



Scheduling on a Stochastic Number of Machines

Moritz Buchem  

Technische Universität München, Germany

Franziska Eberle  

Technische Universität Berlin, Germany

Hugo Kooki Kasuya Rosado  

Technische Universität München, Germany

Kevin Schewior  

University of Southern Denmark, Odense, Denmark

Andreas Wiese  

Technische Universität München, Germany

Abstract

We consider a new scheduling problem on parallel identical machines in which the number of machines is initially not known, but it follows a given probability distribution. Only after all jobs are assigned to a given number of bags, the actual number of machines is revealed. Subsequently, the jobs need to be assigned to the machines without splitting the bags. This is the stochastic version of a related problem introduced by Stein and Zhong [SODA 2018, TALG 2020] and it is, for example, motivated by bundling jobs that need to be scheduled by data centers. We present two PTASs for the stochastic setting, computing job-to-bag assignments that (i) minimize the expected maximum machine load and (ii) maximize the expected minimum machine load (like in the Santa Claus problem), respectively. The former result follows by careful enumeration combined with known PTASs. For the latter result, we introduce an intricate dynamic program that we apply to a suitably rounded instance.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases scheduling, approximation algorithms, stochastic machines, makespan, max-min fair allocation, dynamic programming

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.14

Category APPROX

Related Version *Full Version:* <https://arxiv.org/abs/2407.15737>

Funding *Franziska Eberle:* Supported by the Dutch Research Council (NWO), Netherlands Vidi grant 016.Vidi.189.087.

Kevin Schewior: Supported by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.

1 Introduction

Stein and Zhong [20] recently introduced scheduling problems in which the number of the given (identical) machines is initially unknown. Specifically, all jobs must be assigned to a given number of bags before the actual number of machines is revealed. When that happens, the bags cannot be split anymore and they have to be assigned to the machines as whole bags, optimizing some objective function. Such problems arise, e.g., when “bundling” jobs to be scheduled in data centers, where the number of available machines depends on external factors such as momentary demand [4, 20].

The aforementioned work (as well as follow-up works [1, 5]) focused on the robustness of a job-to-bag assignment. Specifically, they assumed a worst-case number of machines and compared their solution with the in-hindsight optimum for the respective objective function,



© Moritz Buchem, Franziska Eberle, Hugo Kooki Kasuya Rosado, Kevin Schewior, and Andreas Wiese; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 14; pp. 14:1–14:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

i.e., a direct job-to-machine assignment without bags. In contrast to this information-theoretic question, we assume that a distribution of the number of machines is known (e.g., from historical data) and aim to *efficiently compute* a job-to-bag assignment that optimizes the objective function *in expectation* – a common formulation of the objective function for stochastic (scheduling) problems [4, 9, 10, 14, 15, 17, 18]. In other words, we use a “fairer” benchmark for our algorithms, allowing us to sidestep the strong lower bounds by [20]. We are the first to study this novel type of scheduling problem, already proposed in [20].

We consider two classic objective functions: minimizing the maximum machine load (makespan) and maximizing the minimum machine load (Santa Claus). Both objectives are well-studied in the deterministic setting, the special case of our problem with one-point distributions, i.e., the distributions in which only one event happens with positive probability. These problems are well understood from a classic approximation perspective: both are known to be strongly NP-hard [7] and both admit Polynomial-Time Approximation Schemes (PTASs) [11, 21], i.e., polynomial-time $(1 + \varepsilon)$ -approximation algorithms for any $\varepsilon > 0$. In this paper, surprisingly, we recover the same state for the stochastic versions by designing a PTAS in both cases. In contrast to the deterministic setting, we require different techniques tailored to each objective function. For the makespan minimization objective, our main technical contribution is the application and analysis of techniques that have previously been used in approximation schemes for *deterministic* scheduling and packing problems. Our approach for the Santa Claus objective is technically much more intriguing and requires the careful set-up of a novel dynamic program (DP) in order to control its size.

Our results are in stark contrast to classic stochastic scheduling problems, where in some cases the currently best known approximation algorithms have distribution-dependent or even linear guarantees [14, 18]. Even for better-understood problems such as load balancing of stochastic jobs on deterministic machines, previous approaches [4, 9, 15] rely on concentration bounds which inherently prohibit approximation ratios of $1 + \varepsilon$ for arbitrarily small $\varepsilon > 0$. Moreover, PTASs for stochastic load balancing on deterministic machines are only known for identical machines and Poisson distributed jobs [2, 13]. We hope that our positive results inspire research for other scheduling problems with a stochastic number of machines, even for (in the classic model with jobs with stochastic processing times) notoriously hard objective functions such as expected weighted sum of completion times.

1.1 Our Contribution and Techniques

Our first result is the following.

► **Theorem 1.** *There is a PTAS for the problem of computing the job-to-bag assignment that minimizes the expected maximum machine load.*

We first guess the bag sizes of the optimal solution up to a factor of $1 + \varepsilon$. For each guess, we check whether there is a corresponding assignment of the jobs to the bags (up to a factor of $1 + \varepsilon$), using the PTAS for bin packing with variable sizes [12]. Among the guesses that fulfill this condition, we can select the (approximately) best guess using the PTAS for makespan minimization [11].

For this approach to yield a PTAS, we need to bound the number of guesses by a polynomial (in the input length). First note that it is straightforward to get down to a *quasi-polynomial* number of guesses (and thus a QPTAS). The approach is to disregard jobs of size $(\varepsilon/n) \cdot p_{\max}$ where p_{\max} is the largest processing time; indeed, for any solution, such jobs make up at most an ε -fraction of the objective-function value. The resulting number of possible guesses for a single bag size is then logarithmic in n , leading to a quasi-polynomial

number of guesses for the (multi-)set of bag sizes. To get a *polynomial* bound (and thus a PTAS), we make the following crucial observation. Let C be an estimate for the largest *bag* size, up to a constant factor. While bags of size $\mathcal{O}(\varepsilon C)$ cannot be disregarded, it is enough to know their *number* rather than approximate size. Intuitively, when computing a bag-to-machine assignment, these bags are treated like “sand”, i.e., as infinitesimal jobs of total volume equal to the total volume of those bags. The number of possible (rounded) bag sizes is hence constant, leading to a polynomial number of guesses for the (multi-)set of bag sizes.

Our second result is significantly harder to achieve.

► **Theorem 2.** *There is a PTAS for the problem of computing the job-to-bag assignment that maximizes the expected minimum machine load.*

One may be tempted to try a similar approach as for our first result. Even getting a QPTAS is, however, not possible in the same way as one cannot simply disregard jobs of size $(\varepsilon/n) \cdot p_{\max}$. Consider an instance in which the number of machines is deterministically M and in which there are one job of size 1 and $M - 1$ jobs of size $\varepsilon/(2M)$. Here, ignoring the jobs of size $\varepsilon/(2M)$ leads to $M - 1$ empty bags, yielding an objective function value of 0 instead of the optimal value $\varepsilon/(2M)$.

Also, it is no longer true either that for bags of size $\mathcal{O}(\varepsilon C)$ (where C is the size of a largest bag) it is enough to know their total number: Consider an instance with optimal objective-function value OPT and add one huge job of size OPT/ε^2 to the set of jobs and one machine to each scenario. Clearly, this new instance has maximum bag size $C \geq \text{OPT}/\varepsilon^2$ while the optimal objective-function value does not change since this huge job can safely be packed in its private bag and scheduled on its private machine. (However, crucially for the PTAS for makespan minimization, adding this huge job there would change the objective-function value.) In this example, the probability that scenarios with optimal objective-function value much smaller than εC occur is 1. To obtain a $(1 + \varepsilon)$ -approximation, however, one still has to compute a $(1 + \varepsilon)$ -approximation for the original instance, for which the sizes of bags of size $\mathcal{O}(\varepsilon C)$ are relevant. Of course, the issue with this particular instance could be avoided by removing the huge job in a pre-processing step. However, by concatenating the above original instance at super-constantly many different scales, one can create a new instance where one essentially has to identify the “relevant scales” in a preprocessing step.

