Local Search Breaks 1.75 for Graph Balancing

Klaus Jansen
Department of Computer Science, Christian-Albrechts-Universität, Kiel, Germany
kj@informatik.uni-kiel.de

Lars Rohwedder
Department of Computer Science, Christian-Albrechts-Universität, Kiel, Germany
lro@informatik.uni-kiel.de

Abstract

Graph Balancing is the problem of orienting the edges of a weighted multigraph so as to minimize the maximum weighted in-degree. Since the introduction of the problem the best algorithm known achieves an approximation ratio of 1.75 and it is based on rounding a linear program with this exact integrality gap. It is also known that there is no $(1.5 - \epsilon)$-approximation algorithm, unless P = NP. Can we do better than 1.75?

We prove that a different LP formulation, the configuration LP, has a strictly smaller integrality gap. Graph Balancing was the last one in a group of related problems from literature, for which it was open whether the configuration LP is stronger than previous, simple LP relaxations. We base our proof on a local search approach that has been applied successfully to the more general Restricted Assignment problem, which in turn is a prominent special case of makespan minimization on unrelated machines. With a number of technical novelties we are able to obtain a bound of 1.749 for the case of Graph Balancing. It is not clear whether the local search algorithm we present terminates in polynomial time, which means that the bound is non-constructive. However, it is a strong evidence that a better approximation algorithm is possible using the configuration LP and it allows the optimum to be estimated within a factor better than 1.75.

A particularly interesting aspect of our techniques is the way we handle small edges in the local search. We manage to exploit the configuration constraints enforced on small edges in the LP. This may be of interest to other problems such as Restricted Assignment as well.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases graph, approximation algorithm, scheduling, integrality gap, local search

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.74

Category Track A: Algorithms, Complexity and Games

Related Version A full version can be found at `https://arxiv.org/abs/1811.00955`.

Funding Research was supported by German Research Foundation (DFG) project JA 612/15-2.

1 Introduction

In this paper we consider weighted, undirected multigraphs that may contain loops. We write such a multigraph as $G = (V, E, r, w)$, where V is the set of vertices, E is the set of edge identities, and r is a function $E \to \{\{u, v\} : u, v \in V\}$ that defines the endpoints for every edge. Note that in the definition above we allow $u = v$, which describes a loop. E is often defined as a set of vertex pairs. We use the function r instead, since it avoids some issues due to multigraphs. The weight function $w : E \to \mathbb{R}_{>0}$ assigns positive weights to the edges. In the Graph Balancing problem we want to compute an orientation of the edges, i.e., one of the ways to turn the graph into a directed graph. The goal is to minimize the maximum weighted in-degree over all vertices, that is $\max_{v \in V} \sum_{e \in \delta^-(v)} w(e)$, where $\delta^-(v)$ are the incoming edges of vertex v in the resulting digraph. Apart from being an arguably natural problem, Graph Balancing has been of particular interest to the scheduling community.

© Klaus Jansen and Lars Rohwedder;
licensed under Creative Commons License CC-BY

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 74; pp. 74:1–74:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

It is one of the simplest special cases of makespan minimization on unrelated machines for which an inapproximability bound of $1.5-\epsilon$ is known, which is already the best that is known in the general problem. In the interpretation as a scheduling problem, machines correspond to vertices and jobs to edges, i.e., each job has only two potential machines to which it can be assigned. The problem was introduced by Ebenlendr, Krčál, and Sgall [9] and they gave a polynomial time 1.75-approximation for it. Indepenently and under a different name, Asahiro et al. [4] studied the problem and showed that no $(1.5-\epsilon)$-approximation is possible unless P = NP. The algorithm by Ebenlendr et al. rounds the solution of a particular linear programming formulation. This appears to be the best one can hope for using their techniques, since the ratio between integral optimum and fractional optimum of the LP, the integrality gap, can be arbitrarily close to 1.75 [9]. Using a completely different approach to [9], Huang and Ott developed a purely combinatorial algorithm for the problem [10]. With $5/3+4/21 \approx 1.857$, however, their approximation ratio is inferior to the original algorithm. Another algorithm for Graph Balancing, developed by Wang and Sitters [21], achieves an approximation ratio of $11/6 \approx 1.833$, i.e., also worse than the original, but notable for being simpler. For the special case of only two different edge weights, three independent groups found a tight 1.5-approximation [7, 10, 17].

A good candidate for a stronger linear program to that from [9] is the configuration LP. It was introduced by Bansal and Sviridenko for the more general problem Scheduling on Unrelated Machines and the closely related Santa Claus problem [5]. It is easy to show that this LP is at least as strong as the LP from [9] (see the same paper), i.e., the integrality gap must be at most 1.75 as well. The best lower bound known is 1.5 (see for instance [11], this holds even for the case of Graph Balancing). In recent literature, the configuration LP has enabled breakthroughs in the restricted variants for both of the problems above [3, 19]. The restricted variant of Scheduling on Unrelated Machines (also known as Restricted Assignment) can be seen as Graph Balancing with hyperedges. In particular, it contains the Graph Balancing problem as a special case. In this setting, the configuration LP was shown first to have an integrality gap of at most $33/17 \approx 1.941$ [19], which was improved to $11/6 \approx 1.833$ by us [13]. This non-constructive proof is by a local search algorithm that is not known to terminate in polynomial time. In this paper, we present a sophisticated local search algorithm for Graph Balancing and obtain the following result.

▶ **Theorem 1.** *The integrality gap of the configuration LP is at most* 1.749 *for* Graph Balancing.

In other words, it is stronger than the LP from [9]. Although this does not give a polynomial time approximation algorithm, it is strong evidence that such an algorithm can be developed using the configuration LP. Furthermore, the optimal solution can be estimated in polynomial time within a factor of $1.749+\epsilon$ for any $\epsilon > 0$ by approximating the configuration LP. We emphasize that the purpose of this paper is to show a separation between the configuration LP and the previously used LP relaxation. The constants in the proof are not optimized. We chose to keep the case analysis (which is already difficult) and constants as simple as possible instead of improving the third decimal place. A summary of results regarding the configuration LP is given in Table 1. In fact, for all of the problems except for Graph Balancing it was known whether the configuration LP improves over the previous state-of-the-art. Our work in [15] indicates that earlier bounds on the integrality gap of the configuration LP in related problems disregard many constraints enforced on small edges/jobs, but that without them the LP might be much weaker. More precisely, these proofs used only properties that are already enforced by a weaker configuration LP that allows small edges/jobs to appear fractionally in a configuration. This weaker LP, however, has an integrality gap strictly

Table 1 Integrality gap of the configuration LP for various problems.

