Graph Balancing with Orientation Costs

Roy Schwartz
Technion – Israel Institute of Technology, Haifa, Israel
schwartz@cs.technion.ac.il

Ran Yeheskel
Technion – Israel Institute of Technology, Haifa, Israel
ran.yeheskel11@gmail.com

Abstract

Motivated by the classic Generalized Assignment Problem, we consider the Graph Balancing problem in the presence of orientation costs: given an undirected multi-graph $G = (V, E)$ equipped with edge weights and orientation costs on the edges, the goal is to find an orientation of the edges that minimizes both the maximum weight of edges oriented toward any vertex (makespan) and total orientation cost. We present a general framework for minimizing makespan in the presence of costs that allows us to: (1) achieve bicriteria approximations for the Graph Balancing problem that capture known previous results (Shmoys-Tardos [Math. Progrm. '93], Ebenlendr-Krcál-Sgall [Algorithmica '14], and Wang-Sitters [Inf. Process. Lett. '16]); and (2) achieve bicriteria approximations for extensions of the Graph Balancing problem that admit hyperedges and unrelated weights. Our framework is based on a remarkably simple rounding of a strengthened linear relaxation. We complement the above by presenting bicriteria lower bounds with respect to the linear programming relaxations we use that show that a loss in the total orientation cost is required if one aims for an approximation better than 2 in the makespan.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms; Mathematics of computing → Approximation algorithms

Keywords and phrases Graph Balancing, Generalized Assignment Problem

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.82

Funding *Roy Schwartz*: Supported by ISF grant 1336/16 and NSF-BSF grant number 2016742.

1 Introduction

We consider the Graph Balancing problem (Gb) where we are given an undirected multi-graph $G = (V, E)$ equipped with edge weights $p : E \to \mathbb{R}^+$. The goal is to orient all the edges of the graph, where each edge can be oriented to one of its endpoints. Given an orientation of the edges the load of a vertex u is the sum of weights of edges oriented toward it. The goal is to find an orientation of the edges that minimizes the maximum load over all vertices.

Gb was first introduced by Ebenlendr et al. [3] and since its introduction it has attracted much attention (see, e.g., [10, 18, 6, 2, 12]). Besides being a natural graph optimization problem on its own, a main motivation for considering Gb is the well known Generalized Assignment Problem (Gap) (see, e.g., [15, 3, 16, 19]). In Gap we are given a collection $\mathcal{M}$ of m machines and a collection $\mathcal{J}$ of n jobs, along with processing times $p_{i,j}$ (the processing time of job j on machine i) and assignment costs $c_{i,j}$ (the cost of assigning job j to machine i). Each job must be assigned to one of the machines. The processing time of machine i is the sum of processing times $p_{i,j}$ over all jobs j that are assigned to i, and the makespan of an assignment is the maximum over all machines i of its processing time. Additionally, the total assignment cost of an assignment is the sum of assignment costs $c_{i,j}$ over all machines i and jobs j that are assigned to i. Given a target makespan T, we denote by $C(T)$ the minimum total assignment cost over all assignments with makespan at most T. If there are no assignments with makespan at most T, then $C(T) = \infty$. The goal in Gap, given a

target makespan T, is to find an assignment with makespan at most T and total assignment cost at most $C(T)$, or declare that no such assignment exists. We note that only T is given to the algorithm whereas $C(T)$ is not. For this bicriteria problem, the celebrated result of Shmoys and Tardos [15] provides an approximation algorithm that finds an assignment with makespan at most $2T$ and total assignment cost at most $C(T)$.

GB is a captured by GAP since one can: (1) set $\mathcal{M}$ to be V and $\mathcal{J}$ to be E; and (2) for each job $j \in \mathcal{J}$ (which corresponds to an edge $e \in E$) set its processing time to be p_e for the two machines that correspond to the endpoints of e and ∞ for all other machines. Note that assigning job j to machine i corresponds to orienting the edge e toward its endpoint that corresponds to machine i. There are two important things to note. First, GB was originally defined as a *single criterion* optimization problem as opposed to GAP which is a bicriteria optimization problem. Second, the weights p in GB, which represent the processing times of the jobs, are *related*, *i.e.*, the processing times do not depend on the vertex the edge is oriented to. Ebenlendr et al. [3] introduced a novel linear relaxation and rounding algorithm that achieves an approximation of 1.75 with respect to the optimal makespan. They also proved that even for this special case, no polynomial time algorithm can achieve an approximation less than 1.5 unless $P = NP$, thus extending the hardness of GAP to GB.

In this work we consider the *bicriteria* GB problem, where we are also given orientation costs, the equivalent to the assignment costs in GAP. Formally, an edge $e = (u, v)$ has orientation costs $c_{e,u}$ and $c_{e,v}$ and orienting it to u incurs a cost of $c_{e,u}$. Similarly to GAP, given a target makespan T, the goal is to find an orientation of the edges with total orientation cost at most $C(T)$ and makespan at most T.[1] To the best of our knowledge, the bicriteria GB problem was not previously considered. We say that an algorithm is a (α, β)-approximation if given a target makespan T, it outputs an orientation with makespan at most αT and total orientation cost at most $\beta C(T)$. Thus, [15] is a $(2, 1)$-approximation to GB. We note that the algorithm of [3] cannot handle orientation costs and is in fact a $(1.75, \infty)$-approximation for GB. A result by Wang and Sitters [18] implicitly gives a $(11/6, 3/2)$-approximation for GB.

We study the bicriteria tradeoff between makespan and total orientation cost in GB, presenting both upper and lower bounds (the latter are with respect to the linear programming relaxations used in this work). We employ a remarkably simple general framework that allows us to achieve bicriteria approximations for GB that capture and extend known results. Furthermore, we consider extensions of GB that allow for: (1) hyperedges to be present, *i.e.*, a job can be assigned to more than two machines; and (2) processing times can be unrelated, *i.e.*, the processing time of a job might depend on the machine it is assigned to. Our results regarding these extensions improve upon the previously best known algorithms, and are also based on the general framework presented in this paper. We believe this framework might be of independent interest to other related scheduling problems.