In some sense, the first step of proving Theorem 2 is addressing precisely the problem of identifying the correct scales: We show that, at a loss of $1 + \mathcal{O}(\varepsilon)$ in the approximation guarantee, the problem can be reduced to the case of polynomially bounded processing times that are all powers of $1 + \varepsilon$. To do so, we define suitable (non-trivial) subproblems and assemble them to a global solution with a dynamic program (DP). This approach can be seen as a simpler version of our approach for polynomially bounded processing times, which we focus on in the following.

Our general approach is to divide the range of possible bag sizes into intervals that contain $\mathcal{O}_\varepsilon(1)$ possible approximate bag sizes each. For each such interval, it is then possible to guess the set of bags of the respective size in polynomial time. Considering the same range for sizes of *jobs*, rather than bags, we would also be able to guess the assignment of these jobs to these bags. Observe that a job may, of course, be assigned to a bag whose size lies in a different interval than the job’s size. However, we argue that the precise assignment is only relevant when these intervals are neighboring intervals and, hence, can be guessed in polynomial time. If this is not the case, i.e., if jobs are assigned to a bag whose size lies in a much larger interval, then such jobs are sufficiently small for us to only consider their total volume. The resulting parameters, such as number of bags created so far and the assignment of

smaller jobs to larger intervals, through which the subproblems corresponding to the intervals interact, are kept track of by a DP. While it is straightforward to keep track of all bags from larger intervals as well as the assignment of jobs from these intervals to the bags, it is not clear how to do this with a polynomial-time DP. In particular, when we consider bag sizes of one interval, we still need to remember previously defined bags of much larger sizes with a super-constant number of possibilities for these sizes. In fact, we show that it is sufficient to keep track of a constant number of parameters that capture all necessary information about larger intervals. Using that the processing times are polynomially bounded, we can bound the size of the DP table by a polynomial in the encoding of the input.

When considering a DP cell corresponding to some interval and guessing bag sizes along with the other parameters implied by the discussion above, we need to evaluate the quality of this guess. To do so, we guess additional parameters (also kept track of by the DP). The main observation is as follows. Suppose that, in addition to the aforementioned parameters, we know the relevant range of the number of machines and the bag sizes from the next-lower interval. For each number of machines in the aforementioned range, we assign each bag

- (i) from a higher interval to one machine each,
- (ii) from the two currently relevant intervals optimally, and
- (iii) from the lower intervals fractionally (the total volume of these bags can be approximated).

The reason we may do so is that the bags assigned in (i) are large enough to assign enough load to an entire machine and the bags assigned in (iii) are small enough to be considered fractional.

We remark that for both problems considered, a PTAS is the best possible approximation algorithm achievable when the number of bags (and machines) is part of the input, unless $P = NP$: Since their strongly NP-hard deterministic counterparts [8] are special cases of the stochastic problems, neither makespan minimization nor Santa Claus on stochastic machines admits fully polynomial time approximation schemes (FPTASs) unless $P = NP$. If, however, the number of bags (and machines) is not part of the input, i.e., a constant, a FPTAS can be designed by directly guessing the bag sizes approximately, i.e., up to a factor of $1 + \varepsilon$, in polynomial time and using known FPTASs to compute a job-to-bag assignment based on these bag sizes [6, 16].

Stein and Zhong [20] also considered a third objective function, minimizing the difference between the maximum and the minimum machine load. Any polynomial-time approximation algorithm (in the multiplicative sense) is, however, impossible here unless $P = NP$. Indeed, already in the deterministic case, it is strongly NP-hard to decide whether the optimal objective-function value is 0 (as can be seen, e.g., by a straightforward reduction from 3-Partition).

1.2 Further Related Work

We first review the literature on the aforementioned information-theoretic question in which one compares with the in-hindsight optimum. Since this benchmark is stronger than ours and the upper bounds are obtained through polynomial-time algorithms, the upper bounds carry over to our setting as guarantees of polynomial-time approximation algorithms. Specifically, for makespan, Stein and Zhong [20] showed how to compute for any $\varepsilon > 0$ a job-to-bag assignment whose cost is guaranteed to be a factor of at most $5/3 + \varepsilon$ away from the cost of the in-hindsight optimum. They also showed an impossibility of $4/3$. When all jobs have infinitesimal size, the best-possible guarantee is $(1 + \sqrt{2})/2 \approx 1.207$ [5, 20]. For Santa Claus and infinitesimal jobs, the best-possible guarantee is $2 \ln 2 \approx 1.386$ [20].

This model has been generalized in two directions. First, Eberle et al. [5] considered arbitrary machine *speeds* (rather than just 0 or 1) that are revealed after the bags have been created. They gave a guarantee of $2 - 1/m$ with respect to the in-hindsight optimum and improved guarantees for special cases. Second, Balkanski et al. [1] considered the problem with arbitrary speeds in the algorithms-with-predictions framework.

In the majority of the scheduling literature, stochastic uncertainty refers to uncertainty in the processing times of the jobs (see [17] for a survey and [4, 9, 10] for some recent works). The literature on stochastic uncertainty, even uncertainty in general, in the machines is much more scattered. Stadjc considered the unrecoverable breakdown on a single machine caused by stochastic jobs [19]. Temporary machine unavailability has also been studied in [3].

2 Preliminaries

Formally, we are given a job set $J = [n] := \{1, \dots, n\}$, where each job has a *processing time* p_j , and a maximum number of machines M . For each $1 \leq m \leq M$, we are given its *probability* q_m , where $\sum_{m=1}^M q_m = 1$. We want to find a partition of the job set J into M sets, called *bags*. We denote the set of bags by \mathcal{B} . For a bag $B \in \mathcal{B}$, let $p(B) = \sum_{j \in B} p_j$ denote its *size*. We typically say for $j \in B$ that j is *packed* in bag B .

Clearly, if $M \geq n$, we can pack every job in its own private bag, and the problem becomes trivial. Hence, we assume from now on that $M < n$.

We denote by $\text{OPT}(\mathcal{B}, m)$ the optimal objective function value for a given set of bags \mathcal{B} and a scenario with m machines, that is, $\text{OPT}(\mathcal{B}, m)$ denotes the maximum or minimum machine load of an optimal bag-to-machine assignment, or *schedule*, respectively. The objective is to find a partition or set of bags \mathcal{B} that optimizes $\sum_{m=1}^M q_m \text{OPT}(\mathcal{B}, m)$. We denote a fixed optimal set of bags by \mathcal{B}^* and its objective function value by $\text{OPT} := \sum_{m=1}^M q_m \text{OPT}(\mathcal{B}^*, m)$.

As discussed above, the problems we consider are generalizations of strongly NP-hard problems. Thus, unless $P = NP$, we cannot expect to find \mathcal{B}^* in polynomial time. Hence, we are interested in *polynomial-time approximation schemes* (PTASs), i.e., for each $\varepsilon > 0$, a polynomial-time $(1 + \varepsilon)$ -*approximation algorithm*. Such an algorithm is required to return a partition of the job set J into M bags, denoted by \mathcal{B} , that satisfies $\sum_{m=1}^M q_m \text{OPT}(\mathcal{B}, m) \leq (1 + \varepsilon) \text{OPT}$ for makespan, and $\sum_{m=1}^M q_m \text{OPT}(\mathcal{B}, m) \geq \frac{1}{1 + \varepsilon} \text{OPT}$ for Santa Claus.

3 Minimizing the maximum machine load

In this section we design and analyze our polynomial-time approximation scheme for the setting of makespan minimization: For a given number of machines m and a set \mathcal{B} of bags, we want to find an assignment of bags to machines that minimizes the maximum total size of bags assigned to any machine.

3.1 Algorithm

Let $\varepsilon > 0$; we will give a polynomial-time algorithm that achieves an approximation ratio of $1 + \mathcal{O}(\varepsilon)$. This algorithm finds a good estimate of the optimal bag sizes in \mathcal{B}^* . To this end, we show later that the maximum size of a bag in \mathcal{B}^* is at most $4C$, where

$$C := \sum_{m=1}^M q_m \max \left\{ \max_{j \in J} p_j, \frac{1}{m} \sum_{j \in J} p_j \right\}.$$

We say a bag B in \mathcal{B}^* is *regular* if its size is at least εC or if there is at least one job of size at least $\varepsilon^2 C$ packed in B . For $\ell \in \mathcal{L} := \{\lfloor \log_{1+\varepsilon}(\varepsilon^2 C) \rfloor, \lfloor \log_{1+\varepsilon}(\varepsilon^2 C) \rfloor + 1, \dots, \lfloor \log_{1+\varepsilon}(4C) \rfloor\}$, the algorithm *guesses* the number M_ℓ of optimal bags with $p(B) \in [(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})$.

