Graph Balancing with Orientation Costs

Ran Yeheskel
Graph Balancing with Orientation Costs

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Ran Yeheskel

Submitted to the Senate of the Technion — Israel Institute of Technology
Shvat 5780 Haifa February 2020
This research was carried out under the supervision of Roy Schwartz, in the Faculty of Computer Science.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author’s doctoral research period, the most up-to-date versions of which being:


Acknowledgements

I would like to thank my advisor Roy Schwartz. Thank you for you dedicated guidance and care. It was a great pleasure working with you.

The generous financial help of the Technion is gratefully acknowledged.
# Contents

List of Figures

Abstract 1

1 Introduction 3

1.1 Our Results 4

1.2 Our Techniques 7

1.3 Additional Related Work 8

2 Preliminaries 11

3 The General Framework and Graph Balancing 15

4 Graph Balancing — Upper Bound on Tradeoff Between Makespan and Orientation Cost 19

4.1 Graph Balancing — Extending the Tradeoff 22

5 A Second Tradeoff for GB With Orientation Costs 27

6 Lower Bound on The Tradeoff Between Makespan and Cost 31

7 Extending Graph Balancing to Hyperedges and Unrelated Weights 33

7.1 Graph Balancing with Unrelated Light Hyperedges 33

7.2 Graph Balancing with Unrelated Light Hyperedges and Unrelated Heavy Edges 35

7.2.1 Hardness Result Without The Assumption on The Optimal Value 35

7.2.2 An Upper Bound on The Tradeoff Between Makespan and Orientation Costs 36

7.2.3 Integrality Gaps 39

7.3 Semi-Related Graph Balancing 43

8 Conclusion and Open Questions 49

8.1 Open Questions 49

Hebrew Abstract i
List of Figures

1.1 Our bicriteria bounds for Graph Balancing. (a) is given in Shmoys and Tardos [ST93], whereas (b) is implicitly given in Wang and Sitters [WS16]. ............................................................ 7

2.1 The relaxation for GAP by [ST93] ................................................. 11

2.2 The relaxation for GB by [EKS08] .................................................. 12

4.1 Instance with gap of $1.5 + 0.5\alpha$ w.r.t. the makespan, for Lemma 4.0.2. . 21

4.2 Instance with gap of $2.5 - \alpha$ w.r.t. the makespan, for Lemma 4.0.2 . . 22

6.1 Integrality gap instance for orientation cost for $LP_k$. .......... 31

7.1 A single path from the makespan integrality gap instance for GBUH(0.5). There is a dedicated load value of 1/3 on each vertex. All vertices excluding $u_i$ share a light hyperedge of weight 0.5. Above the edges are the weight of the edge depending on the orientation, where the plus sign denotes $1 - 6\epsilon$, and the minus sign denotes $0.5 + \epsilon$. ....................... 40

7.2 The instance of the integrality gap in the proof of Lemma 7.2.4. Each vertex has a load value of $\gamma' + 0.25 - (4\gamma' + 0.5)\epsilon'$, and they share a single hyperedge of weight 0.5. The plus and minus signs of the edges represent the weight with respect to each endpoint. The plus sign represents a weight of $1 - \epsilon'$ and the minus sign represents a weight of $0.5 + \epsilon'$. Orienting $e'$ clockwise, incurs a cost of 1. .................. 41

7.3 Makespan approximation as a function of the value c. ............... 44
Abstract

In the well known Generalized Assignment Problem we are given a set of $n$ machines and $m$ jobs that need to be processed on these machines. Additionally, for each pair of job $j$ and machine $i$ we are given the time it takes for machine $i$ to process job $j$, and cost of assigning job $j$ to machine $i$. The goal is to find an assignment of the jobs to the machines that minimizes (1) the total assignment costs and (2) the makespan of the assignment, i.e., the time it takes to process all the jobs by the machines (where each machine, in parallel with respect to the other machines, processes its assigned job).

Motivated by the classic Generalized Assignment Problem, we consider the Graph Balancing problem in the presence of orientation costs: a special case of Generalized Assignment Problem where every job can be assigned to at most two machines and has an equal processing time on those machines. Graph Balancing was first introduced by Ebenlendr-Krcál-Sgall [Algorithmica ‘14] and can be formulated as a graph optimization problem: 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. We also present bicriteria lower bounds in the form of integrality gaps that show that a loss in the total orientation cost is required if one aims for an approximation better than $2$ in the makespan. Our framework is based on a remarkably simple rounding of a strengthened linear relaxation. We believe this framework might be of independent interest to other related scheduling problems.
Chapter 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. [EKS08] and since its introduction it has attracted much attention (see, e.g., [JR18, WS16, HO16, CS16, PS16]). 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., [ST93, EKS08, Sve11, WS11]). In GAP we are given a collection \( M \) of \( m \) machines and a collection \( J \) 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 [ST93] 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 \( M \) to be \( V \) and \( J \) to be \( E \); and (2) for each job \( j \in J \) (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 \( \infty \) 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. [EKS08] 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 \((\alpha, \beta)\)-approximation if given a target makespan \( T \), it outputs an orientation with makespan at most \( \alpha T \) and total orientation cost at most \( \beta C(T) \). Thus, [ST93] is a \((2,1)\)-approximation to GB. We note that the algorithm of [EKS08] cannot handle orientation costs and is in fact a \((1.75, \infty)\)-approximation for GB. A result by Wang and Sitters [WS16] 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 (in the form of integrality gaps). 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. 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 \( \infty \).
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 1.1. 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 1.2. 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 1.1 captures the \((2, 1)\) and \((11/6, 3/2)\) approximations of [ST93] and [WS16] for \(\gamma = 1/4\) and \(\gamma = 1/12\) respectively, whereas Theorem 1.2 captures the \((1.75, \infty)\)-approximation of [EKS08] for \(\gamma = 0\). Theorem 1.1 is depicted in Figure 1.1.

Lower Bounds: We present bicriteria lower bounds for GB. Our lower bounds apply to a strengthening of the relaxation of [EKS08], which we denote by \(LP_k\) (see subsection 4.1). The lower bound is summarized in the following theorem and is depicted in Figure 1.1.

Theorem 1.3. 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 \(1/(\gamma + 0.75 + \epsilon)OPT_{LP_k}\).

It is important to note that the above result might seem surprising, since one might expect that similarly to the classic result of [ST93] a \((1.75, 1)\)-approximation for GB is possible. We refute this when even a strengthening of the relaxation of [EKS08] is used. 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 [EKS08] (or no relaxation at all, e.g., [HO16]).

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.

\footnote{Recently, Jansen and Rohwedder [JR18] showed that using a different 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.}
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.

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 $\beta$ (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($\beta$)). A special case of this problem was introduced by Huang and Ott in [HO16] who presented a $(5/3 + \beta/3, \infty)$-approximation when $\beta \in [4/7, 1)$. We improve upon [HO16] 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 [HO16] cannot handle orientation costs). Second, we allow any $\beta \in [0, 1]$, where [HO16] 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 [HO16] are related. Our result for this extension is summarized in the following theorem.

**Theorem 1.4.** 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 $\beta$. 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($\beta$)).

**Theorem 1.5.** 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 $\beta$ for which the bicriteria approximation of Theorem 1.5 is tight. Specifically, we prove the latter for $\beta = 1/2$ and LP$_k$. The lower bounds are summarized in the following theorem.

**Theorem 1.6.** 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

---

3The 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.

4While the assumption that the largest weight $p$ equals 1 implies that the optimal makespan is least 1 for GB and GBUH($\beta$), this is not necessarily the case when the edge weights might be unrelated. Thus, the assumption in GBU($\beta$) that the optimal makespan is at least 1 is not without loss of generality.
1/12 ≤ γ ≤ 1/4, target makespan T and ε > 0, there exists an instance for GBU(0.5) that is feasible to LPk and has a value of OPTLPk, and every orientation with makespan at most (1.75 + γ)T has an orientation cost of at least (1−ε)/(2γ+0.5)·OPTLPk.

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 ≥ 1, every edge e = (u, v) satisfies p_{e,u} ≤ c · p_{e,v} and p_{e,v} ≤ c · p_{e,u}. We denote this problem by Semi-Related Graph Balancing (SRGB(c)). The following theorem summarizes our algorithm for SRGB(c).

**Theorem 1.7.** 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:

\[(\frac{1}{c} + \frac{1}{2}) \cdot a^3 + (\frac{5}{2c} - \frac{1}{2}) \cdot a^2 - \frac{7}{2c} \cdot a + \frac{1}{c}.\]

We remark that the approximation guaranteed by Theorem 1.7 is never worse than 2 since it can be proved that a = 1 - Ω(1/c), yielding a (2 - Ω(1/c), 1 + O(1/c))-approximation. It is worth noting that when c = ∞, which corresponds to the most general case, even the configuration LP has an integrality gap of 2 with respect to the makespan (see [EKS14, VW14]).

### 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 [EKS08].

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 \( f : [0,1] \rightarrow [1/2,1] \). 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 [ST93] 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 [EKS08] 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 [EKS08], and thus are required in our analysis of the above two-phase rounding.

1.3 Additional Related Work

Lenstra et al. [LST90] 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 [ST93] 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 [SV05]. If the number of machines is fixed polynomial time approximation schemes are known [HS76, JP01]. 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 [SHS88] 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 [Sve11] 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

\(^5\)This rounding was used in Wang and Sitters [WS16] with a specific “step” threshold function \( f \) to implicitly obtain a \((11/6,3/2)\)-approximation for GB.
Problem [BS06]. This was subsequently improved by Jansen and Rohwedder [JR17] who presented an approximation of \( \frac{11}{6} \). If one further assumes that the processing times have only two possible values [JLM18] presented an improved approximation of \( \frac{5}{3} \). The above results [Sve11, JR17, JLM18] 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 [JR18] 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, [JR18] 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 [HO16, CS16, PS16]).