1.1 Our Results

Our results are of three different flavors: bicriteria upper bounds for GB, bicriteria lower bounds for GB, and both upper and lower bicriteria bounds for extensions of GB (all lower bounds are with respect to the linear programming relaxations we use). Let us now elaborate on each of the above.

[1] As in GAP, the total orientation cost of an orientation is defined as the sum of orientation costs $c_{e,u}$ over all vertices u and edges e oriented toward u. $C(T)$ is defined as the minimum total orientation cost over all orientations with makespan at most T. If no such orientation exists then $C(T)$ is set to ∞.

1.1.1 Upper Bounds

We present a general framework for minimizing makespan in the presence of costs and obtain two algorithms that achieve bicriteria approximations for GB. This is summarized in the following two theorems.

▶ **Theorem 1.** *There exists a polynomial time algorithm that finds an orientation that is a* $(1.75+\gamma, 1/(2\gamma+0.5))$*-approximation for* GRAPH BALANCING*, for every* $1/12 - \epsilon \leq \gamma \leq 1/4$ *where* $\epsilon = \sqrt{33}/4 - 17/12 \approx 0.02$*.*

▶ **Theorem 2.** *There exists a polynomial time algorithm that finds an orientation that is a* $(1.75+\gamma, 1+1/\gamma)$*-approximation for* GRAPH BALANCING*, for every* $0 \leq \gamma \leq 1/4$*.*

Both the above theorems provide a smooth tradeoff between makespan and orientation cost while capturing previous known results for GB as special cases, *i.e.*, Theorem 1 captures the $(2,1)$ and $(11/6, 3/2)$ approximations of [15] and [18] for $\gamma = 1/4$ and $\gamma = 1/12$ respectively, whereas Theorem 2 captures the $(1.75, \infty)$-approximation of [3] for $\gamma = 0$. Theorem 1 is depicted in Figure 1.

1.1.2 Lower Bounds

We present bicirteria lower bounds for GB. As previously mentioned, our lower bounds apply to a strengthening of the relaxation of [3], which we denote by LP_k (see subsection 3.2). The lower bound is summarized in the follwing theorem and is depicted in Figure 1.

▶ **Theorem 3.** *For every* $0 \leq \gamma < 1/4$ *and* $\epsilon > 0$*, there exists an instance for* GRAPH BALANCING *and target makespan* T *such that: (1)* LP_k *is feasible and has value of* OPT_{LP_k}*, and (2) every orientation whose makespan is at most* $(1.75+\gamma)T$ *has orientation cost of at least* $1/(\gamma+0.75+\epsilon) OPT_{LP_k}$*.*

To the best of our knowledge, all algorithms for GB that find an orientation that achieves an approximation better than 2 with respect to the makespan use the relaxation of [3] (or no relaxation at all, e.g., [6]). [2]

1.1.3 Extensions

Using our general framework, we present bicriteria algorithms for extensions of GB. The extensions of GB we consider allow hyperedges and unrelated weights to the edges. It is important to note that all the upper bounds presented below hold for the single criterion versions of these problems as well. In particular, we achieve an approximation strictly better than 2, with respect to the makespan, to several problems that capture GB and are not captured by the RESTRICTED ASSIGNMENT problem (RA).[3] To the best of our knowledge, this is the first polynomial time algorithm with approximation factor better than 2 to the makespan for problems that capture GB and are not captured by RA. Let us now elaborate on these extensions.

[2] Recently, Jansen and Rohwedder [10] showed that using a different stronger relaxation called the configuration LP one can achieve an approximation of less than 1.75 to the makespan. However, this result does not produce a polynomial time algorithm that orients the edges but rather only approximates the *value* of the optimal makespan. Moreover, this result has an unbounded loss with respect to the orientation cost.

[3] The RA is a special case of GAP where each job has a set of machines it can be assigned to, and has an equal processing time on each of them.

The first extension allows for light unrelated hyperedges. Formally, given $\beta \in [0,1]$, the input can contain hyperedges whose weight with respect to the vertices it shares may vary, as long as it does not exceed β (we may assume without loss of generality that the largest weight in p equals 1). We denote this problem by Graph Balancing with Unrelated Light Hyper Edges (Gbuh(β)). A special case of this problem was introduced by Huang and Ott in [6] who presented a $(5/3 + \beta/3, \infty)$-approximation when $\beta \in [4/7, 1)$. We improve upon [6] in three aspects. First, we consider the general bicriteria problem, *i.e.*, orientation costs are present, and achieve bounded loss with respect to the total orientation cost (recall that [6] cannot handle orientation costs). Second, we allow any $\beta \in [0,1]$, where [6] allows for $\beta \in [4/7, 1)$ only. Third, we allow the hyperedges to be unrelated, *i.e.*, different weights to different endpoints, where hyperedge weights in [6] are related. Our result for this extension is summarized in the following theorem.

▶ **Theorem 4.** *Let $0 \leq \beta \leq 1$. For every $\max\{1/12, \beta/3 - 1/12\} \leq \gamma \leq 1/4$, there exists a polynomial time algorithm that finds an orientation that is a $(1.75 + \gamma, 1/(2\gamma+0.5))$-approximation to* Gbuh($\beta$).

The second extension further generalizes the first one, and it also allows edges to have unrelated weights as long as the weights are greater than β. Unfortunately, we prove that this problem in its full generality is as hard to approximate as Gap. However, if it is assumed that the optimal makespan is at least 1 (as before we can assume without loss of generality that the largest weight in p equals 1), we can achieve improved results. We denote this problem by Graph Balancing with Unrelated Light Hyper Edges and Unrelated Heavy Edges (Gbu(β)). [4]

▶ **Theorem 5.** *Let $\beta \geq \sqrt{2} - 1$. For every $\beta/3 - 1/12 \leq \gamma \leq 1/4$, there exists a polynomial time algorithm that finds an orientation that is a $(1.75 + \gamma, 1/(2\gamma+0.5))$-approximation to* Gbu($\beta$).