Further, it *enumerates* all possible numbers M_{sand} of bags of size at most $(1+\varepsilon)\varepsilon C$, called *sand bags*. These sand bags do not directly correspond to optimal bags, but instead can pack all jobs not packed in regular bags in OPT.

Clearly, a guess $(M_\ell)_{\ell \in \mathcal{L}}$ combined with M_{sand} sand bags does not necessarily guarantee that it is *feasible*, i.e., that $M_{\text{sand}} + \sum_{\ell \in \mathcal{L}} M_\ell \leq M$ and that there is a partition of J into bags such that there are at most M_ℓ bags with sizes in $[(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})$ and M_{sand} bags of size at most $(1+\varepsilon)\varepsilon C$. Thus, the algorithm ignores all combinations of $(M_\ell)_{\ell \in \mathcal{L}}$ and M_{sand} with more than M bags. If the total number of bags is at most M , the algorithm uses the PTAS by Hochbaum and Shmoys [12] for bin packing with variable bin sizes to check if there is a feasible packing of jobs into the bags as follows: The input is ε as approximation parameter, an item of size p_j for each job $j \in J$, M_ℓ bins of size $(1+\varepsilon)^{\ell+1}$ for any $\ell \in \mathcal{L}$, and M_{sand} bins of size $(1+\varepsilon)\varepsilon C$. If the guess is feasible, the PTAS is guaranteed to return an item-to-bin (here a job-to-bag) assignment that violates the bin sizes by at most a factor $(1+\varepsilon)$.

If all jobs can be packed by the above PTAS, the algorithm evaluates the current guess by computing a $(1+\varepsilon)$ -approximation of $\text{OPT}((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$, where we overload notation and let $\text{OPT}((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$ denote the minimum makespan for a set of bags consisting of M_ℓ bags of size $(1+\varepsilon)^{\ell+1}$, for any $\ell \in \mathcal{L}$, and M_{sand} bags of size $(1+\varepsilon)\varepsilon C$. We denote the makespan of this $(1+\varepsilon)$ -approximation by $z((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$ and compute it by running the PTAS by Hochbaum and Shmoys [11] for makespan minimization on identical machines with approximation parameter ε , M_ℓ jobs with processing time $(1+\varepsilon)^{\ell+1}$, M_{sand} jobs with processing time $(1+\varepsilon)\varepsilon C$, and m machines.

The algorithm returns a feasible minimizer of $\sum_{m=1}^M q_m z((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$.

3.2 Analysis

In this section, we analyze the algorithm designed in the previous section. We start by justifying our bound on the maximum bag size before we argue that there exists a guess that is similar to the optimal set \mathcal{B}^* in terms of the bag size and objective-function value. Last, we evaluate the running time of the algorithm and conclude with the proof of Theorem 1. For formal proofs we refer to the full version.

We begin by justifying our assumption to only consider bags of size at most $4C = 4 \sum_{m=1}^M q_m \max \{ \max_{j \in J} p_j, \frac{1}{m} \sum_{j \in J} p_j \}$: By [20], $4C$ is an upper bound on OPT. As the largest bag size lower bounds $\text{OPT}(\mathcal{B}, m)$ in scenario m , this implies the next lemma.

► **Lemma 3.** *No optimal solution uses bags of size greater than $4C$.*

Fix a set of bags \mathcal{B}^* with objective-function value OPT. By Lemma 3, the maximum bag size is at most $4C$. The algorithm guesses a set of bag sizes similar to the bag sizes in \mathcal{B}^* .

Based on \mathcal{B}^* we define a “good” guess $(\hat{M}_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}$, i.e., a set of possible bag sizes, as follows: Let \mathcal{B}_R^* denote the set of regular bags, i.e., the set of bags in \mathcal{B}^* that pack at least one job of size at least $\varepsilon^2 C$ or have size at least εC .

- For $\ell \in \mathcal{L}$, let \hat{M}_ℓ be the number of regular bags in \mathcal{B}_R^* with $p(B) \in [(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})$.
- Set $\hat{M}_{\text{sand}} = \left\lceil \frac{\sum_{B \in \mathcal{B}^* \setminus \mathcal{B}_R^*} p(B)}{\varepsilon C} \right\rceil$; recall that sand bags have size at most $(1+\varepsilon)\varepsilon C$.

Since the sizes of regular bags are only rounded up, the bags in $(\hat{M}_\ell)_{\ell \in \mathcal{L}}$ can pack the same subset of jobs as \mathcal{B}_R^* . Since the volume of any sand bag has been increased by a $(1+\varepsilon)$ -factor as opposed to εC and the size of any job not packed in a regular bag is at most $\varepsilon^2 C$, we show that the bag-size vector $\hat{\mathcal{M}} := (\hat{M}_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}$ is a possible and feasible guess.

► **Lemma 4.** *The above defined vector $\hat{\mathcal{M}}$ is a feasible guess of the algorithm. The jobs can be packed into a set of bags consisting of \hat{M}_ℓ bags with size $(1 + \varepsilon)^{\ell+1}$ for $\ell \in \mathcal{L}$ and \hat{M}_{sand} bags with size $(1 + \varepsilon)\varepsilon C$.*

Combining this lemma with Theorem 2 of [12], we get the next corollary.

► **Corollary 5.** *The PTAS by [12] returns a packing of all jobs into a set of bags with M_ℓ bags of size $(1 + \varepsilon)^{\ell+2}$ for $\ell \in \mathcal{L}$ and M_{sand} bags of size $(1 + \varepsilon)^2\varepsilon C$.*

To prove the next lemma, we first assign the regular bags in the same way as in the optimal solution and assign the sand bags one by one to the currently least loaded machine. For bounding the makespan in a given scenario m , we distinguish whether a regular bag determines the makespan (which increases the makespan by at most a factor $(1 + \varepsilon)$ compared to $\text{OPT}(\mathcal{B}^*, m)$) or whether a sand bag determines the makespan. In the latter case, we use a volume bound to upper bound this sand bag's starting time (losing at most a factor $(1 + \varepsilon)$) and amortize its maximum size, i.e., $(1 + \varepsilon)\varepsilon C$, over all scenarios, using that $\sum_{m=1}^M q_m = 1$.

► **Lemma 6.** *$\hat{\mathcal{M}} = (\hat{M}_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}$ satisfies $\sum_{m=1}^M q_m \text{OPT}(\hat{\mathcal{M}}, m) \leq (1 + 5\varepsilon)\text{OPT}$.*

Proof of Theorem 1. Using that we return the cheapest guess and that the number of distinct rounded sizes of regular bags is bounded by $\mathcal{O}(\frac{1}{\varepsilon^2})$, we can show the following two lemmas. Combined, they complete the proof of Theorem 1.

► **Lemma 7.** *The set of bags returned by the algorithm guarantees an objective function value of at most $(1 + \mathcal{O}(\varepsilon))\text{OPT}$.*

► **Lemma 8.** *For $\varepsilon \in (0, \frac{1}{2})$, the algorithm runs in time $\mathcal{O}\left(\left(\frac{n}{\varepsilon}\right)^{\mathcal{O}(1/\varepsilon^2)}\right)$.*

4 Maximizing the minimum machine load

In this section, we present our polynomial-time $(1 + \varepsilon)$ -approximation algorithm for the setting in which we want to maximize the minimum machine load. We refer to the full version for the formal proofs for the results presented in this section.

Polynomially bounded processing times

First, we show that we can reduce our problem to the case of polynomially bounded job processing times that are all essentially powers of $1 + \varepsilon$, while losing at most a factor of $1 + \mathcal{O}(\varepsilon)$ in our approximation guarantee. The main concepts of the reduction can be summarized by the following three ideas.

