Efficient Approximation Schemes for Scheduling on a Stochastic Number of Machines

Leah Epstein
Department of Mathematics, University of Haifa, Israel

Asaf Levin
Faculty of Data and Decisions Science, The Technion, Haifa, Israel

Abstract

We study three two-stage optimization problems with a similar structure and different objectives. In the first stage of each problem, the goal is to assign input jobs of positive sizes to unsplittable bags. After this assignment is decided, the realization of the number of identical machines that will be available is revealed. Then, in the second stage, the bags are assigned to machines. The probability vector of the number of machines in the second stage is known to the algorithm as part of the input before making the decisions of the first stage. Thus, the vector of machine completion times is a random variable. The goal of the first problem is to minimize the expected value of the makespan of the second stage schedule, while the goal of the second problem is to maximize the expected value of the minimum completion time of the machines in the second stage solution. The goal of the third problem is to minimize the $\ell_\mathfrak{p}$ norm for a fixed $\mathfrak{p} > 1$, where the norm is applied on machines' completion times vectors. Each one of the first two problems admits a PTAS as Buchem et al. showed recently. Here we significantly improve all their results by designing an EPTAS for each one of these problems. We also design an EPTAS for $\ell_\mathfrak{p}$ norm minimization for any $\mathfrak{p} > 1$.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Approximation algorithms, Approximation schemes, Two-stage stochastic optimization problems, Multiprocessor scheduling

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.31

Related Version *Full Version*: `https://arxiv.org/abs/2409.10155`

Funding *Asaf Levin*: Partially supported by ISF – Israel Science Foundation grant number 1467/22.

1 Introduction

We consider scheduling problems where the goal is to assign jobs non-preemptively to a set of identical machines. Unlike traditional scheduling problems [22], the number of identical machines available to the scheduler, denoted as k, is not a part of the input, but it is drawn from a known probability distribution on the set of integers $\{1, 2, \ldots, m\}$ for an integer $m \geq 2$ that is a part of the input, and it is given together with the probabilities. As a first stage, the decision maker completes an initial set of decisions, namely it assigns the jobs to m bags, forming a partition of all input jobs. Later, once the realization of the number of machines becomes known, it packs the bags to the machines, where the packing of a bag to a machine means that all its jobs are scheduled together to this machine. That is, in this later step, every pair of jobs that were assigned to a common bag will be assigned to a common machine, and the decision of the first stage (that is, the partition of input jobs into m bags) is irrecoverable. This two-stage stochastic scheduling problem was recently introduced by Buchem et al. [9].

Formally, the input consists of a set of jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ where each job $j \in \mathcal{J}$ has a positive rational size p_j associated with it. We are given an integer $m \geq 2$ and let $\mathcal{B} = \{1, 2, \ldots, m\}$ denote the set of bags. We are also given a probability measure q over $\mathcal{B}$, where q_k is a rational non-negative number denoting the probability that the number of identical machines in the resulting second stage instance is k. Here, we will use the property that $\sum_{k=1}^{m} q_k = 1$. In the first stage of our problem, the jobs are split into m bags by an algorithm. Namely, a feasible solution of the first stage is a function $\sigma_1 : \mathcal{J} \to \mathcal{B}$. In the second stage, after the value of k has been revealed, the bags are assigned to machines, such that all jobs of each bag are assigned together. Namely, the algorithm also computes assignment functions σ_2^k defined for every realization k in the support of q of the bags to the set of integers $\{1, 2, \ldots, k\}$ denoting the indexes of machines in the instance of the second stage problem (corresponding to the realization of q that is equal to k). So if the realization of q is k, a job $j \in \mathcal{J}$ will be assigned to the machine of index $\sigma_2^{(k)}(\sigma_1(j))$. The first function σ_1 maps jobs to bags, and the second function which is based on the value of k, σ_2^k, maps bags to machines. The number of bags is m (and σ_1 is independent of k, so the partition into bags does not depend on k which is not known yet at the time of assignment into bags), and the number of machines is k, so for every realization of k the schedule of the jobs to the k machines is a feasible (non-preemptive) schedule for k identical machines.

We use the terminology of such scheduling problems and let W_i be the work (also called load) of machine i which is the total size of jobs assigned to machine i in a schedule. This is also the completion time of a unit speed machine processing continuously the set of jobs assigned to machine i starting at time 0 (in some order). The *makespan of the schedule in realization* k is the maximum work of a machine in the second stage solution in the realization k of q, and the *Santa Claus value of the schedule of realization* k is the minimum work of a machine (among the k machines) in the second stage solution in the realization k of q.

The expected value of the makespan is the expected value of the random variable of the makespan of the schedule in realization k. The expectation is computed based on all possible values of k. The first problem that we consider is the problem of minimizing the expected value of this random variable. We denote it as PR-MAKESPAN and refer to it as the *makespan minimization problem.* The expected value of the Santa Claus value is the expected value of the random variable of the Santa Claus value of the schedule in realization k. The second problem we consider is the problem of maximizing the expected value of this random variable. We denote it as PR-SANTACLAUS and refer to it as the *Santa Claus maximization problem.* While the term makespan is used in the scheduling literature in this meaning the term Santa Claus is not traditional, and it is usually referred to as the minimum work of a machine in the schedule. Given the length of the resulting terminology for our settings, we prefer to use the non-traditional terminology of Santa Claus value. The expected value of the random variable is called cost or value for minimization problems, and it is called profit or value for maximization problems. In the $\ell_{\mathfrak{p}}$ norm minimization problem, the objective for k machines is $(\sum_{i=1}^{k} W_i^{\mathfrak{p}})^{1/\mathfrak{p}}$. The cost of a solution of the third problem is the expected cost based on the random variable. This last problem is denoted by PR-NORM.

Since the stochastic nature of the problem results only from the different options for k (and once k is known, the problem is deterministic), we say that if the realization of q is k, then this is the scenario of index k or simply scenario k. We let OPT_k denote the objective function value of the second stage problem in scenario k for an optimal solution for the studied problem, and let $\text{OPT} = \sum_{k=1}^{m} q_k \cdot \text{OPT}_k$ denote the value of an optimal solution to our problem. An optimal solution needs to balance all scenarios, and we denote this optimal solution by OPT as well (that is, we use the same notation as the cost or the value). We stress that OPT_k is not necessarily the optimal objective function value for the input jobs and k machines, since the solution with a fixed set of bags may be inferior. We will assume that $\frac{1}{\varepsilon}$ is a positive integer such that $\frac{1}{\varepsilon} > 100$ (this can be done without loss of generality as one can first decrease ε to the minimum between its original value and $\frac{1}{100}$, and then it is always possible to decrease the value of ε by a factor below 2 to satisfy the integrality condition on $\frac{1}{\varepsilon}$). The assumptions on ε will be used for all schemes (and in particular we will use $\varepsilon < \frac{1}{100}$). Let $p_{\max} = \max_{j \in \mathcal{J}} p_j$.

An overview of our results. Here, we study both minimization problems and a maximization problem. All types of algorithms defined here are required to have polynomial running times (and our algorithms will in fact have strongly polynomial running times). For a minimization problem, an $\mathcal{R}$-approximation algorithm is an algorithm that always finds a solution that is feasible and has a cost at most $\mathcal{R}$ times the cost of an optimal solution. For a maximization problem, an $\mathcal{R}$-approximation algorithm is an algorithm that always finds a feasible solution of value at least $\frac{1}{\mathcal{R}}$ times the value of an optimal solution. The approximation ratio of a given algorithm is the infimum value of the parameter $\mathcal{R}$ such that this algorithm is an $\mathcal{R}$-approximation algorithm.

A PTAS is a class of approximation algorithms such that for any $\varepsilon > 0$ the class has a $(1+\varepsilon)$-approximation algorithm. An efficient polynomial time approximation scheme (EPTAS) is a stronger concept. This is a PTAS whose running time is of the form $h(\varepsilon) \cdot poly(N)$ where h is some (computable but not necessarily polynomial) function and $poly(N)$ is a polynomial function of the length N of the (binary) encoding of the input. We will justify the assumption $m \leq n$ later, and therefore the running time can be rewritten in this form when the polynomial is on m and n. A fully polynomial time approximation scheme (FPTAS) is an even stronger concept, defined like an EPTAS, but the function h must be a polynomial in $\frac{1}{\varepsilon}$. In this paper, we are interested in EPTAS's.