	Lower bound	Upper bound
Scheduling on Unrelated Machines	2	2 [16]
⊃ Restricted Assignment	1.5	1.833.. [13]
⊃ Graph Balancing	1.5	**1.749**
⊃ Unrelated Graph Balancing	2 [20]	2
Santa Claus	∞ [5]	∞ [5]
⊃ Restricted Santa Claus	2	3.833.. [14, 8]
⊃ Max-Min Unrelated Graph Balancing	2	2 [20]

higher than 1.5, whereas for the configuration LP it is still open whether 1.5 is the correct answer. The backbone of our new proof is the utilization of these small edge constraints (see end of Section 3.1). This may be relevant to other related local search based proofs as well.

Other related work. The problem of minimizing the maximum out-degree is equivalent to the maximum in-degree. The very similar problem of maximizing the minimum in- or out-degree has been settled by Wiese and Verschae [20]. They gave a 2-approximation and this is the best possible assuming P $\neq$ NP. Surprisingly, this holds even in the unrelated case when the value of an edge may be different on each end. They do not use the configuration LP, but it is easy to also get a bound of 2 on its integrality gap using their ideas. For the restricted case (a special case of the unrelated one) this bound of 2 was already proven by [6]. We are not aware of any evidence that Graph Balancing is easier on simple weighted graphs (without multiedges and loops). The same reduction for the state-of-the-art lower bound holds even in that case. A number of recent publications deal with the important question on how related local search algorithms can be turned into efficient algorithms [1, 2, 12, 18].

2 Preliminaries

Notation. For some $v \in V$ we will denote by $\delta(v)$ the incident edges, i.e. those $e \in E$ with $v \in r(e)$. When a particular orientation is clear from the context, we will write $\delta^-(v)$ for the incoming edges and $\delta^+(v)$ for the outgoing edges of a vertex. For some $F \subseteq E$ we will denote by $\delta_F(v)$ the incident edges of v restricted to F and $\delta_F^-(v)$, $\delta_F^+(v)$ accordingly. For some $e \in E$ we will describe by $t(e) \in r(e)$ the vertex it is oriented towards and by $s(e) \in r(e)$ the vertex it is leaving. For a loop e, i.e., $r(e) = \{v\}$ for some $v \in V$, it always holds that $t(e) = s(e)$. For a subset of edges $S \subseteq E$ we will write $w(S)$ for $\sum_{e \in S} w(e)$ and similar for other functions over the edges.

LP relaxations. The following linear programs have no objective functions. Instead they parameterized by τ, the makespan. The optimum is the lowest τ for which it is feasible. This will be denoted by OPT^* (referring to the configuration LP in the rest of the paper). First we look at the assignment LP by Lenstra, Shmoys, and Tardos [16]. It has a variable $x_{e,v}$ for every vertex v and incident edge $e \in \delta(v)$, which indicates whether e is oriented towards v.

The assignment LP.

$$\sum_{e \in \delta(v)} w(e) \cdot x_{e,v} \leq \tau \quad \forall v \in V, \qquad \sum_{v \in r(e)} x_{e,v} = 1 \quad \forall e \in E, \qquad x_{e,v} \in [0,1]$$

The first constraint ensures that no vertex has more than weight τ of edges oriented towards it. The second one describes that each edge is oriented towards one vertex. The assignment LP has an integrality gap of 2. Let B denote the big edges e, which have $w(e) > 0.5 \cdot \tau$. It is clear that an integral orientation can assign at most one such edge to each vertex. Ebenlendr et al. [9] show that adding the constraint $\sum_{e \in \delta_B(v)} x_{e,v} \leq 1 \quad \forall v \in V$ improves the integrality gap to 1.75. They also give other LP relaxations, but show that none of them have an integrality gap strictly better than 1.75.

Now we will introduce the configuration LP. A configuration is a subset of edges that can be oriented towards a particular vertex without exceeding a particular makespan τ. Formally, we define the configurations of a vertex v and a makespan τ as $\mathcal{C}(v, \tau) := \{C \subseteq \delta(v) : w(C) \leq \tau\}$. The configuration LP now assigns fractions of configurations to each machine.

Primal of the configuration LP.	Dual of the configuration LP.
$\sum_{v \in V} \sum_{C \in \mathcal{C}(v,\tau)} x_{v,C} \leq 1 \quad \forall v \in V$	$\min \sum_{v \in V} y_v - \sum_{e \in E} z_e$
$\sum_{v \in r(e)} \sum_{C \in \mathcal{C}(v,\tau): e \in C} x_{v,C} \geq 1 \quad \forall e \in E$	s.t. $\sum_{e \in C} z_e \leq y_v \quad \forall v \in V, C \in \mathcal{C}(v,\tau)$
$x_{v,C} \geq 0$	$y, z \geq 0$

Although the configuration LP has exponential size, a $(1+\epsilon)$-approximation can be computed in polynomial time for every $\epsilon > 0$ [5]. We are particularly interested in the dual of the configuration LP (which is constructed after adding the objective function $\min (0, \ldots, 0)^T x$). Note that τ is considered a constant in both the primal and the dual. A common idea for proving τ is lower than the optimum is to show that the dual is unbounded for τ (instead of directly showing that the primal is infeasible for τ).

▶ **Lemma 2.** *If there exists $y : V \to \mathbb{R}_{\geq 0}$, and $z : E \to \mathbb{R}_{\geq 0}$, such that $\sum_{e \in C} z(e) \leq y(v)$ for all $v \in V, C \in \mathcal{C}(v, \tau)$ and $\sum_{v \in V} y(v) < \sum_{e \in E} z(e)$, then $\tau < \mathrm{OPT}^*$.*

This holds because y, z is a feasible solution with negative objective value for the dual and so are the same values scaled by any $\alpha > 0$. This way an arbitrarily low objective value can be obtained. Lemma 2 can be seen as a generalization of a space argument: Consider $y(v) = \tau$ and $z(e) = w(e)$. Then for all $v \in V, C \in \mathcal{C}(v, \tau)$, $\sum_{e \in C} z(e) = \sum_{e \in C} w(e) \leq \tau = y(v)$. Thus, by the Lemma we have that $|V| \cdot \tau = \sum_{v \in V} y(v) < \sum_{e \in E} z(e) = w(E)$ implies $\tau < \mathrm{OPT}^*$.

3 Graph Balancing in a special case

To introduce our techniques, we first consider a simplified case where $w(e) \in (0, 0.5] \cup \{1\}$ for each $e \in E$ and the configuration LP is feasible for 1. We will show that there exists an orientation with maximum weighted in-degree $1 + R$ where $R = 0.74$. The proof for the general case (with a slightly worse rate) can be found in the full version of the paper.