The first idea is to disregard scenarios whose contribution to the expected objective function value is very small. W.l.o.g., assume that $p_j \in \mathbb{N}$ and let d be an integer such that $\text{OPT}(\mathcal{B}^*, m)$ falls in the interval $[1, (\frac{n}{\varepsilon})^d]$ for every scenario m . Then, for some offset $a \in \{0, 1, \dots, \frac{1}{\varepsilon} + 3\}$ we “split” the interval $[1, (\frac{n}{\varepsilon})^d]$ into a polynomial number of pairwise disjoint intervals $\tilde{I}_i = \left[\left(\frac{n}{\varepsilon}\right)^{3i + \frac{i-1}{\varepsilon} + a}, \left(\frac{n}{\varepsilon}\right)^{3i + \frac{i}{\varepsilon} + a} \right)$. Observe that any two consecutive intervals have a multiplicative gap of $\left(\frac{n}{\varepsilon}\right)^3$. Using probabilistic arguments, we show that there is an offset a such that the scenarios with $\text{OPT}(\mathcal{B}^*, m)$ in the gaps contribute very little to the expected objective function value. Hence, such scenarios can be neglected by losing a factor of at most $1 + \mathcal{O}(\varepsilon)$ in the approximation ratio. As there is only a polynomial number of possible offsets a , we may assume that we correctly choose such a by enumeration.

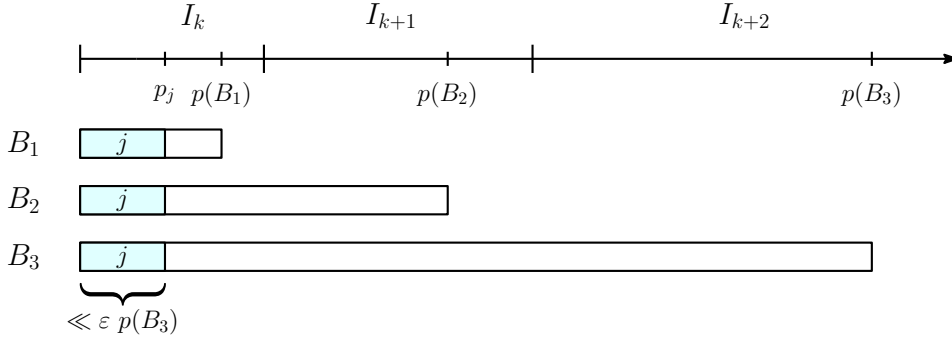
The second idea is to observe that the gaps enable us to actually *ignore* a carefully chosen subset of jobs. Let $\tilde{I}_i^+ = \left[\left(\frac{n}{\varepsilon}\right)^{3i + \frac{i-1}{\varepsilon} + a - 3}, \left(\frac{n}{\varepsilon}\right)^{3i + \frac{i-1}{\varepsilon} + a} \right) \cup \tilde{I}_i$ denote the extended interval obtained by the union of the interval \tilde{I}_i and the smaller of its adjacent gaps, and let m be a scenario such that $\text{OPT}(\mathcal{B}^*, m) \in \tilde{I}_i$. We show that, by losing a factor of at most $1 + \mathcal{O}(\varepsilon)$ in the approximation ratio, we may assume that a machine with minimum load in the schedule that achieves $\text{OPT}(\mathcal{B}^*, m)$ is assigned only bags with jobs whose processing time is in \tilde{I}_i^+ . Then, based on this assumption, we show that we may assume that there are no jobs whose processing times fall in the gaps by losing at most another factor of $1 + \mathcal{O}(\varepsilon)$ in the approximation ratio. Observe that we are now facing an instance where neither $\text{OPT}(\mathcal{B}^*, m)$ nor p_j belong to the just created gaps in the interval $\left[1, \left(\frac{n}{\varepsilon}\right)^d\right]$.

The third idea is then to solve the problem restricted to the intervals \tilde{I}_i individually by using the fact that within each interval \tilde{I}_i the processing times are polynomially bounded and combine the obtained solutions into a single one with a dynamic program. We show that rounding up the processing times and solving each subproblem that arises in the dynamic program costs a factor of at most $1 + \mathcal{O}(\varepsilon)$ in the approximation ratio. Formalizing this proof sketch proves Lemma 9.

► **Lemma 9.** *By losing a factor of at most $1 + \mathcal{O}(\varepsilon)$ in the approximation ratio, we can assume for each job $j \in J$ that $p_j = \lceil (1 + \varepsilon)^{k_j} \rceil$ for some $k_j \in \mathbb{N}_0$ and $p_j \in [1, n^{c(\varepsilon)}]$ where $c(\varepsilon)$ is some global constant.*

Algorithmic overview. Based on Lemma 9, we assume that each job $j \in J$ satisfies $p_j \in [1, n^{c(\varepsilon)}]$. The high-level idea of our algorithm is to partition $[1, n^{c(\varepsilon)}]$ into intervals of the form $I_k := \left[\left(\frac{1}{\varepsilon}\right)^{3k}, \left(\frac{1}{\varepsilon}\right)^{3k+3}\right)$ for $k \in \mathbb{N}$ and only consider bags, jobs, and scenarios relevant for a *single* interval. More precisely, we use these intervals to partition the processing times $\{p_j\}_{j \in J}$, the bag sizes in \mathcal{B}^* and in our solution as well as the values $\{\text{OPT}(\mathcal{B}^*, m)\}_m$. Let K such that $\sum_{j \in J} p_j \in I_K$; we ignore intervals I_k with $k > K$. For $k \in [K]$, let $J_k := \{j \in J : p_j \in I_k\}$, let $L_k := \{\ell \in \mathbb{N} : \lceil (1 + \varepsilon)^\ell \rceil \in I_k\}$, and let $\mathcal{B}_k^* := \{B \in \mathcal{B}^* : p(B) \in I_k\}$.

Our algorithm recursively considers the intervals in the order I_K, I_{K-1}, \dots, I_1 and, step by step, defines bags that correspond to $\mathcal{B}_K^*, \mathcal{B}_{K-1}^*, \dots, \mathcal{B}_1^*$. When considering interval I_k , the algorithm enumerates all possible bag sizes of bags in \mathcal{B}_k^* and all possible assignments of a subset of the jobs in $J_k \cup J_{k-1}$ to those bags; the remaining jobs in J_k are implicitly assigned to bags in $\bigcup_{k'=k+1}^K \mathcal{B}_{k'}^*$. Here, we use the fact that by definition of our intervals only a constant number of jobs in $J_k \cup J_{k-1}$ can be assigned to any bag in \mathcal{B}_k^* while jobs in J_k are tiny compared to bags in $\bigcup_{k'=k+2}^K \mathcal{B}_{k'}^*$ (see Figure 1 for visualization) and, hence, the assignment of jobs J_k to bags $\bigcup_{k'=k+2}^K \mathcal{B}_{k'}^*$ cannot be guessed in polynomial time. The remaining jobs in J_{k-1} will be assigned when interval I_{k-1} is considered. We embed this recursion into a polynomial-time dynamic program (DP). Since our DP is quite technical, we first describe the algorithmic steps that correspond to the root subproblem of the recursion, i.e., I_K , before we define the DP cells and argue about their solution. Defining the DP cells and solving their corresponding subproblem involves enumerating all possible values of several quantities and storing an approximation of the objective-function value of the (approximately) best combination in the DP cell. When arguing about the correctness of our DP, we show that there is a chain of DP cells that represent some (fixed) optimal solution. Hence, we use X^* for some parameter X when referring to the correct value, i.e., the value of this parameter in this optimal solution. We refer to this process as *guessing X^** . In general, we use \hat{X} to refer to an arbitrary guess.



■ **Figure 1** Visualization of relation between jobs in J_k and bags in \mathcal{B}_k^* , \mathcal{B}_{k+1}^* and $\bigcup_{k'=k+2}^K \mathcal{B}_{k'}^*$.

4.1 Guessing initial quantities

In the following, we describe how to guess and evaluate initial parameters corresponding to a (partial) solution of our root subproblem. Intuitively, we construct a partial assignment of jobs to bags \mathcal{B}_K^* and the parameters representing this partial assignment define the DP cell corresponding to the remaining problem.

4.1.1 Algorithm

The algorithm to compute a partial solution to a root subproblem can be essentially split into two phases: (1) *guessing* key quantities and (2) *evaluating* these guesses.