A PTAS may have time complexity of the form $n^{g(\varepsilon)}$, where g can be any function of $\frac{1}{\varepsilon}$ and even a power tower function. However, an EPTAS cannot have such a running time, which makes it more efficient. The concept of an EPTAS is related to fixed parameter tractable (FPT) algorithms [16].

In this paper, we focus on finding EPTAS's for the three objectives. We develop an EPTAS for the makespan minimization problem in Section 2. Then, in Section 3 we establish an EPTAS for the Santa Claus maximization problem PR-SANTACLAUS. Last, in the full version of this work we modify our scheme for PR-MAKESPAN in order to obtain an EPTAS for PR-NORM. Due to space limitations, some of the proofs of the first two results are given in the full version, and the last EPTAS for PR-NORM is presented in the full version as well. In the recent work [9], a PTAS was designed for PR-MAKESPAN, and another PTAS was designed for PR-SANTACLAUS. Those PTAS's are of forms that do not allow a simple modification of the PTAS into an EPTAS, and in particular for PR-SANTACLAUS a dynamic program over a state space whose size is exponential is used (a function of ε appears in the exponent of the input size), and for both schemes enumeration steps of such sizes are applied.

For all objectives, we assume that $n > m$ holds. If $n \leq m$, this means that every job can be assigned to its own bag. For each scenario, it is possible to apply an EPTAS for the corresponding problem (see for example [2], the details for previous work are discussed further below) and thereby get an EPTAS for the required problem.

The problems studied here generalize strongly NP-hard problems, and therefore one cannot expect to obtain an FPTAS (unless $P = NP$), and thus our results are the best possible. Specifically, the special case of each one of the three problems where the number of machines is known, i.e., the probability function has a single value q_m that is equal to 1, and other probabilities are equal to zero, is known to be NP-hard in the strong sense. The three objectives studied here are the three main objectives previously studied for scheduling on identical machines.

Related work. Stein and Zhong [33] introduced the scheduling problem with an unknown number of machines but in their model the number of machines is selected later by an adversary. The problem was mostly studied with respect to makespan minimization. In the deterministic variant [33], the goal is to assign a set of jobs with known properties to identical machines. However, only an upper bound m on the number of machines is given. Jobs have to be partitioned into m subsets, such that every subset will act as an inseparable bag. Then, the number of machines k (where $2 \leq k \leq m$) becomes known, and the schedule is created using the bags, without unpacking them, as in the problem that we study here, and after the number of machines is revealed, the bags are assigned to the machines, as in the problem that we study. Thus, this problem also has two steps or levels of computation, but the worst case out of all possible values of k is analyzed, where the comparison for each value of m is to an optimal offline solution for k identical machines and arbitrary bags. It is not hard to see that a constant approximation ratio (of at most 2) can be obtained using a round-robin approach even for jobs of arbitrary sizes (via pre-sorting). An improved upper bound of $\frac{5}{3} + \varepsilon$ was shown using a much more careful assignment to bags [33].

A variant where machines have speeds (similar to uniformly related machines) was defined and studied by [18] with respect to makespan. In that version, the number of machines m is known, but not their speeds. The number of required bags is equal to the number of machines, but some machines may have speed zero, so the case of identical machines and the makespan objective is seen as binary speeds in this work. There are several input types of interest, which are arbitrary jobs, unit size jobs, sand (one large job that can be cut into any required parts), and pebbles (jobs that have arbitrary sizes, but they are relatively small) [18, 30]. Tight bounds were proved for the case of sand and makespan with and without speeds [33, 18], and for the Santa Claus objective without speeds [33]. All these values are strictly smaller than 1.6. For sand, since any partition of the input jobs is allowed, linear programming can be used to find the best partition. In the case with speeds for arbitrary sizes of jobs the algorithm of [18] has an approximation ratio of at most $2 - \frac{1}{m}$, while special cases allow smaller ratios [18, 30].

In [5], Balkanski et al. relate the problem of scheduling with an unknown number of machines and an unknown set of speeds to online algorithms with predictions. In online problems with predictions [31], the algorithm receives information on the input, where such information could have been computed via machine learning. This model takes into account both the situation where the input matches the prediction exactly and the case where it does not. It is required for the algorithm to have a relatively good performance for every input, but the performance has to be better if the input was predicted correctly. In many algorithms the performance improves as the input gets closer to the predicted input. The work of [5] provides results of this flavor.

Optimizing the worst scenario is pessimistic and does not allow one to take information learned from previous data into account, while the study of the expected value as in our stochastic problem allows us to prefer the typical scenarios in the bag assignment algorithm. See also [4, 8, 32, 17] for related work.

As we mentioned, the most relevant work to our work on the stochastic problem is [9], where two PTAS's are provided. It is also mentioned in that work that FPTAS's for the case where m is seen as a constant can be designed using standard methods. The problem is called *stochastic* due to the probability distribution on scenarios. However, since this distribution is simply a convex combination of the costs for different scenarios, the methods are related to those often used for deterministic algorithms and worst case analysis. The convex combination complicates the problem and it requires carefully tailored methods for the algorithmic design. We follow the standard worst case studies of two-stage stochastic optimization problems, and we study the optimization problems of optimizing the expected value of the random variable.

All three objectives were studied initially for known numbers of identical machines, for which simple greedy approximation algorithms were presented first. Some time later PTAS's were designed. Finally EPTAS's were designed or it was observed that one of the known PTAS's is an EPTAS or can be converted into an EPTAS, which is the best possible result for each one of the three cases unless $P = NP$. We provide additional details for each one of the objectives. The makespan objective was introduced by Graham [22, 23], where greedy algorithms were studied (see also [13, 20]). Twenty years later a PTAS was designed [25]. Hochbaum [24] mentions that one of the approaches of assigning large jobs of rounded sizes (namely, solving an integer linear program in fixed dimension) gives an EPTAS, and attributes this approach to D. Shmoys (see [26, 11] for recent results). The Santa Claus objective (where the name was introduced much later [7]) was studied with respect to greedy heuristics [21, 15, 14]. A PTAS which is actually an EPTAS was designed by Woeginger [34]. The $\ell_{\mathfrak{p}}$ norm objective function was studied with respect to greedy heuristics [10, 12, 29], and a PTAS and an EPTAS were designed [1]. The same authors generalized their results for other objectives that satisfy natural conditions [2]. The $\ell_{\mathfrak{p}}$ norm of a vector is seen as a more suitable measure of fairness of a schedule compared to bottleneck measures [3, 6], and therefore we study this objective additionally to those studied in [9].

Our methods. We develop new techniques to address the uncertainty in the problems. We expect these techniques to be used in later work on approximating related variants.

We use rounding and discretization as in other work on scheduling, though we apply it not only on job sizes but also on bag sizes. The difficulty in relating new sizes to optimal solutions lies in the difference in objectives, since here the optimal value is not defined by a single schedule. However, we are still able to find such a relation using enumeration of sets of values, that is, via guessing. Since jobs are assigned to bags, we introduce configuration integer programs (IP's) on collections of bags, which can be seen as a generalization of previously known approaches. Our configuration IP's are based on the use of templates, where a template defines the contents of a bag, and configurations, where a configuration is an allocation of bags to one machine in a given scenario. We solve this IP using an algorithm for solving an IP in fixed dimension. All earlier steps of our schemes are highly motivated given the goal of creating such an IP and ensuring that it has a fixed dimension.

Machine numbers are split to intervals such that the most difficult interval for each guessed information is solved using an IP, while the others are solved using greedy approaches. The number of intervals for scenarios differs based on the specific problem.

While the sketched approach allows us to design an EPTAS for makespan minimization, the other objectives are harder to approximate. In order to tackle them, we develop an important tool called approximated histograms. Those are histograms of solution costs or values, and we use them in order to reduce the number of relevant scenarios to a constant. The condition for using the approximated histogram is a monotonicity assumption regarding the feasibility of solutions for different scenarios and the monotonicity of the costs for a given solution among scenarios in which it is feasible.

2 An EPTAS for the makespan minimization problem

We start with the makespan minimization problem, and continue to the other objectives in other sections. The first step will be to apply a discretization on bag sizes, and bound (from above) the set of relevant bag sizes as a function of OPT, where we enumerate possible values of OPT and use rounding for this value as well. Our second step is to round the instance of jobs (which will be assigned to bags). We use standard guessing steps and rounding methods for these steps. Then, we are going to approximate the rounded instance by using templates of the assignment of jobs to bags, and configurations of assigning templates to machines in every realization of q. This will allow us to formulate an integer linear program that has a special structure, namely, it is a *two-stage stochastic integer linear program.* This special structure, as well as trivial bounds on the parameters of this formulation, provides us with an FPT algorithm to solve the problem. This algorithm has a running time dominated by a function of $\frac{1}{\varepsilon}$ times a polynomial in n, and therefore it is an EPTAS.