▶ **Definition 3** (Tiny, small, big edges). *We call an edge e* tiny, *if $w(e) \leq 1 - R$;* small, *if $1 - R < w(e) \leq 1/2$; and* big, *if $w(e) = 1$. We will write for the tiny, small, and big edges $T \subseteq E, S \subseteq E$, and $B \subseteq E$, respectively.*

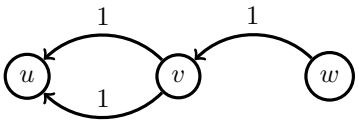
▶ **Definition 4** (Good and bad vertices). *For a given orientation, we call a vertex v* good, *if $w(\delta^-(v)) \leq 1 + R$. A vertex is* bad, *if it is not good.*

The local search algorithm starts with an arbitrary orientation and flips edges until all vertices are good. During this process, a vertex that is already good will never be made bad. It is then proved that (1) the algorithm terminates and (2) when it cannot find a useful edge to flip, the configuration LP also cannot distribute the edges well, i.e., $\mathrm{OPT}^* > 1$, a contradiction.

3.1 Informal overview

Before we give a formal definition of the local search algorithm, we devote this section to giving intuition. We start with very easy algorithms and give challenging instances that motivate the more advanced ideas.

As a toy algorithm consider the following: Start with an arbitrary orientation and repeat until all vertices are good. Whenever there is an edge oriented towards a bad vertex such that its other vertex is good and would remain good even if the edge was flipped (this is called a valid flip), flip this edge. In the following example, both LP and integral optimum are 1. Suppose the algorithm tries to obtain a solution of makespan $2-\epsilon$ with $\epsilon > 0$.



u is bad, v and w are good. However, the algorithm cannot flip one of the edges between u and v, because this would make v bad. It will not flip the edge between v and w, because v is already good. Hence, the algorithm fails. Obviously, in this example we should flip the edge between v and w and then fix u. Let us try to integrate this in the algorithm. We introduce the concept of *pending flips*. When the algorithm wants to flip an edge, but it cannot, because this would make a vertex bad, we add this edge to a list of pending flips. These will be executed once the flip is valid. In the example above, the algorithm could add the edges between u and v to the pending flips, but would not change their orientation, yet. For a pending flip e let us call $s(e)$ the *prospect vertex* and $t(e)$ the *current vertex*. As seen in the example above a sensible local search algorithm should try to move edges away from pending flips' prospect vertices as well (in addition to the bad vertices).

We now state the second toy algorithm. Initialize the pending flips as an empty list. Repeat the following until all vertices are good. Let U be the set of vertices that are either bad or prospect vertices of pending flips. Find an edge from V/U to U and add it to the pending flips. As long as there is a valid pending flip, (1) execute it and (2) delete all pending flips added after it. (2) is to ensure obsolete pending flips are removed. Without it there can be situations where a pending flip has its current vertex in $V \setminus U$, because the pending flip that initially led us to adding it has been executed. This algorithm always succeeds for makespan 1.75 in the special case where weights are in $(0, 0.5] \cup \{1\}$ and $\mathrm{OPT}^* = 1$. We will quickly go over the arguments, since it gives a good idea on how to use the dual of the configuration LP. At this point we will only argue that the algorithm does not get stuck. Normally, we would also have to prove that it terminates (we omit this for sake of brevity). Suppose toward contradiction that the algorithm gets stuck, i.e., there is no valid pending flip and there are no edges from V/U to U. Let F be the big edges e with $t(e) \in U$. We set

$$z(e) = \begin{cases} w(e) & \text{if } t(e) \in U, \\ 0 & \text{otherwise.} \end{cases}$$

$$y(v) = \begin{cases} w(\delta^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| & \text{if } v \in U \text{ and } v \text{ is good,} \\ w(\delta^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| - 0.1 & \text{if } v \in U \text{ and } v \text{ is bad,} \\ \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| & \text{if } v \in V \setminus U. \end{cases}$$

What is left to do is to check that for these values the premise of Lemma 2 is fulfilled. Let $v \in V$ and $C \in \mathcal{C}(v, 1)$. Recall that by definition, $C \subseteq \delta(v)$ and $w(C) \leq 1$. We have to verify that $z(C) \leq y(v)$.

Case 1: $v \in V \setminus U$. Since the algorithm is stuck, there is no edge $e \in \delta^+(v)$ with $t(e) \in U$. Hence, $z(e) = 0$ for all $e \in \delta(v)$ and $z(C) = 0$. By definition of F we have that $\delta_F^-(v) = \emptyset$. Thus, $y(v) \geq 0 = z(C)$.

Case 2: $v \in U$. If v is bad, then $w(\delta^-(v)) > 1.75$ and

$$y(v) \geq \begin{cases} w(\delta_F^-(v)) - \frac{1}{4}|\delta_F^-(v)| - 0.1 \geq \frac{3}{4}|\delta_F^-(v)| - 0.1 \geq 1.4 > z(C) & \text{if } |\delta_F^-(v)| \geq 2, \\ w(\delta^-(v)) - \frac{1}{4}|\delta_F^-(v)| - 0.1 > 1.75 - \frac{1}{4} - 0.1 \geq 1.4 > z(C) & \text{if } |\delta_F^-(v)| \leq 1. \end{cases}$$

If $v \in U$ is good, then it is the prospect vertex of some pending flip e. An invariant of the algorithm is that all pending flips have their current vertex in U. In particular, $z(e) = w(e)$. Furthermore, e is not a valid flip. Thus, $w(\delta^-(v)) + w(e) > 1.75$. Also note that $|\delta_F^-(v)| \leq 1$, since v is good. If $w(e) \leq 0.5$, then

$$y(v) \geq w(\delta^-(v)) - \frac{1}{4}|\delta_F^-(v)| > 1.75 - w(e) - \frac{1}{4} \geq 1 \geq z(C).$$

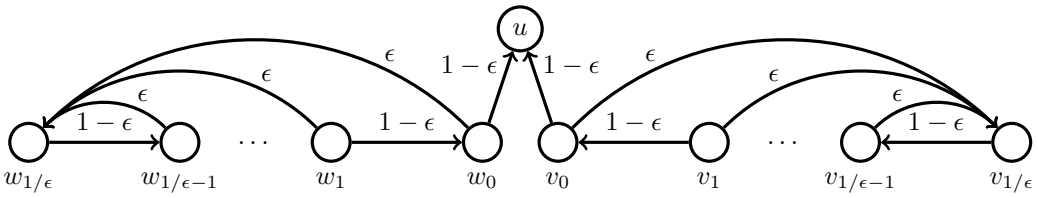
If $w(e) = 1$, then $e \in \delta_F^+(v)$. Hence,

$$y(v) \geq \begin{cases} w(\delta^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| \geq 1.75 - w(e) + \frac{1}{4} \geq 1 \geq z(C) & \text{if } |\delta_F^-(v)| = 0, \\ w(\delta_F^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| \geq 1 + \frac{1}{4} - \frac{1}{4} \geq 1 \geq z(C) & \text{if } |\delta_F^-(v)| = 1. \end{cases}$$