Guessing phase. We start by guessing $|\mathcal{B}_K^*|$ and $|\mathcal{B}_{K-1}^*|$, the number of bags in \mathcal{B}_K^* and in \mathcal{B}_{K-1}^* , before we guess $(1 + \varepsilon)$ -approximations for the bags sizes in $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$. Formally, for each bag $B \in \mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ we guess a value $\ell(B) \in \mathbb{N}$ such that $p(B) \in [(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1})$; we say that such a value $\ell(B)$ is the *size-estimate* for B . Next, we guess an assignment of all jobs in J_K and a subset of the jobs in J_{K-1} to the bags \mathcal{B}_K^* and an assignment of the remaining jobs in J_{K-1} and of a subset of the jobs in J_{K-2} to the bags \mathcal{B}_{K-1}^* . Finally, we guess $m_{\max}^{(K)}$ which we define to be the largest value $m \in M$ for which $\text{OPT}(\mathcal{B}^*, m) \in I_K$.

Evaluation phase. In contrast to the previous section, maximizing the minimum machine load asks for “covering” a machine or, in our case, a bag. To this end, we potentially have to assign jobs from $\bigcup_{k'=1}^{K-2} J_{k'}$ to the bags in \mathcal{B}_K^* . Formally, for $B \in \mathcal{B}_K^*$ let $p^+(B)$ be the total size of the jobs from J_K and J_{K-1} already assigned to B . We define $S := \sum_{B \in \mathcal{B}_K^*} \max\{[(1 + \varepsilon)^{\ell(B)}] - p^+(B), 0\}$. Our DP also stores this value in order to guarantee that, in the remainder, jobs from $\bigcup_{k'=1}^{K-2} J_{k'}$ with total size S are assigned to bags in \mathcal{B}_K^* . Let $J_{K-1}(\mathcal{B}_{K-1}^*)$ and $J_{K-2}(\mathcal{B}_{K-1}^*)$ be the subsets of J_{K-1} and J_{K-2} already assigned to bags in \mathcal{B}_{K-1}^* . Then, $\bar{S} := \sum_{B \in \mathcal{B}_{K-1}^*} [(1 + \varepsilon)^{\ell(B)}] - p(J_{K-1}(\mathcal{B}_{K-1}^*) \cup J_{K-2}(\mathcal{B}_{K-1}^*))$ is the total volume of bags in \mathcal{B}_{K-1}^* that needs to be covered with jobs from $\bigcup_{k'=1}^{K-3} J_{k'}$.

For evaluating our current guess, we fix some $m \leq m_{\max}^{(K)}$ and create a set J_T of dummy jobs, each with processing time 1 and total size $T := \sum_{k=1}^{K-2} \sum_{j \in J_k} p_j - S - \bar{S}$. Now, we guess the assignment of the bags $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ to the machines. Based on the load guaranteed by these bags, we now greedily distribute these dummy jobs as follows. Assume w.l.o.g. that the machines are sorted non-decreasingly by their loads and consider the prefix of the machines which all have the smallest load. We assign to each of these machines the same

number of dummy jobs such that their new load is equal to the load of the machines with the second smallest load. We repeat this procedure until all dummy jobs in J_T are assigned. At the end, among all possibilities to assign the bags $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$, we choose the one which maximizes the minimum machine load after the distribution of J_T . We define $\text{ALG}(m)$ as the load of the least loaded machine for this fixed candidate solution.

Among all guesses with the same set of bag sizes for bags in $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$, the same value $m_{\max}^{(K)}$ and the same values S and \bar{S} , we keep the guess which maximizes our proxy for the (partial) objective function, $\sum_{m=1}^{m_{\max}^{(K)}} q_m \cdot \text{ALG}(m)$.

4.1.2 Analysis

Observing that $|\mathcal{B}_K^*| \leq M \leq n$ and $|\mathcal{B}_{K-1}^*| \leq M \leq n$ implies that we can enumerate all possible combinations in time $O(n^2)$. Since the relative length, i.e., the ratio of the left interval border to the right interval border, of $I_K \cup I_{K-1}$ is bounded by $(\frac{1}{\varepsilon})^6$, there are at most $\mathcal{O}_\varepsilon(1)$ possibilities for each size-estimate $\ell(B)$. By guessing the number of bags with a given size estimate, we can guess the size-estimates of all bags in $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ in time $n^{\mathcal{O}_\varepsilon(1)}$. Further, each bag in \mathcal{B}_K^* can be assigned at most $\mathcal{O}_\varepsilon(1)$ many jobs from $J_K \cup J_{K-1}$ and, similarly, each bag in \mathcal{B}_{K-1}^* can be assigned at most $\mathcal{O}_\varepsilon(1)$ many jobs from $J_{K-1} \cup J_{K-2}$. Hence, there is only a constant number of possible assignments for each bag, up to permutations of jobs with the same size. We formalize these observations in the next lemma.¹

► **Lemma 10.** *In time $n^{\mathcal{O}_\varepsilon(1)}$, we can guess the size-estimate $\ell(B)$ for each bag $B \in \mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ as well as the assignment of the jobs in J_K to the bags \mathcal{B}_K^* , of the jobs in J_{K-1} to the bags $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ and of a subset of jobs in J_{K-2} to the bags in \mathcal{B}_{K-1}^* , up to a permutation of bags.*

First, observe that for each bag $B \in \mathcal{B}_K^*$, the value $\max\{[(1+\varepsilon)^{\ell(B)}] - p^+(B), 0\} \in \mathbb{N}_0$ since $p_j \in \mathbb{N}$ for each $j \in J$ by Lemma 9. Hence, S accurately captures the total volume missing to ensure that each $B \in \mathcal{B}_K^*$ packs jobs with a total size of at least $(1+\varepsilon)^{\ell(B)} \geq \frac{p(B)}{1+\varepsilon}$. Using that each job in $\bigcup_{k=1}^{K-2} J_k$ is very small compared to a bag in \mathcal{B}_K^* , we can argue that knowing S is actually sufficient to cover $B \in \mathcal{B}_K^*$ with jobs of total size of at least $\frac{p(B)}{(1+\varepsilon)^2}$.

Similarly, for a bag in \mathcal{B}_{K-1}^* , each job in $\bigcup_{k=1}^{K-3} J_k$ is very small compared to its size. Hence, we can again argue that knowing $\bar{S} \in \mathbb{N}_0$ is sufficient to pack jobs of total size at least $\frac{p(B)}{(1+\varepsilon)^2}$ into bag $B \in \mathcal{B}_{K-1}^*$.

Observing that no bag in $\bigcup_{k=1}^{K-2} \mathcal{B}_k^*$ can pack a job from $J_K \cup J_{K-1}$ by definition of their sizes, we conclude that T indeed represents the total volume of bags in $\bigcup_{k=1}^{K-2} \mathcal{B}_k^*$. In fact, we can show that for scenarios with $m \leq m_{\max}^{(K)}$ machines *any* assignment of the remaining jobs in $\bigcup_{k'=1}^{K-2} J_{k'}$ of total volume at most $\frac{T}{1+\varepsilon}$ to at most $M - |\mathcal{B}_K^*| - |\mathcal{B}_{K-1}^*|$ bags of size at most $\varepsilon(\frac{1}{\varepsilon})^{3K}$ yields the same objective function value (up to a factor of $1 + \mathcal{O}(\varepsilon)$). These observations are formalized in the next lemma where some jobs are set aside in bags B_S and $B_{\bar{S}}$, corresponding to the values S and \bar{S} .

Recall that we use \hat{X} to denote a possible guess for parameter X considered by our algorithm.

► **Lemma 11.** *Let the guessed quantities be as defined above. Let $\mathcal{B}' \cup \{B_S, B_{\bar{S}}\}$ be a partition of the jobs $\bigcup_{k'=1}^{K-2} J_{k'}$ into $M - |\hat{\mathcal{B}}_K| - |\hat{\mathcal{B}}_{K-1}| + 2$ bags such that*

- *for each bag $B \in \mathcal{B}'$, $p(B) \leq \varepsilon \cdot (\frac{1}{\varepsilon})^{3K}$,*

¹ For the initial guesses, one could give tighter bounds by observing that $|\mathcal{B}_K^*| + |\mathcal{B}_{K-1}^*| = \mathcal{O}_\varepsilon(1)$. However, we give polynomial bounds which are sufficient and of the same kind as the bounds we will use later in the DP.