We will be able to reduce the number of variables such that we can apply Lenstra's algorithm [28, 27] on the integer linear program, and the running time for solving this program will be smaller than that of solving a two-stage stochastic integer linear program. This is the case also for the other two EPTAS's that we show, that is, we will apply Lenstra's algorithm for all objectives. The most difficult step in the current section is to reduce the number of different values of makespan for different scenarios. Without this step, the running time will not satisfy the requirements of an EPTAS. The reduction is based on linear grouping, which is typically used for bin packing [19] and not for scheduling. The other EPTAS's that we design require additional non-trivial steps, and we will discuss them later.

The optimal solution of this integer program is transformed into a set of decisions defining the functions σ_1 and $\sigma_2^{(k)}$ $\forall k$. Next, we present the details of the scheme together with its analysis. In what follows we will present steps where in each of those steps the resulting approximation ratio of the scheme increases by at most a multiplicative factor of $1 + \Theta(\varepsilon)$. By scaling ε prior to applying the scheme, we get that the approximation ratio of the entire scheme will be at most $1 + \varepsilon$. We will use the next lemma for our proof.

▶ **Lemma 1.** *It holds that* $\textsc{opt}_k \in [p_{\max}, n\cdot p_{\max}]$ *for any* $k \le m$. *Thus,* $\textsc{opt} \in [p_{\max}, n\cdot p_{\max}]$.

Guessing opt. Our first step is to guess the value of OPT within a multiplicative factor of $1 + \varepsilon$. That is, we would like to enumerate a polynomial number of candidate values, where the enumerated set will contain (at least one) value that is in the interval $[\textsc{opt}, (1 + \varepsilon) \cdot \textsc{opt})$. We will show that for a guess in this interval we indeed find a solution of cost $(1+\Theta(\varepsilon)) \cdot \textsc{opt}$. The next lemma shows that it is possible to enumerate such a set of candidate values since $\log_{1+\varepsilon} n \le \frac{n}{\varepsilon}$. Our algorithm performs this enumeration.

▶ **Lemma 2.** *There is a set consisting of* $O(\log_{1+\varepsilon} n)$ *values such that this set of values has at least one value in the interval* $[\textsc{opt}, (1 + \varepsilon)\textsc{opt})$.

In what follows, with a slight abuse of notation, we let OPT be the value of this guessed candidate value (which is not smaller than OPT and it is larger by at most a factor of $1 + \varepsilon$ if the guess is correct). We will show that if there is a feasible solution to PR-MAKESPAN whose expected value of the cost is at most OPT, then we will construct a feasible solution with expected value of the cost being at most $(1 + \Theta(\varepsilon)) \cdot \textsc{opt}$. By the above lemma, this means that the problem admits an EPTAS as desired.

Rounding bag sizes. Now, we provide a method to gain structure on the set of feasible solutions that we need to consider for finding a near optimal solution. Given a feasible solution, the size of bag i denoted as $P(i)$ is the total size of jobs assigned to this bag, that is, $P(i) = \sum_{j:\sigma_1(j)=i} p_j$.

▶ **Lemma 3.** *There exists a solution of cost at most* $(1 + 3\varepsilon) \cdot \textsc{opt}$ *in which the size of every non-empty bag is in the interval* $[\varepsilon \cdot \textsc{opt}, (1 + \varepsilon) \cdot \textsc{opt}]$.

In what follows, we will increase the size of a bag to be the next value of the form $(\varepsilon + r\varepsilon^2) \cdot \text{OPT}$ for a non-negative integer r (except for empty bags for which the allowed size remains zero). Thus, we will not use precise sizes for bags but upper bounds on sizes, and we call them *allowed sizes*. We let $P'(i)$ be this increased (allowed) size of bag i, that is, $P'(i) = \min_{r:(\varepsilon+r\varepsilon^2)\text{OPT}\geq P(i)}(\varepsilon + r\varepsilon^2)\text{OPT}$. Since for every subset of bags, the total allowed size of the bags in the set is at most $1 + \varepsilon$ times the total size of the bags in the subset, we conclude that this rounding of the bag sizes will increase the cost of any solution by a multiplicative factor of at most $1 + \varepsilon$ (and therefore the expected value also increases by at most this factor). Thus, in what follows, we will consider only bag sizes that belong to the set $B = \{(\varepsilon + r\varepsilon^2)\text{OPT} : r = 0, 1, \ldots, \frac{1}{\varepsilon^2}\} \cup \{0\}$. We use this set by Lemma 3. We conclude that the following holds.

▶ **Corollary 4.** *If the guessed value (OPT) is at least the optimal expected value of the makespan, then there is a solution of expected value of the makespan of at most $(1 + \varepsilon)(1 + 3\varepsilon) \cdot \text{OPT}$ that uses only bags with allowed sizes in B.*

Note that the allowed size of a bag may be slightly larger than the actual total size of jobs assigned to the bag. Later, the allowed size acts as an upper bound (without a lower bound), and we take the allowed size into account in the calculation of machine completion times in some cases (instead of the size).

Rounding job sizes. We apply a similar rounding method for job sizes. Recall that by our guess, every job j has size at most OPT (see Lemma 1). Next, we apply the following modification to the jobs of sizes at most $\varepsilon^2\text{OPT}$. Whenever there is a pair of jobs, each of which has size at most $\varepsilon^2\text{OPT}$, we unite the two jobs (i.e., we delete the two jobs, adding a new job whose size is the sum of the two sizes of the two deleted jobs). This is equivalent to restricting our attention to solutions of the first stage where we add the constraint that the two (deleted) jobs must be assigned to a common bag. We repeat this process as long as there is such a pair. If there is an additional job of size at most $\varepsilon^2\text{OPT}$, we delete it from the instance, and in the resulting solution (after applying the steps below on the resulting instance) we add the deleted job to bag 1. This is done after the entire algorithm completes running, so it may increase the makespan of every scenario and therefore the expected value of the makespan, but we do not take it into account in the algorithm or in calculating allowed sizes of bags. This addition of the deleted job increases the makespan of every scenario by at most $\varepsilon^2\text{OPT}$, so the resulting expected value of the makespan will be increased by a multiplicative factor of at most $1+\varepsilon^2$. We consider the instance without this possible deleted job, and we prove the following.

▶ **Lemma 5.** *The optimal expected value of the makespan of the instance resulting from the above modification of the job sizes among all solutions with allowed bag sizes in the following modified set $B' = \{(\varepsilon + r\varepsilon^2)\text{OPT} : r = 0, 1, \ldots, \frac{1}{\varepsilon^2} + 2\} \cup \{0\}$ is at most $(1 + 2\varepsilon)(1 + \varepsilon)(1 + 3\varepsilon) \cdot \text{OPT}$.*

We next round up the size of every job j to be the next value that is of the form $(1 + \varepsilon)^r \cdot \text{OPT}$ for an integer r. The size of every job cannot decrease and it may increase by a multiplicative factor of at most $1 + \varepsilon$. Job sizes are still larger $\varepsilon^2 \cdot \text{OPT}$ and the sizes do not exceed $(1 + \varepsilon) \cdot \text{OPT}$. Thus, the number of different job sizes is $O(\log_{1+\varepsilon} \frac{1+\varepsilon}{\varepsilon^2}) \leq \frac{2}{\varepsilon^3} - 1$. We increase the allowed size of each non-empty bag by another multiplicative factor of $1 + \varepsilon$, and round it up again to the next value of the form $\varepsilon + r\varepsilon^2$. Thus, the expected value of the makespan of a feasible solution increases by a multiplicative factor of at most $(1 + \varepsilon)^2$, where one factor of $1+\varepsilon$ is due to the rounding of jobs, and the second one is due to bringing allowed sizes back to the form $\varepsilon + r\varepsilon^2$. So by our guessing step, there is a feasible solution with expected value of the makespan of at most $(1+\varepsilon)^3(1+2\varepsilon)(1+3\varepsilon)\text{OPT}$ that uses only bags with allowed size in $B'' = \{(\varepsilon + r\varepsilon^2)\text{OPT} : r = 0, 1, \ldots, \frac{1+2\varepsilon}{\varepsilon^2} + 2\} \cup \{0\}$.