To the best of our knowledge, no work on GB considered orientation costs and in particular the tradeoff between makespan and orientation cost.
Chapter 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 [ST93] 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 [ST93] 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 2.1 is solved, where \( J \) is the set of jobs and \( M \) is the set of machines. The variable \( x_{i,j} \), for each \( i \in M \) and \( j \in J \), 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 \( x \) to \( LP_{GAP} \), the algorithm of [ST93] constructs a weighted bipar-

\[
\min \sum_{j \in J} \sum_{i \in M} x_{i,j} c_{i,j} \\
\text{s.t.} \quad \sum_{i \in M} x_{i,j} = 1 \quad \forall j \in J \quad \text{(Job)} \\
\sum_{j \in J} x_{i,j} p_{i,j} \leq 1 \quad \forall i \in M \quad \text{(Load)} \\
x_{i,j} = 0 \quad \forall i \in M, j \in J : p_{i,j} > 1 \\
x_{i,j} \geq 0 \quad \forall i \in M, j \in J
\]

Figure 2.1: The relaxation for \( GAP \) by [ST93]
\[
\begin{align*}
\min & \sum_{e \in E} \sum_{u \in e} x_{e,u}c_{e,u} \\
\text{s.t} & \sum_{u \in e} x_{e,u} = 1 & \forall e \in E & \quad (\text{Edge}) \\
& \sum_{e \in \delta(u)} x_{e,u}p_e \leq 1 & \forall u \in V & \quad (\text{Load}) \\
& \sum_{e \in \delta(u): p_e > 0.5} x_{e,u} \leq 1 & \forall u \in V & \quad (\text{Star}) \\
& x_{e,u} \geq 0 & \forall u \in V, e \in \delta(u) 
\end{align*}
\]

Figure 2.2: The relaxation for GB by [EKS08]

tite graph \( G = (J, S, E) \), which will be described shortly. Afterwards, the algorithm finds a minimum cost perfect matching to the side \( J \), i.e., each vertex in \( J \) 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 \( J = \{1, 2, \ldots, n\} \) is the set of jobs and \( S \) is a collection of “slots”. Machine \( i \) is allocated \( k_i = \left\lceil \sum_{j=1}^{n} x_{i,j} \right\rceil \) slots which we denote by \( \text{slot}(i, 1), \ldots \text{slot}(i, k_i) \), 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 \( \text{slot}(i, 1) \)). 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 \( (i, \ell) \) is added to \( E \) if some of the \( x_{i,j} \) units of \( j \) were added to the slot \( (i, \ell) \), and its cost is set to \( c_{i,j} \). A description of [ST93] appears in Algorithm 2.1.

**Algorithm 2.1 Shmoys-Tardos \((x, p, c)\)**

1: Construct the bipartite graph \( G = (J, S, E) \) as described above.
2: Find in \( G \) a minimum cost perfect matching with respect to \( J \).
3: For each job \( j \in J \), assign job \( j \) 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 2.1 is \( 1 + p_{i,1} \), where \( p_{i,1} \) is the processing time of the job on top of \( \text{slot}(i, 1) \), 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 \( OPT_{LP_{cap}} \), 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 $(\alpha, \beta)$-approximation for the bicriteria problem implies an approximation of $\alpha$ for the single criteria problem.
Chapter 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 [EKS08], which we denote by $LP$, with the addition of an objective function that minimizes the orientation cost.\(^\dagger\) This relaxation appears in Figure 2.2. 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 $f : [0,1] \rightarrow [1/2,1]$. 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 2.1. The framework is described in Algorithm 3.1. It receives as an input: (1) the graph $G = (V,E,p,c)$; (2) $x$ a solution to the relaxation; (3) a threshold function $f$.

Note that the Local Step of Algorithm 3.1 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

\[^\dagger\]In Section 7 we also need the constraint that appears in the relaxation of [ST93] which states that $x_{e,u} = 0$ if $p_{e,u} > 1$, for every $e \in E$ and $u \in e$. 

Algorithm 3.1 Framework\((G = (V, E, p, c), x, f)\)

1: for all \(e \text{ and } u \in e\) do
2: \[\text{if } x_{e,u} > f(p_e) \text{ then orient } e \text{ to } u \text{ and remove } e \text{ from } E.\] (Local Step)
3: end for
4: Execute Algorithm 2.1. (Global Step)

\[p \in [0, 1]: f(p) \geq \frac{1}{2}.\] Note that the framework captures Algorithm 2.1 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.

**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 3.0.1.** 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 2.1 by: \(\text{slot}(u, 1), \ldots, \text{slot}(u, k)\) or alternatively by \(s_1, \ldots, s_k\). For \(i \in \{1, 2, \ldots, k\}\) let \(e_i\) be the edge on top of \(\text{slot}(u, i)\) and denote its processing time by \(p_i\). We assume without loss of generality that \(p_{k+1} \equiv 0\) and \(x_{e_{k+1}, u} = 1\) (one can simply add a 0 weight edge that is fully oriented toward \(u\)).\(^2\) 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., \(\sum_{e \in \text{slot}(u, 1)} y_{e,s_1} P_e\). This observation will be useful in bounding the load on \(u\).

**Observation 3.0.2.** The fractional load in the first slot of \(u\) is at least:

\[\sum_{e \in \text{slot}(u, 1)} y_{e,s_1} P_e \geq (1 - f(p_1))p_1 + f(p_1)p_2.\]

\(^2\)Alternatively, we can also assume \(p_{k+2} = 0\) and \(x_{e_{k+2}, u} = 1\) as well.
Proof. From Observation 3.0.1 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 \( \text{slot}(u,1) \). Therefore, \( y_{e_1,s_1} = x_{e_1,u} \geq 1 - f(p_1) \). Recall that \( p_2 \leq p_1 \). Since \( \text{slot}(u,1) \) is full and its capacity equals 1, we can conclude: \( \sum_{e \in \text{slot}(u,1)} y_{e,s_1}p_e \geq f(p_1)p_1 + (1 - f(p_1))p_2. \)

Now we introduce a lemma that is inspired by [ST93] and upper bounds the load on \( u \).

**Lemma 3.0.3.** 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,

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

**Proof.** First, recall that for every \( 1 \leq s \leq k - 1 \), \( \text{slot}(u,s) \) 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 \( 1 \leq i \leq k - 1 \):

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

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 3.0.2, we can conclude that:

\[
\begin{align*}
\sum_{i=1}^{t} q_i + \sum_{i=1}^{k} p_i & = \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{i=1}^{k} p_i \leq \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{i=2}^{k-1} \sum_{e \in \text{slot}(u,i)} y_{e,s_i}p_e \\
& \leq \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{i=1}^{k} \sum_{e \in \text{slot}(u,i)} y_{e,s_i}p_e - \sum_{e \in \text{slot}(u,1)} y_{e,s_1}p_e \\
& \leq \sum_{i=1}^{t} q_i + p_1 + p_2 + \sum_{e \in \delta(u)} x_{e,u}p_e - ((1 - f(p_1))p_1 + f(p_1)p_2) \\
& \leq \sum_{i=1}^{t} q_i + f(p_1)p_1 + (1 - f(p_1))p_2 + (1 - \sum_{i=1}^{t} f(q_i)q_i) \\
& = 1 + \sum_{i=1}^{t} (1 - f(q_i))q_i + f(p_1)p_1 + (1 - f(p_1))p_2.
\end{align*}
\]

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. \( \blacksquare \)

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 3.0.4.** Let \( e \) be an edge in \( \text{slot}(u,i) \) such that \( i > 1 \). Then, \( p_e \leq 1/2 \).
Proof. Assume for the sake of contradiction that $p_e > \frac{1}{2}$. 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 $\frac{1}{2}$. Therefore, $\sum_{e \in \delta(u)}: p_e > \frac{1}{2} x_{e,u} > 1$, which contradicts the Star constraint on $u$. ■

Orientation Cost: The following lemma upper bounds the orientation cost of the orientation produced by Algorithm 3.1.

Lemma 3.0.5. Given $f : [0, 1] \to [1/2, 1]$, let $c \triangleq (\inf \{ f(p) | p \in [0, 1] \})^{-1}$. Then Algorithm 3.1 with $f$ outputs an orientation with a cost of at most $c \cdot C(T)$.

Proof. Let $S$ be the set of edges that were oriented to $u$ in the Local Step of Algorithm 3.1. Let $\eta$ be the orientation the algorithm outputs, i.e., $\eta(e)$ equals the vertex that $e$ is oriented to. Since Algorithm 2.1 does not lose in the assignment cost:

$$\sum_{e \in E \setminus S} c_{e,\eta(e)} \leq \sum_{e = (u,v) \in E \setminus S} (c_{e,u} x_{e,u} + c_{e,v} x_{e,v}).$$

For each $e \in S$: $x_{e,\eta(e)} > f(p_e) \geq \frac{1}{c}$. So we deduce: $c \cdot x_{e,\eta(e)} \geq 1$. Therefore, $c_{e,\eta(e)} \leq c \cdot x_{e,\eta(e)} c_{e,\eta(e)}$. Then we can conclude:

$$\sum_{e \in S} c_{e,\eta(e)} \leq c \sum_{e = (u,v) \in S} (c_{e,u} x_{e,u} + c_{e,v} x_{e,v}).$$

Since $1 \leq c \leq 2$, we can deduce:

$$\sum_{e \in E} c_{e,\eta(e)} \leq c \sum_{e = (u,v) \in E} (c_{e,u} x_{e,u} + c_{e,v} x_{e,v}) \leq c \cdot OPT_{LP},$$

where $OPT_{LP}$ is the optimal value of feasible solution to the relaxation. This implies that the orientation outputted by Algorithm 3.1 with a rounding function $f$, has a cost of at most $c \cdot C(T)$. ■
Chapter 4

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 1.1, and only in the next subsection we show how to extend this tradeoff to fully achieve Theorem 1.1.

**Theorem 4.1.** There exists a threshold function $f$ such that Algorithm 3.1 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 $f_\alpha$ we use in the proof of Theorem 4.1 is the following:

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

(4.1)

where $2/3 \leq \alpha \leq 1$. The following lemma upper bounds the makespan of Algorithm 3.1 with the above $f_\alpha$.

**Lemma 4.0.1.** The makespan of the orientation produced by Algorithm 3.1 with $f_\alpha$ 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 3.0.3 and Observation 3.0.4 the load on $u$ is at most:

$$\sum_{i=1}^{k} p_i \leq 1 + f_\alpha(p_1)p_1 + (1 - f_\alpha(p_1))p_2$$

$$\leq 1.5 + f_\alpha(p_1)(p_1 - 0.5)$$

$$\leq 1.5 + \alpha \cdot (1 - 0.5)$$

$$= 1.5 + 0.5\alpha,$$

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 3.0.1 implies that $x_{e_1',u} + x_{e_1,u} > \alpha + 1 - \alpha = 1$, which contradicts the Star constraint for $u$). Therefore, from Lemma 3.0.3 the load on $u$ in the output of Algorithm 3.1 is at most:

$$q_1 + \sum_{i=1}^{k} p_i \leq 1 + (1 - f_\alpha(q_1))q_1 + f_\alpha(p_1)p_1 + (1 - f_\alpha(p_1))p_2$$

$$\leq 1 + (1 - \alpha)q_1 + p_1$$

$$\leq 1 + (1 - \alpha) + 0.5$$

$$= 2.5 - \alpha$$

$$\leq 1.5 + 0.5\alpha.$$

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 4.1:

**Proof of Theorem 4.1.** Applying Lemma 4.0.1, Lemma 3.0.5 and choosing $\gamma = 0.5\alpha - 0.25$ finishes the proof. ■

**Tightness of Analysis:** We now show that the analysis of Algorithm 3.1 with a threshold function $f_\alpha$ is tight. Formally, we prove the following lemma:

**Lemma 4.0.2.** For every $1/2 \leq \alpha < 1$ there exists an instance such that the output of Algorithm 3.1 with $f_\alpha$ has makespan at least $\max \{1.5 + 0.5\alpha, 2.5 - \alpha\}$ and orientation cost at least $1/\alpha \cdot OPT_{LP}$.

