Bi-Criteria Approximation Algorithms for Load Balancing on Unrelated Machines with Costs

Trung Thanh Nguyen[1]
ORLab, Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam
thanh.nguyentrung@phenikaa-uni.edu.vn

Jörg Rothe
Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, Germany
rothe@hhu.de

Abstract

We study a generalized version of the load balancing problem on unrelated machines with cost constraints: Given a set of m machines (of certain types) and a set of n jobs, each job j processed on machine i requires $p_{i,j}$ time units and incurs a cost $c_{i,j}$, and the goal is to find a schedule of jobs to machines, which is defined as an ordered partition of n jobs into m disjoint subsets, in such a way that some objective function of the vector of the completion times of the machines is optimized, subject to the constraint that the total costs by the schedule must be within a given budget B. Motivated by recent results from the literature, our focus is on the case when the number of machine types is a fixed constant and we develop a bi-criteria approximation scheme for the studied problem. Our result generalizes several known results for certain special cases, such as the case with identical machines, or the case with a constant number of machines with cost constraints. Building on the elegant technique recently proposed by Jansen and Maack [15], we construct a more general approach that can be used to derive approximation schemes to a wider class of load balancing problems with constraints.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases bi-criteria approximation algorithm, polynomial-time approximation algorithm, load balancing, machine scheduling

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.14

Funding *Jörg Rothe*: Supported in part by DFG under grants RO-1202/14-2 and RO-1202/21-1.

Acknowledgements We thank the anonymous ISAAC 2020 reviewers for helpful comments.

1 Introduction

In the load balancing problem (LBP), we are given a set of jobs, denoted by $[n] = \{1, \ldots, n\}$, and a set of unrelated parallel machines, $[m] = \{1, \ldots, m\}$. Each job j has *processing time* (or *size*) $p_{i,j}$ on machine i. An *assignment* (or *schedule*) of jobs to machines is defined as a partition of the set $[n]$ into m disjoint subsets, each corresponding to some machine. We assume that at any time, every machine processes no more than one job and each job is processed without interruption on one of the machines. The goal is to find a job assignment that optimizes a certain objective on the vector $(L_1, \ldots, L_m)$ of the completion times (or loads) of the machines. Motivated by the work of Alon et al. [1], Epstein and Sgall [10], and Epstein and Levin [9], we study the LBP with the following four major types of objectives:

- **(I)** *minimizing* the machine *maximum* load: $\max_{i=1}^{m} f(L_i)$ (*min-max* objective),
- **(II)** *maximizing* the machine *minimum* load: $\min_{i=1}^{m} f(L_i)$ (*max-min* objective),
- **(III)** *minimizing* $\sum_{i=1}^{m} f(L_i)$ (*min-sum* objective), and
- **(IV)** *maximizing* $\sum_{i=1}^{m} f(L_i)$ (*max-sum* objective),

[1] Corresponding author

where $f : \mathbb{R}_+ \to \mathbb{R}_+$ is some fixed function satisfying certain conditions, which will be specified later. In particular, we consider a more general version of LBP with cost constraint, denoted as LBC, in which the assignment of a job j to a machine i has, besides the processing time $p_{i,j}$, a cost $c_{i,j}$, and we want to find a schedule that optimizes one of the above objectives, while ensuring that the total cost derived from that schedule does not exceed a prescribed budget B. Practical applications of this problem can be found in various domains, such as in vehicle routing [11, 14], distribution systems [5], and facility location [22].

It is well-known that, unless P = NP, LBP (and thus LBC) cannot be solved in polynomial time for any of the objectives (I)–(IV), leading to a huge body of developments of approximation algorithms for the problem over the past decades. An *α-approximation algorithm* (for some $\alpha \in (0, 1]$) for a maximization problem is a polynomial-time algorithm that produces, for any given problem instance $\mathcal{I}$, a solution whose value is at least α times the optimum value. In particular, a *polynomial-time approximation scheme* (PTAS) is a family of $(1 - \epsilon)$-approximation algorithms for all $\epsilon > 0$. The running time of a PTAS is polynomial in the input size for every fixed ϵ, but the exponent of the polynomial may depend on $1/\epsilon$. An *efficient polynomial-time approximation scheme* (EPTAS) is a PTAS whose running time is $f(1/\epsilon) \cdot \mathsf{poly}(|\mathcal{I}|)$, where f is some computable function and $|\mathcal{I}|$ is the binary encoding length (or input size) of instance $\mathcal{I}$. An even stronger notion is a *fully polynomial-time approximation scheme* (FPTAS), whose time complexity is polynomial in both the input size and $1/\epsilon$. The notion of approximation algorithms for minimization problems can be defined similarly. Observe that, even without costs, LBC is strongly NP-hard even for identical machines. One could therefore consider to relax the cost constraint to be able to get better approximations. For $\alpha \in (0, 1]$ and $\beta \geq 1$, an (α, β) *(bi-criteria) approximation algorithm* for the maximization version of LBC gives a schedule A with objective value at least αV and with cost at most βC, where V and C, respectively, are the value and the cost of an optimal schedule. In case of minimization, we have $\alpha \geq 1$ and the objective value of A is required to be at most αV. A *bi-criteria polynomial-time approximation scheme* (bi-PTAS) is defined like a PTAS but based on bi-criteria approximation algorithms. We can give a formal definition of a bi-PTAS for the problem LBC as follows.

▶ **Definition 1** (bi-PTAS). *A bi-PTAS for the maximization version of* LBC *is a PTAS which is a $(1 - \epsilon)$-approximation in the objective, and a $(1 + \epsilon)$-approximation in the constraint. Similarly, a bi-PTAS for the minimization version of* LBC *is a PTAS which is a $(1 + \epsilon)$-approximation in the objective, and a $(1 + \epsilon)$-approximation in the constraint.*

1.1 Related Work

Most of the previous work on approximation algorithms for LBC has focused on the study of minimizing the min-max objective (a.k.a. the *makespan*), for the identity function f (i.e., $f(x) = x$ for all $x \in \mathbb{R}_+$). Lin and Vitter [20] proposed a $(2 + \frac{1}{\epsilon}, 1 + \epsilon)$-approximation algorithm for the problem LBC for any $\epsilon > 0$, and these factors were then improved to $(2, 1)$ by Shmoys and Tardos [24]. Regarding inapproximability, Lenstra et al. [18] ruled out the existence of an $(\alpha, 1)$-approximation, for any $\alpha > 3/2$. Better algorithms were known for the case when there is only a constant number of machines [16, 2, 8]. Jansen and Porkolab [16] presented a linear-programming-based $(1 + \epsilon, 1 + \epsilon)$-approximation algorithm with running time $O(n(m/\epsilon)^{O(m)})$ (recall that m is the number of machines and n the number of jobs). Angel et al. [2] followed a dynamic-programming-based approach and significantly improved this result by exhibiting a $(1 + \epsilon, 1)$-approximation algorithm, with a worse complexity than the previous one. Efraimidis and Spirakis [8] generalized the result of Jansen and Porkolab [16] to the case of two cost constraints.