A bag is called *tight* if the set of jobs of this bag cannot use a bag of a smaller allowed size. Note that any solution where some bags are not tight can be converted into one where all bags are tight without increasing the cost. Note that the allowed size of a non-zero bag of allowed size $(\varepsilon + r\varepsilon^2)\text{OPT}$ can be written in the form $(r + \frac{1}{\varepsilon}) \cdot \varepsilon^2 \cdot \text{OPT}$, and since $\frac{1}{\varepsilon}$ is integral, the allowed size of each bag is an integer multiple of $\varepsilon^2 \cdot \text{OPT}$.

▶ **Lemma 6.** *For any tight bag it holds that the allowed size of the bag is at most $\frac{1}{\varepsilon}$ times its actual size (the total size of jobs assigned to the bag, where their rounded sizes are considered).*

Computing the maximum number of machines for which the assignment to bags does not matter. In our algorithm, we would like to focus on values of k for which the structure of bags is crucial. Now, we will detect values of k for which there always exists a good assignment of bags to machines (for reasonable sets of bags). Let P be the total size of all jobs (after the transformation applied on jobs for a fixed value of OPT). The total size (not the allowed size) of all bags is P, and we can compute the makespan based on the actual sizes of bags which are total sizes of jobs (after merging and rounding) assigned to the bags. The motivation is that for very small values of k the maximum bag size is so small compared to $\frac{P}{k}$ that bags can be seen as very small jobs, and one can apply a greedy algorithm [22] for assigning the bags, and still obtain a solution with a small makespan (based on bag sizes).

▶ **Lemma 7.** *If $k \le \frac{\varepsilon \cdot P}{2 \cdot OPT}$, then any set of bags (such that every job is assigned to a bag) where every bag has an allowed size (and size) not exceeding 2OPT, leads to at least one schedule for k machines with makespan in $[\frac{P}{k}, (1+\varepsilon) \cdot \frac{P}{k})$. For other values of k, the makespan of an optimal schedule using tight bags (based on their allowed sizes) is at most $(3/\varepsilon^2) \cdot \text{OPT}$.*

We use the property that no bag has an allowed size above $2 \cdot \text{OPT}$. Our algorithm computes the maximum value of k for which $2\text{OPT} \le \varepsilon \cdot \frac{P}{k}$ holds, and afterwards it excludes values of k for which $2\text{OPT} \le \varepsilon \cdot \frac{P}{k}$ holds. We will have that the cost of the solution obtained for the scenario is at most $(1+\varepsilon) \cdot \frac{P}{k}$, since bag sizes satisfy the condition for bags in the statement of Lemma 7, and therefore we can use the first part of this lemma. This is a valid approach as adding a scenario where $2\text{OPT} \le \varepsilon \cdot \frac{P}{k}$ to the calculation of the approximation ratio will not increase the approximation ratio beyond the value $1+\varepsilon$, and the running time for computing a good solution for such a value of k is polynomial in m, n. Thus, we assume that $2\text{OPT} > \varepsilon \cdot \frac{P}{k}$ holds for every k.

We consider the remaining scenarios. In what follows, we consider the assignment of jobs to bags and the assignment of the bags to machines in each scenario k subject to the condition that $2\text{OPT} > \varepsilon \cdot \frac{P}{k}$ and the optimal makespan of the scenario is at most $(3/\varepsilon^2) \cdot \text{OPT}$. In particular, it means that the number of bags of positive allowed sizes assigned to a machine in such a scenario is at most $3/\varepsilon^3$, since we consider allowed bag sizes not smaller than $\varepsilon \cdot \text{OPT}$ (if we exclude bags of allowed size zero).

Guessing an approximated histogram of the optimal makespan on all scenarios. Our next guessing step is motivated by linear grouping of the OPT_k values. To illustrate this step, consider a histogram corresponding to the costs of a fixed optimal solution satisfying all above structural claims, where the width of the bar corresponding to scenario k is q_k and its height is the value of the makespan in this scenario (that is, OPT_k). Observe that when k

is increased, the optimal makespan does not increase (without loss of generality, since the same assignment of bags can be used), so by plotting the bars in increasing order of k we get a monotone non-decreasing histogram, where the total area of the bars is the expected value of the makespan of the optimal solution and the total width of all the bars is 1 (since we assume that all scenarios are included in the histogram and every scenario κ satisfies the condition $2\textsc{opt} > \varepsilon \cdot \frac{P}{\kappa}$ by scaling the probabilities of the remaining scenarios such that the total probability becomes 1). We sort the indexes of scenarios with (strictly) positive probabilities (q_k values) and we denote by K the resulting sorted list. For every $k \in K$, we compute the total width of the bars with indexes at most k (using the ordering in K, i.e., in increasing indexes), and denote this sum by Q_k. We also let $Q'_k = Q_k - q_k$. Thus, Q_k is the total probability of scenarios in $\{1, 2, \dots, k\}$, and Q'_k is the total probability of scenarios in $\{1, 2, \dots, k-1\}$. The set K does not contain scenarios with probability zero, but if $q_{k-1}, q_k > 0$, it holds that $Q_{k-1} = Q'_k$. According to the definitions, it holds that $Q_k - Q'_k = q_k$ for $k \in K$, and the interval $[Q'_k, Q_k)$ on the horizontal axis corresponds to scenario k.

The idea of the next parts is to get an overestimator for the histogram by extending the value of k to the right, and an underestimator by extending it to the left. The two areas below them will differ only by at most $O(\varepsilon \cdot \textsc{opt})$, so the overestimator can also differ from the original histogram area by at most this amount.

Next, we compute an upper bound histogram W as follows. We start with selecting a sublist K' of K. The motivation is that schedules will be computed later only for scenarios in K', and they will be copied to scenarios of some of the larger indexes. The set K' acts as a representative set, and we show that it is possible to restrict ourselves to such a set. Since we increase upper bounds on the makespan for some scenarios, the feasibility is not harmed.

An index $k \in K$ belongs to K' if there exists an integer ℓ such that $Q'_k \le \varepsilon^3 \ell < Q_k$. The motivation is that we would like to consider points which are integral multiples of ε^3 and the set K' contains all values of $k \in K$ for which such special values belong to the interval $[Q'_k, Q_k)$. Values of makespan will be increased such that the makespan function will still be piecewise linear, but it will have a smaller number of image values.

For every $k \in K'$, the new upper bound histogram is defined as $\textsc{opt}_k$ for all points with horizontal value between Q'_k and up to $Q'_{k'}$ where $k' > k$ is the index just after k in the sublist K' or up to the last point where the histogram is defined (Q_t for the maximum value $t \in K$) if k is the largest element of K'. Since the original histogram was monotone non-increasing, the new histogram is pointwise not smaller than the original histogram. The possible modification is in the interval $[Q_k, Q'_{k'})$ if k is not the maximum value of K', and in this case there may be a change for $k+1, \dots, k'-1$ (that is, if $k' \ge k+2$). If k is the largest element of K' but not of K, there may be a change for all $t \in K$ such that $t \ge k+1$. Thus, an upper bound on the total area below the histogram W is not smaller than the expected value of the makespan of the optimal solution.

We use the modified histogram W to obtain another bound. For that we see W as a step function whose values are the corresponding points in the upper edge of the histogram. We define a new histogram by letting it be $W(x + \varepsilon^3)$ for all $x \in [0, 1]$ (and zero for cases that W is undefined due to an argument above 1). In this way we delete a segment of length ε^3 from W and we shift the resulting histogram to the left. Since the makespan for every scenario never exceeds $(3/\varepsilon^2) \cdot \textsc{opt}$, every point in W has a height of at most $\frac{3\textsc{opt}}{\varepsilon^2}$ and we deleted a segment of width of ε^3, the total area that we delete is at most $3\varepsilon\textsc{opt}$. The resulting histogram is pointwise at most the original histogram (and thus also not larger than W). This property holds since every value $\textsc{opt}_k$ was extended to the right for an interval not longer than ε^3. Thus, if we consider W instead of the original histogram, we increase the cost of the solution by a multiplicative factor not larger than $(1 + 3\varepsilon)$.

The guessing step that we use is motivated by this function W. We let W_k be the height of the histogram W in the bar corresponding to the scenario k. The relevant values of k are those in K', and the histogram W only has values of the form OPT_k for $k \in K'$. We guess the histogram W. This means to guess the optimal makespan of $O(\frac{1}{\varepsilon^3})$ scenarios, where makespans can be one of at most $\frac{3}{\varepsilon^4}$ different values. This holds since all allowed bag sizes are integer multiples of $\varepsilon^2 \cdot \text{OPT}$, so makespans are also integer multiples of $\varepsilon^2 \cdot \text{OPT}$, and the makespan of each scenario is at most $\frac{3\text{OPT}}{\varepsilon^2}$. Thus, the number of possibilities of this guessing step is upper bounded by a function of $\frac{1}{\varepsilon}$ which is $O((\frac{1}{\varepsilon})^{O(1/\varepsilon^3)})$.