**Proof.** We introduce two instances, both of them feasible to $LP$. The output of Algorithm 3.1 on the first instance has makespan of $1.5 + 0.5\alpha$ and orientation cost of $1/\alpha \cdot OPT_{LP}$. The output of Algorithm 3.1 on the second instance has the same orientation cost, however the makespan is $2.5 - \alpha$. This proves the desired result.
Let $\epsilon > 0$. The first instance is shown in Figure 4.1. The processing time of an edge is written above it. Furthermore, there is a load value, i.e., a self loop, on each vertex with a specified weight. These load values are denoted by $q$, e.g., $q_u$ denotes the load value of vertex $u$. We note that in order to ensure the Star constraints of $LP$ are feasible, we split each self loop into two self loops (each with half the original load value). In addition to the information in Figure 4.1, there are orientation costs to each edge: $c_{(u,v_1),v_1} = c_{(u,v_2),v_2} = \epsilon$ and $c_{(u',v'),v'} = 1$. The rest of the orientation costs are 0.

The only feasible solution to $LP$ is $x_{(u,v_1),u} = 1 - \alpha$, $x_{(u,v_2),u} = \alpha + \epsilon$, and $x_{(u',v'),u'} = \alpha + \epsilon$. The reason that this is the only feasible solution is that the fractional load on each vertex is exactly 1 and there are no cycles in the graph. Moreover, the fractional cost of the orientation is:

$$OPT_{LP} = \alpha + \epsilon + \epsilon(\alpha + 1 - \alpha - \epsilon) \leq \alpha + 2\epsilon.$$ 

Algorithm 3.1 with threshold function $f_\alpha$ will orient $(u',v')$ toward $u'$ in the Local Step. Moreover, since Algorithm 2.1 finds minimum cost matching the edges $(u,v_1)$ and $(u,v_2)$ are oriented toward $u$ in the Global Step. This is due to the fact that the cost of the fractional orientation is $\epsilon(\alpha + 1 - \alpha - \epsilon) = \epsilon(1 - \epsilon)$, so orienting an edge toward $v_1$ or $v_2$ will incur a cost of at least $\epsilon$. In conclusion, the output of Algorithm 3.1 with $f_\alpha$ as a threshold function, has makespan of $1.5 + 0.5\alpha - 0.5\epsilon$. Additionally, the orientation cost is $1 = \frac{1}{(\alpha+\epsilon)} \cdot OPT_{LP}$.

The second instance is shown in Figure 4.2. Similar to the previous instance, the processing time of an edge is written above it, and the load value of a vertex is written next to it. In addition, the orientation costs are $c_{(u,v_1),v_1} = 1$ and $c_{(u,v_2),v_2} = \epsilon$ (all
Figure 4.2: Instance with gap of $2.5 - \alpha$ w.r.t. the makespan, for Lemma 4.0.2.

other orientation costs are 0).

The only feasible solution to $LP$ is $x_{(u,v),u} = \alpha + 0.5\epsilon$ and $x_{(u,v_2),u} = \epsilon$. Note that in this case the load on each vertex is exactly 1. Since there are no cycles in this graph this is the only feasible solution to $LP$. The fractional orientation cost is:

$$OPT_{LP} = \alpha + 0.5\epsilon + \epsilon(0.5 + 0.5\epsilon).$$

Algorithm 3.1 with threshold function $f_\alpha$, will orient the edge $(u, v_1)$ toward $u$ in the Local Step. In addition, since Algorithm 2.1 finds a minimum cost matching, $(u, v_2)$ is oriented toward $u$ in the Global Step. To conclude, the makespan is $2.5\alpha - \epsilon$ and the orientation cost is $1 = 1/(\alpha + (1+0.5\epsilon))OPT_{LP}$. This concludes the proof. ■

Lemma 4.0.2 shows the analysis of Algorithm 3.1 with a threshold function $f_\alpha$ is tight. Consequently, in order to extend the bicriteria tradeoff of Theorem 4.1, and obtain Theorem 1.1, we require a different threshold function and a stronger relaxation.

4.1 Graph Balancing – Extending the Tradeoff

It is important to note that Lemma 4.0.2 implies that using Algorithm 3.1 with $LP$ and the threshold function $f_\alpha$ 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):

$$\sum_{e \in S} x_{e,u} \leq |S| - 1 \quad \forall u \in V, \forall S \subseteq \delta(u) : S \notin \mathcal{F}(u) \text{ and } |S| \leq k \quad (Set)$$
We call the new relaxation $LP_k$. 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 4.2.** There exists a rounding function $f$ such that Algorithm 3.1 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 4.1, and together both theorems achieve the tradeoff of Theorem 1.1. The threshold function $f_\epsilon$ we use in the proof of Theorem 4.2 is defined as follows:

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

where $0 \leq \epsilon \leq \sqrt{33/2} - 17/6$. The following lemma upper bounds the makespan (we give the proof later on in this chapter).

**Lemma 4.1.1.** The output of Algorithm 3.1 with the threshold function $f_\epsilon$, has a makespan of at most $11/6 - \epsilon/2$.

Now we conclude with the proofs of Theorems 4.2 and 1.1.

**Proof of Theorem 4.2.** Follows immediately from Lemmas 4.1.1 and 3.0.5, and choosing $\gamma = 1/12 - \epsilon/2$. ■

**Proof of Theorem 1.1.** Follows immediately from Theorems 4.1 and 4.2. ■

Before proving Lemma 4.1.1, we first show a simple but crucial observation that follows directly from the Set constraints of our strengthened relaxation $LP_k$.

**Observation 4.1.2.** Let $u \in V$ and $S \subseteq \delta(u)$. If $\sum_{e \in S} x_{e,u} > |S| - 1$, then $\sum_{e \in S} p_e \leq 1$.

**Proof.** Follows immediately from the Set constraint of $u$ and $S$. ■

Now we give the proof of Lemma 4.1.1:

**Proof of Lemma 4.1.1.** Fix a vertex $u \in V$. We start with the following simple observation:

---

1Similarly 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 $u \in e$).
Observation 4.1.3. The number of edges oriented to \( u \) in the Local Step of Algorithm 3.1 is at most 2.

Proof. Assume for the sake of contradiction that \( e'_1, e'_2, e'_3 \) were oriented to \( u \) in the Local Step. First, denote the weight of \( e'_i \) by \( q_i \). From the Star constraint on \( u \), there is at most one edge \( e'_i \) (among the three edges) such that \( q_i > \frac{1}{2} \). Therefore we can deduce:

\[
x_{e'_1,u} + x_{e'_2,u} + x_{e'_3,u} > f(p_{e'_1}) + f(p_{e'_2}) + f(p_{e'_3}) \\
\geq \frac{2}{3} - \epsilon + \frac{2}{3} + \epsilon + \frac{2}{3} + \epsilon \\
> 2.
\]

From Observation 4.1.2, we get that: \( q_1 + q_2 + q_3 \leq 1 \), but on the other hand from the definition of \( f_\epsilon \) (see 4.2), \( q_i > \frac{1}{3} \), for each \( i \), which is a contradiction. ■

Therefore we denote by \( e'_1, e'_2 \) the edges oriented to \( u \) in the Local Step, and their weights by \( q_1, q_2 \). If there is one (or no) edges oriented to \( u \) in the Local Step, then both \( x_{e'_2,u} \) and \( q_2 \) equal 0 (or all of \( x_{e'_1,u}, q_1, x_{e'_2,u}, q_2 \) equal 0) respectively.

The second observation states the sum \( q_1 + q_2 \) cannot be too large. Formally:

Observation 4.1.4. It must be the case that \( q_1 + q_2 \leq 1 \).

Proof. If \( q_2 = 0 \) or \( q_1 = q_2 = 0 \) then trivially, \( q_1 + q_2 \leq 1 \). Otherwise, \( q_1, q_2 > 0 \), i.e., both \( e'_1, e'_2 \) are edges that were oriented to \( u \) in the Local Step. Therefore,

\[
x_{e'_1,u} + x_{e'_2,u} > f_\epsilon(q_1) + f_\epsilon(q_2) > \frac{1}{2} + \frac{1}{2} = 1.
\]

From Observation 4.1.2 (choosing \( S = \{e'_1, e'_2\} \)), we conclude: \( q_1 + q_2 \leq 1 \). ■

We now consider three cases regarding the value of \( p_1 \) and bound the load on \( u \) in each case concisely (the bound we achieve is \( \frac{11}{6} - \epsilon/2 \)).

Case 1: In this case \( p_1 \leq \frac{1}{3} \). From Lemma 3.0.3 we get that the load on \( u \) is at most:

\[
1 + (1 - f_\epsilon(q_1))q_1 + (1 - f_\epsilon(q_2))q_2 + f_\epsilon(p_1)p_1 + (1 - f_\epsilon(p_1))p_2 \leq 1 + \left( \frac{1}{3} + \epsilon \right) (q_1 + q_2) + p_1 \\
\leq 1 + \left( \frac{1}{3} + \epsilon \right) + \frac{1}{3} \\
= \frac{5}{3} + \epsilon \\
< \frac{11}{6} - \frac{\epsilon}{2}.
\]

The first inequality follows from the definition of \( f_\epsilon \) (and the fact that \( p_2 \leq p_1 \)). Additionally, the second inequality follows from Observation 4.1.4.
**Case 2:** In this case \( p_1 > 1/2 \). Note that it must hold that \( q_1, q_2 \leq 1/2 \) (otherwise Observation 3.0.1 implies that \( x_{e_1,u} + x_{e_1',u} > 1 - f_\epsilon(p_1') + f_\epsilon(p_1) = 1 - f_\epsilon(p_1) + f_\epsilon(p_1) = 1 \), which contradicts the Star constraint on \( u \)). Hence, \( 1/3 < q_1, q_2 \leq 1/2 \).

Moreover it holds that \( p_1 + q_i \leq 1 \) (Observation 4.1.2 applied to the set \( S = \{ e_1', e_1 \} \) along with Observation 3.0.1). From Lemma 3.0.3 we derive that the load on \( u \) is at most:

\[
1 + (1 - f_\epsilon(q_1)) q_1 + (1 - f_\epsilon(q_2)) q_2 + f_\epsilon(p_1) p_1 + (1 - f_\epsilon(p_1)) p_2 \\
= 1 + \left( \frac{1}{3} - \frac{\epsilon}{2} \right)(q_1 + q_2) + \left( \frac{2}{3} - \epsilon \right) p_1 + \left( \frac{1}{3} + \epsilon \right) p_2 \\
= 1 + \left( \frac{1}{3} - \frac{\epsilon}{2} \right)(q_1 + q_2) + \left( \frac{1}{3} - \frac{\epsilon}{2} \right)(q_2 + p_1) + \left( \frac{1}{3} + \epsilon \right) p_2 \\
\leq 1 + \left( \frac{1}{3} - \frac{\epsilon}{2} \right) + \left( \frac{1}{3} - \frac{\epsilon}{2} \right) + \left( \frac{1}{3} + \epsilon \right) \cdot \frac{1}{2} \\
= 1 + \frac{2}{3} - \epsilon + \frac{1}{6} + \frac{\epsilon}{2} \\
= \frac{11}{6} - \frac{\epsilon}{2}.
\]

Note that the second equality follows since \( \epsilon \geq 0 \).