We prove that there are values of β for which the bicriteria approximation of Theorem 5 is tight. Specifically, we prove the latter for $\beta = 1/2$ and LP_k. The lower bounds are summarized in the following theorem.

▶ **Theorem 6.** *For every $\epsilon > 0$, there exists an instance of* Gbu(0.5) *that is feasible to LP_k and every orientation has a makespan of at least $11/6 - \epsilon$. Moreover, for every $1/12 \leq \gamma \leq 1/4$, target makespan T and $\epsilon > 0$, there exists an instance for* Gbu(0.5) *that is feasible to LP_k and has a value of OPT_{LP_k}, and every orientation with makespan at most $(1.75 + \gamma)T$ has an orientation cost of at least $(1-\epsilon)/(2\gamma+0.5) \cdot OPT_{LP_k}$.*

In the third and final extension we allow the edges in Gb to be unrelated, but the weights cannot vary arbitrarily. Formally, given a parameter $c \geq 1$, every edge $e = (u, v)$ satisfies $p_{e,u} \leq c \cdot p_{e,v}$ and $p_{e,v} \leq c \cdot p_{e,u}$. We denote this problem by Semi-Related Graph Balancing (Srgb(c)). The following theorem summarizes our algorithm for Srgb(c).

▶ **Theorem 7.** *There exists a polynomial time algorithm to* Srgb(c)*, that finds an orientation that is a $(1.5 + 0.5a, 1/a)$-approximation, where a is the root in the range $[0.5, 1]$ of the polynomial:*

$$(1/c + 1/2) \cdot a^3 + (5/(2c) - 1/2) \cdot a^2 - 7/(2c) \cdot a + 1/c.$$

[4] While the assumption that the largest weight p equals 1 implies that the optimal makespan is least 1 for Gb and Gbuh(β), this is not necessarily the case when the edge weights might be unrelated. Thus, the assumption in Gbu(β) that the optimal makespan is at least 1 is not without loss of generality.

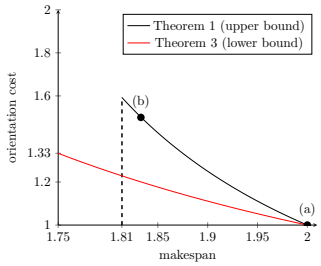

Figure 1 Our bicriteria bounds for Graph Balancing. (a) is given in Shmoys and Tardos [15], whereas (b) is implicitly given in Wang and Sitters [18].

We remark that the approximation guaranteed by Theorem 7 is never worse than 2 since it can be proved that $a = 1 - \Omega(1/c)$, yielding a $(2 - \Omega(1/c), 1 + O(1/c))$-approximation. It is worth noting that when $c = \infty$, which corresponds to the most general case, even the configuration LP has an integrality gap of 2 with respect to the makespan (see [4, 17]).

1.2 Our Techniques

We present a remarkably simple framework that allows us to provide bicriteria upper bounds for both Gb and its extensions, *i.e.*, Gbuh(β), Gbu(β), and Srgb(c). The framework is based on rounding of a strengthening of the linear relaxation of [3].

The rounding is comprised of two complementary steps, the first *local* and the second *global*. Intuitively, in the first local step, each edge can be oriented to one of its endpoints in case the relaxation indicates a strong (fractional) inclination toward that endpoint. We note that in order to quantify this inclination the weight of the edge is taken into account, where lighter edges are less likely to be oriented. Specifically, denote by $x_{e,u} \in [0, 1]$ how much the relaxation fractionally orients edge $e = (u, v)$ toward its endpoint u. The local step orients e toward u if $x_{e,u} > f(p_e)$ for some non-increasing threshold function $f : [0, 1] \to [1/2, 1]$. As previously mentioned, this step is considered *local* since only $x_{e,u}$ and p_e are used to determine whether to orient e, and if so to which of its two endpoints.[5] In the second global step of the rounding, we consider the remaining edges which were not yet oriented in the first local step and apply the algorithm of Shmoys and Tardos [15] which finds a minimum cost perfect matching in a suitable bipartite graph. As previously mentioned, this step is considered *global* since all edges which are not yet oriented are taken into consideration when computing the matching.

The above two-phase rounding is not sufficient on its own to obtain our claimed results, and we further strengthen the relaxation of [3] by forcing additional new constraints. Intuitively, for every vertex u our constraints state that if a collection of edges S touching u has total weight of more than T then not all edges in S can be chosen. We enforce the above constraints for all subsets of size at most k, for some fixed parameter k, resulting in a strengthened linear relaxation which we denote by LP_k. It is important to note that these constraints cannot be inferred from the original relaxation of [3], and thus are required in our analysis of the above two-phase rounding.

[5] This rounding was used in Wang and Sitters [18] with a specific "step" threshold function f to implicitly obtain a $(11/6, 3/2)$-approximation for Gb.

1.3 Additional Related Work

Lenstra et al. [11] introduced the classic well known 2-approximation to the single criterion GAP. They also proved that no polynomial time algorithm can approximate the makespan within a factor less than 1.5 unless $P = NP$. This was followed by Shmoys and Tardos [15] who introduced the bicriteria GAP and presented a $(2, 1)$-approximation for it. A slightly improved approximation of $2 - 1/m$ for the makespan was given by Shchepin and Vakhania [14]. If the number of machines is fixed polynomial time approximation schemes are known [5, 8]. For the case of uniformly related machines (each machine i has speed s_i and assigning job j to machine i takes p_j/s_i time) Hochbaum and Shmoys [13] presented a polynomial time approximation scheme. The RESTRICTED ASSIGNMENT problem (RA) is a special case were each job has an equal processing time on the machines it can be assigned to (for every job j and machine i: $p_{i,j} \in \{p_j, \infty\}$). For this special case, Svensson [16] proved that one can approximate the value of the optimal makespan by a factor of $33/17$ using the configuration LP, that was first introduced by Bansal et al. for the SANTA CLAUS PROBLEM [1]. This was subsequently improved by Jansen and Rohwedder [9] who presented an approximation of $11/6$. If one further assumes that the processing times have only two possible values [7] presented an improved approximation of $5/3$. The above results [16, 9, 7] do not present polynomial time algorithms that produce a schedule with the promised makespan, but only approximate the value of the makespan.