In what follows, we consider the iteration of the algorithm defined below when we use the value of the guess corresponding to the optimal solution. Algorithmically, we will try all possibilities, check the subset of those for which we can provide a feasible solution, and output the best feasible solution obtained in this exhaustive enumeration. The running time of the next algorithm is multiplied by the number of possibilities.

The template-configuration integer program. A *template* of a bag is a multiset of job sizes assigned to a common bag. We consider only multisets for which the total size of jobs is at most $(1+6\varepsilon) \cdot \text{OPT}$ since the largest allowed size of any bag is smaller. Since the number of distinct job sizes (in the rounded instance) is upper bounded by $\frac{2}{\varepsilon^3} - 1$ and there are at most $\frac{2}{\varepsilon^2}$ jobs assigned to a common bag (since the rounded size of each job is above $\varepsilon^2 \cdot \text{OPT}$), we conclude that the number of templates is at most $(\frac{2}{\varepsilon^3})^{(2/\varepsilon^2)}$, which is a constant depending only on $\frac{1}{\varepsilon}$. The reason is that each bag has $\frac{2}{\varepsilon^2}$ slots for jobs, such that each one may be empty or contain a job of one of the sizes. This calculation proves an upper bound on the number of possible templates, and we use a single template for every multiset of job sizes even when one multiset can be found in more than one way in the last calculation.

We are going to have a non-negative counter decision variable y_t for every template t, where this variable stands for the number of bags with template t. Let τ be the set of templates, and assume that a template is represented by a vector. The length of such a vector is the number of different (rounded) job sizes, and for every index ℓ, the ℓ-th component of the vector of the template (denoted by t_ℓ) is the number of jobs with size equal to the ℓ-th job size that are packed into a bag with this template. In order for such an assignment of the first stage to be feasible, we have the following constraints where n_ℓ denotes the number of jobs in the rounded instance whose size is the ℓ-th job size of the (rounded) instance.

$$\sum_{t\in\tau} y_t \le m \ , \qquad \sum_{t\in\tau} t_\ell \cdot y_t = n_\ell \ , \forall \ell.$$

The first constraint states that the number of bags is at most m. If this number is strictly smaller than m, then the remaining bags have allowed sizes of zero and they will not contain jobs. The second constraint, which is a family of constraints, states that the correct numbers of jobs of each size are assigned to templates, according to the number of copies of each template. Observe that the number of constraints in this family of constraints is a constant depending on ε (it is at most $\frac{2}{\varepsilon^3}$), and all coefficients in the constraint matrix are (non-negative) integers not larger than $\frac{2}{\varepsilon^2}$.

We augment K' with the minimum index in K if this index does not already belong to K', in order to satisfy the condition that for every index of K there is some index of K' that is not larger. Consider a scenario $\kappa \in K'$ (satisfying the condition that $2\text{OPT} > \varepsilon \cdot \frac{P}{\kappa}$), and the optimal makespan of scenario κ, which is at most $\frac{3\text{OPT}}{\varepsilon^2}$. Recall that in scenario κ the number of non-empty bags assigned to each machine is at most $\frac{3}{\varepsilon^3}$. Define a configuration of a machine in scenario κ to be a multiset of templates such that the multiset has at most $\frac{3}{\varepsilon^3}$

templates (counting multiple copies of templates according to their multiplicities) and the total size of the templates encoded in the multiset is at most W_κ, which is the guess of a value of the histogram W. The number of configurations is at most $((\frac{2}{\varepsilon^3})^{(2/\varepsilon^2)})^{\frac{3}{\varepsilon^3}}$, and this is also an upper bound on the number of suitable configurations (whose total allowed size of bags does not exceed W_κ). Since our mathematical program will not limit the makespan of any scenario, we use an upper bound for it by not allowing configurations whose total allowed sizes exceed the planned makespan for the scenario. We let $C^{(\kappa)}$ denote the set of configurations for scenario κ, where $c \in C^{(\kappa)}$ is a vector of $|\tau|$ components where component c_t for $t \in \tau$ is the number of copies of template t assigned to configuration c. Components are non-negative integers not larger than $\frac{3}{\varepsilon^3}$. For scenario $\kappa \in K'$, we will have a family of non-negative decision variables $x_{c,\kappa}$ (for all $c \in C^{(\kappa)}$) counting the number of machines with this configuration.

For each such scenario κ, we have a set of constraints each of which involves only the template counters (acting as a family of global decision variables) and the family of the κ-th local family of decision variables, namely the ones corresponding to this scenario. The family of constraints for the scenario $\kappa \in K'$ are as follows.

$$\sum_{c \in C^{(\kappa)}} x_{c,\kappa} = \kappa \ , \qquad \sum_{c \in C^{(\kappa)}} c_t \cdot x_{c,\kappa} - y_t = 0 \ , \ \forall t \in \tau \ .$$

Observe that the number of constraints in such a family of constraints is upper bounded by a constant depending only on ε (which is the number of possible templates plus 1) and the coefficients in the constraint matrix are again all integers of absolute value at most a constant depending only on ε (at most $\frac{3}{\varepsilon^3}$.).

All decision variables are forced to be non-negative integers and we would like to solve the feasibility integer program defined above (with all constraints and decision variables). The right hand side vector consists of integers that are at most n (using $m \le n$). Such an integer linear program is a two-stage stochastic IP. Using the property that the number of scenarios in K' is upper bounded by a function of $\frac{1}{\varepsilon}$, we conclude that the integer linear program has a fixed dimension. Thus, we use Lenstra's algorithm to solve it instead of an algorithm for two-stage stochastic IP.

We obtain a feasible solution (x, y) if such a solution exists. Based on such a feasible solution, we assign jobs to bags using the y variables. That is, for every $t \in \tau$, we schedule y_t bags using template t. By the constraint $\sum_{t \in \tau} y_t \le m$ there are at most m bags, and the other bags will be empty. Next, for every bag and every size p of jobs in the rounded instance, if the template assigned to the bag has α jobs of this size, we will assign α jobs of this size to this bag. Doing this for all sizes and all bags is possible by the constraints $\sum_{t \in \tau} t_\ell \cdot y_t = n_\ell \ , \forall \ell$, and these constraints ensure that all jobs are assigned to bags. Note that the modification for jobs consisted of merging small jobs. When such a merged job is assigned, this means that a subset of jobs is assigned. Reverting jobs to their original sizes does not increase any of the costs, since no rounding steps decreased any sizes. Next consider the assignment of bags to machines in each scenario. Consider a scenario $\kappa' \in K$, and let κ be the largest index in K' such that $\kappa \le \kappa'$. Assign $x_{c,\kappa}$ machines to configuration c (for all $c \in C^{(\kappa)}$). It is possible by the constraint $\sum_{c \in C^{(\kappa)}} x_{c,\kappa} = \kappa \le \kappa'$. If a configuration c assigned to machine i is supposed to pack c_t copies of template t, we pick a subset of c_t bags whose assigned template is t and assign these bags to machine i. We do this for all templates and all machines. In this way we assign all bags by the constraints $\sum_{c \in C^{(\kappa)}} c_t \cdot x_{c,\kappa} - y_t = 0 \ , \ \forall t \in \tau$. If $\kappa' > \kappa$, at least one machine will not receive any configuration and therefore it will not receive any bags or jobs, and we refer to such a machine as empty. Since the configurations we used in scenario κ have a total size of jobs of at most W_κ, we conclude the following.

▶ **Corollary 8.** *For every value of the guessed information for which the integer program has a feasible solution, there is a linear time algorithm that transform the feasible solution to the integer program into a solution to the rounded instance of* PR-MAKESPAN *of cost at most* $\sum_{\kappa} q_{\kappa} \cdot W_{\kappa}$.

Our scheme is established by noting that an optimal solution satisfying the assumptions of the guessed information provides us a multiset of templates all of which are considered in τ and a multiset of configurations (and all of them have total size of templates not larger than W) for which the corresponding counters satisfy the constraints of the integer program. Thus, we conclude our first main result.