**Case 3:** In this case, \( \frac{1}{3} < p_1 \leq 1/2 \). From Lemma 3.0.3 we can bound the load on \( u \) by:

\[
1 + (1 - f_\epsilon(q_1)) q_1 + (1 - f_\epsilon(q_2)) q_2 + f_\epsilon(p_1) p_1 + (1 - f_\epsilon(p_1)) p_2 \\
\leq 1 + \left( \frac{1}{3} + \epsilon \right)(q_1 + q_2) + \left( \frac{2}{3} + \frac{\epsilon}{2} \right) p_1 + \left( \frac{1}{3} - \frac{\epsilon}{2} \right) p_2.
\]

Instead of using Observation 3.0.4 we use a different bound on \( p_2 \) that comes from the Load constraint on \( u \) and from Observation 3.0.2:

\[
\left( \frac{2}{3} - \epsilon \right)(q_1 + q_2) + \left( \frac{1}{3} - \frac{\epsilon}{2} \right) p_1 + \left( \frac{2}{3} + \frac{\epsilon}{2} \right) p_2 \leq x_{e_1',u} q_1 + x_{e_2'} q_2 + \sum_{e \in \delta(u,1)} y_{e,s_1} p_e \\
\leq \sum_{e \in \delta(u)} x_{e,u} p_e \leq 1.
\]

Therefore, we can deduce:

\[
p_2 \leq \left( \frac{2}{3} + \frac{\epsilon}{2} \right)^{-1} \left( 1 - \left( \frac{2}{3} - \epsilon \right)(q_1 + q_2) - \left( \frac{1}{3} - \frac{\epsilon}{2} \right) p_1 \right).
\]
This bound on $p_2$ and Lemma 3.0.3 give us the following bound on the load on $u$:

\[
1 + \left(\frac{1}{3} + \epsilon\right)(q_1 + q_2) + \left(\frac{2}{3} + \frac{\epsilon}{2}\right)p_1 \\
+ \left(\frac{1}{3} - \frac{\epsilon}{2}\right) \cdot \left(\frac{2}{3} + \frac{\epsilon}{2}\right)^{-1} \left(1 - \left(\frac{2}{3} - \epsilon\right)(q_1 + q_2) - \left(\frac{1}{3} - \frac{\epsilon}{2}\right)p_1\right) \\
= 1 + \frac{1}{3} - \frac{\epsilon}{2} + \left(\frac{1}{3} + \epsilon - \frac{1}{3} - \frac{\epsilon}{2}\right)(q_1 + q_2) + \left(\frac{2}{3} + \frac{\epsilon}{2} - \frac{(1/3 - \epsilon)^2}{2 + \frac{\epsilon}{2}}\right)p_1.
\]

It can be easily be verified that the coefficients of $(q_1 + q_2)$ and $p_1$ are non-negative for the range of $\epsilon$. Thus, the above expression is maximized when $p_1 = 1/2$ and $q_1 + q_2 = 1$.

Moreover, it can be easily verified that in that case the above expression (when setting $q_1 + q_2 = 1$ and $p_1 = 1/2$) equals:

\[
1 + \frac{2}{3} + \frac{5}{4}\epsilon - \frac{5}{8}\epsilon^2 - \frac{1}{3}\epsilon - \frac{1}{18}.
\]

Recalling that $\epsilon \leq \sqrt{\frac{33}{2}} - \frac{17}{6}$ implies that the above is at most: $\frac{11}{6} - \epsilon/2$, concluding the proof.

\[\blacksquare\]
Chapter 5

A Second Tradeoff for GB With Orientation Costs

In this section we present a simple rounding algorithm to LP. This algorithm achieves an approximation of $(1.75 + \gamma, 1 + 1/\gamma)$ to GB, where $0 < \gamma \leq 1$.

Our approach is similar to the one of Algorithm 3.1, which consists of two steps. The first local and second global. Similarly to Algorithm 3.1, in the local step if an edge is significantly fractionally oriented to a vertex we orient that edge to that vertex. However, there are two differences from Algorithm 3.1. First, the threshold function is constant, i.e. $f(p) \equiv \alpha$. Second, after the first Local Step, instead of Algorithm 2.1, we use the rounding algorithm by Ebenlendr et al. [EKS08]. This is summarized in the following algorithm.

Algorithm 5.1 ($G = (V, E, p, c), x, \alpha$)
1: for all $e \in E$ and $u \in e$ do
2: if $x_{e,u} > \alpha$, then orient $e$ to $u$ and remove $e$ from $E$. \hfill (Local Step)
3: end for
4: Execute the algorithm of Ebenlendr et al. [EKS08]. \hfill (Global Step)

We now elaborate on the reasons for choosing a constant threshold function to Algorithm 5.1. Note that in order to have a bounded loss in the orientation cost, for every $p$, it must hold that $f(p) < 1$. Otherwise, if there exists a value $p$ such that $f(p) = 1$, then an edge $e$ of weight $p$ is never rounded in the Local Step of Algorithm 5.1. Therefore, in the Global Step it might be oriented to a vertex $u$ in a Leaf Assignment rule (as defined in [EKS08]). It is also possible that $x_{e,u} = \epsilon$ for $\epsilon > 0$ arbitrarily small. Therefore, the Leaf Assignment step might incur an unbounded loss in the orientation cost.

Thus, we limit the threshold function to a constant function in order to make the analysis simpler. We present two lemmas that bound the makespan and orientation cost of Algorithm 5.1. The first lemma bounds the makespan of the output of Algorithm 5.1.
Lemma 5.0.1. The makespan of the orientation Algorithm 5.1 outputs is at most \( \frac{1}{\alpha} + 0.75 \).

The second lemma bound the orientation cost of the output of Algorithm 5.1:

Lemma 5.0.2. The orientation cost of the orientation Algorithm 5.1 outputs is at most \( \frac{1}{(1-\alpha)} \cdot C(T) \).

From the two lemmas above, the proof of Theorem 1.2 follows immediately:

Proof of Theorem 1.2. Follows immediately from Lemmas 5.0.1, 5.0.2 and choosing \( \alpha = \frac{1}{(1+\gamma)} \).

We now proceed to the proofs of Lemmas 5.0.1, 5.0.2.

Proof of Lemma 5.0.1. Fix a vertex \( u \). Denote by \( X \) the fractional load of edges that were oriented to \( u \) in the Local Step:

\[
X \triangleq \sum_{e \in E_{\text{orient}}} x_{e,u}p_e,
\]

where \( E_{\text{orient}} \) is the set of edges that were oriented to \( u \) in the Local Step. The load on \( u \) from edges oriented in the Local Step of Algorithm 5.1 is:

\[
\sum_{e \in E_{\text{orient}}} p_e \leq \sum_{e \in E_{\text{orient}}} \frac{1}{\alpha} x_{e,u}p_e = \frac{1}{\alpha} \sum_{e \in E_{\text{orient}}} x_{e,u}p_e = \frac{1}{\alpha} X.
\]

The first inequality is derived from the definition of the Local Step. In addition, the load on \( u \) from edges oriented in the Global Step is at most: \( 1 - X + 0.75 \). Note that the algorithm by Ebenlendr et al. [EKS08] adds at most 0.75 to the initial fractional load of each vertex (see the proof in [EKS08]). In this case the initial load is \( 1 - X \), since the initial fractional load equals 1 and a total of \( X \) was removed from \( E \) in the Local Step of Algorithm 5.1.

Adding the load on \( u \) from edges oriented in the Local Step and the Global Step of Algorithm 5.1 yields:

\[
\frac{1}{\alpha} X + (1 - X + 0.75) = 1.75 + \left( \frac{1}{\alpha} - 1 \right) X \\
\leq 1.75 + \left( \frac{1}{\alpha} - 1 \right) = \frac{1}{\alpha} + 0.75,
\]

which concludes the proof.
Proof of Lemma 5.0.2. Let \( e = (u, v) \in E \). Assume without loss of generality that \( e \) is oriented to \( u \) in the output of Algorithm 5.1. We prove that:

\[
c_{e,u} \leq \frac{1}{1 - \alpha} \cdot (x_{e,u}c_{e,u} + x_{e,v}c_{e,v}).
\]

(5.1)

We consider two cases depending whether \( e \) was oriented to \( u \) in the Local Step or the Global Step of Algorithm 5.1. Assume \( e \) was oriented in the Local Step, then \( x_{e,u} > \alpha \). Thus,

\[
c_{e,u} \leq \frac{1}{\alpha} x_{e,u}c_{e,u}
\]

\[
\leq \frac{1}{\alpha} (x_{e,u}c_{e,u} + x_{e,v}c_{e,v})
\]

\[
\leq \frac{1}{1 - \alpha} (x_{e,u}c_{e,u} + x_{e,v}c_{e,v}).
\]

The last inequality follows from the fact that \( \alpha \geq 0.5 \).

Assume \( e \) was oriented to \( u \) in the Global Step. From Observation 7.3.2 (which still holds in this case) \( x_{e,u} \geq 1 - \alpha \). Hence we deduce:

\[
c_{e,u} \leq \frac{1}{1 - \alpha} x_{e,u}c_{e,u}
\]

\[
\leq \frac{1}{1 - \alpha} (x_{e,u}c_{e,u} + x_{e,v}c_{e,v}).
\]

Hence, we conclude that Inequality 5.1 holds. Summing Inequality 5.1 over all edges we obtain that the orientation cost of the output of Algorithm 5.1 is at most:

\[
\sum_{e \in E} c_{e,u} \leq \sum_{e \in E} \frac{1}{1 - \alpha} (x_{e,u}c_{e,u} + x_{e,v}c_{e,v})
\]

\[
= \frac{1}{1 - \alpha} \cdot OPT_{LP}.
\]

The proof is concluded since \( OPT_{LP} \leq C(T) \).
Chapter 6

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 [ST93] 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 1.3.

Proof of Theorem 1.3. Let $\epsilon > 0$ arbitrarily small and $k \in \mathbb{N}$. We show that for every $0 \leq \gamma < 0.25$, there is an instance that is feasible to $LP_k$, and any integral solution with makespan at most $1.75 + \gamma$ has a cost of at least $1/(\gamma+0.75+2\epsilon) \cdot OPT_{LP_k}$.

The instance we consider is shown in Figure 6.1. The weight of the edge $e = (u, v)$ is $1 - \epsilon$. Furthermore, the load values (self loops weights) are denoted by $q_u, q_v$. Note that a load value $q$ (whether it is $q_u$ or $q_v$) is split into $(q/\epsilon)$ self loops each with a weight of $1/\epsilon$. Additionally, we set the assignment costs to be $c_{e,v} = 1$ and $c_{e,u} = 0$.

There is exactly one feasible solution to $LP_k$, which is $x_{e,u} = 0.25 - \gamma - 2\epsilon$ and $x_{e,v} = 0.75 + \gamma + 2\epsilon$. The cost of this fractional solution is $OPT_{LP_k} = 0.75 + \gamma + 2\epsilon$.