- $p(B_S) \geq S$,
 - $p(B_{\bar{S}}) \geq \bar{S}$, and
 - $p(\mathcal{B}') := \sum_{B \in \mathcal{B}'} p(B) \geq (1 + \varepsilon)^{-1} T$.
- Suppose that $B \in \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}$ has size in $[(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1}]$. We can compute the vector $(ALG(m))_{m=1}^{m_{\max}^{(K)}}$ in polynomial time and $OPT(\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}, m) \in [(1 + \varepsilon)^{-5} ALG(m), (1 + \varepsilon) ALG(m)]$ for each $m \leq m_{\max}^{(K)}$.

Note that the lemma does not state anything about the relationship of $OPT(\mathcal{B}^*, m)$ and $OPT(\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}, m)$; it relates our proxy function $ALG(m)$ and $OPT(\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}, m)$, the best possible assignment for $\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}$.

In the remaining problem it suffices to focus on scenarios in which we have $m > m_{\max}^{(K)}$ machines and which hence satisfy $OPT(\mathcal{B}^*, m) < (\frac{1}{\varepsilon})^{3K}$. Note that for each bag $B \in \mathcal{B}_K^*$ we have that $p(B) \geq (\frac{1}{\varepsilon})^{3K}$. Therefore, if we are given $m > m_{\max}^{(K)}$ machines, it is optimal to assign each bag $B \in \mathcal{B}_K^*$ to a separate machine without any further bags assigned to that machine. Hence, if $m > m_{\max}^{(K)}$, then the bags in $\mathcal{B}_1^*, \dots, \mathcal{B}_{K-1}^*$ need to ensure only that the remaining $m - |\mathcal{B}_K^*|$ machines get enough load. This insight and the above lemma allow us to decouple our decisions for scenarios with $m \leq m_{\max}^{(K)}$ machines from scenarios with $m > m_{\max}^{(K)}$ machines. This is the key idea for our DP.

4.2 Dynamic program

After our initial guesses above, it remains to

- pack the jobs in $\bigcup_{k'=1}^{K-1} J_{k'}$ into the bags in \mathcal{B}_{K-1}^* ,
- compute the bag sizes in $\mathcal{B}_1^*, \dots, \mathcal{B}_{K-2}^*$, and
- select jobs from $\bigcup_{k=1}^{K-2} J_k$ with total size at least S for filling \mathcal{B}_K^* .

For each $m \geq m_{\min}^{(K-1)} := m_{\max}^{(K)} + 1$, our goal is to obtain a value close enough to $OPT(\mathcal{B}^*, m)$ so that, overall, we achieve a value of $(1 - \mathcal{O}(\varepsilon))OPT$.

To this end, we design a dynamic program (DP) that solves the remaining problem from above. Each DP cell corresponds to some subproblem. We show that for each possible guess of the initial quantities in Section 4.1 there is a DP cell corresponding to the remaining subproblem. In order to solve each subproblem, we guess similar quantities as in the previous section and reduce the resulting remaining problem to the subproblem of another DP cell.

4.2.1 Algorithm

Following the same idea as for the root subproblem, our dynamic program proceeds as follows: for each DP cell we first *guess* key quantities defining a partial solution as well as the transition to the next DP cell and then we *evaluate* this guessed partial solution.

DP cell and its subproblem. Formally, each DP cell \mathcal{C} is specified by

- $k \in \mathbb{N}$ with $k < K$ specifying that we still need to define $\mathcal{B}_1, \dots, \mathcal{B}_k$,
- $M_{k+1}, \dots, M_K \in \mathbb{N}$ counting the previously defined (large) bags $\mathcal{B}_{k+1}, \dots, \mathcal{B}_K$,
- $M_k \in \mathbb{N}$ representing our decision $|\mathcal{B}_k| = M_k$,
- $m_{\min}^{(k)} \in \mathbb{N}$ indicating the minimal number of machines we consider,
- $s_\ell \in \mathbb{N}$ for $\ell \in L_k$ counting $B \in \mathcal{B}_k$ with $\ell(B) = \ell$,
- $a_\ell \in \mathbb{N}$ for $\ell \in L_k$ counting the jobs j with $p_j = \lceil (1 + \varepsilon)^\ell \rceil$ that are assigned to bags in \mathcal{B}_{k+1} ,

14:12 Scheduling on a Stochastic Number of Machines

- $S \in \mathbb{N}$, defining the total size of jobs in $\bigcup_{k'=1}^k J_{k'}$ that must not be assigned to bags $\bigcup_{k'=1}^k \mathcal{B}_{k'}$ and that are neither assigned to bags in \mathcal{B}_{k+1} via the values a_ℓ ; instead they will be assigned to $\bigcup_{k'=k+1}^K \mathcal{B}_{k'}$. (Note that we will make sure that even though jobs from J_k might contribute to S , such jobs will not be used to cover bags in \mathcal{B}_{k+1} .)

The goal of the subproblem of \mathcal{C} is to pack a subset of the jobs $\bigcup_{k'=1}^k J_{k'}$ into the bags $\bigcup_{k'=1}^k \mathcal{B}_{k'}$ and define a size-estimate $\ell(B)$ for $B \in \bigcup_{k'=1}^k \mathcal{B}_{k'}$ such that

- $|\mathcal{B}_k| = M_k$,
- $p(B) \in I_{k'}$ for each $k' \in [k]$ and each $B \in \mathcal{B}_{k'}$,
- $p(B) \in [(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1}]$ for each $B \in \bigcup_{k'=1}^k \mathcal{B}_{k'}$,
- there are s_ℓ bags $B \in \mathcal{B}_k$ with $\ell(B) = \ell$ for each $\ell \in L_k$,
- each job $j \in J_{k'}$ for $k' \in [k]$ is either assigned to some bag in $\mathcal{B}_{k'} \cup \dots \cup \mathcal{B}_k$ or not at all,
- there are a_ℓ jobs j with $p_j = \lceil (1 + \varepsilon)^\ell \rceil$ that are not assigned to any bag for each $\ell \in L_k$,
- the jobs in $\bigcup_{k'=1}^k J_{k'}$ not packed in any bag have total size at least S .

For each DP cell \mathcal{C} , we compute a solution and a corresponding objective function value which we denote by $\text{profit}(\mathcal{C})$. This objective function corresponds to the expected profit from scenarios in $\{m_{\min}^{(k)}, \dots, M\}$ that we achieve with the solution stored in the DP cell and M_{k+1}, \dots, M_K “large” bags, i.e., bags B with $p(B) \in \bigcup_{k'=k+1}^K I_{k'}$.

Guessing phase. By definition of the DP cell, $|\mathcal{B}_k^*| = M_k$. For each $\ell \in L_k$, there are s_ℓ many bags $B \in \mathcal{B}_k^*$ with $\ell(B) = \ell$ (and hence with $p(B) \in [(1 + \varepsilon)^\ell, (1 + \varepsilon)^{\ell+1}]$). We start by guessing the assignment of the jobs $J_{k-1} \cup J_k$ to the bags in \mathcal{B}_k^* . We only consider guesses satisfying the values a_ℓ and S of our current DP cell, i.e., for every possible processing time in I_k enough jobs are left to be assigned to \mathcal{B}_{k+1}^* and enough total volume of jobs in $\bigcup_{k'=1}^k J_{k'}$ is left to be assigned to bags in $\bigcup_{k'=k+1}^K \mathcal{B}_{k'}$. Finally, we guess $m_{\max}^{(k)}$ which is the largest value m for which $\text{OPT}(\mathcal{B}^*, m) \in I_k$.

Evaluation phase. In order to calculate the proxy objective function value $\text{profit}(\mathcal{C})$, we need to combine \mathcal{C} with a cell $\hat{\mathcal{C}}$ corresponding to a DP cell for the remaining problem. To this end, let us define the parameters of this cell $\hat{\mathcal{C}}$. Clearly, we only need to define $\mathcal{B}_1, \dots, \mathcal{B}_{k-1}$. Hence, the first parameter of $\hat{\mathcal{C}}$ is $k - 1$. Further, the total number of previously defined bags is given by $\hat{M}_{k, \dots, K} = M_k + M_{k+1}, \dots, M_K$. As we do not ignore scenarios, we choose $m_{\min}^{(k-1)} := m_{\max}^{(k)} + 1$. Since we have already guessed the assignment of jobs in J_{k-1} to bags in \mathcal{B}_k^* , we can simply calculate the values \hat{a}_ℓ for $\ell \in L_{k-1}$ that indicate the number of jobs j with $p_j = \lceil (1 + \varepsilon)^\ell \rceil$ to be assigned to bags in \mathcal{B}_k^* .