We study LBC in the setting when the number of different types of machines is a fixed constant, where machines are of the same type if the processing time of every job is identical on these machines. This naturally generalizes previous special cases in the literature such as the case of identical machines and the case of a constant number of machines. Focusing on this setting, attempts have been devoted to designing approximation schemes for LBC without constraints, for different objectives (assuming f is the identity function). An overview of these results is given in Table 1. The best approximation results for the min-max and max-min objectives are EPTASes due to Jansen and Maack [15]. Kones and Levin [17] have significantly generalized these results to an objective that is a convex combination of min-max and min-sum. More interestingly, their results can be applied to a variety of well-known machine scheduling problems. However, a straightforward application of their technique does not seem to work for the case of max-min and max-sum objectives, with cost constraints. This motivates the study of an improved technique for designing approximation schemes for the LBC problem, with a general class of functions f.

Table 1 A brief overview of previously known and our novel results for LBC. The parameters k, n, m, and T are the number of constraints, jobs, machines, and machine types, respectively, and $\epsilon > 0$ is any constant less than 1. Our results are highlighted in bold letters; previous results are grey. Note that the EPTAS provided by Kones and Levin [17] is actually for minimizing both min-max and min-sum. Our boldfaced results as well as previous results marked with * hold for functions f fulfilling condition (†) (see Section 1.2).

Objectives	min-max	max-min	min-sum	max-sum
$k = 0$	2 [18]	$\tilde{O}(n^{-\epsilon})$ [7]	2 [3]	unknown
$k = 1$	$(2 + \frac{1}{\epsilon}, 1 + \frac{1}{\epsilon})$ [20] $(2 + \epsilon, 1)$ [24]	unknown	unknown	unknown
$T = 1$, $k = 0$	PTAS [13]	EPTAS [25]	EPTAS* [1]	EPTAS* [1]
$m = O(1)$, $k = 0$	FPTAS [14]	FPTAS [26]	FPTAS [4]	FPTAS [4]
$m = O(1)$, $k = 1$	bi-PTAS [16] PTAS [2]	bi-PTAS	bi-PTAS	bi-PTAS
$m = O(1)$, $k = 2$	bi-PTAS [8]	**bi-PTAS**	**bi-PTAS**	**bi-PTAS**
$T = O(1)$, $k = 0$	PTAS [6, 12] EPTAS [15]	EPTAS [15]	PTAS [6] EPTAS [17]	PTAS
$T = O(1)$, $k = O(1)$	**bi-PTAS**	**bi-PTAS**	**bi-PTAS**	**bi-PTAS**

1.2 Our Contribution

Suppose that the function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is computable and fulfills the following condition (†): For all $\epsilon > 0$, there exists a $\delta > 0$ (whose value depends only on ϵ) such that for all $x, y \geq 0$, $|y - x| \leq \delta x$ implies $|f(y) - f(x)| \leq \epsilon f(x)$. The function f is *convex* (respectively, *concave*) if and only if

$$f(x + \delta) + f(y - \delta) \leq f(x) + f(y) \quad \text{(respectively, } f(x + \delta) + f(y - \delta) \geq f(x) + f(y)\text{)}$$

holds for all x, y, δ with $0 \leq x \leq y$ and $0 \leq \delta \leq y - x$. Our contributions are as follows:

▶ **Theorem 2.** *If the number of machine types is fixed, then there is a bi-PTAS for* LBC *with max-min or max-sum objectives, for any computable concave function f satisfying condition* (†).

▶ **Theorem 3.** *If the number of machine types is fixed, then there is a bi-PTAS for* LBC *with min-max or min-sum objectives, for any computable convex function f satisfying condition* (†).

All of our bi-criteria PTASes are achieved based on a novel algorithmic framework which combines some existing ideas due to Alon et al. [1] and Jansen and Maack [15], and our framework is in fact strongly built on them. Their basic idea is as follows:

- First, round all the job sizes using the geometric rounding approach;
- second, formulate the rounded problem instance as a mixed integer linear program (MILP) with a small number of integral variables that can be efficiently solved;
- and, finally, round the obtained fractional solution to an integer solution, which is then converted into a near-optimal schedule to the original instance.

This idea has been successfully applied to the cases of max-min and min-max objectives with identity functions f, as shown by Jansen and Maack [15]. However, major changes are required to make their approach applicable not only to other objectives such as max-sum and min-sum but also to non-identity functions f. A crucial change is the way how to deal with both small jobs and huge jobs of sizes bigger than a threshold K – the guessed optimal value of the objective. Jansen and Maack [15] show, for every huge job, how one can round its size to K without changing the objective value. However, this argument is no longer valid when the objective function is measured as a sum of machine loads.

In our framework, huge jobs are considered together with other types of jobs in the MILP. In addition, we will find a way of replacing small jobs assigned to a machine by a number of *dummy jobs* all of the same reasonable size, which could help in defining configurations. Another major change is the method of rounding the fractional solution to the MILP. In our case the flow network utilizing flow integrality technique used by Jansen and Maack [15] does not help and we instead follow a more general linear-programming rounding combined with a partial enumeration technique.

Finally, we mention that this approach has also turned out to be useful in the different context of approximating Pareto sets for fair and efficient allocation of indivisible goods when there are only a few agent types or a few types of goods [21].

1.3 Organization

Our paper is structured as follows. First, in Section 2, we describe an algorithmic framework for designing bi-criteria approximation schemes for LBC with respect to the max-sum objective. In Section 3, we then show how to modify it to obtain similar results for other types of objectives. We conclude in Section 4 and discuss interesting open problems for future work.

2 Bi-Criteria Approximation Schemes for the Max-Sum Objective

Let us fix some notation to be used throughout this paper. For a positive integer z, we use $[z]$ to denote the set $\{1, 2, \ldots, z\}$. A schedule of jobs to machines is denoted by $A = (A_i)_{i \in [T]}$, where A_i is the set of machine i's jobs, and we denote by L_i the load of machine i, i.e., the total of the sizes of the jobs in A_i. We use $\phi_{\mathcal{I}}(A)$ to denote the objective value of schedule A for a problem instance $\mathcal{I}$. Let T be the number of different machine types and let m_t be the number of machines of type $t \in [T] = \{1, \ldots, T\}$. Two different machines i and i' are

of the same type if $p_{i,j} = p_{i',j}$ for all jobs $j \in [n]$. The size and the cost of a job j on a machine of type t is denoted by $p_{t,j}$ and $c_{t,j}$, respectively. Note that every job j has the same cost $c_{t,j}$ for all machines of type t. We first present an algorithmic framework for designing a bi-PTAS for LBC with respect to the max-sum objective, and then show how to modify it to obtain similar results for other types of objectives. Fix a constant $\epsilon \in (0, 1)$; without loss of generality, we may assume that $\epsilon < 1/2$. Assume that f is a nonnegative concave function satisfying condition (†), that is, one can choose δ (depending on ϵ) such that $|f(y) - f(x)| \leq \epsilon \cdot f(x)$ for every $x, y \geq 0, |y - x| \leq \delta x$. Let $\alpha = \min\{\frac{1}{9}, \frac{\delta}{6}\}$. Let $\mathcal{I}$ be a given problem instance and A^* be its optimal schedule. The following lemma, due to Alon et al. [1], exhibits the existence of another optimal schedule A of a *nice structure*, which is essentially the starting point of our framework.