There are two integral solutions:

1. Orienting $e$ toward $u$. This solution has makespan of $1.75 + \gamma + \epsilon$.

2. Orienting $e$ toward $v$. This solution has an orientation cost of $1 = 1/(0.75+\gamma+2\epsilon) \cdot$

![Figure 6.1: Integrality gap instance for orientation cost for $LP_k$.](image-url)
Therefore, every integral solution with makespan at most 1.75 + \gamma has an orientation cost of at least \frac{1}{(0.75+\gamma+2\epsilon)} \cdot OPT_{LP_k}. Choosing \epsilon' = \epsilon/2, and replacing \epsilon with \epsilon' in this proof, concludes the proof.
Chapter 7

Extending Graph Balancing to Hyperedges and Unrelated Weights

In this chapter we discuss about three extensions to Graph Balancing. In these three extensions we allow a combination of hyperedges and unrelated weights in the input graph, under some conditions. These extensions and their related results appear in the introduction as well as in this chapter.

7.1 Graph Balancing with Unrelated Light Hyperedges

Let us recall the definition of GBUH(\(\beta\)), 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 every \(u \in e\). 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(\(\beta\)) is summarized in Theorem 1.4, which improves upon the previous result of [HO16] (refer to Section 1 for a thorough discussion on how our result improves upon [HO16]). 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. We now give the proof of Theorem 1.4:

Proof of Theorem 1.4. Note that every instance of $\text{GBUH}(\beta)$, where $\beta < 1/2$, is an instance of $\text{GBUH}(0.5)$. Therefore it sufficient to prove the theorem for $\beta \geq 1/2$. Moreover, we say $e \in E$ is a heavy edge if it is not a light hyperedge and $p_e > \beta$. Note that each edge is either a light hyperedge or a heavy edge.

The proof is very similar to the one of Theorem 4.1. We use Algorithm 3.1 with $f_\alpha$ (see 4.1) as a threshold function (the same threshold function used in the proof of Theorem 4.1). The bounding of the orientation cost remains the same as in Lemma 3.0.5 (which still holds in this case). Therefore, the orientation cost is at most $\frac{1}{\alpha} \cdot C(T)$.

However, the makespan bounding in the proof of Theorem 4.1 does not hold. The main reason that the analysis is not the same is due to Observation 3.0.1. This observation holds only for edges and not for hyperedges. However, since the hyperedges have a bounded processing time, we can achieve a similar result. In the makespan bounding in the proof of Theorem 4.1, we consider two cases. For simplicity of presentation, we repeat the analysis as in the proof of Theorem 4.1 and elaborate on the differences.

Case 1: No edge was oriented toward $u$ in the Local Step of Algorithm 3.1. Using the same notations in the proof of Theorem 4.1, if $e_1$ (the edge on top of slot$(u, 1)$) is heavy, then the analysis remains valid. This is since $e_1$ is an edge and thus Observation 3.0.1 holds. Otherwise, $e_1$ is a light hyperedge, and thus $p_1 \leq \beta$. According to the original analysis of [ST93] (or Lemma 3.0.3) the makespan is at most $1 + \beta \leq 1 + 3\gamma + 0.25 \leq 1.75 + \gamma$ (this follows from the fact that $\beta/3 - 1/12 \leq \gamma \leq 1/4$).

Case 2: In the second case, an edge was oriented to $u$ in the Local Step. In constrast to the proof of Theorem 4.1, it is no longer necessarily true to determine that $p_1 \leq 1/2$. This is because we cannot apply Observation 3.0.1 to the edge $e_1$ if it is a light hyperedge (if it is an edge the proof remains valid). If $e_1$ is a light hyperedge, then the best bound on $p_1$ is $p_1 \leq \beta$ (rather than $1/2$). Using the bound on the load on $u$ from the proof of Theorem 4.1, we bound the load on $u$ by:

$$1 + (1 - \alpha)q_1 + f_\alpha(p_1)p_1 + (1 - f_\alpha(p_1))p_2 \leq 1 + (1 - \alpha) + p_1 \leq 2 - \alpha + \beta = 1.5 + \beta - 2\gamma \leq 1.75 + \gamma.$$  

The equality follows from our choice of $\alpha$ to be $\alpha = 2\gamma + 0.5$. Moreover, the last inequality follows from the fact that $\beta/3 - 1/12 \leq \gamma$. This concludes bounding the makespan, and thus concludes the proof. ■
7.2 Graph Balancing with Unrelated Light Hyperedges and Unrelated Heavy Edges

The problem of $\text{GBU}(\beta)$ further generalizes the above $\text{GBUH}(\beta)$ 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).

We partition this section into three subsections:

1. The proof that without the assumption that the value of the optimal makespan (in a $\text{GBU}(\beta)$ instance) is at least 1, $\text{GBU}(\beta)$ is as hard as GAP. This result is formulated in Lemma 7.2.1.

2. An upper bound on the tradeoff between makespan and orientation costs, as specified in Theorem 1.5.

3. Integality gaps with respect to $LP_k$, as presented in Theorem 1.6.

7.2.1 Hardness Result Without The Assumption on The Optimal Value

The following lemma states that without the assumption that the value of the optimal makespan (in a $\text{GBU}(\beta)$ instance) is at least 1, $\text{GBU}(\beta)$ is as hard as GAP:

**Lemma 7.2.1.** For every $0 < \beta \leq 1$, if there is a $c$-approximation to $\text{GBU}(\beta)$, then there is a $c$-approximation to the Generalized Assignment Problem$^1$

**Proof of Lemma 7.2.1.** We give an approximation preserving reduction from GAP to $\text{GBU}(\beta)$ (the single criterion version of these problems). For simplicity of presentation, we use the notations of jobs and machines instead of edges and vertices. Let ALG be a $c$-approximation to $\text{GBU}(\beta)$.

Given an instance of GAP, denote the different (sorted) values of the processing times $p$ by $\infty = w_0 > w_1 > w_2 > \cdots > w_k$. We present in Algorithm 7.1 our approximation preserving reduction. As previously mentioned in the paper, without loss of generality, we scale all processing times such that $\max\{p_{i,j} | p_{i,j} \neq \infty\} = 1$.

We now show that Algorithm 7.1 is a $c$-approximation to the given GAP instance. Denote by $OPT$ the optimal makespan of the original GAP instance. We note that there exists some $\ell$ such that $w_{\ell} \leq OPT < w_{\ell-1}$. We prove that for this $\ell$, the $\ell$th

$^1$We remark that by using a similar reduction, it is possible to show approximation equivalence for the bicriteria versions. Formally, if there is an $(a, b)$-approximation to $\text{GBU}(\beta)$, there is an $(a, b)$-approximation to GAP. We give further details after the proof of this lemma.
Algorithm 7.1 Reduction(p)

1: for \( \ell = 1 \) upto \( k \) do
2:   If \( p_{i,j} > w_\ell \), set \( p_{i,j} \leftarrow \infty \).
3:   Add a new job \( j' \) and two new machines \( i_1 \) and \( i_2 \) such that: \( p_{i_1,j'} = w_\ell + \epsilon \), \( p_{i_2,j'} = \frac{1}{\beta} \cdot w_\ell \) and for every \( i \notin \{i_1, i_2\} \) set \( p_{i,j'} = \infty \).
4:   Run ALG on the new instance.
5:   Reset all changes made to \( p \) in this iteration.
6: end for
7: Return the best schedule among all iterations (without the job \( j' \) and machines \( i_1, i_2 \)).

iteration of Algorithm 7.1 produces a schedule with makespan at most \( c \cdot OPT + \epsilon \) where \( \epsilon > 0 \) is arbitrarily small.

First, note that the input is valid to ALG. This is because the only heavy job is \( j' \) (the new job added in step 3), and the rest of the input (the jobs of the GAP instance), are light hyperedges. Thus, step 4 of Algorithm 7.1 is valid. Second, denote by \( OPT' \) the optimal makespan of the instance created in iteration \( \ell \). We show that \( OPT' \leq OPT + \epsilon \), which concludes the proof. A feasible orientation can orient the new added job \( j' \) toward \( i_1 \), and the rest of the input can be oriented like in the optimal solution to the GAP instance (since \( OPT < w_{\ell-1} \)). The makespan of this orientation is at most \( OPT' \), and thus:

\[
OPT' \leq \max\{OPT, w_\ell + \epsilon\} \leq \max\{OPT, OPT + \epsilon\} = OPT + \epsilon
\]

which concludes the proof.

We remark that the bicriteria reduction from \( GBU(\beta) \) to GAP is similar to the one in Algorithm 7.1. Namely, by ignoring steps 1, 6 in Algorithm 7.1 (but not ignoring steps 2 to 5), and setting \( w_\ell \triangleq T \), (where \( T \) is the target makespan), the modified algorithm is the desired reduction. i.e., if \( ALG \) is an \((a,b)\)-approximation to \( GBU(\beta) \), then by modifying Algorithm 7.1 as described above, we have an \((a,b)\)-approximation to GAP as well.

7.2.2 An Upper Bound on The Tradeoff Between Makespan and Orientation Costs

Surprisingly, by adding a mild constraint on the value of the optimal makespan we can give an approximation better than 2 for \( GBU(\beta) \). The assumption is that \( OPT \geq 1 \), and we assume it from this point onwards. Notice that \( OPT \geq 1 \) holds naturally in GB, and \( GBUH(\beta) \). Therefore, \( GBU(\beta) \) with the assumption that the value of the optimal makespan is at least 1, captures these problems. We extend our previous results to \( GBU(\beta) \) and achieve the same approximation factors for \( \beta \geq \sqrt{2} - 1 \). To be precise, in the case that \( \beta \geq \sqrt{2} - 1 \), we get the same result as in Theorem 1.4. This
result is summarized in Theorem 1.5.\footnote{We remark that makespan approximation of less than 2 is still possible with this technique for \( \beta < \sqrt{2} - 1 \). The required threshold function can be \( f(p) = 1 - \frac{1}{p} \) for \( p > \frac{1}{\epsilon} \) and 1 otherwise, for sufficiently big \( x \) (depends on \( \beta \)).}

We use Algorithm 3.1 to prove Theorem 1.5. Recall that given a target makespan \( T \) the algorithm scales all the processing times by \( T \). Similarly to GB, if \( T < 1 \), the algorithm outputs that there is no orientation with makespan at most \( T \). This is true since \( \text{OPT} \geq 1 \). Hence, similarly to GB we know that \( p_{e,u} \leq 1 \) for every \( e \in E \) and \( u \in e \).

The only difference in the algorithm with respect to GB is that if \( p_{e,u} \leq 0.75 + \gamma \) for every \( e \in E \) and \( u \in e \), we use a rounding function \( f \equiv 1 \). This will result in a \((1.75 + \gamma, 1)\)-approximation as proved in [ST93] (or Lemma 3.0.3). Therefore, we assume the processing times are scaled by a multiplicative factor of at least \( 0.75 + \gamma \), i.e. the target makespan \( T \) satisfies \( T \geq \frac{1}{0.75 + \gamma} \).

We choose the following threshold function to prove Theorem 1.5:

\[
f_\alpha(p_{i,j}) = \begin{cases} 
1 & \text{if } p_{i,j} \leq \frac{1}{3} \\
\alpha & \text{if } p_{i,j} > \frac{1}{3}
\end{cases}
\] (7.1)

where \( \max \{\frac{1}{3} + (2\beta)/3, \frac{2}{3}\} \leq \alpha \leq 1 \).