When considering GB, Jansen and Rohwedder [10] recently showed a similar flavor result: using the configuration LP one can estimate the value of the optimal makespan by a factor of $1.75 - \epsilon$, for some small constant $\epsilon > 0$. However, as before, [10] does not produce an orientation in polynomial time. The special case of GB where only two processing times are present admits a (tight) 1.5-approximation (given independently by [6, 2, 12]).

To the best of our knowledge, no work on GB considered orientation costs and in particular the tradeoff between makespan and orientation cost.

1.4 Paper Organization

Section 2 contains the required preliminaries. In Section 3 we present our general framework and apply it to GB to obtain bicriteria algorithms. Section 4 contains our bicriteria lower bound for GB. Finally, in Section 5 we consider the mentioned extensions of GB and apply the framework to these extensions to obtain improved algorithms.

2 Preliminaries

Given a multi-graph $G = (V, E)$ and a vertex $u \in V$ denote by $\delta(u) \triangleq \{e \in E \mid u \in e\}$ the collection of edges incident to u. In addition define: $\mathcal{F}(u) \triangleq \{S \subseteq \delta(u) \mid \sum_{e \in S} p_e \leq 1\}$, *i.e.*, the collection of feasible subsets of edges incident to u (for simplicity of presentation we further assume without loss of generality that $T = 1$ since we can scale all processing times by T). Moreover, we denote by OPT_{LP} and OPT_{LP_k} the optimal value of a feasible solution to the relaxation LP and LP_k respectively.

The algorithm of Shmoys and Tardos [15] is a key ingredient in our framework, thus we present it not only for completion but also since understanding its inner-working helps in analyzing our algorithms. Recall that [15] is a $(2, 1)$-approximation for GAP. We assume without loss of generality that $T = 1$ since one can scale the processing times by T. First, the relaxation in Figure 2 is solved, where $\mathcal{J}$ is the set of jobs and $\mathcal{M}$ is the set of machines.

(LP_{GAP})

$$\begin{aligned}
\min \quad & \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{M}} x_{i,j}c_{i,j} \\
\text{s.t} \quad & \sum_{i\in\mathcal{M}} x_{i,j} = 1 && \forall j \in \mathcal{J} && (Job) \\
& \sum_{j\in\mathcal{J}} x_{i,j}p_{i,j} \leq 1 && \forall i \in \mathcal{M} && (Load) \\
& x_{i,j} = 0 && \forall i \in \mathcal{M}, j \in \mathcal{J} : p_{i,j} > 1 \\
& x_{i,j} \geq 0 && \forall i \in \mathcal{M}, j \in \mathcal{J}
\end{aligned}$$

Figure 2 The relaxation by Shmoys and Trados [15] to GAP.

The variable $x_{i,j}$, for each $i \in \mathcal{M}$ and $j \in \mathcal{J}$, indicates whether job j is scheduled on machine i. Note that if there is no feasible solution to the relaxation, then the algorithm declares there is no schedule with makespan at most T.

Given a solution $\mathbf{x}$ to LP_{GAP}, the algorithm of [15] constructs a weighted bipartite graph $G = (\mathcal{J}, S, E)$, which will be described shortly. Afterwards, the algorithm finds a minimum cost perfect matching to the side $\mathcal{J}$, *i.e.*, each vertex in $\mathcal{J}$ is matched to a vertex in S. Using this matching the algorithm assigns each job to a machine. The bipartite graph G is constructed as follows, where we assume that $\mathcal{J} = \{1, 2, \ldots, n\}$ is the set of jobs and S is a collection of "slots". Machine i is allocated $k_i \triangleq \lceil \sum_{j=1}^{n} x_{i,j} \rceil$ slots which we denote by $slot(i, 1), \ldots slot(i, k_i)$, each having a capacity of 1. For each machine i sort the jobs in a non-increasing order of their processing time $p_{i,j}$, and for each job j in this order add $x_{i,j}$ units of job j to the next non-full slot of machine i (starting from $slot(i, 1)$). If $x_{i,j}$ is larger than the remaining capacity of the slot, which we denote by r, add r units of job j to that slot and $x_{i,j} - r$ units of job j to the next slot. An edge connecting job j and a slot (i, ℓ) is added to E if some of the $x_{i,j}$ units of j were added to the slot (i, ℓ), and its cost is set to $c_{i,j}$. A description of [15] appears in Algorithm 1.

Algorithm 1 Shmoys-Tardos $(\mathbf{x}, \mathbf{p}, \mathbf{c})$.

1. Construct the bipartite graph $G = (\mathcal{J}, S, E)$ as described above.
2. Find in G a minimum cost perfect matching with respect to $\mathcal{J}$.
3. For each job $j \in \mathcal{J}$, assign j to machine i if the slot that is matched to j belongs to i.

We say a slot is *full* if the remaining capacity of that slot is 0. Additionally, we say a job j is on *top* of a slot if j is the first job to be inserted to that slot. It can be proved that the load on machine i in the output of Algorithm 1 is at most $1 + p_{i,1}$, where $p_{i,1}$ is the processing time of the job on top of $slot(i, 1)$, *i.e.*, the largest processing time of a job that is fractionally scheduled on machine i. Since, $p_{i,1} \leq 1$, the makespan of the assignment is at most 2. Furthermore, it can be shown that the cost of the assignment is at most $OPT_{LP_{GAP}}$, and thus at most $C(T)$.

We remark that when one is aiming to solve the single criterion version of this problem, *i.e.*, finding an assignment that minimizes the makespan, a binary search could be preformed to find the smallest T such that the linear relaxation is feasible. In general, any (α, β)-approximation for the bicriteria problem implies an approximation of α for the single criteria problem.