▶ **Lemma 4** (Alon et al. [1]). *There is an optimal schedule A and there are nonnegative values $V_1, \ldots, V_T$ such that for every machine type $t \in [T]$, if a job of size at least V_t is assigned to some machine then it is the only job it receives; those machines which are not assigned such a job are assigned a subset of jobs of total sizes in $(\frac{1}{2}V_t, 2V_t)$.*

▶ Remark 5. Regarding the applicability of Lemma 4, note that Observations 2.1 and 2.2 provided by Alon et al. [1] do not aim at addressing cost constraints, which is part of our problem model LBC. However, since we assume that every job has the same cost for all machines of the same type, exchanging the jobs between these machines does not affect the total cost of the whole assignment.

The basic idea in the proof of Lemma 4 is to convert the schedule A^* into a new schedule A with the desired structure, by a suitable reassignment of jobs among machines of the same type. The existence of such an optimal assignment A paves a way for (approximately) solving $\mathcal{I}$ via a simple enumeration technique: First, guess approximate values of V_t for all t and, second, find a schedule of maximum value among those having a nice structure w.r.t. these guessed values. The guessing step can be done by enumerating all possible intervals $I_k = [(1 + \alpha)^k, (1 + \alpha)^{k+1})$, for nonnegative integers k, and in each interval $(1 + \alpha)^k$ can serve as an estimated value of V_t. Consequently, the resulting schedule may not be optimal, but its value should be within a small factor of $1 + \alpha$ of the optimum.

The next idea is to decompose our problem instance $\mathcal{I}$ into a bounded number of subproblems, denoted as $\textsc{Sub}(\mathcal{I}; W)$, each being associated with a vector $W = (W_t)_{t\in[T]}$ that can be seen as the approximate values of $V = (V_t)_{t\in[T]}$. The final solution to the original instance is the one of maximum value among all solutions to subproblems. Here, a *feasible* solution to the subproblem $\textsc{Sub}(\mathcal{I}; W)$ is defined as a solution that gives to each machine of type t only one job of size at least $(1 + \alpha)W_t$ (later on, we will call this a *huge* job) or a set of jobs of total size in $(\frac{1}{2}W_t, 2(1 + \alpha)W_t)$; and the total costs of all machines is at most B. Our goal is to find an approximate solution (in terms of its objective value) to each subproblem rather than solving it exactly. For doing so, we present a polynomial-time algorithm, Oracle, which can either report "NO," meaning that there is no feasible solution, or which produces a *relaxed feasible* solution that has a value of at least $(1 - \rho\epsilon)$ times that of an optimal solution, for some small integer $\rho > 0$, and has cost at most $(1 + \epsilon)B$. We call such a schedule a *ρ-schedule*. Having the algorithm Oracle, one can solve the original instance $\mathcal{I}$ via Algorithm 1. The correctness of the algorithm is stated in Lemma 6.

▶ **Lemma 6** (Correctness of Algorithm 1). *Algorithm 1 returns a schedule that has value at least $(1 - O(\epsilon))$ times the optimum, and has total costs of at most $(1 + \epsilon) \cdot B$. The running time of the algorithm is a polynomial in the input size, provided that the algorithm Oracle has the same running time.*

Algorithm 1 MAIN.

$\mathcal{P} \leftarrow \emptyset$
for each $t \in [T]$ **do**
    $\omega_t \leftarrow \lceil \log_{1+\alpha}(\sum_{j\in[n]} p_{t,j})\rceil$; $\Xi_t = \{0\} \cup \{(1+\alpha)^k, k = 0, \ldots, \omega_t\}$
for each vector $W \in \Xi_1 \times \cdots \times \Xi_T$ **do**
    Run ORACLE$(\mathcal{I}; W)$ and return a ρ-schedule A (if there exists one)
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{A\}$
Return $\bar{A} \leftarrow \arg\max_{A\in\mathcal{P}} \phi(A)$

The proof of Lemma 6 is omitted due to space limitations. In the rest of this section we describe the algorithm ORACLE to solve the subprobem SUB$(\mathcal{I}; W)$, for a given nonnegative vector $W = (W_t)_{t\in[T]}$. Our approach is based on solving a *configuration mixed integer linear program* (MILP) that can be viewed as a relaxed version of the subproblem. If the MILP is infeasible then our algorithm returns "NO"; otherwise, it outputs a solution with some nice properties by which the solution can be transformed back into a schedule without much affecting any of the objective value and the total cost. We sketch the main steps of ORACLE in Algorithm 2, and give some more detailed explanations in the following paragraphs.

Algorithm 2 ORACLE$(\mathcal{I}; W)$.

$\mathcal{X} \leftarrow \emptyset$; $\Delta \leftarrow 2(3T + \frac{1}{\alpha}\log\frac{1}{\alpha}\cdot T + 1)$; $q \leftarrow \max\{\lceil \Delta/\alpha \rceil, \lceil \Delta/\epsilon \rceil\}$;
Classify jobs into small, large, and huge ones according to their sizes
Enumerate sets $\mathcal{R}_t^1$ of the q jobs of largest size among all small jobs, and sets $\mathcal{R}_t^2$ of the q jobs of highest cost, which are assigned to machine type t, for all $t \in [T]$
for each valid guess $(\mathcal{R}_t^1, \mathcal{R}_t^2)_{t\in[T]}$ **do**
    Round down job sizes $p_{t,j}$, based on geometric rounding
    Define a MILP of small number of integer variables
    Find an optimal basic feasible solution (BFS) $(\mathbf{x}^*, \mathbf{y}^*)$ of at most Δ fractional components to MILP (if there exists any)
    Round $(\mathbf{x}^*, \mathbf{y}^*)$ to an integer solution $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ via solving a totally unimodular linear program, LP
    Transform $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ back into a schedule A
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{A\}$
return $\bar{A} = \arg\max_{A\in\mathcal{X}} \phi(A)$

2.1 Job Classification

We classify jobs in $[n]$ as *small*, *large*, and *huge* jobs, according to their sizes. A job is called

- *small*, w.r.t. machine type t, if its size is at most αW_t,
- *large*, w.r.t. machine type t, if its size is in $(\alpha W_t, (1+\alpha)W_t)$, and
- *huge*, w.r.t. machine type t, if its size is at least $(1+\alpha)W_t$.