The second condition of Lemma 2 is that $z(E) > y(V)$. This holds because

$$z(E) = \sum_{v \in U} w(\delta^-(v)) = \sum_{v \in U} w(\delta^-(v)) + \frac{1}{4} \sum_{v \in V} [|\delta_F^+(v)| - |\delta_F^-(v)|] > y(V).$$

The strict inequality follows from the fact that there is at least one bad vertex. In the general case this algorithm does not get better than 2 as can be seen in the example below. In a similar, but more complicated way one can also show that in the special case with weights in $(0, 0.5] \cup \{1\}$, the algorithm does not succeed for $1.75 - \epsilon$, where $\epsilon > 0$ is arbitrary. In other words, the analysis for 1.75 is tight.



The LP and integral optima are again 1. Suppose the algorithm tries to find a solution with makespan $2 - 3\epsilon$. The only bad vertex is u. Hence, the algorithm will add the $(1 - \epsilon)$-edges to the pending flips one after another. However, at the time it reaches $v_{1/\epsilon}$ or $w_{1/\epsilon}$ it will get stuck, since there is a load of $1/\epsilon \times \epsilon$ on them. The way to fix this is to allow edges (in this case those of weight ϵ) to also be flipped towards vertices in U, i.e., vertices where we wanted to reduce the load. We cannot simply allow arbitrary flips back and forth between U,

because we have to take care that the algorithm eventually terminates. This is where the concept of vertices *repelling* edges comes in: Depending on the current orientation of the edges and the list of pending flips we will define a binary relation between vertices and edges. The exact definition has to be chosen carefully. For a pair (v, e) of this relation we write vertex v repels edge e. When a vertex repels an edge, it means it is undesirable that the edge is oriented towards this vertex. When it does not repel the edge, we do not care. This will be used in the algorithm when adding pending flips: An edge e may only be added to the pending flips, when it is repelled by its current vertex and not repelled by its prospect vertex.

In the earlier toy algorithms a vertex either repels all edges (when it is in U) or none (when it is not in U). By adding a fine-grained strategy of repelled edges, we gain much more flexibility. A strategy that appears particularly simple and powerful is the following. (1) We let bad vertices repel all edges. Moreover, (2) for every pending flip e with prospect vertex v we find maximum threshold W such that all edges in $\delta^-(v)$ with weight at least W are already enough to prevent the flip from being executed, i.e., their total weight is greater than $1 + R - w(e)$. We let v repel all edges of weight at least $\min\{w(e), W\}$.

Now the third toy algorithm is to repeat the following until all vertices are good. Find an edge e that is repelled by $t(e)$, but not by $s(e)$ and add it to the list of pending flips. As long as there is a valid pending flip, execute it and delete all pending flips added after it. This algorithm succeeds in the previous example. It will add the $(1-\epsilon)$-edges to the pending flips, but the vertices $u, v_1, \ldots, v_{1/\epsilon-1}, w_1, \ldots, w_{1/\epsilon-1}$ repel only the $(1-\epsilon)$-edges and not the ϵ-edges. Once the pending flips reach $w_{1/\epsilon}$ or $v_{1/\epsilon}$, these vertices will repel the ϵ-edges and start flipping them. Finally, there will be enough space on $w_{1/\epsilon}$ or $v_{1/\epsilon}$ and the $1-\epsilon$ edges can be flipped one after another.

For the simple case at least this is very close to the algorithm that gives us the bound of 1.74. However, to make the analysis work, we add a couple of tweaks. One of them is the idea of critical vertices. When the threshold of repelled edges, i.e., $\min\{w(e), W\}$, reaches a low value, e.g., 0.26, we make the vertex repel all edges. We call such a vertex a *critical* vertex. This tweak (together with some technicalities) allows us to argue that at least half of the critical vertices are prospect vertices of pending flips for tiny edges. This is helpful, since (unless the pending flip is valid) such prospect vertices have a lot of weight oriented towards them and this makes it easier to argue that the configuration LP also cannot distribute this weight very well; thereby reaching a contradiction.

An important novelty in this paper compared to earlier local search algorithms is that we are able to repel only a subset of edges small/tiny edges. In the Restricted Assignment proofs [19, 13] it is always the case that a machine (vertex) repels all small/tiny jobs (edges) or none. To utilize the flexibility in the small/tiny edges we scale the z values of tiny edges up by a factor $\beta > 1$. It is not obvious at all that this works out and the proof is quite tricky.

In a sense, we push the threshold for repelling all edges down to some value less than 0.5 (see description of critical vertices above). A logical conclusion to draw would be that pushing it down even more (or removing it completely) should give an even better algorithm. This would bring us back to toy algorithm 3. In fact, we are not aware of bad instances. It is an intriguing question whether this can also be proved that this simple algorithm is good.

3.2 Algorithm

The central data structure we use is an ordered list of pending flips $P = (e_1^P, e_2^P, \ldots, e_\ell^P)$. Here, every component e_k^P, stands for an edge the algorithm wants to flip. If P is clear from the context, we simply write *flip* when we speak of a pending flip e_k^P. A *tiny flip* is a flip where e_k^P is tiny. In the same way we define *small* and *big flips*. The prospect vertex of a

flip e_k^P is the vertex $s(e_k^P)$ to which we want to orient it. The algorithm will not perform the flip, if this would create a bad vertex, i.e., $w(\delta^-(s(e_k^P))) + w(e_k^P) > 1 + R$. If it does not create a bad vertex, we say that the flip e_k^P is a *valid flip*. For every $0 \leq k \leq \ell$ define $P_{\leq k} := (e_1^P, \ldots, e_k^P)$, i.e., the first k elements of P (with $P_{\leq 0}$ being the empty list).

At each point during the execution of the algorithm, the vertices repel certain edges. This can be thought of as a binary relation between vertices and their incident edges, i.e., a subset of $\{(v, e) : v \in r(e)\}$, and this relation changes dynamically as the current orientation or P change. The definition of which vertices repel which edges is given in Section 3.3. The algorithm will only add a new pending flip e to P, if e is repelled by the vertex it is oriented towards and not repelled by the other.