▶ **Theorem 9.** *Problem* PR-MAKESPAN *admits an EPTAS.*

3 An EPTAS for the Santa Claus problem

In this section we apply additional ideas to obtain an EPTAS for the second problem. We present the details of the scheme together with its analysis. In what follows, and similarly to the scheme to PR-MAKESPAN, we will present steps, and in each of those steps the resulting approximation ratio of the scheme increases by at most a multiplicative factor of $1 + \Theta(\varepsilon)$.

Preprocessing steps. We consider an optimal solution OPT, and analyze the information that one can guess in order to be able to approximate it. We assume without loss of generality that $\text{OPT}_k \leq \text{OPT}_{k'}$ for all $k > k'$, since a solution for scenario k can be used for scenario k' (by assigning the bags of $k - k'$ machines in an arbitrary way). Recall that we can assume that for every value of k in the support of q the instance has at least k non-zero sized jobs, since we assume $n > m$. We will use the next lemma for our proof.

▶ **Lemma 10.** *For any scenario k (such that $1 \leq k \leq m$) and an assignment of jobs to bags according to the solution* OPT*, there exists a job j^k for which it holds that $p_{j^k} \leq \text{OPT}_k \leq n \cdot p_{j^k}$.*

We would like to split the sequence of scenarios into four parts (where some of the parts may be empty). The suffix (with the largest numbers of machines) will contain scenarios for which the assignment of bags is unimportant since the gain from them in the objective function value OPT is small either because the probability is zero or because the number of machines is large. This suffix may be empty. There will also be a prefix (which could be empty) of machines for which the specific assignment to bags is unimportant because the number of machines is small and any reasonable assignment into bags allows a good schedule. For this prefix we will use different arguments from the ones we used for PR-MAKESPAN. For the remaining scenarios, the prefix will consist of scenarios for which the gain is also small, and a suffix of the most important scenarios.

The first step is to guess the maximum scenario $k_{\max}$ for which $\text{OPT}_{k_{\max}} \geq \varepsilon \cdot \text{OPT}$ holds, out of scenarios with strictly positive probabilities, that is, such that $q_{k_{\max}} > 0$ holds. This index is well-defined since there exists an index k for which $q_k > 0$ and $\text{OPT}_k \geq \text{OPT}$. There are at most m possible values for this guess. While OPT still denotes an optimal solution, we do not guess or use its value as a part of the algorithm. Let LB be equal to $\text{OPT}_{k_{\max}}$ rounded down to the next integer power of $1 + \varepsilon$. For a fixed job $j^{k_{\max}}$ (see Lemma 10), the number of possible values for LB is $O(\frac{n}{\varepsilon})$. Since the index $j^{k_{\max}}$ is also not known, the number of possible values for LB is $O(\frac{n^2}{\varepsilon})$.

The proof of the next lemma is obtained by selecting a set of consecutive scenarios minimizing the total weighted profit out of $\frac{1}{\varepsilon}$ disjoint sequences of scenarios of similar forms. The proof of the lemma requires the knowledge not only of OPT but of all values of the form

$\textsc{opt}_i$. However, we use the lemma to obtain the information that given the correct value LB (this is a rounded value, so we will be able to guess it), there is a pair of values k, k' and a value ρ that specify the only properties of $\textsc{opt}$ that we use for our algorithm.

▶ **Lemma 11.** *There is a value of ρ which is an integer power of $\frac{1}{\varepsilon}$ that satisfies $\frac{1}{\varepsilon^2} \le \rho \le (\frac{1}{\varepsilon})^{1/\varepsilon+1}$ and there exist two indexes k, k' such that $0 \le k' < k \le k_{\max}$, where every non-zero index has to be a scenario in the support of q, such that the following conditions hold. $\textsc{opt}_k \le \rho \cdot LB$, if $k' \ge 1$, then $\textsc{opt}_{k'} \ge \frac{\rho}{\varepsilon} \cdot LB$,$\sum_{\kappa=k'+1}^{k-1} q_\kappa \cdot \textsc{opt}_\kappa \le \varepsilon \cdot \textsc{opt}$.*

The next guessing step. In this step we guess several additional values. As mentioned above, we guess the value of LB (which is a power of $1+\varepsilon$ and it is equal to $\textsc{opt}_{k_{\max}}$ within a multiplicative factor of $1+\varepsilon$ since $\textsc{opt}_{k_{\max}} \in [LB, LB \cdot (1+\varepsilon))$), the value of ρ (by guessing r' from the proof of Lemma 11, that is, we guess the power of $\frac{1}{\varepsilon}$), and two indexes $k' < k \le k_{\max}$. That is, we would like to enumerate a set of polynomial number of candidate values of four components vectors that contains at least one vector with first component value that is LB, with a second component whose value is ρ, and the two indexes $k' < k$ in the third and fourth components of that vector. The next lemma shows that it is possible to enumerate such a set of candidate values. Our algorithm performs this enumeration.

▶ **Lemma 12.** *There is a set consisting of $O((\frac{m \cdot n}{\varepsilon})^2)$ vectors such that this set of vectors contains at least one vector with the required properties. Thus, the number of guesses including the guess of $k_{\max}$ is at most $O((\frac{n}{\varepsilon})^2 \cdot m^3)$.*

In what follows, we let LB be the first component value of the guessed information, ρ be the second component value of the guessed information, and k, k' be the two guessed scenarios (where it is possible that $k' = 0$ is not an actual scenario). We will show that if there is a feasible solution to Pr-SantaClaus for which the guessed information describes the solution and its expected value of the objective is $\textsc{opt}$, then we will construct a feasible solution with expected value of the objective being at least $(1 - \Theta(\varepsilon)) \cdot \textsc{opt}$. This means that the problem admits an EPTAS as desired.

We let $UB = LB \cdot \rho$. Furthermore, for every scenario κ with $k' < \kappa < k$, we set $q_\kappa = 0$. From now on we drop the assumption that the sum of all q values is 1 and instead of that we use the assumption that these are non-negative numbers with sum at most 1. This modification of the vector (q) decreases the expected value of the Santa Claus value of the optimal solution by at most $\varepsilon \cdot \textsc{opt}$ (and it does not increase the objective function value of any feasible solution to our problem). The justification for this is as follows. From Lemma 11 it follows that there is a quadruple of integers $k, k', \rho, \log_{1+\varepsilon} LB$ (where each integer belongs to the set that we test for this integer) for which there is a solution of profit at least $\frac{\text{OPT}}{1+\varepsilon} - \varepsilon \cdot \textsc{opt} \ge (1-\varepsilon)^3 \cdot \textsc{opt}$ with the first two properties stated in the lemma, and for every scenario κ such that $k' < \kappa < k$ the profit is at least zero.

Partitioning the input into two independent problems. Next, we use the above guessing step in order to partition the input into two independent inputs. First, a job $j \in \mathcal{J}$ is called huge if its size is strictly larger than UB, that is, if $p_j > UB$ (and otherwise it is non-huge).

We pack every huge job into its own bag and we let h denote the number of huge jobs (where we will show that $h \le m-1$ holds). Every huge job can cover one machine in every scenario $\kappa \ge k$ regardless of its exact size (because for all these scenarios we consider solutions for which $\textsc{opt}_\kappa$ is not larger than $\rho \cdot LB = UB$). On the other hand, the non-huge jobs (i.e., jobs of sizes at most UB) will be packed into bags of size at most $3 \cdot UB$. The packing of the non-huge jobs into bags will be optimized according to the scenarios of indexes at least k

(and at most $k_{\max}$), and we will ignore the scenarios of indexes smaller than k when we pack the non-huge jobs into bags. The resulting bags of the non-huge jobs together with the set of the huge jobs (which are bags consisting of a single job) will be packed into $\kappa \le k'$ machines (in the corresponding scenario κ) based on existing EPTAS for the Santa Claus problem on identical machines (seeing bags as jobs). We will show that if indeed we use bags of sizes at most $3 \cdot UB$ when we pack the non-huge jobs into bags, then the scenarios of indexes at most k' can be ignored. We show that this holds for any collection of bags of this form, that is, where every huge job has its own bag and other bags have total sizes of jobs not exceeding $3 \cdot UB$. Thus, even though the bags are defined based on scenarios where the number of machines is at least k, still the scenarios with at most k' machines have good solutions. We will also show that it is possible to restrict ourselves to these types of collections of bags.