In addition to the jobs above, we create *dummy jobs*, each of the same size αW_t w.r.t. machine type t. We call the jobs in $[n]$ *real jobs* to distinguish them from the dummy jobs. The dummy jobs are not included as real jobs in the instance $\mathcal{I}$, and are only used for modeling a mixed integer linear program later on.

2.2 Guessing Jobs of Specified Sizes and Costs in an Optimal Schedule

Let $q = \max\{\lceil \Delta/\alpha \rceil, \lceil \Delta/\epsilon \rceil\}$, where Δ is some constant that will be specified later. We guess a set $\mathcal{R}_t^1$ of the q jobs of largest size among all small jobs and a set $\mathcal{R}_t^2$ of q jobs of highest costs, which are assigned to machines of type t in an optimal schedule. For each set $\mathcal{R}_t^1$ we define an associated set $\bar{\mathcal{R}}_t^1 = \{j \in [n] \setminus \mathcal{R}_t^1 \mid \alpha W_t \geq p_{t,j} > \min_{j \in \mathcal{R}_t^1} p_{t,j}\}$, meaning that the (small) jobs in $\bar{\mathcal{R}}_t^1$ will not be assigned to any machine of type t. Similarly, we define an associated set $\bar{\mathcal{R}}_t^2 = \{j \in [n] \setminus \mathcal{R}_t^2 \mid c_{t,j} > \min_{j \in \mathcal{R}_t^2} c_{t,j}\}$, for each $\mathcal{R}_t^2$. It is essential to make sure that the sets $\mathcal{R}_t^1$ and $\mathcal{R}_t^2$, $t \in [T]$, are *consistent* with each other. For doing so, we give a definition of a *valid guess* below. Let $\mathcal{R}_t = \mathcal{R}_t^1 \cup \mathcal{R}_t^2$ and $\bar{\mathcal{R}}_t = \bar{\mathcal{R}}_t^1 \cup \bar{\mathcal{R}}_t^2$ for each $t \in [T]$.

▶ **Definition 7** (Valid guess). *A tuple $(\mathcal{R}_t^1, \mathcal{R}_t^2)_{t \in [T]}$ is a valid guess if the following hold:*

- $\mathcal{R}_t^1 \cap \bar{\mathcal{R}}_t^2 = \emptyset$ *and* $\bar{\mathcal{R}}_t^1 \cap \mathcal{R}_t^2 = \emptyset$ *for every* $t \in [T]$,
- $\mathcal{R}_t \cap \mathcal{R}_{t'} = \emptyset$ *for every* $t, t' \in [T]$ *with* $t \neq t'$.

To find exactly $(\mathcal{R}_t^1, \mathcal{R}_t^2)_{t \in [T]}$, Algorithm 2 considers all possibilities of choosing such sets of jobs of size q, with a running time bounded by $(Tn)^{2q} = n^{O(1)}$, as T and q are constants. In what follows we can thus assume, without loss of generality, that $(\mathcal{R}_t^1, \mathcal{R}_t^2)_{t \in [T]}$ are given.

2.3 Rounding the Job Sizes

Following the geometric rounding approach, the size of job j for machines of type $t \in [T]$ is rounded down to

$$\hat{p}_{t,j} = (1-\alpha)^{\ell} \cdot \alpha W_t, \quad \text{where } \ell = \left\lceil \log_{1-\alpha} \left({}^{p_{t,j}}/_{\alpha W_t} \right) \right\rceil. \tag{1}$$

Let $\mathcal{L}_t$ be the set of large rounded job sizes w.r.t. machine type t. The main purpose of using the geometric rounding approach is to bound the size of $\mathcal{L}_t$ by a small constant. Indeed, since the size of every large job is less than $(1+\alpha)W_t$, from (1) it follows that the number of large rounded sizes must be at most $\log_{1-\alpha} \frac{\alpha}{1+\alpha} \leq \frac{1}{\alpha} \log \frac{1}{\alpha} = O(1)$. On the negative side, the size of every (real) job may be decreased because of the rounding, but the rounded size is guaranteed to be within a factor of $1-\alpha$ of the original size. Formally, we have that $\hat{p}_{t,j} \geq (1-\alpha) \cdot p_{t,j}$ for all $t \in [T], j \in [n]$. Besides, note that the size of dummy jobs is not affected at all. Let $\hat{\mathcal{I}}$ be the new instance obtained from $\mathcal{I}$ by replacing $p_{t,j}$ by $\hat{p}_{t,j}$.

▶ **Proposition 8.** *For any schedule A, it holds that $|\phi_{\hat{\mathcal{I}}}(A) - \phi_{\mathcal{I}}(A)| \leq \epsilon \cdot \phi_{\mathcal{I}}(A)$.*

The proofs of Propositions 8 above and 9 below are omitted due to space limitations.

We define a subproblem $\textsc{Sub}(\hat{\mathcal{I}}; W)$ in a similar way as we have done for $\textsc{Sub}(\mathcal{I}; W)$ before, except that the interval $(\frac{1}{2}W_t, 2(1+\alpha)W_t)$ is now slightly changed to $\left(\frac{1-\alpha}{2}W_t, 2(1+\alpha)W_t\right)$.

▶ **Proposition 9.** *If there is a feasible schedule A to $\textsc{Sub}(\mathcal{I}; W)$, then A is also a feasible schedule to $\textsc{Sub}(\hat{\mathcal{I}}; W)$, and if A is a ρ-schedule to $\textsc{Sub}(\hat{\mathcal{I}}; W)$, then A is a $(\rho+2)$-schedule to $\textsc{Sub}(\mathcal{I}; W)$.*

By Proposition 9, it is sufficient to compute a ρ-schedule $\hat{A}$ to $\textsc{Sub}(\hat{\mathcal{I}}; W)$ for some small constant $\rho > 0$. This is done by Steps 6–9 of Algorithm 2.

2.4 Configuration Mixed Integer Linear Program (C-MILP)

At the heart of the MILP is a so-called *configuration*, an integer-coordinate vector encoding a subset of jobs received by a machine. In more detail, each coordinate of the vector encodes the number of jobs of a particular size contained in the subset. Unlike the configurations

considered in the model of Jansen and Maack [15], which involve only information about large job sizes, we allow configurations in our model to involve information about small ones. This, however, raises some obstacle: The number of small-job sizes can be as large as the number of small jobs, making the number of configurations as large as well.

To overcome this difficulty, one can instead use a number of dummy jobs of total size approximating the total size of small jobs. Formally, we can express a configuration as a $(|\mathcal{L}_t|+1)$-dimensional vector $\Phi = (\theta, \Phi_v)_{v\in\mathcal{L}_t} \in \mathbb{Z}_+^{|\mathcal{L}_t|+1}$, where Φ_v denotes the number of jobs of processing time $v \in \mathcal{L}_t$, and θ denotes the number of dummy jobs. The *value of a configuration* Φ *for a machine of type* t is defined as $\sigma_t(\Phi) = \sum_{v\in\mathcal{L}_t} v\cdot\Phi_v + \theta\cdot\alpha W_t$.