(LP)

$$\min \quad \sum_{e \in E} \sum_{u \in e} x_{e,u} c_{e,u}$$

$$\text{s.t} \quad \sum_{u \in e} x_{e,u} = 1 \qquad \forall e \in E \qquad (Edge)$$

$$\sum_{e \in \delta(u)} x_{e,u} p_e \leq 1 \qquad \forall u \in V \qquad (Load)$$

$$\sum_{e \in \delta(u):\ p_e > 0.5} x_{e,u} \leq 1 \quad \forall u \in V \qquad (Star)$$

$$x_{e,u} \geq 0 \qquad \forall u \in V, e \in \delta(u)$$

Figure 3 The relaxation by Ebenlendr et al. [3] to GB.

3 The General Framework and Graph Balancing

We start by describing the general framework in the setting of GB. For simplicity of presentation, given a target makespan T, if there exists an edge e such that $p_e > T$ the algorithm immediately declares that there is no orientation with makespan at most T. Otherwise, we scale the processing times by T. Thus, without loss of generality, $T = 1$ and $p_e \leq 1$ for every $e \in E$.

Currently, we consider the relaxation of [3], which we denote by LP, with the addition of an objective function that minimizes the orientation cost.[6] This relaxation appears in Figure 3.

Note that the Star constraint of LP implies that at most a total fraction of 1 of *big* edges, *i.e.*, edges whose weight is larger than $1/2$, can be oriented toward u. Moreover, we note that later we strengthen this relaxation by adding additional constraints.

Once the processing times are scaled by T, the algorithm solves the relaxation LP. If there is no feasible solution to the relaxation, then the algorithm declares that there is no orientation with makespan at most T. Thus, from this point onward we assume that LP is feasible and focus on the rounding.

Recall that the rounding consists of only two steps, the first local and the second global. In the first step, some of the edges might be oriented, where an edge e is oriented toward u if $x_{e,u} > f(p_e)$ for a given threshold function $f : [0, 1] \to [1/2, 1]$. We employ the framework for threshold functions f which are monotone non-increasing, thus making lighter edges less likely to be oriented compared to heavier edges. In the second step, the remaining un-oriented edges are oriented using Algorithm 1. The framework is described in Algorithm 2. It receives as an input: (1) the graph $G = (V, E, \mathbf{p}, \mathbf{c})$; (2) $\mathbf{x}$ a solution to the relaxation; (3) a threshold function f.

Algorithm 2 Framework$(G = (V, E, \mathbf{p}, \mathbf{c}), \mathbf{x}, f)$.

1 For each edge e and $u \in e$: if $x_{e,u} > f(p_e)$ then orient e to u and remove e from E. (*Local Step*)
2 Execute Algorithm 1. (*Global Step*)

[6] In Section 5 we also need the constraint that appears in the relaxation of [15] which states that $x_{e,u} = 0$ if $p_{e,u} > 1$, for every $e \in E$ and $u \in e$.

Note that the *Local Step* of Algorithm 2 is well defined, *i.e.*, no edge is oriented to both its endpoints. This is due to the Edge constraints and the fact that for each $p \in [0,1]$: $f(p) \geq 1/2$. Note that the framework captures Algorithm 1 as a special case since one can choose $f \equiv 1$. We now focus on bounding the makespan and orientation cost produced by the framework, for a general threshold function f. This analysis will be useful for the rest of the paper.

Let us start by focusing on bounding the makespan. We start by presenting a simple but crucial observation. The observation states that if an edge $e = (u, v)$ was *not* oriented at the *Local Step* then $x_{e,u}$ and $x_{e,v}$ cannot vary much. It is important to note that this is the *only* place in our proof we use the fact that e is an edge, *i.e.*, the job that corresponds to e can be assigned to only two machines u and v (otherwise our algorithm could have been applied to the more general problem of Ra).

▶ **Observation 1.** *Let $e = (u, v) \in E$ such that e was not oriented to either u or v in the Local Step. Then $1 - f(p_e) \leq x_{e,u} \leq f(p_e)$.*

Proof. e was not oriented toward u in the *Local Step*, and therefore $x_{e,u} \leq f(p_e)$. Additionally, the Edge constraint implies that $x_{e,v} = 1 - x_{e,u}$, and since e was not oriented toward v in the *Local Step* then $1 - x_{e,u} \leq f(p_e)$. This concludes the proof. ◀

Now we focus on bounding the makespan. Fix a vertex $u \in V$, and denote the slots that were allocated to u in Algorithm 1 by: $slot(u,1), ..., slot(u,k)$ or alternatively by $s_1, ..., s_k$. For $i \in \{1, 2, \ldots, k\}$ let e_i be the edge on top of $slot(u,i)$ and denote its processing time by p_i. We assume without loss of generality that $p_{k+1} \triangleq 0$ and $x_{e_{k+1},u} = 1$ (one can simply add a 0 weight edge that is fully oriented toward u).[7] Additionally, denote by $e'_1, \ldots, e'_t$ the edges that were oriented to u in the *Local Step*, and denote by $q_1, \ldots, q_t$, their processing times respectively. Lastly, for a slot s and edge e we denote by $y_{e,s}$ the fraction that e is assigned to s.

We now introduce a new observation that lower bounds the fractional load in the first slot of u, *i.e.*, $\sum_{e \in slot(u,1)} y_{e,s_1} p_e$. This observation will be useful in bounding the load on u.

▶ **Observation 2.** *The fractional load in the first slot of u is at least:*

$$\sum_{e \in slot(u,1)} y_{e,s_1} p_e \geq (1 - f(p_1))p_1 + f(p_1)p_2.$$

Proof. From Observation 1 we know that $x_{e_1} \geq 1 - f(p_1)$. Moreover, since e_1 is the first edge to be inserted to the first slot, then it is contained fully in $slot(u,1)$. Therefore, $y_{e_1,s_1} = x_{e_1,u} \geq 1 - f(p_1)$. Recall that $p_2 \leq p_1$. Since $slot(u,1)$ is full and its capacity equals 1, we can conclude: $\sum_{e \in slot(u,1)} y_{e,s_1} p_e \geq f(p_1)p_1 + (1 - f(p_1))p_2$. ◀

Now we introduce a lemma that is inspired by [15] and upper bounds the load on u.