It remains to calculate the value $\bar{S} \in \mathbb{N}$, the total size of jobs in $\bigcup_{k'=1}^{k-1} J_{k'}$ assigned as very small jobs to bags in $\bigcup_{k'=k}^K \mathcal{B}_{k'}$. To this end, we calculate the total size of jobs from $\bigcup_{k'=1}^{k-2} J_{k'}$ that need to be packed in \mathcal{B}_k^* . For each $B \in \mathcal{B}_k^*$, let $p^+(B)$ be the total size of the jobs from $J_{k-1} \cup J_k$ that B already packs. We define $S_k := \sum_{B \in \mathcal{B}_k^*} \max\{\lceil (1 + \varepsilon)^{\ell(B)} \rceil - p^+(B), 0\}$. Denote by $J_k(\mathcal{B}_k^* \cup \mathcal{B}_{k+1}^*)$ the set of jobs from J_k assigned to bags \mathcal{B}_k^* and \mathcal{B}_{k+1}^* . Then, \bar{S} is defined as $\bar{S} := S - \sum_{j \in J_k \setminus J_k(\mathcal{B}_k^* \cup \mathcal{B}_{k+1}^*)} p_j + S_k$, where S is defined by the current DP cell \mathcal{C} . (Note that \bar{S} does not contain jobs from J_{k-1} to be assigned to \mathcal{B}_k^* as they are accounted for by \hat{a}_ℓ .)

Hence, the remaining problem corresponds to some DP cell $\hat{\mathcal{C}}$ satisfying

$$\hat{\mathcal{C}} = \left(k - 1, M_{k+1}, \dots, M_K + M_k, \hat{M}_{k-1}, m_{\max}^{(k)} + 1, \hat{S} \geq \bar{S}, (\hat{s}_\ell)_{\ell \in L_{k-1}}, (\hat{a}_\ell)_{\ell \in L_{k-1}} \right), \quad (1)$$

where \hat{s}_ℓ is a possible number of bags with size-estimate $\ell \in L_{k-1}$, i.e., with size $(1 + \varepsilon)^\ell$, the number of bags $|\mathcal{B}_{k-1}^*|$ is given by $\hat{M}_{k-1} = \sum_{\ell \in L_{k-1}} \hat{s}_\ell$, and we require that \hat{S} is at least \bar{S} .

Given $\text{profit}(\hat{\mathcal{C}})$ for some $\hat{\mathcal{C}}$, we can now calculate $\text{profit}(\mathcal{C})$ as follows: For each value $m \in \{m_{\min}^{(k)}, \dots, m_{\max}^{(k)}\}$, we compute an estimate $\text{ALG}(m)$ of the objective value of an optimal bag-to-machines assignment of $\bigcup_{k'=1}^k \hat{\mathcal{B}}_{k'}$ and M_{k+1}, \dots, K large bags to m machines. To this end, we use a variant of Lemma 11, which is explained in detail in the full version. Then, the profit of the candidate combination of \mathcal{C} with $\hat{\mathcal{C}}$ is given by $\sum_{m=m_{\min}^{(k)}}^{m_{\max}^{(k)}} q_m \text{ALG}(m) + \text{profit}(\hat{\mathcal{C}})$. Among all these candidate combinations, we choose the one with the largest profit and set $\text{profit}(\mathcal{C}) = \sum_{m=m_{\min}^{(k)}}^{m_{\max}^{(k)}} q_m \text{ALG}(m) + \text{profit}(\hat{\mathcal{C}})$.

4.2.2 Analysis

Observe that there are at most $\mathcal{O}_\varepsilon(1)$ many distinct processing times of jobs $J_{k-1} \cup J_k$ and each bag $B \in \mathcal{B}_k^*$ contains at most $\mathcal{O}_\varepsilon(1)$ many jobs from each of these processing times because of the definition of \mathcal{B}_k^* , J_{k-1} , and J_k .

We guess all possible assignments of jobs to a single bag of size at most $(\frac{1}{\varepsilon})^{3k+3}$; typically such an assignment is called a *configuration*. There are at most $\mathcal{O}_\varepsilon(1)$ such configurations. Then, for each configuration and each ℓ with $\lceil (1 + \varepsilon)^\ell \rceil \in I_k$, we guess how often the configuration is assigned to a bag B with $\ell(B) = \ell$. Following this sketch, the next lemma proves that we can in fact guess the job-to-bag assignment in polynomial time.

► **Lemma 12.** *In time $n^{\mathcal{O}_\varepsilon(1)}$ we can guess the assignment of jobs from J_{k-1} and J_k to the bags \mathcal{B}_k^* up to permuting jobs and bags.*

During the evaluation phase, we try all possible combinations of the current DP cell \mathcal{C} with $\hat{\mathcal{C}}$ satisfying (1), i.e., DP cells corresponding to the remaining subproblems matching the parameters of \mathcal{C} . We now give a proof sketch of why our guesses combined \mathcal{C} indeed give a feasible solution to the subproblem for k . Let $\hat{\mathcal{B}}_1, \dots, \hat{\mathcal{B}}_{k-1}$ be the bags given by the solution to $\hat{\mathcal{C}}$ and $\hat{\mathcal{B}}_k$ be the bags corresponding to our guess. (We do not change $\bigcup_{k'=1}^{k-1} \hat{\mathcal{B}}_{k'}$.)

We need to assign jobs from $\bigcup_{k'=1}^{k-1} J_{k'}$ to $\hat{\mathcal{B}}_k$ satisfying

- (i) for every bag $B \in \hat{\mathcal{B}}_k$, $p(B) \geq (1 + \varepsilon)^{\ell(B)-1}$ and
- (ii) the total processing time of
 - all jobs in $\bigcup_{k'=1}^{k-1} J_{k'}$ not assigned to bags in $\bigcup_{k'=1}^k \hat{\mathcal{B}}_{k'}$ and
 - all jobs in J_k neither assigned to $\hat{\mathcal{B}}_k$ nor reserved by the values a_ℓ for the bags with size in I_{k+1}
is at least S .

Each $B \in \hat{\mathcal{B}}_k$ has already jobs from J_k and J_{k-1} of total size $p^+(B)$ assigned to it. Let $p^-(B) := \max\{\lceil (1 + \varepsilon)^{\ell(B)} \rceil - p^+(B), 0\}$ be the missing volume in B to cover B to the desired level of $\lceil (1 + \varepsilon)^{\ell(B)} \rceil$. If $p^-(B) = 0$, no additional job from $\bigcup_{k'=1}^{k-2} J_{k'}$ needs to be assigned to B . Otherwise, we greedily add jobs from $\bigcup_{k'=1}^{k-2} J_{k'}$ not packed in $\bigcup_{k'=1}^{k-1} \hat{\mathcal{B}}_{k'}$ until assigning the next job would exceed $p^-(B)$. Hence, the total size of jobs $\bigcup_{k'=1}^{k-2} J_{k'}$ assigned to B by this routine is at least $p^-(B) - (\frac{1}{\varepsilon})^{3k-6}$. Thus,

$$p(B) \geq p^+(B) + p^-(B) - \left(\frac{1}{\varepsilon}\right)^{3k-6} = (1 + \varepsilon)^{\ell(B)} - \left(\frac{1}{\varepsilon}\right)^{3k-6} \geq (1 + \varepsilon)^{-1} (1 + \varepsilon)^{\ell(B)}$$

since by definition $(1 + \varepsilon)^\ell \geq \frac{1}{\varepsilon} \cdot (\frac{1}{\varepsilon})^{3k-6}$ for all $\ell \in L_k$.