First we upper bound the assignment cost of Algorithm 3.1 with the above threshold function 7.1.

**Corollary 7.1.** For the specific choice of a threshold function \( f_\alpha \) as in 7.1, the output of Algorithm 3.1 has an assignment cost of at most \( \frac{1}{\alpha} \cdot C(T) \).

**Proof.** The proof follows immediately from Lemma 3.0.5, which still holds for GBU(\( \beta \)).\qed

Now we focus on bounding the makespan. We prove that the makespan is at most \( 1.5 + 0.5\alpha \), and choosing \( \alpha = 2\gamma + 0.5 \) gives the desired result.

Fix a machine \( i \). We use the same notations and definition as in the proof of Theorem 4.2. Moreover, notice that observations 4.1.2, 4.1.3, and 4.1.4, are valid in this case as well. However, note that Observation 3.0.1 does not trivially hold, even for heavy edges. This is due to the fact that the heavy edges are unrelated. We now formulate the exact observation (similar to Observation 3.0.1) that holds for GBU(\( \beta \)).

**Observation 7.2.2.** Let \( j \) be a heavy job that isn’t assigned in the Local Step of Algorithm 3.1. Additionally, let \( i \) be a machine such that \( j \) could be assigned to \( i \). Then, for our choice of a rounding function \( f_\alpha \) it holds that \( 1 - \alpha = 1 - f_\alpha(p_{i,j}) \leq x_{i,j} \leq f_\alpha(p_{i,j}) = \alpha \).

**Proof.** Let \( j \) be a heavy job that isn’t assigned in the Local Step, and \( i \) a machine such that \( j \) could be assigned to \( i \). The definition of a heavy job implies that before
scaling the processing times (in the binary search for the optimal makespan): \( p_{i,j} > \beta \).
Moreover, since we scale by a factor of at least \( 0.75 + \gamma \), we know that after scaling the processing times satisfy \( p_{i,j} > \beta(0.75 + \gamma) \). Since \( \gamma \geq \frac{\beta}{3} - \frac{1}{12} \), we can deduce:

\[
\begin{align*}
p_{i,j} &> \beta \left( 0.75 + \frac{1}{3} \beta - \frac{1}{12} \right) \\
&= \beta \left( \frac{2}{3} + \frac{1}{3} \beta \right) \\
&\geq 0.5 \left( \frac{2}{3} + \frac{1}{3} \cdot 0.5 \right) \\
&\geq 0.3.
\end{align*}
\]

Therefore, if \( x_{i,j} > \alpha \) according to Algorithm 3.1 and the definition of \( f_\alpha \) (see 7.1), \( j \) should be assigned to \( i \) in the Local Step, which is a contradiction. Moreover, if \( x_{i,j} < 1 - \alpha \), then \( x_{i',j} > \alpha \) where \( i' \) is the only machine different from \( i \), that \( j \) could be assigned to. Since \( p_{i',j} > 1/3 \) as well, \( j \) should be assigned to \( i' \) in the Local Step, again a contradiction.

Now we continue with bounding the makespan. Fix a machine \( i \). We show that the load on \( i \) is at most \( 1.5 + 0.5\alpha \). We use the same definitions and notations as in the previous sections. Recall that \( j_1, \ldots, j_k \) are the jobs on top of the slots of machine \( i \), and \( j'_1, j'_2 \) are the jobs assigned to machine \( i \) in the Local Step. For simplicity of notation we use \( p_\ell \) and \( q_\ell \) do denote \( p_{i,j_\ell} \) and \( p_{i,j'_\ell} \) respectively. We consider two cases, whether \( j_1 \) is a heavy job or not.

**Case 1:** In this case \( j_1 \) is a heavy job. Therefore, from Lemma 3.0.3 and Observation 7.2.2 the load on \( i \) is at most:

\[
1 + (1 - f_\alpha(q_1))q_1 + (1 - f_\alpha(q_2))q_2 + f_\alpha(p_1)p_1 + (1 - f_\alpha(p_1))p_2 \\
= 1 + (1 - \alpha)(q_1 + q_2) + \alpha p_1 + (1 - \alpha)p_2 \\
= 1 + (1 - \alpha)(q_1 + p_1) + (1 - \alpha)(q_2 + p_1) + (3\alpha - 2)p_1 + (1 - \alpha)p_{j_2} \\
\leq 1 + (1 - \alpha) + (1 - \alpha) + (3\alpha - 2) + (1 - \alpha) \cdot 0.5 \\
= 1.5 + 0.5\alpha.
\]

The last inequality follows from Observations 4.1.4 and 3.0.4, and the fact that \((3\alpha - 2)p_1\) is maximized when \( p_1 = 1 \) (this is true since \( \alpha \geq \frac{2}{3} \)).

**Case 2:** In this case \( j_1 \) is a light job. Therefore, \( p_{j_1} \leq \beta \).\(^3\) From Lemma 3.0.3, the

\[\text{Since we assume that the optimal makespan is at least 1, when scaling the processing times they do not increase.}\]

\[\text{38}\]
load on $i$ is at most:

$$1 + (1 - f_\alpha(q_1))q_1 + (1 - f_\alpha(q_2))q_2 + f_\alpha(p_1)p_1 + (1 - f_\alpha(p_1))p_2$$

$$\leq 1 + (1 - \alpha)(q_1 + q_2) + p_1$$

$$\leq 1 + (1 - \alpha) + \beta$$

$$= 2 - \alpha + \beta$$

$$\leq 1.5 + 0.5\alpha.$$

The first inequality follows from the fact that $p_2 \leq p_1$. Additionally, the second inequality follows from Observation 4.1.4. Lastly, the third inequality follows from the fact that $\alpha \geq 1/3 + 2/3\beta$.

From the above we can deduce the following corollary:

**Corollary 7.2.** The makespan of the assignment produced by Algorithm 3.1 with a threshold function $f_\alpha$ as in 7.1) is at most $1.5 + 0.5\alpha$.

Now we are ready to prove Theorem 1.5.

**Proof of Theorem 1.5.** The proof follows from Corollaries 7.1 and 7.2, and choosing $\alpha = 2\gamma + 0.5$. ■

### 7.2.3 Integrality Gaps

We now present a matching lower bound for Theorem 1.5. Specifically, we prove Theorem 1.6 by providing two integrality gaps to $LP_k$. Intuitively, the first instance shows that using $LP_k$ it is impossible to achieve an approximation less than $11/6$ with respect to the makespan. Intuitively, the second instance shows that using $LP_k$ an approximation of $1.75 + \gamma$ with respect to the makespan must incur a loss of at least $1/(2\gamma + 0.5) \cdot OPT_{LP_k}$ with respect to the orientation cost. This is formulated in the following two lemmas.

**Lemma 7.2.3.** There exists an instance of $GBUH(0.5)$ that is feasible to $LP_k$, such that every integral solution has makespan of at least $11/6 - \epsilon$, for every $\epsilon > 0$.

**Lemma 7.2.4.** There exists an instance of $GBUH(0.5)$ that is feasible to $LP_k$, such that every integral solution with makespan of at most $1.75 + \gamma$ has an assignment cost of at least $(1-\epsilon)/(2\gamma+0.5) \cdot OPT_{LP_k}$, for every $\epsilon > 0$ and $1/12 \leq \gamma < 1/4$.

The proof of Theorem 1.6 follows immediately from Lemmas 7.2.3 and 7.2.4. We now proceed to the proofs of these lemmas.

**Proof of Lemma 7.2.3.** Let $\epsilon > 0$. The instance consist of $1/\epsilon$ identical copies of a path of $1/\epsilon$ vertices. We describe the $i$th path, as it is identical to the others. We denote by $u_i$ the leftmost vertex in the $i$th path. All the edges of the path are heavy. The edge incident to $u_i$ has a weight of $1 - 6\epsilon$, and the rest of the edges have a processing time
Figure 7.1: A single path from the makespan integrality gap instance for GBUH(0.5). There is a dedicated load value of $\frac{1}{3}$ on each vertex. All vertices excluding $u_i$ share a light hyperedge of weight 0.5. Above the edges are the weight of the edge depending on the orientation, where the plus sign denotes $1 - 6\epsilon$, and the minus sign denotes $0.5 + \epsilon$.

of $0.5 + \epsilon$ to the left endpoint and a processing time of $1 - 6\epsilon$ to the right endpoint. Additionally, each vertex has a load value (self loop) of $\frac{1}{3}$. In fact, the load value is split into several self loops in the input, and the exact number will be determined later in the proof.

All the vertices in the $i$th path, except $u_i$, share a light hyperedge of weight 0.5, denoted by $e_i$. This concludes the description of a single path. In addition, all the vertices $u_i$ over all paths, share a light hyperedge of weight 0.5, denoted by $e'$. A single path from the instance is shown in Figure 7.1.

Now we show that this instance is feasible to $LP_k$. Consider the following fractional solution, denoted by $x$: each edge in each path is oriented $\frac{2}{3}$ toward the left endpoint (and thus $\frac{1}{3}$ toward the right endpoint), and each hyperedge is uniformly oriented between all the vertices it shares (i.e. each hyperedge is oriented a fraction of $\epsilon$ to each of the $\frac{1}{\epsilon}$ vertices it shares). Note that $x$ is a feasible solution to $LP_k$. First, the edge constraints trivially hold from the definition of $x$. Now we show the Load constraints hold for each vertex. Since the fractional solution is symmetric over all the paths, we consider a single path $i$. The fractional load on $u_i$ is:

$$\frac{1}{3} + \frac{2}{3} \cdot (1 - 6\epsilon) + \frac{1}{\epsilon} \cdot \frac{1}{2} = 1 - 3.5\epsilon.$$  

Additionally, the fractional load on the rest of the vertices in the path is at most:

$$\frac{1}{3} + \frac{2}{3} \cdot (0.5 + \epsilon) + \frac{1}{3} \cdot (1 - 6\epsilon) + \frac{1}{\epsilon} \cdot \frac{1}{2} = 1 - \frac{5\epsilon}{6}.$$  

Thus, the Load constraints hold for all vertices. The Star constraints also trivially hold from the definition of $x$. Lastly we show the Set constraints hold. We split the load value to $\frac{k}{\epsilon}$ self loops each with weight $\epsilon/(3k)$. Notice that this is sufficient so that Set constraints with respect to sets of size at most $k$ hold. This is true since for every vertex $v$ and $S \subseteq \delta(v)$, such that $\sum_{e \in S} p_{e,v} > 1$, $S$ must contain at least two edges that are not self loops. Hence, from the definition of $x$ we can deduce that $\sum_{e \in S} x_{e,v} \leq |S| - 1$.