▶ **Lemma 8.** *Let $e'_1, \ldots, e'_t$ be the edges that were oriented to u in the Local Step, and let $q_1, \ldots, q_t$ be their processing times respectively. Then,*

$$\sum_{i=1}^{t} q_i + \sum_{i=1}^{k} p_i \leq 1 + \sum_{i=1}^{t} (1 - f(q_i))q_i + f(p_1)p_1 + (1 - f(p_1))p_2.$$

[7] Alternatively, we can also assume $p_{k+2} = 0$ and $x_{e_{k+2},u} = 1$ as well.

Proof. First, recall that for every $1 \leq s \leq k-1$ $slot(u,s)$ has a capacity exactly 1. Moreover, the slots are filled with edges in decreasing order of processing time. Therefore, we can deduce that for each $1 \leq i \leq k-1$:

$$\sum_{e \in slot(u,i)} y_{e,s_i} p_e \geq \sum_{e \in slot(u,i)} y_{e,s_i} p_{i+1} = p_{i+1} \sum_{e \in slot(u,i)} y_{e,s_i} = p_{i+1}.$$

Since at most one edge from each slot can be selected in the *Global Step*, the load on u from edges that oriented to u in the *Global Step* is at most $\sum_{i=1}^{k} p_i$. From the above inequality, along with Observation 2, we can conclude that:

$$\begin{aligned}
\sum_{i=1}^{t} q_i + \sum_{i=1}^{k} p_i &= \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{i=3}^{k} p_i \leq \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{i=2}^{k-1} \sum_{e \in slot(u,i)} y_{e,s_i} p_e \\
&\leq \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{i=1}^{k} \sum_{e \in slot(u,i)} y_{e,s_i} p_e - \sum_{e \in slot(u,1)} y_{e,s_1} p_e \\
&\leq \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{e \in \delta(u)} x_{e,u} p_e - ((1 - f(p_1))p_1 + f(p_1)p_2) \\
&\leq \sum_{i=1}^{t} q_i + f(p_1)p_1 + (1 - f(p_1))p_2 + \left(1 - \sum_{i=1}^{t} f(q_i)q_i\right) \\
&= 1 + \sum_{i=1}^{t} (1 - f(q_i))q_i + f(p_1)p_1 + (1 - f(p_1))p_2.
\end{aligned}$$

The last inequality follows from the Load constraint on u, and the fact that the edges $e'_1, \ldots, e'_t$ were removed from E at the end of the *Local Step*. ◀

Lastly, we observe that all of the *big* edges, *i.e.*, edges whose weight is larger than $1/2$, that are not oriented toward u in the *Local Step* are assigned to the first slot. This is summarized in the following observation.

▶ **Observation 3.** *Let e be an edge in $slot(u,i)$ such that $i > 1$. Then, $p_e \leq 1/2$.*

Proof. Assume for the sake of contradiction that $p_e > 1/2$. Since the slots are filled in a non-increasing weight order, all edges in slots $1, 2, \ldots, i-1$ are filled with fractions of edges whose processing time is greater than $1/2$. Therefore, $\sum_{e \in \delta(u):\ p_e > 1/2} x_{e,u} > 1$, which contradicts the Star constraint on u. ◀

Let us now focus on the orientation cost. The following lemma upper bounds the orientation cost of the orientation produced by Algorithm 2.

▶ **Lemma 9.** *Given $f : [0,1] \to [0, 1/2]$, let $c \triangleq (inf\{f(p) | p \in [0,1]\})^{-1}$. Then Algorithm 2 with f outputs an orientation with a cost of at most $c \cdot C(T)$.*

3.1 Graph Balancing – Upper Bound on Tradeoff Between Makespan and Orientation Cost

Let us now focus on applying the framework, with an appropriate threshold function f, to GB. First, we present a theorem that achieves part of the tradeoff claimed in Theorem 1, and only in the next subsection we show how to extend this tradeoff to fully achieve Theorem 1.

▶ **Theorem 10.** *There exists a threshold function f such that Algorithm 2 finds an orientation that is a $(1.75+\gamma, 1/(2\gamma+0.5))$-approximation, for every $1/12 \leq \gamma \leq 1/4$.*

The function f_α we use in the proof of Theorem 10 is the following:

$$f_\alpha(p_e) = \begin{cases} 1 & \text{if } p_e \leq 1/2 \\ \alpha & \text{if } p_e > 1/2 \end{cases} \tag{1}$$

where $2/3 \leq \alpha \leq 1$. The following lemma upper bounds the makespan of Algorithm 2 with the above f_α.

▶ **Lemma 11.** *The makespan of the orientation produced by Algorithm 2 with f_α is at most: $1.5+0.5\alpha$, where $2/3 \leq \alpha \leq 1$.*

Proof. Consider the number of edges that were oriented toward u in the *Local Step*. First, we note that from the Star constraint on u, at most one edge can be oriented toward u in the *Local Step*. If this is not the case then let e'_1 and e'_2 be edges oriented to u in the *Local Step*. Then, $p_{e_1}, p_{e_2} > 1/2$. However, $x_{e_1,u} + x_{e_2,u} > \alpha + \alpha \geq 2/3 + 2/3 > 1$, which contradicts the Star constraint on u. Hence, there are only two cases to consider.

Case 1: Assume no edge is oriented toward u in the *Local Step*. Therefore, using Lemma 8 and Observation 3 the load on u is at most:

$$\begin{aligned} \sum_{i=1}^{k} p_i &\leq 1 + f_\alpha(p_1)p_1 + (1-f_\alpha(p_1))p_2 \leq 1.5 + f_\alpha(p_1)(p_1 - 0.5) \\ &\leq 1.5 + \alpha \cdot (1-0.5) = 1.5 + 0.5\alpha, \end{aligned}$$

where the last inequality follows from the fact that the expression: $f_\alpha(p_1)(p_1 - 0.5)$ is maximized when $p_1 = 1$ (and thus $f_\alpha(p_1) = \alpha$).