▶ **Lemma 13.** *If all guesses are correct, any partition into bags has at least one bag with a total size of jobs no larger than UB. In particular, there are at most $m-1$ huge jobs.*

▶ **Lemma 14.** *An EPTAS for κ machines where $\kappa \le k'$ which is applied on bags (instead of jobs) such that every huge job has its own bag and any additional bag has total size of jobs not exceeding $3 \cdot UB$ finds a solution of profit at least $\frac{OPT_\kappa}{(1-3\varepsilon)\cdot(1-\varepsilon)}$.*

Proof. Consider an optimal solution for κ machines and the original jobs. We show that this schedule can be modified into a schedule of the bags, such that the profit of the schedule is smaller by a factor not exceeding $1-3\varepsilon$. Thus, an optimal schedule of the bags is not worse, and by using an EPTAS the profit may decrease by another factor of $1-\varepsilon$. The adaptation of the schedule is as follows. The huge jobs are assigned as before, since each of them has its own bag. All non-huge jobs are removed, and each machine receives bags until it is not possible to add another bag without exceeding the previous load or no unassigned bags remain. Given the property that the total size of bags is equal to the total size of jobs, it is either the case that the loads are equal to previous loads (in which case the value is unchanged) or there is at least one unassigned bag. In the latter case, no machine load decreased by more than an additive term of $3 \cdot UB$, and therefore the value is at least the previous one minus $3 \cdot UB$. To complete the assignment, all remaining bags are assigned arbitrarily. Since $\text{OPT}_\kappa \ge \frac{\rho \cdot LB}{\varepsilon} = \frac{UB}{\varepsilon}$ and the resulting value is at least $\text{OPT}_\kappa - 3 \cdot UB$, we find by $3 \cdot UB \le 3\varepsilon \cdot \text{OPT}_\kappa$ that $\text{OPT}_\kappa - 3 \cdot UB \ge (1-3\varepsilon) \cdot \text{OPT}_\kappa$. ◀

▶ **Lemma 15.** *Consider the instance of our problem when we let $q_\kappa = 0$ for all $\kappa < k$ and for $\kappa > k_{\max}$. There exists a partition into bags where a profit at least OPT_κ can be obtained for any $\kappa \in [k, k_{\max}]$, the set of bags satisfies that every huge job is packed into its own bag, and each other bag consists of non-huge jobs of total size at most $2 \cdot UB$.*

Proof. The number of partitions into bags is finite for fixed m, n (it does not exceed m^n). Consider the set of partitions for which a solution of profit at least OPT_κ can be obtained for any $\kappa \in [k, k_{\max}]$. There is at least one such partition for the correct guess. For every partition it is possible to define a vector of m components, such that total sizes of bags appear in a non-increasing order. Consider the partition among the considered partitions for which the number of huge jobs that have their own bags is maximum, and out such partitions, one where the vector is lexicographically minimal. We claim that this partition satisfies the requirements.

Assume by contradiction that the number of huge jobs that do not have their own bags is not h. Consider a bag with a huge job that contains at least one additional job. This will be named the first bag. Consider a bag whose total size is at most UB, which must exist due to Lemma 13. This last bag does not have a huge job since the size of a huge job is above

UB, and we call it the second bag. Move all contents of the first bag into the second bag excluding one huge job. For any $\kappa \in [k, k_{\max}]$, since $\textsc{opt}_\kappa \le UB$, the assignment of bags to machines does not reduce the objective function value below $\textsc{opt}_\kappa$. This holds because there is at most one machine whose total size was reduced (for every $\kappa \in [k, k_{\max}]$), but if such a machine exists, it still has a huge job whose size is above UB. Thus, the new partition is also one of the considered partitions. The new partition has a larger number of huge jobs assigned into their own bags, because the second bag was not such a bag and the first bag became such a bag. This contradicts the choice of assignment to bags. Thus, the partition into bags consists of h bags with huge jobs and $m - h \ge 1$ bags with non-huge jobs.

Next, consider only the components of the vector which do not correspond to bags of huge jobs. This vector is also minimal lexicographically out of vectors for partitions of the non-huge jobs into $m - h$ bags. Assume by contradiction that the first component is above $2 \cdot UB$. Consider the bag of the first component (called the first bag now) and a bag with a total size at most UB (called the second bag now). Move one job from the first bag to the second bag. The second bag now has a total size of at most $2 \cdot UB$ (since a non-huge job was moved). The first bag now has a smaller total size, but still larger than UB. The sorted vector is now smaller lexicographically, and it is still possible to obtain a solution of profit at least $\textsc{opt}_\kappa$ for any $\kappa \in [k, k_{\max}]$, similarly to the proof given here for huge jobs, which is a contradiction. ◀

In summary, we can focus on the scenarios interval $[k, k_{\max}]$, assume that huge jobs have their own bags, and remaining bags have total sizes not exceeding $2 \cdot UB$.

Modifying the input of scenarios with indexes at least k and at most $k_{\max}$. Motivated by the last partitioning of the original input into four parts, and the fact that only scenarios with indexes at least k and at most $k_{\max}$ need to be considered, we apply the following transformation. First, for every $\kappa < k$ we let $q_\kappa = 0$, in the sense we can augment every solution to the remaining instance (with only a subset of scenarios) into an EPTAS for the original instance before this change to q. Furthermore, for every $\kappa > k_{\max}$ we let $q_\kappa = 0$ in the sense we can ignore the profit of such scenarios. Then, every huge job is packed into a separate bag. Such a bag suffices to cover one machine in every remaining scenario. Therefore, our second step in the transformation is the following one. We delete the huge jobs from $\mathcal{J}$, we decrease the index of each remaining scenario by h (in particular the new indexes in the support of q will be in the interval $[k - h, k_{\max} - h]$), and we enforce the condition that every bag size is at most $2 \cdot UB$.

As one can see, the transformations here are more complicated compared to those used in the previous section, and they have to be carefully designed and analyzed. However, now we are ready to apply methods that resemble the previous section. Using the fact that for every remaining scenario we have that $\textsc{opt}_\kappa$ is between LB and UB, and the fact that $\frac{UB}{LB} = \rho \le (\frac{1}{\varepsilon})^{1/\varepsilon+1}$ we are able to apply the methods we have developed for the makespan minimization problem to obtain an EPTAS for Pr-SantaClaus, as we do next.

Rounding bag sizes. We next provide a method to gain structure on the set of feasible solutions that we need to consider for finding a near optimal solution. Given a feasible solution, the size of bag i denoted as $P(i)$ is the total size of jobs assigned to this bag, that is, $P(i) = \sum_{j:\sigma_1(j)=i} p_j$.

▶ **Lemma 16.** *There exists a solution of value at least $(1 - 3\varepsilon) \cdot \textsc{opt}$ in which the size of every non-empty bag is in the interval $[\varepsilon \cdot LB, 2.5 \cdot UB]$.*

In what follows, we will decrease the size of a bag to be the next value of the form $(\varepsilon + r\varepsilon^2) \cdot LB$ for an integer $r \geq 0$ (except for empty bags for which the allowed size remains zero). Thus, we will not use precise sizes for bags but lower bounds on sizes, and we call them "allowed sizes" in what follows. For bags of allowed size zero the meaning remains that such a bag is empty. We let $P'(i)$ be this decreased (allowed) size of bag i, that is, $P'(i) = \max_{r:(\varepsilon+r\varepsilon^2)\cdot LB \leq P(i)}(\varepsilon + r\varepsilon^2) \cdot LB$. The allowed size is not smaller than $\varepsilon \cdot LB$ and it is smaller than the size by an additive term of at most $\varepsilon^2 \cdot LB$. Thus, for every subset of bags, the total allowed size of the bags in the set is at least $\frac{1}{1+\varepsilon}$ times the total size of the bags in the subset, we conclude that this rounding of the bag sizes will decrease the value of any solution by a multiplicative factor of at most $1 + \varepsilon$ (and therefore the expected value also decreases by at most this factor), and it will not increase the value of a solution for any scenario. Thus, in what follows, we will consider only bag sizes that belong to the set $B = \{(\varepsilon + r\varepsilon^2) \cdot LB : r \in \mathbb{Z}, r \geq 0, (\varepsilon + r\varepsilon^2) \cdot LB \leq 2.5 \cdot UB\} \cup \{0\}$. We use this set by Lemma 16. We conclude that the following holds.

▶ **Corollary 17.** *If the guessed information vector is satisfied by an optimal solution, then there is a solution of expected value of the Santa Claus value of at least $\frac{1-3\varepsilon}{1+\varepsilon} \cdot \textsc{opt}$ that uses only bags with allowed sizes in B.*

Note that the allowed size of a bag may be slightly smaller than the actual total size of jobs assigned to the bag. Later, the allowed size acts as a lower bound (without an upper bound), and we sometimes take the allowed size into account in the calculation of machine completion times.