For every $t \in [T]$, all the possible configurations of value in $\left(\frac{1-3\alpha}{2}W_t,\ (2+3\alpha)W_t\right)$ are collected into a set $\mathcal{C}_t$. Also, we denote by $\mathcal{C}'_t \subset \mathcal{C}_t$ the set of configurations Φ with $\theta > 0$.

▶ **Proposition 10.** *$|\mathcal{C}_t|$ is upper-bounded by $O(1)$.*

The proof of Proposition 10 is again omitted due to space limitations.

Now, let $\varphi_1, \varphi_2, \varphi_3$ be nonnegative real numbers with $\varphi_1 \le 1$ and $\varphi_2, \varphi_3 \ge 1$. We formulate a mixed integer linear program, denoted by $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$, that consists of both fractional and integer variables. Denote $\mathbf{x} = (x_{t,j})_{t\in[T], j\in[n]}$ and $\mathbf{y} = (y_{t,\Phi})_{t\in[T],\Phi\in\mathcal{C}_t}$, where $x_{t,j} \in [0,1]$ indicates the fraction of job j assigned to machines of type t and $y_{t,\Phi} \in \mathbb{N}$ the number of configurations $\Phi \in \mathcal{C}_t$ assigned to machines of type t. Some variables $x_{t,j}$ are set to be either 0 or 1 according to our sets $(\mathcal{R}_t^1, \mathcal{R}_t^2)_{t\in[T]}$. In fact, for $j \in \mathcal{R}_t = \mathcal{R}_t^1 \cup \mathcal{R}_t^2$ we set $x_{t,j} = 1$ and $x_{t',j} = 0$ for every $t' \ne t$. In addition, we set $x_{t,j} = 0$ for all $j \in \bar{\mathcal{R}}_t = \bar{\mathcal{R}}_t^1 \cup \bar{\mathcal{R}}_t^2$.

For $t \in [T]$, let $\mathcal{O}_{t,v} = \{j \in [n] \mid \hat{p}_{t,j} = v\}$ be the set of jobs of rounded size v w.r.t. machine type t, and let $\mathcal{H}_t$ and $\mathcal{S}_t$, respectively, be the sets of huge jobs and of small jobs. The model of $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$ is as follows:

$$\text{maximize} \quad g(\mathbf{x},\mathbf{y}) = \sum\nolimits_{t\in[T]}\sum\nolimits_{\Phi\in\mathcal{C}_t} f(\sigma_t(\Phi))\cdot y_{t,\Phi} + \sum\nolimits_{t\in[T]}\sum\nolimits_{j\in\mathcal{H}_t} f(\hat{p}_{t,j})\cdot x_{t,j}$$

subject to

$$\sum\nolimits_{j\in\mathcal{H}_t} x_{t,j} + \sum\nolimits_{\Phi\in\mathcal{C}_t} y_{t,\Phi} = m_t, \qquad t \in [T], \tag{2}$$

$$\sum\nolimits_{t\in[T]} x_{t,j} = 1, j \in [n], \tag{3}$$

$$\sum\nolimits_{j\in\mathcal{O}_{t,v}} x_{t,j} \ge \sum\nolimits_{\Phi\in\mathcal{C}_t} \Phi_v\cdot y_{t,\Phi}, t \in [T], v \in \mathcal{L}_t, \tag{4}$$

$$\varphi_1 \sum_{\Phi\in\mathcal{C}'_t} (\theta-1) y_{t,\Phi} \le \frac{1}{\alpha W_t} \sum_{j\in\mathcal{S}_t} \hat{p}_{t,j}\cdot x_{t,j} \le \varphi_2 \sum_{\Phi\in\mathcal{C}'_t} (\theta+1) y_{t,\Phi}, t \in [T], \tag{5}$$

$$\sum\nolimits_{t\in[T]}\sum\nolimits_{j\in[n]} c_{t,j}\cdot x_{t,j} \le \varphi_3\cdot B. \tag{6}$$

The constraints (2) require that, for every type $t \in [T]$, the total number of configurations and huge jobs assigned to machines of this type is exactly the number of machines of this type. By the constraints (3), the variables $x_{t,j}$ indicate the fractional assignment of job j to machine types $t \in [T]$, for every $j \in [n]$. The constraints (4) guarantee that, for every $t \in [T]$ and for every $v \in \mathcal{L}_t$, the total number of jobs of size v that are assigned to machines of type t equals the total number of jobs of the same size used in the chosen configurations. The constraints (5) upper- and lower-bound the total sizes of small jobs assigned to machines of type t, according to the total size of dummy jobs used by the chosen configurations in $\mathcal{C}'_t$. By the constraints (6), the total cost of the schedule does not exceed $\varphi_3 B$. Finally, notice that the total number of variables and the total number of constraints are bounded by $O(n)$ and by $h = 1 + 3T + \sum_{t\in[T]} |\mathcal{L}_t| + n$, respectively. This completes the description of $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$.

2.5 Computing and Rounding the Solution to MILP

The next lemma summarizes important properties of the above constructed $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$, which are necessary in analyzing the correctness of Algorithm 2.

▶ **Lemma 11.** *The following three statements are true:*

1. *One can find an optimal* basic feasible solution *(BFS)* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *to* $\mathsf{MILP}[1,1,1]$ *(if there exists any), which has at most* $\Delta = 2\left(3T + \frac{1}{\alpha}\log\frac{1}{\alpha}\cdot T + 1\right)$ *fractional components.*
2. *Solution* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *can be rounded to an integer solution* $(\bar{\boldsymbol{x}}^*, \boldsymbol{y}^*)$ *with* $g(\bar{\boldsymbol{x}}^*, \boldsymbol{y}^*) \geq g(\boldsymbol{x}^*, \boldsymbol{y}^*)$.
3. *The rounded solution* $(\bar{\boldsymbol{x}}^*, \boldsymbol{y}^*)$ *is feasible to* $\mathsf{MILP}[1-\alpha, 1+\alpha, 1+\epsilon]$.

Proof.