Algorithm 1: Local search algorithm for simplified Graph Balancing.

Input : Weighted multigraph $G = (V, E, r, w)$ with $\mathrm{OPT}^* = 1$ and $w(e) \in (0, 0.5] \cup \{1\}$ for all $e \in E$
Result: Orientation $s, t : E \to V$ with maximum weighted in-degree $1 + R$
let $s, t : E \to V$ map arbitrary source and target vertices to each edge ;
// i.e., $\{s(e), t(e)\} = r(e)$ for all $e \in E$
$\ell \leftarrow 0$; // number of pending edges P to flip
while *there is a bad vertex* **do**
  if *there exists a valid flip* $e \in P$ **then**
    let $0 \leq k \leq \ell$ be minimal such that e is repelled by $t(e)$ w.r.t. $P_{\leq k}$;
    exchange $s(e)$ and $t(e)$;
    $P \leftarrow P_{\leq k}$; $\ell \leftarrow k$; // Forget pending flips $e_{k+1}^P, \ldots, e_\ell^P$
  else
    choose an edge $e \in E \setminus P$ with $w(e)$ minimal and e is repelled by $t(e)$ and not repelled by $s(e)$ w.r.t. P ;
    $P_{\ell+1} \leftarrow e$; $\ell = \ell + 1$; // Append e to P
  end
end

3.3 Repelled edges

Consider the current list of ℓ pending flips $P_{\leq \ell}$. The repelled edges are defined inductively. For some $k \leq \ell$ we will now define the repelled edges w.r.t. $P_{\leq k}$.

(initialization) If $k = 0$, let every bad vertex v repel every edge in $\delta(v)$. Furthermore, let every vertex v repel every loop e where $\{v\} = r(e)$.

(monotonicity) If $k > 0$ and v repels e w.r.t. $P_{\leq k-1}$, then let v repel e w.r.t. $P_{\leq k}$.

The rule on loops is only for a technical reasons. Loops will never appear in the list of pending flips. The remaining rules regard $k > 0$ and the last pending flip in $P_{\leq k}$, e_k^P. The algorithm should reduce the load on $s(e_k^P)$ to make it valid.

Which edges exactly does the prospect vertex of e_k^P, i.e., $s(e_k^P)$, repel? First, we define $\tilde{E}(P_{\leq k-1}) \subseteq E$ where $e \in \tilde{E}(P_{\leq k-1})$ if and only if e is repelled by $s(e)$ w.r.t. $P_{\leq k-1}$. We will omit $P_{\leq k-1}$ when it is clear from the context. $\tilde{E}$ are edges that we do not expect to be able to flip: Recall that when an edge e is repelled by $s(e)$, it cannot be added to P. Moreover, for every W define $E_{\geq W} = \{e \in E : w(e) \geq W\}$. We are interested in values W such that

$$w(\delta^-_{\tilde{E} \cup E_{\geq W}}(s(e_k^P))) + w(e_k^P) > 1 + R. \tag{1}$$

Let $W_0 \in (0, w(e_k^P)]$ be maximal such that (1) holds. To be well-defined, we set $W_0 = 0$, if no such W exists. In that case, however, it holds that $w(\delta^-(s(e_k^P))) + w(e_k^P) \leq 1 + R$. This means that e_k^P is valid and the algorithm will remove it from the list immediately. Hence, the case is not particularly interesting. We define the following edges to be repelled by $s(e_k^P)$:
(uncritical) If $W_0 > 1 - R$, let $s(e_k^P)$ repel every edge in $\delta_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$.
(critical) If $W_0 \leq 1 - R$, let $s(e_k^P)$ repel every edge in $\delta(s(e_k^P))$.
Note that in the cases above there is not a restriction to incoming edges like in (1).

▶ **Fact 1.** *$s(e_k^P)$ repels e_k^P w.r.t. $P_{\leq k}$.*

This is because of $W_0 \leq w(e_k^P)$ in the rules for $P_{\leq k}$. An important observation is that repelled edges are stable under the following operation.

▶ **Fact 2.** *Let $e \notin P_{\leq k}$ be an edge that is not repelled by any vertex w.r.t. $P_{\leq k-1}$, and possibly by $t(e)$ (but not $s(e)$) w.r.t. $P_{\leq k}$. If the orientation of e changes and this does not affect the sets of good and bad vertices, the edges repelled by some vertex w.r.t. $P_{\leq k}$ will still be repelled after the change.*

Proof. We first argue that the repelled edges w.r.t. $P_{\leq k'}$, $k' = 0, \ldots, k-1$ have not changed. This argument is by induction. Since the good and bad vertices do not change, the edges repelled w.r.t. $P_{\leq 0}$ do not change. Let $k' \in \{1, \ldots, k-1\}$ and assume that the edges repelled w.r.t. $k'-1$ have not changed. Moreover, let W_0 be as in the definition of repelled edges w.r.t. $P_{\leq k'}$ before the change. We have to understand that e is not and was not in $\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_{k'}^P))$ (with $\tilde{E} = \tilde{E}(P_{\leq k'-1})$). This means that flipping it does not affect the choice of W_0 and, in particular, not the repelled edges. Let v and v' denote the vertex e is oriented towards before the flip and after the flip, respectively. Since e was not repelled by v and v' w.r.t. $P_{\leq k'}$, it holds that $v, v' \neq s(e_k^P)$ or $w(e) < W_0$: $s(e_k^P)$ repelled all edges greater or equal W_0 in both the case (uncritial) and (critical), but e was not repelled by v or v'.

Therefore $e \notin \delta^-_{E_{\geq W_0}}(s(e_{k'}^P))$ before and after the change. Moreover, since e was not repelled by any vertex w.r.t. $P_{\leq k'-1}$ (and by induction hypothesis, it still is not), it follows that $e \notin \tilde{E}(P_{\leq k'-1})$. By this induction we have that edges repelled w.r.t. $P_{\leq k-1}$ have not changed and by the same argument as before, e is not in $\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$ after the change. This means W_0 has not increased and edges repelled w.r.t. $P_{\leq k}$ are still repelled. It could be that W_0 decreases, if e was in $\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$ before the flip. This would mean that the number of edges repelled by $s(e_k^P)$ increases. ◀

We note that W_0 (in the definition of $P_{\leq k}$) is either equal to $w(e_k^P)$ or it is the maximal value for which (1) holds. Furthermore,