Finally, we show that every integral solution to this instance has makespan of at least $\frac{11}{6} - 6\epsilon$ (and since we can replace $\epsilon$ with $\epsilon/6$ we conclude the proof). If there exists a vertex such that two path edges are oriented toward it, i.e., a collision, then the load
Figure 7.2: The instance of the integrality gap in the proof of Lemma 7.2.4. Each vertex has a load value of $\alpha + 0.25 - (4\alpha' + 0.5)\epsilon'$, and they share a single hyperedge of weight 0.5. The plus and minus signs of the edges represent the weight with respect to each endpoint. The plus sign represents a weight of $1 - \epsilon'$ and the minus sign represents a weight of $0.5 + \epsilon'$. Orienting $e'$ clockwise, incurs a cost of 1.

on that vertex is at least $1/6 - 5\epsilon$. Otherwise in each path, all the edges are oriented either to the right or the left and no collision is formed. First, we examine the case that there exists a path, such that all of the edges are oriented to the right. Without loss of generality assume this is the $i$th path. In this case, if we denote by $v$ the vertex that $e_i$ is oriented to, the load of $v$ is $1/3 + 1 - 6\epsilon + 0.5 = 11/6 - 6\epsilon$. Otherwise, all the edges of all the paths are oriented toward the left. In particular, for every path $i$ the vertex the edge incident to $v_i$ in the path is oriented toward $v_i$. Thus, there exists a vertex $v_i$ such that the hyperedge $e'$ is oriented toward it. Therefore, the load on this $v_i$ is $1/3 + 1 - 6\epsilon + 0.5 = 11/6 - 6\epsilon$.

Hence, we can conclude that $x$ is a feasible solution to $LP_k$, and every integral solution has makespan of at least $11/6 - 6\epsilon$. By replacing $\epsilon$ with $\epsilon/6$ in this proof, we finish the proof.

**Proof of Lemma 7.2.4.** Let $\epsilon > 0$ and $1/12 \leq \gamma < 1/4$. First, define $\epsilon' = \epsilon/4$ and $\gamma' = \gamma/(1-4\epsilon')$. We now present an instance of GBUH(0.5) that is feasible to $LP_k$ and every integral solution with makespan at most $1.75 + \gamma$ has an orientation cost of at least $(1-\epsilon)/(2\gamma+0.5) \cdot OPT_{LP_k}$. The instance is a cycle of $1/\epsilon'$ vertices that share a hyperedge of weight 0.5. The edges in the cycle are heavy and have a processing time of $1 - \epsilon'$ when oriented counter-clockwise and $0.5 + \epsilon'$ when oriented clockwise. Additionally, each vertex has a load value of $\gamma' + 0.25 - (4\gamma' + 0.5)\epsilon'$. Lastly, one edge in the cycle, denoted by $e'$, has orientation cost of 1 when oriented clockwise. The rest of the orientation costs are 0. The instance is shown in Figure 7.2.

Now we show this instance is feasible to $LP_k$. In fact there is exactly one feasible solution, which we denote by $x$. This fractional solution is described as follows: each edge in the cycle is oriented $2\gamma' + 0.5$ clockwise (and thus $0.5 - 2\gamma'$ counter-clockwise). Moreover, the hyperedge is uniformly oriented to all the vertices in the cycle, i.e., it is

41
fractionally oriented $\epsilon'$ to each of the $1/\epsilon'$ vertices it shares.

Now, we show $x$ is feasible to $LP_k$. The Edge and Star constraints trivially hold from the definition of $x$. Since the solution $x$ is symmetric to all the vertices, we show the Load and Set constraints are valid for an arbitrary vertex $u$. The fractional load on $u$ is:

$$\gamma' + 0.25 - (4\gamma' + 0.5)\epsilon' + (2\gamma' + 0.5)(0.5 + \epsilon') + (0.5 - 2\gamma')(1 - \epsilon') + 1/\epsilon' \cdot 0.5 = 1.$$ 

Thus, the Load constraints hold. If we split the load value to $k/\epsilon'$ self loops, then similarly to the proof of Lemma 7.2.3, a set $S \subseteq \delta(u)$ such that $\sum_{e \in S} p_{r,e,u} > 1$ and $|S| \leq k$, must contain at least two edges that are not self loops. Thus, from the definition of $x$ the Set constraints hold, and therefore $x$ is feasible.

Now we show each integral solution with makespan at most $1.75 + \gamma'$ has orientation cost of at least $(1 - 4\epsilon')/(2\gamma' + 0.5) \cdot OPT_{LP_k}$. First, observe that the orientation cost of $x$ is $2\gamma' + 0.5$, and since $x$ is the only feasible solution to the relaxation then $OPT_{LP_k} = 2\gamma' + 0.5$. Any integral solution such that two cycle edges are oriented to the same vertex (collision) has makespan of at least:

$$(\gamma' + 0.25 - (4\gamma' + 0.5)\epsilon') + 1 - \epsilon' + 0.5 + \epsilon' = 1.75 + \gamma' - (4\gamma' + 0.5)\epsilon'$$

$$= 1.75 + (1 - 4\epsilon')\gamma' + 0.5\epsilon'$$

$$= 1.75 + \gamma + \epsilon'/8.$$ 

If there are no collisions, all the edges in the cycle are either oriented clockwise or counter-clockwise. If the edges are oriented clockwise, then in particular, $\epsilon'$ is oriented clockwise. Thus, the orientation cost of this solution is 1. Otherwise, all the cycle edges are oriented counter-clockwise. Let $u$ be the vertex such that the light hyperedge is oriented to it. Therefore, the load on $u$, if all the edges are oriented counter-clockwise, is:

$$(\gamma' + 0.25 - (4\gamma' + 0.5)\epsilon') + 1 - \epsilon' + 0.5 = 1.75 + \gamma' - (4\gamma' + 1.5)\epsilon'$$

$$= 1.75 + (1 - 4\epsilon')\gamma' + 1.5\epsilon'$$

$$= 1.75 + \gamma + 3\epsilon'/8.$$ 

Thus, every integral solution with makespan at most $1.75 + \gamma$ has a cost of 1. We
conclude the proof since:

\[
\frac{1 - \epsilon}{2\gamma + 0.5} \cdot \text{OPT}_{LP_k} = \frac{1 - \epsilon}{2\gamma + 0.5} \cdot (2\gamma' + 0.5) \\
= \frac{1 - \epsilon}{2\gamma + 0.5} \cdot \left( \frac{2\gamma}{1 - 4\epsilon'} + 0.5 \right) \\
\leq \frac{1 - \epsilon}{2\gamma + 0.5} \cdot \frac{2\gamma + 0.5}{1 - 4\epsilon'} \\
= 1.
\]

Proof of Theorem 1.6. Follows immediately from Lemmas 7.2.3, 7.2.4. ■

7.3 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 [VW14, EKS14], 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.

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 1.7. Note that SRGB\((c)\) captures GB when \(c = 1\), and indeed in Theorem 1.7 we achieve a \((11/6, 3/2)\)-approximation for SRGB\((c)\) when \(c = 1\) (similarly to Theorem 4.1). Moreover, when \(c = \infty\) Theorem 1.7 achieves a \((2, 1)\)-approximation for SRGB\((c)\), matching the integrality gap of [VW14, EKS14]. Finally, we also show that in general Theorem 1.7 provides a \((2 - \Omega(1/c), 1 + O(1/c))\)-approximation for SRGB\((c)\). Figure 7.3 shows the makespan approximation obtained in Theorem 1.7 as a function of \(c\).
In order to prove Theorem 1.7 we use Algorithm 3.1 and $LP_k$ (replacing $p_e$ with $p_{e,u}$) with a suitable choice of a threshold function $f$. The threshold function we use is the following:

$$f_{a,b}(p_e) = \begin{cases} 
  a & \text{if } p_e > b \\
  1 & \text{if } p_e \leq b 
\end{cases} \quad (7.2)$$

where parameters $a$ and $b$ will be determined later.

We now focus on the proof of Theorem 1.7. First, we present the following lemma that bounds the makespan of the orientation produced by Algorithm 3.1 with threshold function $f_{a,b}$:

**Lemma 7.3.1.** The makespan of the orientation produced by Algorithm 3.1 with $f_{a,b}$ is at most:

$$\max \left\{ \frac{1}{a} + c \cdot b, \ 1.5 + 0.5a, \ 2 - \left( 2 - \frac{1}{a} \right) b \right\}$$

Using Lemma 7.3.1 (which we prove later on in this section), we can finish the proof of Theorem 1.7:

**Proof of Theorem 1.7.** First, we choose the values of $a$ and $b$ (as in the rounding function $f_{a,b}$) such that the terms in the maximum expression of Lemma 7.3.1 are balanced so as to minimize the makespan. Thus, we choose $a$ and $b$ such that:

$$\frac{1}{a} + c \cdot b = 1.5 + 0.5a = 2 - \left( 2 - \frac{1}{a} \right) b.$$

This is equivalent to finding an $a$ such that:

$$\left( \frac{1}{c} + \frac{1}{2} \right) a^3 + \left( \frac{5}{2c} - \frac{1}{2} \right) a^2 - \frac{7}{2c} a + \frac{1}{c} = 0.$$
We show there is a unique \( a \) in the range \([0.5, 1 - 1/4c]\), that satisfies the above equation. This fact, along with Lemma 3.0.5 (which bounds the orientation cost of the algorithm) finishes the proof.

Thus, we show that the polynomial:

\[
g(a) = \left( \frac{1}{c} + \frac{1}{2} \right) a^3 + \left( \frac{5}{2c} - \frac{1}{2} \right) a^2 - \frac{7}{2c} a + \frac{1}{c}
\]

has a unique root in \([0.5, 1 - 1/4c]\). It can be proved that \( g(0.5) = -1/16 \) and \( g(1 - 1/4c) > 0 \), for every \( c \geq 1 \). Thus, from the Intermediate Value Theorem, we can deduce there is a root of the polynomial \( g(a) \) in the range \([0.5, 1 - 1/4c]\). Furthermore, It can be shown using simple calculus that \( g \) has a unique root in \([0.5, 1]\) (by showing \( g \) has no local maximum in this range). Hence, we can deduce that there is a unique root for \( g(a) \) in the range \([0.5, 1 - 1/4c]\).

Now we prove Lemma 7.3.1 which bounds the makespan of the orientation produced by Algorithm 3.1 with threshold function \( f_{a,b} \).

**Proof of Lemma 7.3.1.** Fix a vertex \( u \in V \). We use definitions similar to the previous sections: denote by \( e_1, \ldots, e_k \) the edges on top of \( \text{slot}(u,1), \ldots, \text{slot}(u,k) \). Additionally, we denote by \( p_1, \ldots, p_k \) the weights (with respect to the vertex \( u \)) of those edges respectively.

Now, we are required to introduce an observation similar to Observation 3.0.1. The reason Observation 3.0.1 does not hold in this case, is due to the fact that an edge does not have the same weight on the two vertices it shares. However, since the weights cannot differ much, we observe the following fact.

**Observation 7.3.2.** Let \( e = (u,v) \in E \) such that \( p_{e,u} > c \cdot b \) and \( e \) is not oriented in the Local Step of Algorithm 3.1. Then, \( x_{e,u} \geq 1 - a \).