1. It suffices to prove that $\mathbf{x}^*$ has at most Δ fractional components. Since the number of nontrivial constraints of $\mathsf{MILP}[1,1,1]$ is $h = 1 + 3T + \sum_{t\in[T]}|\mathcal{L}_t| + n$, $\mathbf{x}^*$ has at most h positive components. Consider the family of constraints (3). In each of these constraints, there are only two cases: (i) there is exactly one positive component $x^*_{t,j} = 1$, and (ii) there are at least two nonintegral components. Denote by ℓ_1 and ℓ_2, respectively, the number of constraints (3) in each case. Then we have that $\ell_1 + \ell_2 = n$. The total number of positive components of $\mathbf{x}^*$ is at least $\ell_1 + 2\ell_2 = 2n - \ell_1$. By an earlier argument, we must have that $2n - \ell_1 \leq h = 1 + 3T + \sum_{t\in[T]}|\mathcal{L}_t| + n$ or, equivalently, $\ell_1 \geq n - 3T - 1 - \sum_{t\in[T]}|\mathcal{L}_t|$. Therefore, the number of nonintegral components of $\mathbf{x}^*$ is at most

$$\begin{aligned} h - \left(n - 3T - 1 - \sum\nolimits_{t\in[T]}|\mathcal{L}_t|\right) &= 2\left(3T + 1 + \sum\nolimits_{t\in[T]}|\mathcal{L}_t|\right) \\ &\leq 2\left(3T + \frac{1}{\alpha}\log\frac{1}{\alpha}\cdot T + 1\right), \end{aligned}$$

   as $|\mathcal{L}_t|$ is upper-bounded by $\frac{1}{\alpha}\log\frac{1}{\alpha}$. Finally, notice that an optimal BFS to $\mathsf{MILP}[1,1,1]$ can be found in polynomial time [23].
2. To round $(\mathbf{x}^*, \mathbf{y}^*)$ to an integer solution, we follow a linear-programming-based approach.[2] As $\mathbf{y}^*$ is already integral, we need only to round $\mathbf{x}^*$ (or, more precisely, the fractional components of $\mathbf{x}^*$). To this end, we find a nonnegative optimal (integer) solution to the following linear program (LP) in which $\boldsymbol{\xi} \in [0,1]^{[T]\times[n]}$ and for all t, j we set $\xi_{t,j} = x^*_{t,j}$ for all $x^*_{t,j}$ that are already integral:

$$\max \quad g(\boldsymbol{\xi}, \mathbf{y}^*) = \sum\nolimits_{t\in[T]}\sum\nolimits_{\Phi\in\mathcal{C}_t} f(\sigma_t(\Phi))\cdot y^*_{t,\Phi} + \sum\nolimits_{t\in[T]}\sum\nolimits_{j\in\mathcal{H}_t} f(\hat{p}_{t,j})\cdot\xi_{t,j}$$

   subject to

$$\begin{aligned} \sum\nolimits_{j\in\mathcal{H}_t}\xi_{t,j} &= m_t - \sum\nolimits_{\Phi\in\mathcal{C}_t} y^*_{t,\Phi}, && t\in[T], \\ \sum\nolimits_{t\in[T]}\xi_{t,j} &= 1, && j\in[n], \\ \sum\nolimits_{j\in\mathcal{O}_{t,v}}\xi_{t,j} &\geq \sum\nolimits_{\Phi\in\mathcal{C}_t}\Phi_v\cdot y^*_{t,\Phi}, && t\in[T], v\in\mathcal{L}_t, \\ \xi_{t,j} &\geq 0, && t\in[T], j\in[n]. \end{aligned}$$

   One can see that LP has the form of $\mathcal{A}x \geq \mathbf{b}$, where the entries of $\mathcal{A}$ are only 0 or 1, and $\mathbf{b}$ is an integer vector. Furthermore, $\mathcal{A}$ has exactly two nonzero entries per column. Hence, $\mathcal{A}$ is a totally unimodular matrix, and thus every optimal basic feasible solution of

[2] Jansen and Maack [15] employ of a flow network utilizing flow integrality to round $(\mathbf{x}^*, \mathbf{y}^*)$ to an integer solution. In our case, the rounding needs to be done without decreasing the optimal objective value, making Jansen and Maack's technique inapplicable.

LP is integral. It is well-known that such a solution $\boldsymbol{\xi}^*$ of LP can be found in polynomial time. To get the second part of the second item, note that the optimal value of LP is an upper bound of that of MILP, and that $(\mathbf{x}^*, \mathbf{y}^*)$ constitutes a feasible solution to LP. By setting $\bar{x}^*_{t,j} = \xi^*_{t,j}$ for all t, j, we get an integer solution $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ with

$$g(\bar{\mathbf{x}}^*, \mathbf{y}^*) = g(\boldsymbol{\xi}^*, \mathbf{y}^*) \geq g(\mathbf{x}^*, \mathbf{y}^*). \tag{7}$$

3. One can check that while the integer solution $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ fulfills the constraints (2), (3), and (4), it may violate the last constraints – (5) and (6) – due to the rounding. We will show that this violation does not affect much the quality of the rounded solution. Consider the constraints (5) and fix a machine type $t \in [T]$. It suffices to consider the case when $n > q$, and thus $\mathcal{R}^1_t \subset [n]$; otherwise, n were constant and the original problem trivial to solve. Let $s = \arg\min_{j \in \mathcal{R}^1_t} p_{t,j}$ and $\mathcal{U}_t \subset [n]$ be the set of jobs j for which $x^*_{t,j}$ was fractional but rounded down to 0 because of the rounding. By Lemma 11, the size of $\mathcal{U}_t$ is at most Δ. Hence, we have that

$$\begin{aligned}
\sum\nolimits_{j \in \mathcal{S}_t} \hat{p}_{t,j} \cdot \bar{x}^*_{t,j} &\geq \sum\nolimits_{j \in \mathcal{S}_t} \hat{p}_{t,j} \cdot x^*_{t,j} - \sum\nolimits_{j \in \mathcal{U}_t} \hat{p}_{t,j} \cdot x^*_{t,j} \\
&\geq \sum\nolimits_{j \in \mathcal{S}_t} \hat{p}_{t,j} \cdot x^*_{t,j} - \Delta \cdot \hat{p}_{t,s} \\
&\geq \sum\nolimits_{j \in \mathcal{S}_t} \hat{p}_{t,j} \cdot x^*_{t,j} - \frac{\Delta}{|\mathcal{R}^1_t|} \cdot \sum\nolimits_{j \in \mathcal{R}^1_t} \hat{p}_{t,j} \\
&= \sum\nolimits_{j \in \mathcal{S}_t} \hat{p}_{t,j} \cdot x^*_{t,j} - \frac{\Delta}{q} \cdot \sum\nolimits_{j \in \mathcal{R}^1_t} \hat{p}_{t,j} \cdot x^*_{t,j}.
\end{aligned}$$

From $q \geq \lceil \Delta/\alpha \rceil$ and the feasibility of $(\mathbf{x}^*, \mathbf{y}^*)$ to MILP$[1,1,1]$, it follows that $\sum_{j \in \mathcal{S}_t} \hat{p}_{t,j} \geq (1-\alpha) \cdot \sum_{j \in \mathcal{S}_t} \hat{p}_{t,j} \cdot x^*_{t,j} \geq (1-\alpha) \sum_{\Phi \in \mathcal{C}'_t} (\theta - 1) \cdot y_{t,\Phi} \cdot \alpha W_t$. Similarly, one can prove that $\sum_{j \in \mathcal{S}_t} \hat{p}_{t,j} \leq (1+\alpha) \cdot \sum_{\Phi \in \mathcal{C}'_t} (\theta + 1) \cdot y_{t,\Phi} \cdot \alpha W_t$ and $\sum_{t \in [T]} \sum_{j \in [n]} c_{t,j} \cdot \bar{x}^*_{t,j} \leq (1+\epsilon) \cdot B$. ◀

2.6 Solution-to-Schedule Transformation

We now present a transformation of the integer solution $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ back into a schedule (not necessarily feasible) to Sub$(\hat{\mathcal{I}}; W)$. The details are provided in Lemma 12 the proof of which is omitted due to space limitations.