▶ **Fact 3.** *Let W_0 be as in the definition of repelled edges w.r.t. $P_{\leq k}$. If $W_0 < w(e_k^P)$, then there is an edge of weight exactly W_0 in $\delta^-(s(e_k^P))$. Furthermore, it is not a loop and it is not repelled by its other vertex, i.e., not $s(e_k^P)$, w.r.t. $P_{\leq k}$.*

Proof. We prove this for $P_{\leq k-1}$. Since the change from $P_{\leq k-1}$ to $P_{\leq k}$ only affects $s(e_k^P)$, this suffices. All edges that are repelled by their other vertex w.r.t. $P_{\leq k-1}$ (in particular, loops) are in $\tilde{E}$. Recall that $w(\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))) + w(e_k^P) > 1 + R$. Assume toward contradiction there is no edge of weight W_0 in $\delta^-(s(e_k^P))$, which is not in $\tilde{E}$. This means there is some $\epsilon > 0$ such that $\delta^-_{\tilde{E} \cup E_{\geq W_0 + \epsilon}}(s(e_k^P)) = \delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$. Hence, W_0 is not maximal. ◀

3.4 Analysis

The following analysis holds for the values $R = 0.74$ and $\beta = 1.1$. β is a central parameter in the proof. These can be slightly improved, but we refrain from this for the sake of simplicity. The proof consists of two parts. We need to show that the algorithm terminates and that there is always either a valid flip in P or some edge can be added to P.

▶ **Lemma 5.** *The algorithm terminates after finitely many iterations of the main loop.*

Proof. We consider the potential function $s(P) = (g, |\tilde{E}(P_{\leq 0})|, \ldots, |\tilde{E}(P_{\leq \ell})|, -1)$, where g is the number of good vertices. We will argue that this vector increases lexicographically after every iteration of the main loop. Intuitively $|\tilde{E}(P_{\leq k})|$, $k \leq \ell$, is an measure for progress. When an edge e is repelled by a vertex v, then we want that $s(e) = v$. $|\tilde{E}(P_{\leq k})|$ counts exactly these situations. Since ℓ is bounded by $|E|$, there can be at most $|V| \cdot |E|^{O(|E|)}$ possible values for the vector. Thus, the algorithm must terminate after at most this many iterations. In an iteration either a new flip is added to P or a flip is executed.

If a flip e is added as the $(\ell+1)$-th element of P, then clearly $\tilde{E}(P_{\leq i})$ does not change for $i \leq \ell$. Furthermore, the last component of the vector is replaced by some non-negative value.

Now consider a flip $e \in P$ that is executed. If this flip turns a bad vertex good, we are done. Hence, assume otherwise and let v and v' be the vertex it was previously and the one it is now oriented towards. Furthermore, let ℓ' be the length of P after the flip. Recall that ℓ' was chosen such that before the flip is executed v repels e w.r.t. $P_{\leq \ell'}$, but not w.r.t. $P_{\leq k}$ for any $k \leq \ell' - 1$. Also, e is not repelled by v' w.r.t. $P_{\leq k}$ for any $k \leq \ell'$ or else the flip would not have been added to P in the first place. By Fact 2 this means that repelled edges w.r.t. $P_{\leq k}$, $k \leq \ell'$, are still repelled after the flip. Because of this and because the only edge that changed direction, e, is not in $\tilde{E}(P_{\leq k})$ for any $k \leq \ell'$, $|\tilde{E}(P_{\leq k})|$ has not decreased. Finally, e has not been in $\tilde{E}(P_{\leq \ell'})$ before the flip, but now is. Thus, the first $\ell' - 1$ components of the vector have not decreased and the ℓ'-th one has increased. ◀

▶ **Lemma 6.** *If there at least one bad vertex remaining, then there is either a valid flip in P or a flip that can be added to P.*

Proof. We assume toward contradiction that there exists a bad vertex, no valid pending flip and no edge that can be added to P. We will show that this implies $\mathrm{OPT}^* > 1$.

Like above we denote by $\tilde{E} = \tilde{E}(P)$ those edges e that are repelled by $s(e)$. In particular, if an edge e is in P or repelled by $t(e)$ it must also be in $\tilde{E}$: If such an edge is in P, this follows from Fact 1. Otherwise, such an edge must be repelled also by $s(e)$ or else it could be added to P. For every $e \in E \setminus \tilde{E}$, set $z(e) = 0$. For every $e \in \tilde{E}$, set

$$z(e) = \begin{cases} 1 & \text{if } w(e) = 1, \\ w(e) & \text{if } 1 - R < w(e) \leq 1/2, \text{ and} \\ \beta w(e) & \text{if } w(e) \leq 1 - R. \end{cases}$$

In general, we would like to set each $y(v)$ to $z(\delta^-(v))$ (or equivalently, $z(\delta^-_{\tilde{E}}(v))$). However, there are two kinds of amortization between vertices that we include in the values of y.

As can be seen in the definition of repelled edges, a vertex v repels either edges with weight at least a certain threshold $W > 1 - R$ and edges in $\delta^-(v)$ that are repelled by their other vertex; or they repel all edges. We will call vertices of the latter kind *critical*. In other words, a vertex v is critical, if in the inductive definition of repelled edges at some point $v = s(e_k^P)$ and the rule (critical) applies. There might be a coincidence where only rule (uncritical) applies for v, but this already covers all edges in $\delta^-(v)$. This is not a critical vertex. We note that every vertex that is prospect vertex of a tiny flip e_k^P must be critical: In the inductive definition when considering e_k^P we have that $W_0 \leq w(e_k^P) \leq 1 - R$ by definition.

Critical vertex amortization. If v is a good vertex and critical, but is not a prospect vertex of a tiny flip, set $a_v := \beta - 1$. If v is a good vertex and prospect vertex of tiny flip (in particular, v is critical), set $a_v := -(\beta - 1)$. Otherwise, set $a_v = 0$.

Big edge amortization. Let $F \subseteq P$ denote the set of big flips. Then in particular $F \subseteq \tilde{E}$ (Fact 1). We define for all $v \in V$, $b_v := (|\delta_F^+(v)| - |\delta_F^-(v)|) \cdot (1 - R)$.

We conclude the definition of y by setting $y(v) = z(\delta^-(v)) + a_v + b_v$ for all good vertices v and $y(v) = z(\delta^-(v)) + a_v + b_v - \mu$ for all bad vertices v, where $\mu = 0.01$.

▷ Claim 7. It holds that $\sum_{v \in V} y(v) < \sum_{e \in E} z(e)$.

▷ Claim 8. $y(v), z(e) \geq 0$ f.a. $v \in V, e \in E$ and f.a. $v \in V$, $C \in \mathcal{C}(v, 1)$, $\sum_{e \in C} z(e) \leq y(v)$.