Rounding job sizes. We apply a similar rounding method for job sizes. Recall that by our transformation, every job j has size at most UB (since huge jobs were removed from the input). Next, we apply the following modification to the jobs of sizes at most $\varepsilon^2 \cdot LB$. While there is a pair of jobs of sizes at most $\varepsilon^2 \cdot LB$, we unite such a pair of jobs. If there is an additional job of size at most $\varepsilon^2 \cdot LB$, we delete it from the instance, and in the resulting solution (after applying the algorithm below on the resulting instance) we add the deleted job to an arbitrary bag. This deletion of the deleted job decreases the Santa Claus value of every scenario by at most $\varepsilon^2 \cdot LB$, so the resulting expected value of the Santa Claus value will be decreased by at most $\varepsilon^2 \cdot LB$. Adding the job back does not decrease the value. We consider the instance without this possibly deleted job, and we prove the following.

▶ **Lemma 18.** *The optimal expected value of the Santa Claus value of the instance resulting from the above modification of the job sizes among all solutions with allowed bag sizes in the following modified set $B' = \{(\varepsilon + r\varepsilon^2) \cdot LB : r \in \mathbb{Z}, r \geq -2, (\varepsilon + r\varepsilon^2) \cdot LB \leq 3 \cdot UB\} \cup \{0\}$ is at least $(1 - 2\varepsilon) \cdot \frac{1-3\varepsilon}{1+\varepsilon} \cdot \textsc{opt}$.*

Next, we round down the size of every job j to be the next value that is of the form $(1 + \varepsilon)^r$ for an integer r. The size of every job cannot increase and it may decrease by a multiplicative factor of at most $1 + \varepsilon$. Job sizes are still larger than $\varepsilon^3 \cdot LB$ and not larger than $UB \leq (\frac{1}{\varepsilon})^{1/\varepsilon+1} \cdot LB$. Thus, the number of different job sizes is $O(\log_{1+\varepsilon}(\frac{1}{\varepsilon})^{1/\varepsilon+4}) \leq \frac{2}{\varepsilon^3} - 1$. We decrease the allowed size of each bag by another multiplicative factor of $1 + \varepsilon$. Bags sizes are rounded down again to the next value of the form $\varepsilon + r\varepsilon^2$, and since the smallest allowed size was $(\varepsilon - 2\varepsilon^2) \cdot LB$ and $\frac{\varepsilon - 2\varepsilon^2}{1+\varepsilon} \geq \varepsilon - 3\varepsilon^2$, the allowed bag sizes become $B'' = \{(\varepsilon + r\varepsilon^2) \cdot LB : r \in \mathbb{Z}, r \geq -3, (\varepsilon + r\varepsilon^2) \cdot LB \leq 3 \cdot UB\} \cup \{0\}$, and the expected value of the Santa Claus value of a feasible solution decreases by a multiplicative factor of at most $\frac{1-2\varepsilon}{1-3\varepsilon}$. By our guessing step and our transformation, there is a feasible solution with expected value of the Santa Claus value of at least $(1 - \varepsilon)(1 - 3\varepsilon)^2 \cdot \textsc{opt}$.

A bag is called *tight* if the set of jobs of this bag cannot use a bag of a larger allowed size. Note that any solution where some bags are not tight can be converted into one where all bags are tight without decreasing the expected value of the objective.

▶ **Lemma 19.** *For any tight bag it holds that the allowed size of the bag is at least ε^2 times its actual size (the total size of jobs assigned to the bag, such that their rounded sizes are considered).*

The final steps of the EPTAS for Pr-SantaClaus. Our next step is to guess an approximated histogram of the optimal Santa Claus value in all scenarios. This step and the next step of formulating a template-configuration integer program of fixed dimension are similar to the ones we have established for Pr-makespan. Using these additional steps we manage to prove our second main result.

▶ **Theorem 20.** *Problem Pr-SantaClaus admits an EPTAS.*

References

1 N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 493–500, 1997.

2 N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

3 B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load balancing in the l_p norm. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 383–391, 1995.

4 Y. Azar, L. Epstein, and R. van Stee. Resource augmentation in load balancing. *Journal of Scheduling*, 3(5):249–258, 2000.

5 E. Balkanski, T. Ou, C. Stein, and H.-T. Wei. Scheduling with speed predictions. In *Proc. of the 21st International Workshop on Approximation and Online Algorithms (WAOA'23)*, pages 74–89, 2023.

6 N. Bansal and K. R. Pruhs. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM Journal on Computing*, 39(7):3311–3335, 2010. `doi:10.1137/090772228`.

7 N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proc. of the 38th Annual ACM Symposium on Theory of Computing (STOC'06)*, pages 31–40, 2006.

8 M. Brehob, E. Torng, and P. Uthaisombut. Applying extra-resource analysis to load balancing. *Journal of Scheduling*, 3(5):273–288, 2000.

9 M. Buchem, F. Eberle, H. K. Kasuya Rosado, K. Schewior, and A. Wiese. Scheduling on a stochastic number of machines. In *Proc. of the 27th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'24)*, pages 14:1–14:15, 2024.

10 A. K. Chandra and C. K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM Journal on Computing*, 4(3):249–263, 1975. `doi:10.1137/0204021`.

11 L. Chen, K. Jansen, and G. Zhang. On the optimality of exact and approximation algorithms for scheduling problems. *Journal of Computer and System Sciences*, 96:1–32, 2018. `doi:10.1016/J.JCSS.2018.03.005`.

12 R. A. Cody and E. G. Coffman Jr. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *Journal of the ACM*, 23(1):103–115, 1976. `doi:10.1145/321921.321933`.

13 E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1):1–17, 1978. `doi:10.1137/0207001`.

14 J. Csirik, H. Kellerer, and G. Woeginger. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters*, 11(5):281–287, 1992. `doi:10.1016/0167-6377(92)90004-M`.

15 B. L. Deuermeyer, D. K. Friesen, and M. A. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Discrete Mathematics*, 3(2):190–196, 1982.

16 R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, Berlin, 1999.

17 S. Dye, L. Stougie, and A. Tomasgard. The stochastic single resource service-provision problem. *Naval Research Logistics*, 50(8):869–887, 2003.

18 F. Eberle, R. Hoeksma, N. Megow, L. Nölke, K. Schewior, and B. Simon. Speed-robust scheduling: sand, bricks, and rocks. *Mathematical Programming*, 197(2):1009–1048, 2023. `doi:10.1007/S10107-022-01829-0`.

19 W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

20 D. K. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13(1):170–181, 1984. `doi:10.1137/0213013`.

21 D. K. Friesen and B. L. Deuermeyer. Analysis of greedy solutions for a replacement part sequencing problem. *Mathematics of Operations Research*, 6(1):74–87, 1981. `doi:10.1287/MOOR.6.1.74`.

22 R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

23 R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

24 D. S. Hochbaum. Various notions of approximations: Good, better, best and more. In D. S. Hochbaum, editor, *Approximation algorithms.* PWS Publishing Company, 1997.

25 D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987. `doi:10.1145/7531.7535`.

26 K. Jansen, K.-M. Klein, and J. Verschae. Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research*, 45(4):1371–1392, 2020. `doi:10.1287/MOOR.2019.1036`.

27 R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. of the 15th annual ACM Symposium on Theory of Computing (STOC'83)*, pages 193–206, 1983.

28 H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. `doi:10.1287/MOOR.8.4.538`.

29 J. Y.-T. Leung and W.-D. Wei. Tighter bounds on a heuristic for a partition problem. *Information Processing Letters*, 56(1):51–57, 1995. `doi:10.1016/0020-0190(95)00099-X`.

30 J. Minařík and J. Sgall. Speed-robust scheduling revisited. In *Proc. of the 27th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'24)*, pages 8:1–8:20, 2024. `doi:10.4230/LIPIcs.APPROX/RANDOM.2024.8`.

31 M. Mitzenmacher and S. Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022. `doi:10.1145/3528087`.

32 K. Rustogi and V. A. Strusevich. Parallel machine scheduling: Impact of adding extra machines. *Operations Research*, 61(5):1243–1257, 2013. `doi:10.1287/OPRE.2013.1208`.

33 C. Stein and M. Zhong. Scheduling when you do not know the number of machines. *ACM Transactions on Algorithms*, 16(1):9:1–9:20, 2020. `doi:10.1145/3340320`.

34 G. J. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997. `doi:10.1016/S0167-6377(96)00055-7`.