▶ **Lemma 12.** *Given the integer solution $(\bar{\boldsymbol{x}}^*, \boldsymbol{y}^*)$, one can transform it into a schedule whose value is at least $(1-\epsilon) \cdot g(\bar{\boldsymbol{x}}^*, \boldsymbol{y}^*)$ and whose cost is at most $(1+\epsilon) \cdot B$.*

2.7 Schedule-to-Solution Transformation

▶ **Lemma 13.** *Given an optimal (feasible) schedule A to Sub$(\hat{\mathcal{I}}; W)$, MILP$[1,1,1]$ has a feasible integer solution $(\bar{\boldsymbol{x}}, \boldsymbol{y})$ whose objective value is at least $(1-\epsilon) \cdot \phi_{\hat{\mathcal{I}}}(A)$.*

Proof. Let A be an optimal schedule to Sub$(\hat{\mathcal{I}}; W)$, we construct a feasible integer solution $(\bar{\mathbf{x}}, \mathbf{y})$ to MILP$[1,1,1]$ as follows. For each job j assigned to machines of type t, we set $\bar{x}_{t,j} = 1$. It remains to determine variables $y_{t,\Phi}$ for each $\Phi \in \mathcal{C}_t$. For machines that received large jobs only, one can easily define configurations corresponding to these machines. For machines that received small jobs, their assignments in A do not well establish a configuration. For such machines, we do the following transformation. Let M_t be the set of all machines i of type t which received small jobs. Fix a machine $i \in M_t$ and let δ_i be the total size of small jobs assigned to i. We first simply remove all jobs, and then assign dummy jobs to this machine i, one-by-one, until the total size of jobs assigned is equal to or exceeds δ_i. Let θ_i be the number

of dummy jobs that have been assigned to i. Based on θ_i and the large jobs assigned to i, one can easily determine the configuration $\Phi^i = (\theta_i, \Phi^i_v)_{v \in \mathcal{L}_t}$ corresponding to this machine. Finally, we can define the value of variables $y_{t,\Phi}$, for each $\Phi \in \mathcal{C}_t$, according to the number of configurations Φ that have been determined. One can check that the constraints (2), (3), (4), and (5) are fulfilled for $\varphi_1 = \varphi_2 = 1$. The satisfaction of the last constraint (6) comes from the fact that A is an optimal schedule to $\textsc{Sub}(\hat{\mathcal{I}}; W)$.

By the transformation above, one can see that $|L_i - \sigma_t(\Phi^i)| \leq \alpha W_t$ holds for every $i \in M_t$. Using the fact that $L_i \in \left(\frac{1-\alpha}{2} W_t, 2(1+\alpha)W_t\right)$ (as A is feasible to $\textsc{Sub}(\hat{\mathcal{I}}; W)$), we obtain $\frac{1-3\alpha}{2} W_t \leq \sigma_t(\Phi^i) \leq (2+3\alpha)W_t$. This also holds for every machine i that received large jobs only, implying that $\Phi^i \in \mathcal{C}_t$ for every i.

To prove the approximation factor, i.e., $g(\bar{\mathbf{x}}, \mathbf{y}) \geq (1-\epsilon) \cdot \phi_{\hat{\mathcal{I}}}(A)$, we first notice that

$$|L_i - \sigma_t(\Phi^i)| \leq 4\alpha \cdot L_i \leq \delta \cdot L_i \Longrightarrow f(\sigma_t(\Phi^i)) \geq (1-\epsilon) \cdot f(L_i) \tag{8}$$

holds for every $i \in M_t$, as $L_i \geq \frac{1-\alpha}{2} W_t$. If a machine i gets a configuration Φ^i containing only large jobs or a huge job j then it holds that $f(\sigma_t(\Phi^i)) = f(L_i)$ and $f(\hat{p}_{t,j}) = f(L_i)$. This together with (8) gives the desired inequality $g(\bar{\mathbf{x}}, \mathbf{y}) \geq (1-\epsilon) \cdot \phi_{\hat{\mathcal{I}}}(A)$. ◀

2.8 Correctness of Algorithm 2

We now are ready to prove that Algorithm 2 is correct.

▶ **Lemma 14.** *If A^* is an optimal (feasible) schedule to the subproblem $\textsc{Sub}(\mathcal{I}; W)$, then Algorithm 2 runs in polynomial time in the input size and returns a 4-schedule A, that is, $\phi_{\mathcal{I}}(A) \geq (1-4\epsilon) \cdot \phi_{\mathcal{I}}(A^*)$. Moreover, the cost of A is at most $(1+\epsilon) \cdot B$.*

Proof. Let $\hat{A}$ be an optimal schedule to $\textsc{Sub}(\hat{\mathcal{I}}; W)$. Then, by Lemma 13, there exists a feasible integer solution $(\bar{\mathbf{x}}, \mathbf{y})$ to $\mathsf{MILP}[1,1,1]$ and $g(\bar{\mathbf{x}}, \mathbf{y}) \geq (1-\epsilon) \cdot \phi_{\hat{\mathcal{I}}}(\hat{A})$. From (7) and the fact that $(\mathbf{x}^*, \mathbf{y}^*)$ is an optimal (fractional) solution to $\mathsf{MILP}[1,1,1]$, it follows that:

$$g(\bar{\mathbf{x}}^*, \mathbf{y}^*) \geq g(\mathbf{x}^*, \mathbf{y}^*) \geq g(\bar{\mathbf{x}}, \mathbf{y}) \geq (1-\epsilon) \cdot \phi_{\hat{\mathcal{I}}}(\hat{A}). \tag{9}$$

From Lemma 11.3, we know that $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ is feasible to $\mathsf{MILP}[1-\alpha, 1+\alpha, 1+\epsilon]$. Hence, by Lemma 12, one can compute a schedule A such that

$$\phi_{\hat{\mathcal{I}}}(A) \geq (1-\epsilon) \cdot g(\bar{\mathbf{x}}^*, \mathbf{y}^*). \tag{10}$$

From (9) and (10) it follows that $\phi_{\hat{\mathcal{I}}}(A) \geq (1-\epsilon)^2 \cdot \phi_{\hat{\mathcal{I}}}(\hat{A}) \geq (1-2\epsilon) \cdot \phi_{\hat{\mathcal{I}}}(\hat{A})$. By Proposition 9, we have $\phi_{\mathcal{I}}(A) \geq (1-4\epsilon) \cdot \phi_{\mathcal{I}}(A^*)$. Finally, by the feasibility of $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ to $\mathsf{MILP}[1-\alpha, 1+\alpha, 1+\epsilon]$, the cost of A is at most $(1+\epsilon) \cdot B$.