By Lemma 2 this implies that $\mathrm{OPT}^* > 1$. ◀

Proof of Claim 7. First we note that

$$\sum_{v \in V} b_v = (1 - R) \underbrace{\left(\sum_{v \in V} |\delta_F^+(v)| - \sum_{v \in V} |\delta_F^-(v)|\right)}_{=0} = 0.$$

Moreover, we have that $\sum_{v \in V} a_v \leq 0$: This is because at least half of all good vertices that are critical are prospect vertices of tiny flips. The proof for this is omitted due to space constraints. We conclude,

$$\sum_{e \in E} z(e) \geq \sum_{v \in V} \sum_{e \in \delta^-(v)} z(e) + \sum_{v \in V} [b_v + a_v] > \sum_{v \in V} y(v),$$

where the strict inequality holds because there exists at least one bad vertex. ◁

Proof of Claim 8. Let $v \in V$ and $C \in \mathcal{C}(v, 1)$. We need to show that $z(C) \leq y(v)$. Obviously the z values are non-negative. By showing the inequality above, we also get that the y values are non-negative. First, we will state some auxiliary facts.

▶ **Fact 4.** *For every edge flip $e \in P$, we have $w(\delta_{\tilde{E}}^-(s(e))) + w(e) > 1 + R$.*

This is due to the fact that e is not a valid flip and by definition of repelled edges.

▶ **Fact 5.** *If v is a good vertex and not prospect vertex of a tiny pending flip, then $y(v) \geq z(C) + w(\delta_{\tilde{E}}^-(v)) - 1 + b_v$. In other words, it is sufficient to show that $w(\delta_{\tilde{E}}^-(v)) + b_v \geq 1$.*

If v is critical, then $y(v) = z(\delta^-(v)) + \beta - 1 + b_v \geq w(\delta_{\tilde{E}}^-(v)) + z(C) - 1 + b_v$. If v is uncritical, all edges $e \in C$ with $z(e) > w(e)$ must be in $\delta^-(v)$. Otherwise, the flip e could be added to P. Therefore,

$$\begin{aligned} y(v) = z(\delta^-(v)) + a_v + b_v &\geq z(\delta_{\tilde{E}}^-(v)) - w(\delta_{\tilde{E}}^-(v)) + w(\delta_{\tilde{E}}^-(v)) + b_v \\ &\geq z(\tilde{C}) - w(\tilde{C}) + w(\delta_{\tilde{E}}^-(v)) + b_v \geq z(C) + w(\delta_{\tilde{E}}^-(v)) - 1 + b_v. \end{aligned}$$

▶ **Fact 6.** *For every vertex v it holds that (1) v repels no edges, (2) v repels all edges (critical), or (3) there exists a threshold $W > 1 - R$ such that v repels edges $e \in \delta^-(v)$ if $w(e) \geq W$ or they are also repelled by $s(e)$.*

Furthermore, in (3) we have that $W = \min_{e \in \delta_P^+(v)} w(e)$ or $W < \min_{e \in \delta_P^+(v)} w(e)$ and $W \in \{w(e) : e \in \delta_{\tilde{E}}^-(v)\}$ (see Fact 3).

Case 1: v is a bad vertex. If $|\delta_F^-(v)| \geq 2$,

$$y(v) \geq z(\delta^-(v)) - |\delta_F^-(v)| \cdot (1-R) - \mu \geq |\delta_F^-(v)| \cdot (1-(1-R)) - \mu \geq 2R - \mu \geq \beta \geq z(C).$$

Otherwise, $|\delta_F^-(v)| \leq 1$ and therefore

$$y(v) \geq z(\delta^-(v)) - 1 + R - \mu \geq w(\delta^-(v)) - 1 + R - \mu > 1 + R - 1 + R - \mu = 2R - \mu \geq \beta \geq z(C).$$

Here we use that $w(\delta^-(v)) > 1 + R$ by the definition of a bad vertex. Assume for the remainder that v is a good vertex, in particular that $|\delta_F^-(v)| \leq 1$.

Case 2: v is good and prospect vertex of a tiny flip. Using Fact 4 we get $w(\delta_{\tilde{E}}^-(v)) > 1 + R - (1-R) = 2R$. Since $z(e) \geq w(e)$ for all $e \in \tilde{E}$, we also have $z(\delta^-(v)) \geq 2R$. Moreover, since the vertex is good, $|\delta_F^-(v)| \leq 1$ and consequently $b_v \geq -(1-R)$. We conclude

$$y(v) \geq z(\delta^-(v)) - (\beta - 1) - (1-R) \geq 2R - (\beta - 1) - (1-R) = \beta + \underbrace{3R - 2\beta}_{\geq 0} \geq z(C).$$

Case 3: v is good, not prospect vertex of a tiny flip, but prospect vertex of a small flip. If $|\delta_F^-(v)| = 0$, again with Fact 4 we get $w(\delta_{\tilde{E}}^-(v)) + b_v \geq 1 + R - 0.5 + 0 > 1$. This suffices because of Fact 5. Moreover, if $|\delta_F^-(v)| = 1$ and $\delta_{\tilde{E}}^-(v)$ contains a small edge, it holds that

$$w(\delta_{\tilde{E}}^-(v)) + b_v \geq w(\delta_F^-(v)) + (1-R) + b_v \geq 1 + (1-R) - (1-R) = 1.$$

Here we use that the small edge must have a size of at least $1 - R$. The case that remains is where $\delta_{\tilde{E}}^-(v)$ contains one edge from F and tiny edges. Since the overal weight of $\delta_{\tilde{E}}^-(v)$ is at least $1 + R - 0.5$, the weight of tiny edges is at least $R - 0.5$. Thus, the z-value of the tiny edges is at least $\beta(R - 0.5)$. If v is critical,

$$y(v) \geq z(\delta^-(v)) - (1-R) + (\beta - 1) \geq 1 + \underbrace{\beta(R - 0.5) - (1-R)}_{\geq 0} + (\beta - 1) \geq z(C).$$

Notice that $\beta(R - 0.5) \geq 1 - R$ by choice of R and β. Assume for the remainder of this case that v is uncritical. If C contains a big edge, this must be the only element in C. Therefore,

$$y(v) \geq z(\delta^-(v)) - (1-R) \geq 1 + \beta(R - 0.5) - (1-R) \geq 1 = z(C).$$