**Proof.** From the problem definition, we know that \( p_{e,u} \leq c \cdot p_{e,v} \). Since \( p_{e,u} > c \cdot b \) we can deduce that \( p_{e,v} > b \). In addition, since \( e \) is not oriented in step 1 of Algorithm 3.1, then \( x_{e,v} \leq a \). From the Edge constraint of \( e \), we know that \( x_{e,u} + x_{e,v} = 1 \). Therefore, we conclude that \( x_{e,u} \geq 1 - a \). ■

Before proceeding to the proof of Lemma 7.3.1, we denote by \( E_{\text{orient}} \) the set of edges oriented toward \( u \) in the Local Step. In addition, we denote by \( X \) the fractional load of edges oriented toward \( u \) in the Local Step:

\[
X \triangleq \sum_{e \in E_{\text{orient}}} x_{e,u} p_{e,u}.
\]

The next observation bounds the load from edges oriented toward \( u \) in the Local Step.

**Observation 7.3.3.** The load from edges oriented toward \( u \) in the Local Step of Algorithm 3.1 is at most \( 1/a \cdot X \).
Proof. The load on $u$ from edges oriented in the Local Step is:

$$
\sum_{e \in E_{\text{orient}}} p_{e,u} \leq \sum_{e \in E_{\text{orient}}} \frac{1}{a} \cdot x_{e,u} p_{e,u} = \frac{1}{a} X.
$$

The first inequality holds since for every $e \in E_{\text{orient}}$ it must be the case that $x_{e,u} > a$. ■

We consider three cases, depending on the values of $p_1$ and $X$.

Case 1: $p_1 \leq c \cdot b$. We can use the result by [ST93] (or Lemma 3.0.3), in order to bound the load from edges oriented to $u$ in the Global Step of Algorithm 3.1 by:

$$
\sum_{i=1}^{k} p_i \leq 1 - X + p_1.
$$

Note that this is true since the total fractional load on $u$ in the beginning of the Global Step of Algorithm 3.1 is $1 - X$. From Observation 7.3.3 and the previous inequality we can bound the total load on $u$ as follows:

$$
\frac{1}{a} X + (1 - X) + p_1 \leq 1 + \left(\frac{1}{a} - 1\right) X + c \cdot b
$$

$$
\leq 1 + \left(\frac{1}{a} - 1\right) + c \cdot b
$$

$$
= \frac{1}{a} + c \cdot b.
$$

Case 2: $p_1 > c \cdot b$ and $X = 0$. Note that since $X = 0$, no edge was oriented toward to $u$ in the Local Step. From Observation 7.3.2 we derive that $x_{e_1,u} \geq 1 - a$. Therefore, using Lemma 3.0.3 we obtain the following bound on the load on $u$:

$$
1 + (1 - a)p_1 + a \cdot p_2.
$$

Since no edges were oriented to $u$ in the Local Step, we can conclude that the load on $u$ is at most:

$$
1 + (1 - a)p_1 + a \cdot p_2 \leq 1 + 1 - a + a \cdot 0.5
$$

$$
= 1.5 + 0.5a.
$$

Note that $p_2 \leq 1/2$ from Observation 3.0.4.

Case 3: $p_1 > c \cdot b$ and $X > 0$. We start with an observation that follows from the Set constraints. The observation gives an upper bound to $p_1$.

Observation 7.3.4. $p_1 \leq 1 - b$.

Proof. Due to the case condition $X > 0$, there is an edge $e'$ that was oriented toward $u$ in the Local Step. From the definition of the rounding function 7.2 it holds that
$x'_{e,u} > a$. Moreover, by Observation 7.3.2 we obtain that $x_{e_1,u} \geq 1 - a$. Thus, we can deduce: $x_{e_1,u} + x'_{e,u} > 1$.

From the Set constraint of $S = \{e', e_1\}$ we derive that $p_1 + p'_{e,u} \leq 1$ (using Observation 4.1.2). Moreover, from the definition of the threshold function 7.2 it holds that $p'_{e,u} > b$. Therefore, we can conclude that $p_1 \leq 1 - p'_{e,u} < 1 - b$. ■

The next observation, gives an upper bound on $X$ that follows from the Load constraint on $u$.

**Observation 7.3.5.** $X \leq 1 - (1 - a) \cdot p_1 - (1 - a)p_2$.

**Proof.** From the Load constraint on $u$ we deduce:

$$1 \geq \sum_{e \in \delta(u)} x_{e,u}p_{e,u} \geq X + \sum_{e \in \text{slot}(u,1)} x_{e,u}p_{e,u} \geq X + (1 - a)p_1 + ap_2.$$ 

The last inequality follows from Observation 7.3.2, and the fact that the capacity of $\text{slot}(u,1)$ is exactly 1 (recall that without loss of generality we can add edges with weight zero in order to fill the slot if needed). From the previous inequality we conclude:

$$X \leq 1 - (1 - a)p_1 - a \cdot p_2.$$ 

Now we bound the load on $u$. We use the same bound on the load of the edges oriented toward $u$ in the **Global Step**, as in the previous case. In addition, we bound the load from edges oriented toward $u$ in the **Local Step** by using Observation 7.3.3. Thus, the load on $u$ is at most:

$$\frac{1}{a}X + (1 - X) + a \cdot p_1 + (1 - a)p_2 = 1 + \left(\frac{1}{a} - 1\right)X + a \cdot p_1 + (1 - a)p_2$$ 

$$\leq 1 + \left(\frac{1}{a} - 1\right)(1 - (1 - a)p_1 - ap_2) + a \cdot p_1 + (1 - a)p_2$$ 

$$= \frac{1}{a} + \left(2 - \frac{1}{a}\right)p_1$$ 

$$\leq \frac{1}{a} + \left(2 - \frac{1}{a}\right)(1 - b)$$ 

$$= 2 - \left(2 - \frac{1}{a}\right)b.$$ 

The first and second inequalities follow from Observations 7.3.5 and 7.3.4 respectively. Note that in each case the load on $u$ is at most: $\max \left\{\frac{1}{a} + c \cdot b, \ 1.5 + 0.5a, \ 2 - (2 - \frac{1}{a})b\right\}$, which concludes the proof. ■
Chapter 8

Conclusion and Open Questions

In this work we presented the bicriteria optimization problem of Graph Balancing with Orientation Costs, and some extension of it. Using a simple framework inspired by the algorithm of [ST93], we proved that we can extend the tradeoff of $(2, 1)$-approximation by [ST93] for Graph Balancing. Moreover, we proved that using our linear relaxation, achieving an approximation less than 2 with respect to the makespan, incurs a loss in the orientation costs. Using our framework, we also achieve upper bounds on the tradeoff between makespan and orientation costs for extensions of Graph Balancing. These extensions allow a combination of hyperedges and unrelated weights (with some restrictions), and some of them were previously studied [HO16, EKS14, VW14].

8.1 Open Questions

A major open question related to is whether or not there exists an $(2 - \epsilon)$-approximation with respect to the makespan to the Generalized Assignment Problem, where $\epsilon > 0$ is a constant. Even the powerful Configuration LP, that has integrality gap strictly less than 2 for some special cases of GAP ([Sve11]), has an integrality gap of 2 for GAP ([EKS14, VW14]).

Another related open problem is approximating Graph Balancing with approximation ratio less than 1.75 with respect to the makespan. The relaxation used by [EKS08] has an integrality gap of exactly 1.75. Moreover, the Configuration LP has an integrality gap $1.75 - \epsilon$ for a constant $\epsilon > 0$ [JR18]. However no polynomial time algorithm is known to round its fractional solution in order to obtain an approximation ratio less than 1.75. It might be possible that the relaxation that we presented in this work (see Section 4.1) could be useful in solving this problem (since the integrality presented by [EKS08] do not apply for this relaxation).

Finally, improving the upper bounds on the bicriteria tradeoffs we presented for GB, or extending their range for smaller makespan approximations, is also an interesting open question. Similarly, improving the lower bounds on this tradeoff or presenting an bicriteria-approximation hardness based on some complexity assumption (such as
$P \neq NP$) would be an interesting result.
Bibliography


Using the binomial coefficient method [2, 1], we prove that the problem of balancing the two sets of vertices is equivalent to the problem of finding an equilibrium in a graph similar to the approximation. For the problem of finding an equilibrium in graphs, we show that the equilibrium with high cost can be obtained by using an algorithm with a constant factor of multipliers. Consequently, we cannot achieve a result like the one presented, and we note that every algorithm in the time complexity of the problem of finding an equilibrium in graphs, does not use the approximation presented.

In order to find the equilibrium in graphs, we use the algorithm presented, and we present a two-dimensional approximation for the problem of finding an equilibrium in graphs. We note that the approximation in the above theorem is true when the algorithm is used, or when the multipliers change, but the algorithm presented in the theorem is not unique. Moreover, we note that in order to find the result of the theorem presented, we use a two-dimensional algorithm, and we present two-dimensional algorithms for the problem of finding an equilibrium in graphs.

In order to find the equilibrium in graphs, we use the algorithm presented, and we present two-dimensional approximations for the problem of finding an equilibrium in graphs. We note that the approximation in the above theorem is true when the algorithm is used, or when the multipliers change, but the algorithm presented in the theorem is not unique. Moreover, we note that in order to find the result of the theorem presented, we use a two-dimensional algorithm, and we present two-dimensional algorithms for the problem of finding an equilibrium in graphs.

For the problem of finding an equilibrium in graphs, we show that the equilibrium with high cost can be obtained by using an algorithm with a constant factor of multipliers. Consequently, we cannot achieve a result like the one presented, and we note that every algorithm in the time complexity of the problem of finding an equilibrium in graphs, does not use the approximation presented.
Let \( C(T) \) denote the optimal runtime for a given problem. If we consider two parameters, \( \alpha, \beta \), and their trade-off, we can analyze the relationship between two functions: \( \alpha T \) and \( \beta C(T) \).

Noting (1.75, \( \infty \)) – the optimal time for \( \alpha T \) and \( \beta C(T) \), we can see that the latter is more efficient. In this context, we can use the following theorem:

**Theorem.**

\[
\epsilon = \sqrt{\frac{33}{4} - \frac{17}{12}} \approx 0.02, \quad \frac{\epsilon}{2} \leq \gamma \leq \frac{1}{4}
\]

For \( \gamma \) close to 0, the optimal runtime becomes

\[
(1.75 + \gamma, 1 + \frac{1}{2\gamma} + 0.5)
\]

This theorem provides a bound for the optimal runtime when considering the trade-off between two parameters. The optimal runtime is given by

\[
(1.75 \gamma, 1 + \frac{1}{\gamma})
\]

The above theorem is a generalization of the previous work, and it provides a tighter bound for the optimal runtime when considering the trade-off between two parameters.
תקציר

خلاים

פאוץ

ופאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ

פאוץ
هجוז

ארכי בהוז בלהוז לעיון שוחרי על הניהון המסורתי והמשקה. זה היה מהנה לכלכלテーレד איתיס.

לפי הוז להוז לעיון שוחרי על הניהון המסורתי והמשקה. זה היה מהנה לכלכלテーレד איתיס.
בעיית איזון בגרפים על עליות כיוון

חובר על מחקר

לשם مليולי חלקי של הדרישות לקבלת התואר
מניסור למדעי ברודיינו המחשב

ור תחתאל

הוגש לסנט הפקון — מרכז טכנולוגיה לישראל
شبטי הח"ף והת"ש מבארא 2020
בעיית איזון בגרפים עם עלויות קיומי

ר גייזטל