Case 2: Assume there is exactly one edge that was oriented toward u in the *Local Step*. Recall we denote this edge as e'_1 and its processing time by q_1. Since $q_1 > 1/2$ and $x_{e'_1,u} > \alpha$, then it must be the case that $p_1 \leq 1/2$ (otherwise Observation 1 implies that $x_{e'_1,u} + x_{e_1,u} > \alpha + 1 - \alpha = 1$, which contradicts the Star constraint for u). Therefore, from Lemma 8 the load on u in the output of Algorithm 2 is at most:

$$\begin{aligned} q_1 + \sum_{i=1}^{k} p_i &\leq 1 + (1-f_\alpha(q_1))q_1 + f_\alpha(p_1)p_1 + (1-f_\alpha(p_1))p_2 \leq 1 + (1-\alpha)q_1 + p_1 \\ &\leq 1 + (1-\alpha) + 0.5 = 2.5 - \alpha \leq 1.5 + 0.5\alpha. \end{aligned}$$

The second inequality follows from the fact that $p_2 \leq p_1$ and $f_\alpha(q_1) = \alpha$ (since $q_1 > 1/2$), whereas the third inequality from the fact that $p_1 \leq 1/2$. In addition, the last inequality follows from the fact that $2/3 \leq \alpha \leq 1$. ◀

Now, we are ready to conclude the proof of Theorem 10:

Proof of Theorem 10. Applying Lemma 11, Lemma 9 and choosing $\gamma = 0.5\alpha - 0.25$ finishes the proof. ◀

We now show that the analysis of Algorithm 2 with a threshold function f_α is tight. Formally, we prove the following lemma:

▶ **Lemma 12.** *For every* $1/2 \leq \alpha < 1$ *there exists an instance such that the output of Algorithm 2 with* f_α *has makespan at least max* $\{1.5 + 0.5\alpha, 2.5 - \alpha\}$ *and orientation cost at least* $1/\alpha \cdot OPT_{LP}$.

Lemma 12 shows the analysis of Algorithm 2 with a threshold function f_α is tight. Consequently, in order to extend the bicriteria tradeoff of Theorem 10, and obtain Theorem 1, we require a different threshold function and a stronger relaxation.

3.2 Graph Balancing – Extending the Tradeoff

It is important to note that Lemma 12 implies that using Algorithm 2 with LP and the threshold function f_α cannot achieve an approximation better than $11/6$ with respect to the makespan. To this end we strengthen LP using the following constraint (which we denote by Set constraints):

$$\sum_{e \in S} x_{e,u} \leq |S| - 1 \qquad \forall u \in V, \forall S \subseteq \delta(u) : S \notin \mathcal{F}(u) \text{ and } |S| \leq k \quad (Set)$$

We call the new relaxation LP_k.[8] Intuitively, the Set constraints enforce that given an infeasible set of edges S touching u not all edges of S can be oriented toward u. In fact, for our specific choice of a threshold function f we use $k = 3$. Thus, no separation oracle is required when solving the relaxation. The exact result is formulated in the following theorem:

▶ **Theorem 13.** *There exists a rounding function* f *such that Algorithm 2 finds an orientation that is a* $(1.75 + \gamma, 1/(2\gamma+0.5))$*-approximation, for every* $1/12 - \epsilon/2 \leq \gamma \leq 1/12$, *where* $\epsilon = \sqrt{33}/2 - 17/6$.

Note that this theorem extends the tradeoff achieved in Theorem 10, and together both theorems achieve the tradeoff of Theorem 1. The threshold function f_ϵ we use in the proof of Theorem 13 is defined as follows:

$$f_\epsilon(p_e) = \begin{cases} 2/3 - \epsilon & \text{if } p_e > 1/2 \\ 2/3 + \epsilon/2 & \text{if } 1/3 < p_e \leq 1/2 \\ 1 & \text{if } p_e \leq 1/3 \end{cases} \tag{2}$$

where $0 \leq \epsilon \leq \sqrt{33}/2 - 17/6$. The following lemma upper bounds the makespan:

▶ **Lemma 14.** *The output of Algorithm 2 with the threshold function* f_ϵ, *has a makespan of at most* $11/6 - \epsilon/2$.

Now we conclude with the proofs of Theorems 13 and 1.

Proof of Theorem 13. Follows immediately from Lemmas 14 and 9, and choosing $\gamma = 1/12 - \epsilon/2$. ◀

Proof of Theorem 1. Follows immediately from Theorems 10 and 13. ◀

[8] Similarly to LP, for some of the extensions of GB we add that $x_{e,u} = 0$ if $p_{e,u} > 1$ (for every $e \in E$ and $u \in e$).

4 Lower Bound on The Tradeoff Between Makespan and Cost

We show that using LP_k for every $k \in \mathbb{N}$, one must loose in the total orientation cost when obtaining an approximation for the makespan that is strictly better than 2. This is in contrast to the classic result of [15] for which one can achieve an approximation factor of 2 with respect to the makespan with no loss in the assignment cost. This result is formulated in Theorem 3.