Consider the case where $C \cap \tilde{E}$ contains at most one small edge and no big edge. Since v is uncritical, all tiny edges in C with positive z value must also be in $\delta^-(v)$. Therefore, $z(C) \leq 0.5 + z(\delta_T^-(v))$. Thus,

$$y(v) \geq z(\delta^-(v)) - (1-R) \geq 1 + z(\delta_T^-(v)) - (1-R) > 0.5 + z(\delta_T^-(v)) \geq z(C).$$

In the final case $C \cap \tilde{E}$ contains $k \in \{2, 3\}$ small edges. We let s denote the weight of the smallest edge with a pending flip whose prospect vertex is v. Since v is not critical and there is no small edge in $\delta^-(v)$, v repels exactly those edges with weight at least s. This means the small edges in $C \cap \tilde{E}$ must be of weight at least s: Otherwise, they could be added as pending flips. Therefore,

$$\begin{aligned} y(v) &\geq z(\delta^-(v)) - (1-R) \geq z(\delta_F^-(v)) + z(\delta_T^-(v)) - (1-R) \\ &\geq 1 + \beta(w(\delta_{\tilde{E}}^-(v)) - w(\delta_F^-(v))) - (1-R) = R + \beta(w(\delta_{\tilde{E}}^-(v)) - 1) \geq R + \beta(R - s). \end{aligned}$$

Note that $s \leq 1/k$ and, by choice of β, $k - \beta(k-1) \geq 0$. It follows that

$$\begin{aligned} z(C) &\leq k \cdot s + \beta(1 - k \cdot s) \\ &\leq y(v) + k \cdot s - R + \beta(1 - R - (k-1)s) \leq y(v) + k \cdot \frac{1}{k} - R + \beta\left(1 - R - \frac{k-1}{k}\right) \\ &= y(v) + 1 - R + \beta\left(\frac{1}{k} - R\right) \leq y(v) + 1 - R + \beta(0.5 - R) \leq y(v). \end{aligned}$$

Case 4: v is good and not prospect vertex of a small/tiny flip. If $|\delta_F^+(v)| = 0$, then v does not repel any edges. In particular, $\delta_F^-(v) = \emptyset$ and every $e \in \delta(v)$ with $z(e) > 0$ must be in $\delta^-(v)$. Therefore, $z(C) \leq z(\delta^-(v)) \leq y(v)$. We assume in the remainder that $|\delta_F^+(v)| = 1$.
If $|\delta_F^-(v)| = 1$, we get $w(\delta_{\tilde{E}}^-(v)) + b_v \geq 1 + 0$. Otherwise, it must hold that $|\delta_F^-(v)| = 0$ and $w(\delta_{\tilde{E}}^-(v)) + b_v \geq R + (1 - R) = 1$. ◁

References

1. Chidambaram Annamalai. Lazy Local Search Meets Machine Scheduling. *CoRR*, abs/1611.07371, 2016. arXiv:1611.07371.
2. Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial Algorithm for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms*, 13(3):37:1–37:28, 2017. Previously appeared in SODA'15. doi:10.1145/3070694.
3. Arash Asadpour, Uriel Feige, and Amin Saberi. Santa Claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24:1–24:9, 2012. See Asadpour's homepage for the bound of 4 instead of 5 as in the paper; Previously appeared in APPROX'08. doi:10.1145/2229163.2229168.
4. Yuichi Asahiro, Jesper Jansson, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *Journal of Combinatorial Optimization*, 22(1):78–96, 2011. doi:10.1007/s10878-009-9276-z.
5. Nikhil Bansal and Maxim Sviridenko. The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 31–40, 2006. Previously appeared in STOC'06. doi:10.1145/1132516.1132522.
6. Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On Allocating Goods to Maximize Fairness. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 107–116, 2009. doi:10.1109/FOCS.2009.51.
7. Deeparnab Chakrabarty and Kirankumar Shiragur. Graph Balancing with Two Edge Types. *CoRR*, abs/1604.06918, 2016. arXiv:1604.06918.
8. Siu-Wing Cheng and Yuchen Mao. Integrality Gap of the Configuration LP for the Restricted Max-Min Fair Allocation. *CoRR*, abs/1807.04152, 2018. arXiv:1807.04152.
9. Tomáš Ebenlendr, Marek Krčál, and Jiří Sgall. Graph Balancing: A Special Case of Scheduling Unrelated Parallel Machines. *Algorithmica*, 68(1):62–80, 2014. Previously appeared in SODA'08. doi:10.1007/s00453-012-9668-9.
10. Chien-Chung Huang and Sebastian Ott. A Combinatorial Approximation Algorithm for Graph Balancing with Light Hyper Edges. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 49:1–49:15, 2016. doi:10.4230/LIPIcs.ESA.2016.49.
11. Klaus Jansen, Kati Land, and Marten Maack. Estimating The Makespan of The Two-Valued Restricted Assignment Problem. In *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.SWAT.2016.24.

12 Klaus Jansen and Lars Rohwedder. A Quasi-Polynomial Approximation for the Restricted Assignment Problem. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 305–316, 2017. doi:10.1007/978-3-319-59250-3_25.

13 Klaus Jansen and Lars Rohwedder. On the Configuration-LP of the Restricted Assignment Problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2670–2678, 2017. doi:10.1137/1.9781611974782.176.

14 Klaus Jansen and Lars Rohwedder. A note on the integrality gap of the configuration LP for restricted Santa Claus. *CoRR*, abs/1807.03626, 2018. arXiv:1807.03626.

15 Klaus Jansen and Lars Rohwedder. Compact LP Relaxations for Allocation Problems. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 11:1–11:19, 2018. doi:10.4230/OASIcs.SOSA.2018.11.

16 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation Algorithms for Scheduling Unrelated Parallel Machines. *Math. Program.*, 46:259–271, 1990. doi:10.1007/BF01585745.

17 Daniel R. Page and Roberto Solis-Oba. A 3/2-Approximation Algorithm for the Graph Balancing Problem with Two Weights. *Algorithms*, 9(2):38, 2016. doi:10.3390/a9020038.

18 Lukás Polácek and Ola Svensson. Quasi-Polynomial Local Search for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms*, 12(2):13:1–13:13, 2016. Previously appeared in ICALP'12. doi:10.1145/2818695.

19 Ola Svensson. Santa Claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012. Previously appeared in STOC'11. doi:10.1137/110851201.

20 José Verschae and Andreas Wiese. On the configuration-LP for scheduling on unrelated machines. *Journal of Scheduling*, 17(4):371–383, 2014. Previously appeared in ESA'11. doi:10.1007/s10951-013-0359-4.

21 Chao Wang and René Sitters. On some special cases of the restricted assignment problem. *Information Processing Letters*, 116(11):723–728, 2016. doi:10.1016/j.ipl.2016.06.007.