenly distributed to the two sides of v.

3.4.1 Complexity results

The problem is clearly equivalent to finding an acyclic orientation of G maximizing $\sum_{v\in V} \varrho(v)\delta(v)$. Without the acyclicity condition, the optimal orientations are either the Eulerian (that is, $\varrho(v) = \delta(v)$ for every $v \in V$) or the almost-Eulerian (that is, $|\varrho(v)-\delta(v)| \le 1$ for every $v \in V$) orientations. It is well known that every graph has an Eulerian or almost-Eulerian orientation and we can compute such orientations in polynomial time, hence the problem is polynomial-time solvable.

However, the acyclicity condition makes the problem much more difficult. In [3], the authors considered similar ordering problems in which they wanted to find an order minimizing $\sum_{v\in V} |\overleftarrow{d}(v)-\vec{d}(v)|$. They called an order *perfectly balanced* if $|\overleftarrow{d}(v) - \vec{d}(v)| \le 1$ holds for every vertex v, and proved that deciding whether a given graph has a perfectly balanced vertex order is NP-complete, even for bipartite graphs with maximum degree 6. In [8], it was proven NP-complete even for planar graphs with maximum degree at most 4 and for 5-regular graphs.

For a single vertex v, the value of $\overleftarrow{d}(v)\vec{d}(v)$ is maximized if and only if $|\overleftarrow{d}(v) - \vec{d}(v)| \le 1$. So if G has a perfectly balanced order, then exactly the perfectly balanced orders are maximizing $\sum_{v\in V} \overleftarrow{d}(v)\vec{d}(v)$. Therefore, by the NP-hardness of finding a perfectly balanced order, we obtain the NP-hardness of finding a vertex order maximizing $\sum_{v\in V} \overleftarrow{d}(v)\vec{d}(v)$.

For simple graphs with maximum degree at most 3, however, the problem of finding an order minimizing $\sum_{v\in V} |\overleftarrow{d}(v) - \vec{d}(v)|$ is polynomial-time solvable. The algorithm given in [3] finds an order minimizing the number of vertices for which $\overleftarrow{d}(v) = 0$ or $\vec{d}(v) = 0$. The main idea of the algorithm is, to compute an s-t order of every biconnected component of the graph and combine these orders into a complete vertex order. We generalized their algorithm for finding an order maximizing $\sum_{v\in V}\overleftarrow{d}(v)\overrightarrow{d}(v)$ in case of graphs with maximum degree at most three.

3.4.2 Approximation algorithm

We investigated the expected approximation ratio of a random order of the vertices for maximizing $\sum_{v\in V}\overleftarrow{d}(v)\overrightarrow{d}(v)$ in case of multigraphs, and proved that a random order is a 3-approximate solution in expectation.

We de-randomized the algorithm by fixing the vertices from right to left using the method of conditional expectations. In each step, we fix a vertex for which the conditional expectation of the objective function is the highest. It is easy to see that this algorithm also gives a 3-approximating order for the problem. The main difficulty was to compute the conditional expectations of the vertices. In case of simple graphs, we gave an explicit formula for the conditional expectations, but in case of multigraphs we were only able to compute them by a dynamic programming method.

4 Future plans and open questions

The most intriguing open question from the previous semesters is the complexity of partitioning into an in-arborescence and an acyclic subgraph.

From the topic of egalitarian orderings, there are also some interesting questions. We proved the dec-min and inc-max k bounded ordering problems NP-hard for any $k \geq 3$ and polynomial-time solvable for $k = 1$. But for $k = 2$ we only proved that the same orders are optimal for the dec-min 2-bounded and inc-max 2-bounded ordering problems, and the complexity of finding such an optimal order remains open. As one of our main result, we proved that the min $\sum_{v\in V} h(\overleftarrow{d}(v))$ ordering problem is NP-hard for any discrete strictly convex function $h : \mathbb{Z}_+ \to \mathbb{R}$ if parallel edges are allowed in the graph. However, the complexity remains open in case of simple graphs. For the special case of $h(z) = z^2$, we extended the NP-hardness proof for simple graphs. We examined the approximation ratio of a greedy algorithm for this special case. It is an open question, whether the greedy algorithm approximates the problem within a constant factor.

In this semester, we have been writing a paper from the result of the previous two semesters, in the topic of the $(f, g; \sum w)$-FAS ordering problem. I presented the results related to the egalitarian ordering problems on the faculty TDK conference, and we plan to submit it to the OTDK conference.

References

[1] J. Bang-Jensen, S. Bessy, D. Gonçalves, and L. Picasarri-Arrieta. Complexity of some arc-partition problems for digraphs. *Theoretical Computer Science*, 928:167–182, 2022.

[2] T. Biedl, F. J. Brandenburg, and X. Deng. On the complexity of crossings in permutations. *Discrete Mathematics*, 309(7):1813–1823, 2009.

[3] T. Biedl, T. Chan, Y. Ganjali, M. T. Hajiaghayi, and D. R. Wood. Balanced vertex-orderings of graphs. *Discrete Applied Mathematics*, 148(1):27–48, 2005.

[4] G. Borradaile, J. Iglesias, T. Migler, A. Ochoa, G. Wilfong, and L. Zhang. Egalitarian graph orientations. *Journal of Graph Algorithms and Applications*, 21(4):687–708, 2017.

[5] J. Cheriyan and J. H. Reif. Directed s-t numberings, rubber bands, and testing digraph k-vertex connectivity. *Combinatorica*, 14(4):435–451, 1994.

[6] A. Frank and K. Murota. Decreasing minimization on M-convex sets: algorithms and applications. *Mathematical Programming*, 195(1-2):1027–1068, 2022.

[7] A. Frank and K. Murota. Decreasing minimization on M-convex sets: background and structures. *Mathematical Programming*, 195(1-2):977–1025, 2022.

[8] J. Kára, J. Kratochvíl, and D. R. Wood. On the complexity of the balanced vertex ordering problem. In *International Computing and Combinatorics Conference*, pages 849–858. Springer, 2005.

[9] J. G. Kemény. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

[10] Z. Király and D. Pálvölgyi. Acyclic orientations with degree constraints. *arXiv preprint arXiv:1806.03426*, 2018.

[11] J. Carlos Sancho, A. Robles, and J. Duato. A new methodology to compute deadlock-free routing tables for irregular networks. In *International Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing*, pages 45–60. Springer, 2000.

[12] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker. Autonet: A high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9(8):1318–1335, 1991.

[13] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[14] V. W. Wittorff. Implementation of constraints to ensure deadlock avoidance in networks, May 12 2009. US Patent 7,532,584.