By choice of $\hat{\mathcal{C}}$, the total size \hat{S} of jobs in $\bigcup_{k'=1}^{k-1} J_{k'}$ neither assigned to bags in $\bigcup_{k'=1}^{k-1} \hat{\mathcal{B}}_{k'}$ nor reserved for \mathcal{B}_k via the values $(\hat{a}_\ell)_{\ell \in L_{k-1}}$ is at least \bar{S} . With the definition of \bar{S} , we get

$$\hat{S} \geq \bar{S} = S - \sum_{j \in J_k \setminus J_k(\hat{\mathcal{B}}_k \cup \hat{\mathcal{B}}_{k+1})} p_j + S_k = S - \sum_{j \in J_k \setminus J_k(\hat{\mathcal{B}}_k \cup \hat{\mathcal{B}}_{k+1})} p_j + \sum_{B \in \hat{\mathcal{B}}_k} p^-(B).$$

We remark that the second term indeed corresponds to the contribution of jobs from J_k to filling bags with sizes in $\bigcup_{k'=k+1}^K J_{k'}$ since such jobs cannot be packed into $\bigcup_{k'=1}^{k-1} \hat{B}_k$ by definition of the corresponding sizes. Observe that the greedy procedure described above assigns jobs from $\bigcup_{k'=1}^{k-2} J_{k'}$ with total volume *at most* $p^-(B)$ to $B \in \hat{B}_k$. Hence, the combination of our guess \hat{B}_k (and the guessed partial assignment of $J_{k-1} \cup J_k$ to \hat{B}_k) and the solution for \hat{C} is indeed a feasible solution for \mathcal{C} .

Similar to the proof of Lemma 11, we show that for any candidate solution (consisting of a guess \hat{B}_k , a partial assignment of $J_k \cup J_{k-1}$, and any solution for \hat{C} as defined in (1)), we can calculate the values $\text{ALG}(m)$ for $m \in \{m_{\min}^{(k)}, \dots, m_{\max}^{(k)}\}$ in polynomial time such that $\text{ALG}(m)$ is within a factor $(1 + \mathcal{O}(\varepsilon))$ of the optimal assignment given the same set of bags. This is formalized in the next lemma.

► **Lemma 13.** *Let $\hat{B}_k, m_{\max}^{(k)}$ and the job-to-bag assignment of $J_k \cup J_{k-1}$ to \hat{B}_k be guesses as defined. Further, let \hat{C} satisfy (1) and suppose that the bag sizes in \hat{B}_{k-1} are given by $(\hat{s}_\ell)_{\ell \in L_{k-1}}$. Let $\mathcal{B}' \cup \{B_{\hat{s}}\}$ be a partition of the jobs $\bigcup_{k'=1}^{k-2} J_{k'}$ into $M - M_{k+1, \dots, K} - |\hat{B}_k| - |\hat{B}_{k-1}| + 1$ bags such that*

- for each bag $B \in \mathcal{B}'$ we have that $p(B) \leq \varepsilon \cdot \left(\frac{1}{\varepsilon}\right)^{3k}$,
- $p(B_{\hat{s}}) \geq \hat{S}$,
- $p(\mathcal{B}') := \sum_{B \in \mathcal{B}'} p(B) \geq (1 + \varepsilon)^{-1} T$.

Suppose that each bag $B \in \hat{B}_k \cup \hat{B}_{k-1}$ satisfies $p(B) \in [(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1}]$ and let \hat{B}_L contain $M_{k+1, \dots, K}$ many large bags of size at least $\left(\frac{1}{\varepsilon}\right)^{3k+3}$. There is a polynomial-time algorithm that either asserts that our guess is incorrect and cannot be combined with \hat{C} or that computes a vector $(\text{ALG}(m))_{m=m_{\min}^{(k)}}^{m_{\max}^{(k)}}$ such that $\text{OPT}(\mathcal{B}_L \cup \hat{B}_k \cup \hat{B}_{k-1} \cup \mathcal{B}', m) \in [(1 + \varepsilon)^{-5} \text{ALG}(m), (1 + \varepsilon) \text{ALG}(m)]$ holds for each $m \in \{m_{\min}^{(k)}, \dots, m_{\max}^{(k)}\}$ for which $\text{OPT}(\mathcal{B}_L \cup \hat{B}_k \cup \hat{B}_{k-1} \cup \mathcal{B}', m) \geq (1 + \varepsilon)^{-1} \left(\frac{1}{\varepsilon}\right)^{3k}$.

Further, we can find the best \hat{C} that satisfies (1) and can be combined with our guess in polynomial time.

To compute the final solution, we combine the correct initial guesses with the solution stored in the DP cell corresponding to the remaining subproblem. This yields a global solution to the original problem. In order to prove the correctness of our DP, we observe that for each $k \in [K - 1]$ there is a *special* DP cell for which k is the first parameter and whose other parameters correspond to the optimal solution (e.g., the assignment of jobs in J_k to bags in \mathcal{B}_{k+1}^*). We then prove by induction that, for each $k \in [K - 1]$, the solution stored in the corresponding special DP cell yields a profit that is similar to the optimal profit restricted to scenarios with $m \in \{1, \dots, m_{\max}^{(k)}\}$ machines, using Lemma 13.

5 Conclusion

In this paper, we continue the recent line of research on scheduling with uncertainty in the machine environment [1, 5, 20] by considering a stochastic machine environment in which the number of identical parallel machines is only known in terms of a distribution and the actual number is revealed once the jobs are assigned to bags which cannot be split anymore. Interestingly, we present polynomial time approximation schemes for minimizing the makespan as well as maximizing the minimum machine load, which matches their respective deterministic counterparts from the perspective of approximation algorithms. We believe that our insights open up many interesting questions for future research such as extending the current model to the setting with uniformly related machines in which the uncertainty is modeled in terms of machine speeds as done in [5] from a robustness point-of-view or to the setting with different (job-based) objectives such as sum of weighted completion times.

References

- 1 Eric Balkanski, Tingting Ou, Clifford Stein, and Hao-Ting Wei. Scheduling with speed predictions. In *International Workshop on Approximation and Online Algorithms (WAOA)*, pages 74–89, 2023.
- 2 Anindya De, Sanjeev Khanna, Huan Li, and Hesam Nikpey. An efficient PTAS for stochastic load balancing with poisson jobs. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 37:1–37:18, 2020.
- 3 Florian Diedrich, Klaus Jansen, Ulrich M. Schwarz, and Denis Trystram. A survey on approximation algorithms for scheduling with machine unavailability. In *Algorithmics of Large and Complex Networks*, pages 50–64, 2009.
- 4 Franziska Eberle, Anupam Gupta, Nicole Megow, Benjamin Moseley, and Rudy Zhou. Configuration balancing for stochastic requests. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 127–141, 2023.
- 5 Franziska Eberle, Ruben Hoeksma, Nicole Megow, Lukas Nölke, Kevin Schewior, and Bertrand Simon. Speed-robust scheduling: sand, bricks, and rocks. *Math. Program.*, 197(2):1009–1048, 2023.
- 6 Leah Epstein and Rob van Stee. Maximizing the minimum load for selfish agents. *Theor. Comput. Sci.*, 411(1):44–57, 2010.
- 7 Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975.
- 8 Michael R. Garey and David S. Johnson. “Strong” NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.
- 9 Anupam Gupta, Amit Kumar, Viswanath Nagarajan, and Xiangkun Shen. Stochastic load balancing on unrelated machines. *Math. Oper. Res.*, 46(1):115–133, 2021.
- 10 Anupam Gupta, Benjamin Moseley, and Rudy Zhou. Minimizing completion times for stochastic jobs via batched free times. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1905–1930, 2023.
- 11 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.
- 12 Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988. doi:10.1137/0217033.
- 13 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 81:1–81:20, 2021.
- 14 Sungjin Im, Benjamin Moseley, and Kirk Pruhs. Stochastic scheduling of heavy-tailed jobs. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 474–486, 2015.
- 15 Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217, 2000.
- 16 Sartaj K. Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, 1976.
- 17 Jay Sethuraman. Stochastic scheduling. In *Encyclopedia of Algorithms*, pages 2110–2113. Springer, 2016.
- 18 Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Math. Oper. Res.*, 41(3):851–864, 2016.
- 19 Wolfgang Stadje. Selecting jobs for scheduling on a machine subject to failure. *Discret. Appl. Math.*, 63(3):257–265, 1995.
- 20 Clifford Stein and Mingxian Zhong. Scheduling when you do not know the number of machines. *ACM Trans. Algorithms*, 16(1):9:1–9:20, 2020.
- 21 Gerhard J. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Oper. Res. Lett.*, 20(4):149–154, 1997.