One can see that the overall complexity of Algorithm 2 is bounded by $O(T_1 \cdot T_2)$, where $T_1 = O(n^{2q})$ is the number of valid guesses of $\mathcal{R}_t$, and T_2 is the amount of time needed for solving $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$. To estimate T_2, note that $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$ has many variables but a constant number of integer ones and thus can be solved in time polynomial in the size of the input (see, for example, the work of Lenstra [19]). This completes the proof. ◀

3 Adapting the Framework for Other Objectives

In this section, we will discuss how to adapt the algorithmic framework presented in the previous section to attain similar results for other objectives such as max-min, min-max, and min-sum. First of all, note that Lemma 4 holds for any concave function f in case of the max-min and max-sum objectives, and for any convex function f in case of the min-max

and min-sum objectives. Also, Proposition 8 applies to any of our four objectives above, assuming that the function f fulfills condition (†). Now, consider the max-min objective. The only changes are in the model of the constructed $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$:

$$\begin{aligned}
\text{maximize} \quad & \zeta \\
\text{s.t.} \quad & \text{constraints (2), (3), (4), and (5)} \\
& y_{t,\Phi} \leq m_t \cdot y'_{t,\Phi}, && \Phi \in \mathcal{C}_t, && (11)\\
& y'_{t,\Phi} \cdot \sigma_t(\Phi) \geq \zeta, && \Phi \in \mathcal{C}_t, && (12)\\
& y'_{t,\Phi} \in \{0,1\}, && t \in [T], \Phi \in \mathcal{C}_t && (13)
\end{aligned}$$

are fulfilled. The number of constraints (Δ) now increases due to the added constraints, but still is a constant. Every step in Algorithm 2 remains unchanged and all our calculations are almost the same as for the max-sum objective. For the min-sum and the min-max objectives, there are several straightforward changes. The first change concerns the definition of subproblems, where the output of the algorithm Oracle is now a ρ-schedule whose value is *at most* $(1 + \rho\epsilon)$ times the optimum, for some constant $\rho > 0$. Rounding job sizes down still works well, though rounding them up can lead to a final solution with a slightly better approximation factor. Proposition 8 is unchanged, whilst the second part of Proposition 9 is stated slightly differently: "If A is a ρ-schedule to $\textsc{Sub}(\hat{\mathcal{I}}; W)$, then A is a $(\rho + 4)$-schedule to $\textsc{Sub}(\mathcal{I}; W)$." The proof is similar to that of Proposition 9. Regarding $\mathsf{MILP}[\varphi_1, \varphi_2, \varphi_3]$, the major change is in the objective, where we now are concerned with a minimization problem instead of a maximization one. Due to this change, all our technical results (Lemmas 11–14) can be modified in an appropriate way, without any difficulty. Our method can also be adopted to achieve a bi-PTAS for the case with any constant number of linear cost constraints, which generalizes the problem studied by Efraimidis and Spirakis [8].

▶ **Theorem 15.** *There is a bi-criteria PTAS for the load balancing problem on unrelated parallel machines with any constant number of linear cost constraints, assuming that the number of different types of machines is constant.*

4 Conclusion and Future Work

We have studied a class of load balancing problems on unrelated parallel machines in the presence of linear cost constraints, where the number of machine types is assumed to be bounded by a constant. Building on the work of Alon et al. [1] and Jansen and Maack [15], we have derived a unified approach for designing approximation schemes for the four studied objective types, involving a general nonnegative (convex or concave) function f fulfilling a certain property. Our results significantly generalize several existing results for load balancing problems with various objectives. Table 1 on page 3 gives an overview of our results.

In addition, in the absence of the constraints, we indeed achieve a PTAS for the max-sum objective, provided that the number of machine types is constant. As for future work, it would be interesting to extend our result to the setting where all machines of the same type are uniformly related [9], that is, they may have different speeds, making the processing time of a job different on different machines. Note that this special case has already been settled by Jansen and Maack [15] and Kones and Levin [17] for identity functions f. Another interesting direction is to study the question of whether we can achieve polynomial-time approximation schemes for computing a near-optimal schedule whose cost is not allowed to exceed the budget B.

References

1. N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.
2. Eric Angel, Evripidis Bampis, and Alexander V. Kononov. A FPTAS for approximating the unrelated parallel machines scheduling problem with costs. In *Proceedings of the 9th Annual European Symposium on Algorithms*, volume 2161 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2001.
3. Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 331–337. ACM Press, 2005.
4. Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, volume 2368 of *Lecture Notes in Computer Science*, pages 288–297. Springer, 2002.
5. J.F. Benders and J.A.E.E. van Nunen. A property of assignment type mixed integer linear programming problems. *Operations Research Letters*, 2:47–52, 1982.
6. Vincenzo Bonifaci and Andreas Wiese. Scheduling unrelated machines of few different types. Technical Report arXiv:1205.0974 [cs.DS], ACM Computing Research Repository (CoRR), 2012. `arXiv:1205.0974`.
7. Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 107–116. IEEE Computer Science Press, 2009.
8. Pavlos S. Efraimidis and Paul G. Spirakis. Approximation schemes for scheduling and covering on unrelated machines. *Theoretical Computer Science*, 359(1–3):400–417, 2006.
9. Leah Epstein and Asaf Levin. An efficient polynomial time approximation scheme for load balancing on uniformly related machines. *Mathematical Programming*, 147(1–2):1–23, 2014.
10. Leah Epstein and Jirí Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39(1):43–57, 2004.
11. Marshall L. Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.
12. Jan Clemens Gehrke, Klaus Jansen, Stefan E. J. Kraft, and Jakob Schikowski. A PTAS for scheduling unrelated machines of few different types. *International Journal of Foundations of Computer Science*, 29(4):591–621, 2018.
13. Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.
14. Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.
15. K. Jansen and M. Maack. An EPTAS for scheduling on unrelated machines of few different types. *Algorithmica*, 81(10):4134–4164, 2019.
16. Klaus Jansen and Lorant Porkolab. Improved approximation schemes for scheduling unrelated parallel machines. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 408–417. ACM Press, 1999.
17. I. Kones and A. Levin. A unified framework for designing EPTAS for load balancing on parallel machines. *Algorithmica*, 81(7):3025–3046, 2019.
18. Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
19. H. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.
20. Jyh-Han Lin and Jeffrey Scott Vitter. ϵ-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782. ACM Press, 1992.

21 T. Nguyen and J. Rothe. Approximate Pareto set for fair and efficient allocation: Few agent types or few resource types. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 290–296. ijcai.org, July 2020.

22 G.T. Ross and R.M. Soland. A branch and bound algorithm for the generalized assignment problem. *Management Science*, 24:345–357, 1977.

23 A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, USA, 1986.

24 David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

25 G. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.

26 Gerhard J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.