5 Extending Graph Balancing to Hyperedges and Unrelated Weights

5.1 Graph Balancing with Unrelated Light Hyperedges

Let us recall the definition of GBUH(β), where $\beta \in [0, 1]$. The input consists of a hypergraph, where each vertex represents a machine and each hyperedge represents a job. The jobs are of two types, "light" and "heavy". Every light hyperedge $e \in E$ is associated with weights $p_{e,u}$, one for each vertex $u \in e$ (*i.e.*, e is unrelated since it has a different processing time for each of the machines it can be assigned to). The requirement is that $p_{e,u} \leq \beta$ for evert $u \in e$. On the other hand, every heavy hyperedge $e \in E$ must in fact be an edge, *i.e.*, $|e| = 2$. Such a heavy e is associated with a single weight $p_e \in [0, 1]$ (*i.e.*, e is related since it has the same processing time for each of the two machines it can be assigned to). In the above, as previously mentioned, we assume without loss of generality that the largest weight equals 1. For both types, light and heavy, orienting e toward one of its endpoints is equivalent to assigning the job e represents to the machine that is represented by the vertex e was oriented to. It is important to note that when $\beta = 1$ the problem is exactly GAP, and when $\beta = 0$ the problem is exactly GB.

Our result for GBUH(β) is summarized in Theorem 4, which improves upon the previous result of [6] (refer to Section 1 for a thorough discussion on how our result improves upon [6]). To the best of our knowledge, our result provides the first approximation better than 2 with respect to the makespan of a natural problem that captures GB but is not captured by RA.

5.2 Graph Balancing with Unrelated Light Hyperedges and Unrelated Heavy Edges

The problem of GBU(β) further generalizes the above GBUH(β) as it allows heavy edges to have unrelated weights. Formally, every heavy edge $e = (u, v) \in E$ is associated with two weights $p_{e,u}$ and $p_{e,v}$, *i.e.*, e is unrelated since $p_{e,u}$ indicates the processing time of the job e represents on the machine that is represented by u. The requirement is that $p_{e,u}, p_{e,v} \in (\beta, 1]$. As mentioned earlier, it is assumed that the value of the optimal makespan is at least 1 (otherwise the problem is as hard as GAP). Our results relating to GBU(β) are formulated in Theorems 5 and 6.

5.3 Semi-Related Graph Balancing

Consider the general problem of UNRELATED GRAPH BALANCING, which is identical to GB except that an edge can have a different weight depending on its orientation: $p_{e,u}$ and $p_{e,v}$ for every $e = (u, v) \in E$, *i.e.*, the weights are unrelated. This generalization of GB was already considered in [17, 4], who presented lower bounds for the problem. Specifically, they showed that the even the configuration LP (which captures LP_k) has an integrality gap of 2 with respect to the makespan.

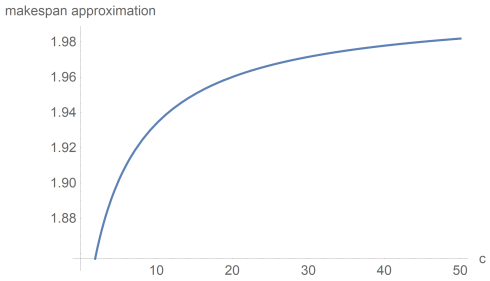

Figure 4 Makespan approximation as a function of the value c.

We consider an interesting special case of the above problem where the weights are still unrelated, but cannot vary arbitrarily. Formally, each edge $e = (u, v) \in E$ has two weights depending on the vertex e is oriented to, which satisfy: $p_{e,u} \leq c \cdot p_{e,v}$ and $p_{e,v} \leq c \cdot p_{e,u}$ (where $c \geq 1$ is a parameter of the problem). We denote this problem by Semi-Related Graph Balancing (Srgb(c)).

Our result for Srgb(c) is formulated in Theorem 7. Note that Srgb(c) captures Gb when $c = 1$, and indeed in Theorem 7 we achieve a $(11/6, 3/2)$-approximation for Srgb(c) when $c = 1$ (similarly to Theorem 10). Moreover, when $c = \infty$ Theorem 7 achieves a $(2, 1)$-approximation for Srgb(c), matching the integrality gap of [17, 4]. Finally, we also show that in general Theorem 7 provides a $(2 - \Omega(1/c), 1 + O(1/c))$-approximation for Srgb($c$).

Figure 4 shows the makespan approximation obtained in Theorem 7 as a function of c.

In order to prove Theorem 7 we use Algorithm 2 and LP_k (replacing p_e with $p_{e,u}$) with a suitable choice of a threshold function f.

References

1. 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. doi:10.1145/1132516.1132522.
2. Deeparnab Chakrabarty and Kirankumar Shiragur. Graph Balancing with Two Edge Types. *CoRR*, abs/1604.06918, 2016. arXiv:1604.06918.
3. Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 483–490, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347135.
4. Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph Balancing: A Special Case of Scheduling Unrelated Parallel Machines. *Algorithmica*, 68(1):62–80, 2014. doi:10.1007/s00453-012-9668-9.
5. Ellis Horowitz and Sartaj Sahni. Exact and Approximate Algorithms for Scheduling Nonidentical Processors. *J. ACM*, 23:317–327, April 1976.
6. 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.
7. Klaus Jansen, Kati Land, and Marten Maack. Estimating the Makespan of the Two-Valued Restricted Assignment Problem. *Algorithmica*, 80(4):1357–1382, 2018. doi:10.1007/s00453-017-0314-4.
8. Klaus Jansen and Lorant Porkolab. Improved Approximation Schemes for Scheduling Unrelated Parallel Machines. *Mathematics of Operations Research*, 26(2):324–338, 2001.

9 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.

10 Klaus Jansen and Lars Rohwedder. Local search breaks 1.75 for Graph Balancing. *CoRR*, abs/1811.00955, 2018. arXiv:1811.00955.

11 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.

12 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.

13 Dorit S. Hochbaum and David Shmoys. A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using the Dual Approximation Approach. *SIAM J. Comput.*, 17:539–551, June 1988.

14 Evgeny V. Shchepin and Nodari Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33(2):127–133, 2005.

15 David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62:461–474, 1993. doi:10.1007/BF01585178.

16 Ola Svensson. Santa Claus schedules jobs on unrelated machines. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 617–626, 2011. doi:10.1145/1993636.1993718.

17 José Verschae and Andreas Wiese. On the configuration-LP for scheduling on unrelated machines. *J. Scheduling*, 17(4):371–383, 2014. doi:10.1007/s10951-013-0359-4.

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

19 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE.