A unified approach to truthful scheduling on related machines*

Leah Epstein† Asaf Levin‡ Rob van Stee§

Abstract

We present a unified framework for designing deterministic monotone polynomial time approximation schemes (PTAS's) for a wide class of scheduling problems on uniformly related machines. This class includes (among others) minimizing the makespan, maximizing the minimum load, and minimizing the ℓ_p-norm of the machine loads vector. Previously, this kind of result was only known for the makespan objective. Monotone algorithms have the property that an increase in the speed of a machine cannot decrease the amount of work assigned to it.

The idea of our novel method is to show that for goal functions that are sufficiently well-behaved functions of the machine loads, it is possible to compute in polynomial time a highly structured nearly optimal schedule. An interesting aspect of our approach is that, in contrast to all known approximation schemes, we avoid rounding any job sizes or speeds throughout. We can therefore find the *exact* best structured schedule using dynamic programming. The state space encodes a sufficient amount of information such that no postprocessing is needed, allowing an elegant and relatively simple analysis without any special cases. The monotonicity is a consequence of the fact that we find the *best* schedule in a specific collection of schedules.

In the game-theoretical setting of these scheduling problems, there is a social goal, which is one of the objective functions that we study. Each machine is controlled by a selfish single-parameter agent. The private information of an agent is its cost of processing a unit-sized job, which is also the inverse of the speed of its machine. Each agent wishes to maximize its own profit, defined as the payment it receives from the mechanism minus its cost for processing all jobs assigned to it, and places a bid which corresponds to its private information. Monotone approximation schemes have an important role in the emerging area of algorithmic mechanism design, as in the case of single-parameter agents, a necessary and sufficient condition for truthfulness with respect to the bids is that the allocation algorithm be monotone. For each one of the problems, we show that we can calculate payments that guarantee truthfulness in an efficient manner. Thus, there exists a dominant strategy where agents report their true speeds, and we show the existence of a truthful mechanism which can be implemented in polynomial time, where the social goal is approximated within a factor of $1 + \varepsilon$ for every $\varepsilon > 0$.

1 Introduction

A major question in algorithmic game theory is whether the presence of selfish agents affects the approximability of various classic optimization problems [32]. Specifically, the following research agenda was suggested by Nisan and Ronen [32]: *"to what extent is incentive compatible efficient computation fundamentally less powerful than "classic" efficient computation?"* (see [18] for this formulation of the agenda of

*An extended abstract of this work appears in Proc. SODA 2013.

†Department of Mathematics, University of Haifa, Haifa, Israel. lea@math.haifa.ac.il.

‡Faculty of Industrial Engineering and Management, The Technion, Haifa, Israel. levinas@ie.technion.ac.il.

§Department of Computer Science, University of Leicester, Leicester, LE1 7RH, United Kingdom. rob.vanstee@leicester.ac.uk. Work performed while this author was at Max Planck Institute for Informatics, Germany.

[32]). Of particular interest are scheduling problems, where jobs are assigned for processing to agents, each controlling one machine, and who have some private information regarding their machines [32, 5, 30, 14]. In this paper, we consider the case of single-parameter agents with scheduling problems on uniformly related machines, which was among the first problems considered in the area of algorithmic mechanism design [5]. The private information of an agent is the cost of processing one unit of work, which is also the inverse of the speed of the machine. We provide an answer to the question raised in [32] for scheduling problems on uniformly related machines, by designing $(1+\varepsilon)$-approximation mechanisms for all these problems.

Non-preemptive scheduling problems on m uniformly related machines are defined as follows. We let the set of machines be denoted by $M = \{1, 2, \ldots, m\}$. We are given a set of jobs $J = \{1, 2, \ldots, n\}$, where each job j has a positive size p_j. We let s_i denote the (actual) speed of machine i, meaning that the processing of job j takes $\frac{p_j}{s_i}$ time units if j is assigned to machine i. The jobs need to be partitioned into m subsets $S_1, \ldots, S_m$, with S_i being the subset of jobs assigned to machine i. For such a solution (also known as a schedule), we let $L_i = (\sum_{j \in S_i} p_j)/s_i$ be the *completion time* or *load* of machine i. The *work* of machine i is $W_i = \sum_{j \in S_i} p_j = L_i \cdot s_i$, that is, the total size of the jobs which are assigned to i. We consider objective functions which are functions of the machine loads, $L_1, L_2, \ldots, L_m$ and a variety of objective functions (social goals). A well-known objective function is the *makespan*, which is the maximum load over all machines. The optimization problem of finding a schedule that minimizes the makespan is a basic one [24, 23, 25, 26, 15]. The problem of finding a schedule that maximizes the minimum load, also known as the *cover value*, is the famous *Santa Claus* problem (or the *machine covering* problem) on uniformly related machines (see e.g. [22, 34, 2, 8, 20, 11, 21]). Both these problems are concerned with the optimization of the extremum values of the set of machine loads. We will also consider the optimization problem of minimizing $\sum_{i=1}^{m} f(L_i)$ where f is a well-behaved function. We say that a function f is *well-behaved* if f is a non-negative, convex, strictly monotonically increasing function satisfying the additional property that if $x \leq (1+\varepsilon)y$, then $f(x) \leq (1 + O(1) \cdot \varepsilon) f(y)$. With regard to the problem of minimizing $\sum_{i=1}^{m} f(L_i)$, we assume that there is an oracle such that given a rational number x it computes $f(x)$ exactly in constant time[1]. The most important example of such a function is $f(x) = x^p$ for $p > 1$ in which case the problem is equivalent to minimizing the ℓ_p-norm of the vector of machines loads (if $p = 1$, then an optimal solution which will satisfy our requirements would be to schedule all jobs on the fastest machine of minimum index). The optimization goal function of minimizing the ℓ_2-norm (and the goal of minimizing the ℓ_p-norm for $p > 1$) of the vector of completion times of the machines has been widely studied (see e.g. [17, 13, 7]). The original motivation was minimization of the average latency in storage allocation applications (rather than worst-case latency, see [17]), and the problem has additional applications in algorithmic game theory [12]. Bansal and Pruhs [10] recently stated: "The standard way to compromise between optimizing for the average and optimizing for the worst case is to optimize the ℓ_p-norm, generally for something like $p = 2$ or $p = 3$."

The setup of mechanism design for single-parameter agents operating uniformly related machines is as follows. Agents present bids to a mechanism, where the bid b_i of an agent i is the claimed cost per unit of work of its machine (the inverse of its claimed speed). Based on these bids, the mechanism allocates the jobs to the machines and also assigns payments to the agents. We assume that each agent is only interested in maximizing its own profit, which is its payment minus its (actual) cost of processing the jobs allocated to it. A mechanism is called *truthful* if reporting their true costs per unit of work is a dominant strategy for the agents. That is, this strategy maximizes the profit for each agent, regardless of the strategies of the other agents. In the case of single-parameter agents, a well-known necessary and sufficient condition

[1] We can loosen this condition by replacing f with a piecewise-linear continuous convex approximation of f (i.e., the approximation is well-behaved as well) without affecting the results. We will assume that f can be computed exactly for simplicity.

for truthfulness is that the allocation algorithm be *monotone* [31, 33, 5, 4], that is, the allocation algorithm must have the property that if an agent i increases its claimed speed (i.e., decreases its bid) while all other bids are unchanged, the work allocated to i does not decrease. More precisely, in such a case there exist simple payment functions that can be coupled with the (monotone) allocation algorithm to give a truthful mechanism. If the allocation algorithm runs in polynomial time, and the payments can be computed in polynomial time as well, then the resulting truthful mechanism can be implemented in polynomial time. Thus, for single-parameter agents, since the problems are typically strongly NP-hard, the primary goal is to design a monotone (polynomial time) approximation algorithm with the smallest possible approximation ratio, and to show how the corresponding payments can be computed in polynomial time for its outputs.

Approximation schemes. An $\mathcal{R}$-approximation algorithm for a minimization problem is a polynomial time algorithm which always finds a feasible solution of cost at most $\mathcal{R}$ times the cost of an optimal solution. An $\mathcal{R}$-approximation algorithm for a maximization problem is a polynomial time algorithm which always finds a feasible solution of value at least $\frac{1}{\mathcal{R}}$ times the value of an optimal solution (we use the convention of approximation ratios greater than 1 for maximization problems). The infimum value of $\mathcal{R}$ for which an algorithm is an $\mathcal{R}$-approximation is called the approximation ratio or the performance guarantee of the algorithm. A polynomial time approximation scheme (PTAS) is a family of approximation algorithms such that the family has a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$ (the running time must be polynomial in the input size). If the running time is polynomial in $\frac{1}{\varepsilon}$ as well, then the PTAS is in fact an FPTAS (fully polynomial time approximation scheme). On the other hand, if the running time is quasi-polynomial (logarithmic factors of the input size may appear in the exponent), then the approximation scheme (which is not a PTAS) is a quasi-polynomial time approximation scheme (QPTAS). Being strongly NP-hard[2], the scheduling problems studied here cannot have an FPTAS unless P=NP.

A classic PTAS for these problems generally works by restricting the set of allowable schedules and optimizing (approximately) over this set (thus obtaining an approximate solution), where the details depend on the specific algorithm and the objective function considered. Typically, a chief method of restricting allowed schedules is to do grouping and rounding of jobs (there is a number of subsets, such that the jobs of each subset are seen as identical) and to treat jobs which are very small compared to the work that a machine should receive as arbitrarily divisible (or sand). Several difficulties arise when trying to modify such schemes to satisfy the monotonicity requirement (some of which were partially dealt with in the past, see below). It is no longer possible to treat similar jobs as "identical", and their exact sizes must be considered. Jobs that are small for the machine receiving them are much more difficult to deal with; such jobs usually do not affect the approximation ratio but nevertheless they must be assigned very carefully in order to satisfy the monotonicity requirement, since even a very small reduction in the work when the machine increases its speed is not allowed. Moreover, it is not known in advance which job is small on which machine because the work of the machine is not known.

Dhangwatnotai et al. [18] used randomization to construct a monotone PTAS for the three main objective functions listed above (makespan, cover value, and ℓ_p-norm), which, combined with an appropriate payment function they give, implies a mechanism which is truthful in expectation. That is, given a choice of $\varepsilon > 0$, their algorithm for this value of ε has an approximation ratio of $1+\varepsilon$ for *any* realization, but the monotonicity is proved for the expected works of machines. In this weaker notion of truthfulness, the agents are not interested in their actual profits (which are random variables, as the allocation algorithm is randomized) but only in the *expected* ones, that is, the agents are risk-neutral. For example, if an agent earns a profit of M with probability $\frac{1}{M}$, then they would see their 'expected' profit as earning nothing at all

[2]Using the standard reduction from the 3-Partition problem, deciding if there exists a schedule of the jobs on identical machines for which all machines have unit load is NP-complete in the strong sense.

(rather than earning 1 in expectation). The solution of [18] to the difficulties above is that when a machine receives a job of a given rounded size, the actual job is chosen uniformly at random from the set of jobs of this rounded size, so the "sizes" of jobs (the expected sizes) are easier to deal with. For jobs that are small, a fractional assignment is found (and rounded using randomization). They also derived deterministic monotone QPTAS's for minimizing the maximum load and the ℓ_p-norm of the loads. A fully deterministic (and hence universally truthful) monotone PTAS for minimizing the makespan was given by Christodoulou and Kovács [15]. They assign jobs that have almost the same size (are in the same group) very carefully in a fixed order (sorted by size) to the machines (where machines are given in a fixed order of their speeds). Moreover, they begin by rounding speeds to powers of $1+\varepsilon$, and round the job sizes to powers of $1+\delta$ for some $\delta \ll \varepsilon$. This ensures that when a speed changes, this change is always relatively large compared to the job classification, so the rounding errors introduced by small jobs are not large compared to the required change in the work. The authors give a long and technical proof to show that it is possible to combine these main ideas and give a deterministic monotone assignment. This approach can be used only for minimizing the makespan, since in the scheme of [15], machines of similar speeds should either receive almost the same work (implied by the makespan), or no small jobs at all, except for a small number of machines. Informally, the small jobs are pushed to the fastest machines. This method does not seem to work even for the similar problem of maximizing the cover value (that is also a bottleneck optimization scheduling problem), but applying the methods of [15] leads to a deterministic monotone $(2+\varepsilon)$-approximation for this last objective, given by Christodoulou, Kovács, and van Stee [16] (the problem was also studied in [21]).

What can be seen from these previous results is that satisfying the monotonicity requirement would become easier if we could simply avoid the notion of small jobs. Then, we could calculate with exact job sizes (and thus exact loads) throughout. An important contribution of this paper is to show that for any given schedule (the "original schedule"), a highly structured *similar* schedule exists, where although the ratio of the maximum size and the minimum size of jobs assigned to a machine is *unbounded*, the jobs' types assigned to this machine are restricted in the sense that these jobs are grouped into a sufficiently small number of classes. This allows us to use dynamic programming *without* introducing a notion of small jobs or inexact calculations even though it does not seem to be possible to actually bound the maximum ratio of sizes of jobs assigned to a machine. The set of highly structured schedules is independent of the possible speeds, which assists in dealing with speed changes, and finally, the work of each machine is very close to its work in the original schedule, which keeps the approximation ratio close to 1. This enables us to deal with *all* of the objective functions mentioned above at once using a dynamic programming formulation represented by a layered graph, having one layer for each machine. Unlike previous approximation schemes that use such graphs, a path in the graph corresponds to one specific schedule (not to a class of schedules, or a schedule for a set of rounded jobs), and the cost of the path (with respect to a goal function) is precisely the cost of the corresponding schedule and not its approximated value. That is, there is no rounding or imprecise calculation with respect to relatively small jobs (or any other jobs). This makes proving monotonicity much more straightforward, as our allocation algorithm satisfies the property of *maximal in range* where the range is a set containing all highly structured schedules (and perhaps some additional ones). Moreover, it even simplifies the proof of the approximation ratio, and the presentation of the algorithm, compared to previous (non-monotone) PTAS's. This simplification arises as we do not need to analyze the impact of re-rounding the total size of small jobs, as the work of machines increases, and the size of every unassigned small job become a smaller fraction of the work of the considered machine. Our construction works in the same way for all inputs and all objectives, and does not require any special cases. Hence we streamline the monotone PTAS for minimizing the makespan [15]. Moreover, we provide the first deterministic monotone PTAS's for maximizing the minimum load and minimizing the ℓ_p-norm, which are our main contributions.

Other related work. For a fixed (constant) number of machines, scheduling problems typically have an

FPTAS [27, 9, 19], and even a (deterministic) monotone one for makespan minimization and for maximizing the minimum load [3, 21]. The QPTAS of [18] for minimizing the ℓ_p-norm is in particular a PTAS for fixed values of m. Prior to the monotone FPTAS of Andelman, Azar, and Sorani [3] for makespan minimization, Auletta et al. [6] gave the first deterministic monotone algorithm for this problem (where the number of machines is fixed), with an approximation ratio of $4 + \varepsilon$.

In what follows we discuss the case where the number of machines is a part of the input. It was shown by Hochbaum and Shmoys that the makespan minimization problem has a PTAS for identical (equal speed) machines [25] and for uniformly related machines [26]. All optimization problems studied here, including maximizing the minimum load and minimizing the ℓ_p-norm, are known to have a PTAS for identical machines [34, 1, 2], and for uniformly related machines [8, 20]. As for monotone algorithms for the makespan minimization problem, before the papers [18, 15] mentioned above, Archer and Tardos [5] gave a randomized 3-approximation mechanism for minimizing the makespan which is truthful in expectation only. The ratio was later improved to 2 [4] (and eventually to $1 + \varepsilon$ [18]). A deterministic monotone algorithm of approximation ratio at most 5 was given in [3], and Kovács improved the ratio to 3 and then to 2.8 [28, 29].

Proof overview. Our proof consists of two parts. In the first one, we define several properties which a structured schedule should have, and show that every schedule has a similar schedule fulfilling these properties. Similarity is measured by allowing only a very small change in the work of every machine. For the proof, we introduce a notion of a fractional schedule, where some (relatively small) jobs may be split over multiple machines. For any (integral or fractional) schedule, we can define a magnitude vector with a component for every machine. Unlike previous work, where the magnitude of a machine corresponded directly to its work (or the largest job assigned to it), we use the magnitude component of a machine as an upper bound for the size of any job which is assigned to it, but if a component of the magnitude vector is different from the previous one, we require that the value of this component matches (approximately) the work of the corresponding machine. If the schedule was already defined, its magnitude vector can be created component by component (for a list of machines sorted by non-decreasing speed); increase the magnitude of the current machine (as opposed to keeping the same magnitude of the previous machine) only if keeping the same magnitude as for the previous machine would result in a violation of the upper bound on the maximum size of any job assigned to the current machine. This novel approach allows additional flexibility in the set of allowed schedules.

For a given integral schedule, where the works of the machines are increasing with the speeds, we show (in Lemma 9) that there exists a fractional schedule where the total size of very small jobs which are (partially) assigned to machines with high work (compared to the sizes of these jobs) is small, and the work on each machine is the same as the work in the integral schedule. We then (in Lemma 12) refine this result by constructing an integral schedule where *no* very small jobs are assigned to machines with high work, the works of the machines are all close to the original works, and an additional technical property holds. However, despite the works being close to the original works, they may no longer be sorted in the resulting schedule (though if the works of two consecutive machines are out of order, then the difference between their works is very small). We cannot search for unsorted structured schedules as sorted schedules are essentially required for monotonicity, while a postprocessing step of sorting may also harm monotonicity. We therefore do one extra step (in Theorem 15) to create a final integral schedule in which the works are sorted again (but still very close to the original works) and several structural properties hold. We do not use rounding, but jobs are partitioned into mega-classes and mini-classes according to their size, and we apply re-assignment of jobs in every class to comply with the required structure. For a given schedule, some classes of jobs can turn out to be too large for some machines, while they are very small compared to the work of all the other machines. These jobs are combined into chunks called "alternative jobs". Since this process can be applied in particular for an optimal schedule (for each one of the studied problems), there exists a structured

schedule where works are very close to the works in an optimal schedule, such that this schedule has an objective value that is close to optimum.

Once we show the existence of such a schedule, we can turn to the design of an algorithm which finds it. We use a dynamic programming formulation that is based on the structural properties. By the structural properties and the existence of a magnitude vector, it is only necessary to have a small number of components of this vector in the state space. A preprocessing step is performed, where all possible types of alternative jobs are created. While a job will belong to a number of sets of alternative jobs, every solution will use it at most once as a part of an alternative job (or possibly it will simply be assigned as a job). Thus, we find an optimal solution out of a given class using a polynomial time algorithm, and this optimal schedule is then guaranteed to be close to an overall optimal schedule, as well as being monotone.

2 Preliminaries

Without loss of generality, let ε be a small constant such that $0 < \varepsilon \leq \frac{1}{32}$ and $\frac{1}{\varepsilon}$ is an integer power of 2, where the exponent is denoted by r, such that $r \geq 5$ (i.e., $\varepsilon = \frac{1}{2^r}$). Throughout the paper, for a solution $\mathcal{A}$ we denote by $\mathcal{A}$ both the solution and the value of the objective function for this solution. Without loss of generality, we assume that $0 < p_1 \leq p_2 \leq \cdots \leq p_n$.

An integral schedule is a function $S : J \to M$ mapping each job to its assigned machine. We let $W_i^S = \sum_{j \in J : S(j)=i} p_j$ (this is the work of machine i in the integral schedule S). A fractional schedule is a function $X : J \times M \to [0, 1]$. The value $X(j, i)$ is the fraction of job j assigned to machine i, and the following condition (that every job is assigned completely) must be satisfied:
(F1) For every $j \in J$, $\sum_{i \in M} X(j, i) = 1$.

Let $W_i^X = \sum_{j \in J} p_j \cdot X(j, i)$ be the total fractional size of jobs of machine i, and let $\tilde{W}_i^X = 2^{\alpha_i^X}$, where $\alpha_i^X = \lceil \log_2 W_i^X \rceil$, be its rounded value (if $W_i^X = 0$ then $\alpha_i^X = -\infty$ and $\tilde{W}_i^X = 0$). We call W_i^X the work of machine i in X (as for integral schedules) and $\tilde{W}_i^X$ is the rounded work (also for integral schedules) of machine i in X.

We next define the notion of a valid fractional schedule. Intuitively, it means that if a job j is assigned fractionally to machine i, then its size is small with respect to the work of i. Formally, a fractional schedule is *valid* if it satisfies condition (F2):

(F2) There is a partition $J = J_{\mathbb{Z}}(X) \cup J_{\mathbb{R}}(X)$ ($J_{\mathbb{Z}}(X) \cap J_{\mathbb{R}}(X) = \emptyset$), such that if $j \in J_{\mathbb{Z}}(X)$, then there is a unique value $i \in M$ such that $X(j, i) > 0$ (and therefore $X(j, i) = 1$), and if $j \in J_{\mathbb{R}}(X)$ and $X(j, i) > 0$, then $p_j \leq \varepsilon \tilde{W}_i^X$.

Note that the partition in (F2) is not necessarily uniquely defined (for a given fractional schedule). Every integral schedule S induces a valid fractional schedule X with the same jobs assigned to every machine as follows: let $X(j, i) = 1$ if $S(j) = i$, else $X(j, i) = 0$. Furthermore, we let $J_{\mathbb{R}}(X) = \{j \in J : p_j \leq \varepsilon \tilde{W}_{S(j)}^S\}$ and $J_{\mathbb{Z}}(X) = J \setminus J_{\mathbb{R}}(X)$ (this is one possible partition of J satisfying (F2), and the partition where $J_{\mathbb{R}}(X)$ is empty is another possible partition). Note that $\tilde{W}_i^S = \tilde{W}_i^X$ for $i = 1, \ldots, m$. Let X be called the (valid) fractional schedule induced by S.

On the other hand, every valid fractional schedule X for which $X(j, i) \in \{0, 1\}$ for all $j \in J, i \in M$ induces an integral schedule S with the same works by setting $S(j) = i$ for the value of i for which $X(j, i) = 1$ (this value of i is unique due to (F1)). Let S be called the integral schedule induced by X. In what follows we use the term *schedule* for an integral schedule. We let $L_i^S = \frac{W_i^S}{s_i}$ be the load of machine i in the schedule S.

Our framework relies on the property that there is an optimal solution for which faster machines have

no smaller work. We next show that the objective functions which we consider satisfy this condition (this claim is similar to an observation in [20]).

Claim 1 *Assume that $s_1 \leq s_2 \leq \cdots \leq s_m$. There exists an optimal schedule S for the problem of minimizing $\sum_{i=1}^{m} f(L_i)$ where f is a well-behaved function, which satisfies $W_1^S \leq W_2^S \leq \cdots \leq W_m^S$. There exists an optimal schedule S_1 for the makespan minimization problem which satisfies $W_1^{S_1} \leq W_2^{S_1} \leq \cdots \leq W_m^{S_1}$. There exists an optimal schedule S_2 for the machine covering problem which satisfies $W_1^{S_2} \leq W_2^{S_2} \leq \cdots \leq W_m^{S_2}$.*

Proof. Consider a schedule S with makespan M and cover value C. Call a pair of machines i, j *reversed* if $1 \leq i < j \leq m$ and $W_i^S > W_j^S$. We show that removing a consecutive reversed pair (that is, $j = i + 1$) by swapping the sets of jobs assigned to them from any schedule S does not increase the makespan or decrease the cover value, which implies the second claim and the third claim (since after a finite number of such steps there will no longer be reversed pairs). Let S' be the schedule resulting from swapping the two job sets of machines i,j. In S', machine j gets more work, but the load remains at most M because we have $W_j^{S'}/s_j = W_i^S/s_j \leq W_i^S/s_i \leq M$. Machine i gets less work, but the cover value remains at least C since we have $W_i^{S'}/s_i = W_j^S/s_i \geq W_j^S/s_j \geq C$. To prove the first claim, we show that swapping the sets of jobs of i and j does not increase the objective function value. By swapping, the objective is changed by an additive factor of $f(\frac{W_i^{S'}}{s_i}) + f(\frac{W_j^{S'}}{s_j}) - f(\frac{W_i^S}{s_i}) - f(\frac{W_j^S}{s_j}) = f(\frac{W_j^S}{s_i}) + f(\frac{W_i^S}{s_j}) - f(\frac{W_i^S}{s_i}) - f(\frac{W_j^S}{s_j})$. As $W_i^S > W_j^S$ and $s_i \leq s_j$, we have $\frac{W_j^S}{s_j} \leq \frac{W_j^S}{s_i} < \frac{W_i^S}{s_i}$. By convexity, $f(\frac{W_i^S}{s_i}) + f(\frac{W_j^S}{s_j}) \geq f(\frac{W_j^S}{s_i}) + f(\frac{W_i^S}{s_i} + \frac{W_j^S}{s_j} - \frac{W_j^S}{s_i})$, and by monotonicity, $f(\frac{W_i^S}{s_i} + \frac{W_j^S}{s_j} - \frac{W_j^S}{s_i}) \geq f(\frac{W_i^S}{s_j})$, as $\frac{W_i^S}{s_i} + \frac{W_j^S}{s_j} - \frac{W_j^S}{s_i} \geq \frac{W_i^S}{s_j}$ is equivalent to $(W_i^S - W_j^S)(s_j - s_i) \geq 0$, which holds by assumption. This shows that by swapping, the objective function value does not increase. ■

For all cases, we conclude that if machines are sorted by non-decreasing speed, it is sufficient to consider optimal schedules where the works are non-decreasing (as a function of the indices). This is the main property of the objective functions that we will use in the next section to show the existence of near optimal highly structured solutions.

3 The existence of near-optimal highly structured solutions

We define a partition of the job set J into mega-classes. For $k \in \mathbb{Z}$, let $\mathcal{I}_k = (2^k, 2^{k+1}]$, and let mega-class k be $\{j \in J : p_j \in \mathcal{I}_k\}$. Recall that $2^r = \frac{1}{\varepsilon}$, and we say that an integer k *dominates* the integer k' if $k > k' + r$. Mega-class k *dominates* mega-class k' if k dominates k'. If j, j' belong to mega-classes k, k', respectively, such that mega-class k dominates mega-class k', then $p_{j'} < \varepsilon p_j$. This holds because $p_j > 2^k \geq 2^{k'+r+1} = \frac{1}{\varepsilon} \cdot 2^{k'+1} \geq \frac{1}{\varepsilon} \cdot p_{j'}$, since $k' + r + 1 \leq k$ and $\varepsilon = 2^{-r}$. We refine this partition of J into mega-classes and consider the partition of J into mini-classes as follows. Denote by $K \subseteq \mathbb{Z}$ the set of indices of non-empty mega-classes (clearly $|K| \leq n$). Let $\lambda = \lceil \log_{1+\varepsilon} 2 \rceil$. For $k \in K$ and $0 \leq \ell \leq \lambda - 1$, let $I_{k,\ell} = (2^k \cdot (1+\varepsilon)^\ell, 2^k \cdot (1+\varepsilon)^{\ell+1}]$. The mini-class (k, ℓ) is the set of jobs of mega-class k whose size is in $I_{k,\ell}$. Note that $(1+\varepsilon)^{\lceil \log_{1+\varepsilon} 2 \rceil} \geq (1+\varepsilon)^{\log_{1+\varepsilon} 2} = 2$, and thus the partition of J into the mini-classes is a refined partition of the partition into the mega-classes.

Given a set of consecutive mega-classes $k_1, k_1 + 1, \ldots, k_2$ where $k_2 \geq k_1$, with the job set $\hat{J}$ consisting of all jobs of J with sizes in the interval $(2^{k_1}, 2^{k_2+1}]$, and letting $\varrho = 2^{k_2}$, we create an alternative set of jobs that will possibly replace $\hat{J}$, as follows. These alternative jobs have sizes in the interval $(\varrho, 2\varrho]$ (except perhaps for one alternative job that may be smaller). A simple way to create these alternative jobs is to partition $\hat{J}$ into subsets each of which has total size at most 2ϱ, such that no two subsets can be united

keeping this condition. A set of subsets satisfying this condition has at most one subset whose total size is at most ϱ. We create these subsets by picking in each step a maximal prefix of the jobs in $\hat{J}$ (where $\hat{J}$ is sorted according to the indices of the jobs, i.e., by non-decreasing size) with total size at most 2ϱ and remove the selected jobs from $\hat{J}$. This algorithm is equivalent to applying the bin packing algorithm Next-Fit Increasing (NFI) using "bins" of size 2ϱ. Our algorithm sometimes decides to replace $\hat{J}$ with the alternative jobs, and in this case we partition these alternative jobs into separate mini-classes which we call alternative mini-classes (for $\hat{J}$). The alternative mini-class (k, ℓ) contains all the alternative jobs with size in $I_{k,\ell} \cap I_k$, resulting in at most $\lambda + 1$ alternative mini-classes (λ mini-classes for the interval $\mathcal{I}_{k_1} = (\varrho, 2\varrho]$ and another one for the remaining smaller alternative job, if it exists). If the algorithm decides to replace $\hat{J}$ with alternative jobs, then in the output of the algorithm each alternative job is replaced with the original jobs that were combined to form it, and this is done just before returning the output (the work of each machine is not affected by this change). Since there are at most n non-empty mega-classes, there are $O(n^2)$ different sets $\hat{J}$ that the algorithm possibly replaces with alternative jobs. Thus, creating all the sets of alternative jobs takes $O(n^3)$ time. Note that one job can be contained in multiple alternative jobs, but at most one of these alternative jobs will be used. Given such a set of alternative jobs, we assign indices to the alternative jobs such that a smaller alternative job is assigned a smaller index (breaking ties arbitrarily).

Definition 2 *An integral schedule* respects *the alternative jobs of mega-classes $k_1, k_1 + 1, \ldots, k_2$, where $k_2 \geq k_1$, if every pair of jobs j, j' with sizes in the interval $(2^{k_1}, 2^{k_2+1}]$ of a common alternative job, are scheduled on a common machine.*

The motivation for this definition is that these jobs can be easily replaced by the alternative job to which they belong without affecting the works of the machines. In what follows we will use the notion of a magnitude vector to direct the algorithm. Roughly speaking, our magnitude vector encodes an upper bound on the maximum (approximated) size of a job scheduled so far, as well as the approximated work of some machines (but not all machines).

Definition 3 *A vector $\bar{a} = (a_0, a_1, \ldots, a_m)$ (of length $m + 1$) whose components belong to $\mathbb{Z} \cup \{-\infty\}$ is called a* magnitude vector *if $a_0 = -\infty$, for $i = 0, 1, \ldots, m - 1$, $a_i \leq a_{i+1}$ and if $a_i \neq a_{i+1}$, then a_{i+1} dominates a_i (i.e., $a_{i+1} \geq a_i + r + 1$).*

We now define the *signature vector* $\bar{b}$ of a magnitude vector $\bar{a}$. The number of components in $\bar{b}$ is the number of distinct values among the components of $\bar{a}$ excluding a_0, which is denoted by $\tau(\bar{a})$. Each component $t = 1, 2, \ldots, \tau(\bar{a})$ of $\bar{b}$ is a pair $b_t = (\xi_t, \nu_t)$ such that $\xi_1 = 1$, and for $1 \leq t \leq \tau(\bar{a})$ and $\xi_t \leq i \leq \xi_{t+1} - 1$ (where $\xi_{\tau(\bar{a})+1} = m + 1$) we have $a_i = \nu_t$. That is, for $t \geq 2$, the value ξ_t is always the index of the first machine which has a larger component of $\bar{a}$ than the previous machine and the value of this component is ν_t. That is, the signature vector $\bar{b}$ of a magnitude vector $\bar{a}$ is a compact representation of $\bar{a}$. For each component ν_t of $\bar{a}$, there may be jobs whose sizes are sufficiently close to 2^{ν_t}, specifically, these are jobs of sizes in $(2^{\nu_t - r}, 2^{\nu_t + r + 1}]$. There may also be jobs whose sizes are not contained in such ranges. For every $t = 1, 2, \ldots, \tau(\bar{a}) - 1$, we let $J^t(\bar{a}) = \{j \in J : 2^{\nu_t + r + 1} < p_j \leq 2^{\nu_{t+1} - r}\}$.

Observation 4 *For every job j and every magnitude vector $\bar{a}$ with its signature vector $\bar{b}$, there are at most two values of $t \in \{1, \ldots, \tau(\bar{a})\}$ for which $p_j \in (2^{\nu_t - r}, 2^{\nu_t + r + 1}]$, and if there exists at least one such value of t, then $j \notin \cup_\theta J^\theta(\bar{a})$.*

Proof. By the definitions above, for every j, there are at most two values of t for which $p_j \in (2^{\nu_t - r}, 2^{\nu_t + r + 1}]$ (since $\nu_{\theta+1} \geq \nu_\theta + r + 1$ for every θ). Moreover, if $p_j \in (2^{\nu_t - r}, 2^{\nu_t + r + 1}]$, then for every $\theta < t$, we have $j \notin J^\theta(\bar{a})$ because $p_j > 2^{\nu_t - r} \geq 2^{\nu_{\theta+1} - r}$, and thus j is too large to be in $J^\theta(\bar{a})$. If $\theta \geq t$, then $p_j \leq 2^{\nu_t + r + 1} \leq 2^{\nu_\theta + r + 1}$, and thus j is too small to be in $J^\theta(\bar{a})$. ■

Example. Let $m = 11$ and $r = 10$. Additionally, let $\bar{a} = (-\infty, 0, 0, 0, 0, 12, 12, 50, 50, 50, 100, 100)$. In this case, $\bar{a}$ has four distinct finite values, and $\tau(\bar{a}) = 4$. Moreover, $\bar{b} = ((1, 0), (5, 12), (7, 50), (10, 100))$. The sets $J^t(\bar{a})$ for $t = 1, 2, 3$ are defined as follows. We have $J^1(\bar{a}) = \emptyset$, $J^2(\bar{a}) = \{j \in J : p_j \in (2^{23}, 2^{40}]\}$, $J^3(\bar{a}) = \{j \in J : p_j \in (2^{61}, 2^{90}]\}$. We also have $(2^{\nu_1 - r}, 2^{\nu_1 + r + 1}] = (2^{-10}, 2^{11}]$, $(2^{\nu_2 - r}, 2^{\nu_2 + r + 1}] = (2^2, 2^{23}]$, $(2^{\nu_3 - r}, 2^{\nu_3 + r + 1}] = (2^{40}, 2^{61}]$, and $(2^{\nu_4 - r}, 2^{\nu_4 + r + 1}] = (2^{90}, 2^{111}]$. Consider jobs j_1, j_2, and j_3, where $p_{j_1} = 10^2$, $p_{j_2} = 10^5$, and $p_{j_3} = 10^{22}$. As $2^6 < p_{j_1} < 2^7$, it belongs to $(2^{\nu_1 - r}, 2^{\nu_1 + r + 1}]$ and $(2^{\nu_2 - r}, 2^{\nu_2 + r + 1}]$. As $2^{16} < p_{j_2} < 2^{17}$, it belongs to $(2^{\nu_2 - r}, 2^{\nu_2 + r + 1}]$. As $2^{73} < p_{j_3} < 2^{74}$, it belongs to $J^3(\bar{a})$. See Figure 1 for an illustration of these ranges.

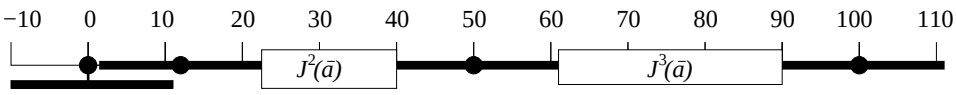

Figure 1: Example (on logarithmic scale). The dots represent the distinct values in the magnitude vector. The thick black bars indicate sizes that are less than a factor 2^r smaller or at most 2^{r+1} larger than the values indicated by the dots. Sizes between two black bars are in some set $J^t(\bar{a})$ as shown.

Definition 5 *A valid fractional schedule X is* consistent *with a magnitude vector $\bar{a}$ if*

1. *for every job j and machine i, if $X(j, i) > 0$, then $p_j \leq 2^{a_i + r + 1}$, that is, machine i does not contain parts of jobs of a mega-class higher than $a_i + r$, and*
2. *if $a_i \neq a_{i-1}$ (for $i \in M$), then $a_i = \alpha_i^X (= \lceil \log_2 W_i^X \rceil)$.*

Observation 6 *If a valid fractional schedule X is consistent with a magnitude vector $\bar{a}$ and $W_1^X \leq W_2^X \leq \cdots \leq W_m^X$, then for every $i \in M$, we have $a_i \leq \alpha_i^X$.*

Favorable pairs. The goal of favorable pairs is to make a first step towards fractional schedules and schedules where there are no very small jobs assigned to machines whose components in the corresponding magnitude vector are large. The following condition ensures, in particular, that the total size of parts of jobs whose mega-class is dominated by mega-class ν_t, assigned to a given machine of index at least ξ_{t+3}, is relatively small compared to the work of that machine. The condition requires that the total size of these parts of jobs assigned to all machines of indices at least ξ_{t+3} is at most $2^{\nu_{t+1} + r + 1} \leq 2^{\nu_{t+2}} < 2^{\nu_{t+3} - r} = \varepsilon \cdot 2^{\nu_{t+3}} = \varepsilon \cdot \tilde{W}_{\xi_{t+3}}^X$.

Definition 7 *A pair $(X, \bar{a})$, where X is a valid fractional schedule, and $\bar{a}$ is a magnitude vector such that X is consistent with $\bar{a}$ is called* favorable *if for $t = 1, 2, \ldots, \tau(\bar{a}) - 3$, we have*

$$\sum_{i=\xi_{t+3}}^{m} \sum_{j : p_j \leq 2^{\nu_t - r}} p_j \cdot X(j, i) \leq 2^{\nu_{t+1} + r + 1} .$$

We define several processes in which a valid fractional schedule is modified into a different valid fractional schedule. These processes are defined algorithmically though they are not a part of the final algorithm, but only of the proof that a highly structured integral schedule must exist (the properties of such a schedule will be discussed later). First consider algorithm Fractional Next Fit Increasing (FNFI) defined as follows.

Algorithm FNFI (Fractional Next Fit Increasing)
Input: A subset of jobs $J' \subseteq J$ and a set of bounds $U_1, \ldots, U_m \geq 0$ (for the m machines) such that $\sum_{j \in J'} p_j = \sum_{i=1}^{m} U_i$.
Output: a fractional allocation of J'.

1. Let $i = 1$ be the first active machine, and for every $j \in J'$ let $q_j = p_j$.
2. While $J' \neq \emptyset$ do:
   (a) Pick the minimum index job $j \in J'$.
   (b) Allocate $\beta = \min\{q_j, U_i\}$ processing time of j to machine i.
   (c) Decrease both U_i and q_j by β.
   (d) If $U_i = 0$, then increase i by 1, and if $q_j = 0$, then remove j from J'.

Algorithm FNFI defines a standard way to schedule jobs that are scheduled fractionally. Observe that since $\sum_{j \in J'} p_j = \sum_{i=1}^{m} U_i$, the event $J' = \emptyset$ occurs at the first iteration where $i = m+1$. FNFI is sometimes used in our proofs to reassign a subset of jobs in a valid fractional schedule such that the total size of jobs of this subset assigned to each machine is unchanged (i.e., the bounds U_i are given by the assignment of the jobs of the subset in the original valid fractional schedule). This is done only in situations where it is ensured that the resulting fractional schedule is valid.

Definition 8 *A valid fractional schedule X is* compatible with FNFI *if running* FNFI *on the input job set $J_{\mathbb{R}}(X)$ with the set of bounds $U_1, \ldots, U_m$ such that $U_i = \sum_{j \in J_{\mathbb{R}}(X)} p_j X(j,i)$ allocates exactly $p_j \cdot X(j,i)$ time units of job j to machine i for every $j \in J_{\mathbb{R}}(X)$ and all $i \in M$, that is, it keeps the valid fractional schedule unchanged.*

We will use the observation that a valid fractional schedule which is compatible with FNFI has at most two fractional jobs assigned to each machine, and these jobs are relatively small with respect to the work of that machine. Thus, it is easy to round such a fractional schedule into an integral one using the procedure which we define now.

Round-FNFI. On several occasions, given a valid fractional schedule X, which is compatible with FNFI, we will apply the following rounding procedure, called Round-FNFI, that creates a schedule. Assign each job $j \in J_{\mathbb{R}}(X)$ completely to the minimum index i such that $X(j,i) > 0$. Since in the assignment process of FNFI each machine receives at most two jobs that are not completely assigned to it, the job of smallest index and the job of largest index, the resulting fractional schedule induces an integral schedule S in which each machine may have additional parts of at most one job (the one of the largest index assigned to this machine by FNFI), and may have less parts of at most one job (the one of the smallest index assigned to this machine by FNFI). Since by condition (F2), the size of each fractional job $j \in J_{\mathbb{R}}(X)$ on machine i (that is, every $j \in J_{\mathbb{R}}(X)$ such that $X(j,i) > 0$) satisfies $p_j \leq \varepsilon \tilde{W}_i^X \leq 2\varepsilon W_i^X$, we conclude that for every $i \in M$ we have $(1-2\varepsilon)W_i^X \leq W_i^S \leq (1+2\varepsilon)W_i^X$. We say that the integral schedule S is created by applying Round-FNFI on X.

Lemma 9 *Given a schedule $S : J \to M$ such that $W_1^S \leq W_2^S \leq \cdots \leq W_m^S$, there exists a favorable pair $(X, \bar{a})$ where $W_i^X = W_i^S$ for $i = 1, 2, \ldots, m$, and X is compatible with* FNFI.

Proof. First, as described in Section 2, S induces a valid fractional schedule, here denoted by X_S, with the same sequence of works. Consider the values α_i^S for $i = 1, 2, \ldots, m$. Since $W_1^S \leq W_2^S \leq \cdots \leq W_m^S$,

we have $\alpha_i^S \leq \alpha_{i+1}^S$ for all $i = 1, 2, \ldots, m-1$. We define a magnitude vector $\bar{a}^S = (a_0^S, a_1^S, \ldots, a_m^S)$ as follows. We let $a_0^S = -\infty$. For $i = 1, 2, \ldots, m$, if $\alpha_i^S \leq a_{i-1}^S + r$, then $a_i^S = a_{i-1}^S$, and otherwise $a_i^S = \alpha_i^S$. The valid fractional schedule X_S is consistent with $\bar{a}^S$ since for every $i \in M$, either $a_i^S = \alpha_i^S$ or $\alpha_i^S \leq a_{i-1}^S + r = a_i^S + r$. In both cases, the size of any job assigned (completely) to machine i cannot exceed $W_i^{X_S} \leq \tilde{W}_i^{X_S} = 2^{\alpha_i^S} \leq 2^{a_i^S + r}$.

We next consider the set of pairs consisting of a valid fractional schedule and a magnitude vector $(X', \bar{a})$ such that X' is consistent with $\bar{a}$, and such that for $i = 1, 2, \ldots, m$, $W_i^{X'} = W_i^S$ and $a_i \leq \alpha_i^{X'}$ (the set is nonempty by the existence of $(X_S, \bar{a}^S)$). Among all the possible choices for X' and $\bar{a}$, we consider one such that the vector $\bar{a}$ has a signature vector with the smallest number of components, and (as a secondary objective, i.e., among such solutions which minimize the number of components in the signature vector) $|J_{\mathbb{R}}(X')|$ is maximized. Such a pair of a magnitude vector and its signature vector is well-defined because the number of values that can be achieved by this bicriteria objective is finite and thus the optimum is attained. Based on X', we will define another fractional schedule X (by applying FNFI on $J_{\mathbb{R}}(X')$), and X will be shown to be a valid fractional schedule satisfying the lemma.

We modify X' by reassigning the jobs of $J_{\mathbb{R}}(X')$ using FNFI with the set of bounds $U_1, \ldots, U_m$ such that $U_i = \sum_{j \in J_{\mathbb{R}}(X')} p_j X'(j, i)$. We denote the resulting fractional schedule which is compatible with FNFI by X where we define $J_{\mathbb{R}}(X) = J_{\mathbb{R}}(X')$. It is obvious that X satisfies (F1), and we argue that X satisfies (F2) as well. To prove this, we will show that if $j \in J_{\mathbb{R}}(X)$ and $i \in M$ satisfy that $X(j, i) > 0$, then $p_j \leq \varepsilon \tilde{W}_i^X$. Since the works of the machines are sorted in a non-decreasing order and X' is valid, it suffices to show that for $j \in J_{\mathbb{R}}(X)$ and i such that $X(j, i) > 0$, there exists $j' \geq j$, $j' \in J_{\mathbb{R}}(X')$ and $i' \leq i$ such that $X'(j', i') > 0$, since in such a case $p_j \leq p_{j'} \leq \varepsilon \tilde{W}_{i'}^{X'} \leq \varepsilon \tilde{W}_i^X$. Assume by contradiction that this claim does not hold for j and i. Thus, in X', machines $1, 2, \ldots, i$ may only have parts of jobs of $J_{\mathbb{R}}(X')$ with indices no larger than $j-1$. In X, as j is assigned to i by FNFI (at least partially), jobs of $J_{\mathbb{R}}(X) = J_{\mathbb{R}}(X')$ of indices at most $j-1$ are only assigned to machines $1, 2, \ldots, i$, and they do not occupy this space completely (due to j occupying some space on machine i). We find $\sum_{j' \in J_{\mathbb{R}}(X'): j' < j} p_{j'} = \sum_{j' \in J_{\mathbb{R}}(X): j' < j} p_{j'} < \sum_{\gamma=1}^{i} U_\gamma$ and $\sum_{j' \in J_{\mathbb{R}}(X'): j' < j} p_{j'} \geq \sum_{\gamma=1}^{i} U_\gamma$. Therefore, X is indeed a valid fractional schedule.

We claim that X is consistent with $\bar{a}$. It suffices to prove that in every prefix of machines $1, 2, \ldots, i$, the maximum size of a job j such that $X(j, \gamma) > 0$ for some $1 \leq \gamma \leq i$ does not increase when we replace X' by X. Let j be a job of maximum size which is assigned in X (possibly fractionally) to a machine $\gamma \in \{1, 2, \ldots, i\}$. If $j \in J_{\mathbb{Z}}(X) = J_{\mathbb{Z}}(X')$ then $X'(j, \gamma) = X(j, \gamma) = 1$, and the claim holds. Otherwise, $j \in J_{\mathbb{R}}(X)$. There exists $j' \in J_{\mathbb{R}}(X')$ and $i' \leq \gamma$ such that $X'(j', i') > 0$ and $j' \geq j$ as we showed above, and the claim holds as well.

Last, we prove that $(X, \bar{a})$ is a favorable pair. Let t be such that $1 \leq t \leq \tau(\bar{a}) - 3$. Let $j \in J$ be such that there is $i \in [\xi_{t+3}, m]$ with $X(j, i) > 0$ and $p_j \leq 2^{\nu_t - r}$. If there is no such job, then we are done. We have $j \in J_{\mathbb{R}}(X) = J_{\mathbb{R}}(X')$ because $\tilde{W}_i^X \geq 2^{a_i} \geq 2^{\nu_{t+3}} > 2^{\nu_t} \cdot 2^{3r} = \frac{1}{\varepsilon^3} \cdot 2^{\nu_t} \geq \frac{1}{\varepsilon^3} p_j$ where the first inequality holds by Observation 6, so if $j \notin J_{\mathbb{R}}(X)$ then $X(j, i) = 1$ and we can add j to $J_{\mathbb{R}}(X)$, contradicting our choice of X'. Consider the machines $A_{t+1} = \{\xi_{t+1}, \ldots, \xi_{t+2} - 1\}$. If all jobs assigned (possibly fractionally) by X to these machines have sizes of at most $2^{\nu_t + r + 1}$, then we can redefine $a_{i'}$ for all $i' \in A_{t+1}$ to be ν_t, contradicting the minimality of the number of components of the signature vector of $\bar{a}$. Consider a job j' such that there is $i' \in A_{t+1}$ for which $X(j', i') > 0$ and $p_{j'} > 2^{\nu_t + r + 1}$. By the existence of $j \in J_{\mathbb{R}}(X)$ with size at most $2^{\nu_t - r}$, such that a part of it is allocated to a machine of higher index, we conclude that $j' \in J_{\mathbb{Z}}(X)$ since X is compatible with FNFI. We also have $p_{j'} \leq 2^{\nu_{t+1} + r + 1} \leq 2^{\nu_{t+3} - r - 1} < \varepsilon \cdot \tilde{W}_{\xi_{t+3}}^X$. If $\sum_{\gamma=\xi_{t+3}}^{m} \sum_{j'': p_{j''} \leq 2^{\nu_t - r}} p_{j''} \cdot X(j'', \gamma) > 2^{\nu_{t+1} + r + 1}$, then $\sum_{\gamma=\xi_{t+3}}^{m} \sum_{j'': p_{j''} \leq 2^{\nu_t - r}} p_{j''} \cdot X(j'', \gamma) > p_{j'}$. In this case, we add j' to $J_{\mathbb{R}}(X)$, and modify X as follows. We consider a replacement of the position of j' with the position of a set of parts of jobs (where

each such job has size at most $2^{\nu_t - r} = \varepsilon 2^{\nu_t} \leq \varepsilon^2 2^{\nu_{t+1}-1} \leq \varepsilon^2 \frac{\tilde{W}^X_\gamma}{2} < \varepsilon^2 W^X_\gamma$ for every $\gamma \in A_{t+1}$, and belongs to $J_{\mathbb{R}}(X'))$ of total size $p_{j'}$ which were previously assigned to machines with index at least ξ_{t+3}. The resulting schedule indeed satisfies (F2) since the jobs which take the place of j' are smaller than $\varepsilon^2 W^X_\gamma$ for every $\gamma \in A_{t+1}$ while $p_{j'} < \varepsilon \cdot \tilde{W}^X_{\xi_{t+3}}$. Thus, the resulting valid fractional schedule is consistent with $\bar{a}$, contradicting our choice of X' since $|J_{\mathbb{R}}(X')|$ is not maximal among valid fractional schedules consistent with $\bar{a}$ (and having the required properties). ■

Good schedules and schedules that are almost consistent with magnitude vectors. In the next step, we would like to transform the fractional schedule X satisfying the properties which we established in the last lemma into an integral schedule satisfying similar properties (some of these are relaxed versions of the properties that X satisfies). One of our goals is to create a relationship between the works of a schedule and its magnitude vector, while the other goal is to schedule jobs only on machines whose components of the magnitude vector are sufficiently related to the sizes of these jobs.

Definition 10 *A schedule S is* almost consistent *with a magnitude vector $\bar{a}$ if for every $i = 1, 2, \ldots, m$, the set of jobs assigned to machine i does not contain any job of a mega-class higher than $a_i + r$, and if $a_i \neq a_{i-1}$ (for $i \in M$) then $|a_i - \alpha^S_i| \leq 1$.*

Observe that if the valid fractional schedule X_S induced by the schedule S is consistent with a magnitude vector $\bar{a}$, then S is almost consistent with $\bar{a}$ (but the other direction does not necessarily hold).

Definition 11 *A schedule $S : J \to M$ is good if the following properties hold.*
1. There exists a magnitude vector $\bar{a}$ such that S is almost consistent with $\bar{a}$, and furthermore for every $t = 1, 2, \ldots, \tau(\bar{a}) - 4$ there is no j and $i \geq \xi_{t+4}$ such that $p_j \leq 2^{\nu_t - r}$ and $S(j) = i$.
2. For every $t = 1, 2, \ldots, \tau(\bar{a}) - 1$, if $J^t(\bar{a}) = \{j \in J : 2^{\nu_t + r + 1} < p_j \leq 2^{\nu_{t+1} - r}\} \neq \emptyset$, then S respects the alternative jobs of mega-classes $\nu_t + r + 1, \nu_t + r + 2 \ldots, \nu_{t+1} - r - 1$.

Lemma 12 *Given a schedule $S : J \to M$ such that $W^S_1 \leq W^S_2 \leq \cdots \leq W^S_m$, there exists a good schedule $S' : J \to M$ such that for $i = 1, 2, \ldots, m$, we have*

$$(1 - 12\varepsilon) \cdot W^S_i \leq W^{S'}_i \leq (1 + 12\varepsilon) \cdot W^S_i. \tag{1}$$

Proof. By Lemma 9, there exists a favorable pair $(X, \bar{a})$ where $W^X_i = W^S_i$ for $i = 1, 2, \ldots, m$, and X is compatible with FNFI. We will transform X into a good schedule, denoted by S', in two steps. First, for every $t = 4, 5, \ldots, \tau(\bar{a})$ (in an increasing order of t), we reschedule all parts of jobs j such that $p_j \leq 2^{\nu_{t-3} - r}$ and for which there exists $i > \xi_t$ such that $X(j, i) > 0$ by moving them to machine ξ_t (from all such machines $i > \xi_t$). We denote the resulting fractional schedule by $\tilde{X}$. We next bound the value of $W^{\tilde{X}}_i$ in terms of W^X_i for every $i \in M$. The work of i may increase (if $i = \xi_t$ for some $t = 4, 5, \ldots, \tau(\bar{a})$). Since $(X, \bar{a})$ is a favorable pair, the amount of this increase is at most $\varepsilon \cdot \tilde{W}^X_i < 2\varepsilon W^X_i$, since $2^{\nu_{t-2} + r + 1} < 2^{\nu_t - r} = \varepsilon \tilde{W}^X_i$. Next, we bound the total size of parts of jobs removed from machine i (for $2 \leq i \leq m$). Let t' be the maximum index such that $\xi_{t'} < i$ (which must exist since $\xi_1 = 1$). Then, for every $t = 4, 5, \ldots, t'$, we may have removed a total size of at most $2^{\nu_{t-2} + r + 1} \leq \frac{\varepsilon}{2} \cdot 2^{\nu_t}$ from machine i (and move these parts of jobs to machine ξ_t). Thus $W^X_i - W^{\tilde{X}}_i \leq \frac{\varepsilon}{2} \cdot \sum_{t=4}^{t'} 2^{\nu_t} \leq \varepsilon \cdot 2^{\nu_{t'}} \leq 2\varepsilon \cdot W^X_i$. We conclude that for every i, we have $(1 - 2\varepsilon) W^X_i \leq W^{\tilde{X}}_i \leq (1 + 2\varepsilon) W^X_i$.

Let $J_{\mathbb{R}}(\tilde{X}) = J_{\mathbb{R}}(X)$. We observe that $\tilde{X}$ is a valid fractional schedule which is compatible with FNFI (similarly to the bounds on such jobs in Lemma 9, if a job is moved to machine i, then its size is at most $2^{\nu_{t-3} - r} \leq \varepsilon^4 2^{\nu_t}$, while $\varepsilon W^{\tilde{X}}_i \geq \varepsilon W^X_i \cdot (1 - 2\varepsilon) \geq \varepsilon(1 - 2\varepsilon) 2^{\nu_t - 1} > \varepsilon^2 2^{\nu_t}$, since $\varepsilon \leq 1/32$). We now

apply Round-FNFI on $\tilde{X}$ to create an integral schedule $\tilde{S}$. Every $j \in J_{\mathbb{R}}(\tilde{X})$ such that $\tilde{X}(j,i) > 0$ has size $p_j \leq \varepsilon \tilde{W}_i^X \leq 2\varepsilon W_i^X$, so for every $i \in M$ we have

$$(1-4\varepsilon)W_i^X \leq W_i^{\tilde{S}} \leq (1+4\varepsilon)W_i^X \ . \tag{2}$$

The maximum size of a job in a prefix of machines in $\tilde{S}$ is the same as in $\tilde{X}$, and a job moved from its position in X to a new position on machine ξ_t in $\tilde{X}$ has size at most $2^{\nu_t - 3} < \varepsilon 2^{\nu_t} = \varepsilon 2^{a_{\xi_t}}$.

Now, we will do the second step of our transformation. Consider the set of jobs $J^t(\bar{a})$. Since X is consistent with $\bar{a}$, for every $j \in J^t(\bar{a})$ and $i < \xi_{t+1}$, we have $X(j,i) = 0$, and since the maximum size of a job in a prefix of machines did not change, $\tilde{S}(j) > i$. Since $\tilde{W}^X_{\xi_{t+1}} = 2^{\nu_{t+1}}$, we have for all $j \in J^t(\bar{a})$ and $i \geq \xi_{t+1}$ that $p_j \leq 2^{\nu_{t+1}-r} = \varepsilon \cdot \tilde{W}^X_{\xi_{t+1}} \leq \varepsilon \cdot \tilde{W}_i^X$. We remove the jobs in $J^t(\bar{a})$ from their positions in $\tilde{S}$, and we will schedule the alternative jobs instead, using FNFI (which gives a schedule of the original jobs which respects the alternative jobs of mega-classes $\nu_t + r + 1, \nu_t + r + 2, \ldots, \nu_{t+1} - r - 1$). For every $i \in M$, we let U_i be the total size of jobs in $J^t(\bar{a})$ which are assigned to machine i by $\tilde{S}$. The set of machines i for which $U_i \neq 0$ is contained in the interval $[\xi_{t+1}, \xi_{t+4}]$ where if $t + 4 > \tau(\bar{a})$, then we let $\xi_{t+4} = m$. We apply FNFI to fractionally schedule the alternative jobs, followed by Round-FNFI. This is done for every value of t for which $J^t(\bar{a}) \neq \emptyset$ sequentially. We denote by S' the resulting integral solution. Let $i \in M$. There are at most four values of t for which i participated in the process of the rescheduling of $J^t(\bar{a})$. As a result of applying Round-FNFI for the alternative jobs for all t, every machine i can have at most four additional parts of alternative jobs and less parts of at most four alternative jobs, all of which have size of at most $\varepsilon \tilde{W}_i^X \leq 2\varepsilon W_i^X$. Thus, $W_i^{\tilde{S}} - 8\varepsilon W_i^X \leq W_i^{S'} \leq W_i^{\tilde{S}} + 8\varepsilon W_i^X$. Using (2), we get (1).

The integral schedule S' is almost consistent with the magnitude vector $\bar{a}$. To see this claim, first observe that no job is too large: if the maximum size of a job on machine i in S' is not the same as in $\tilde{S}$, this maximum size job $j \in J^t(\bar{a})$ is moved from its position in $\tilde{S}$ to a new position on machine i, and therefore $p_j \leq 2^{\nu_{t+1}-r} = \varepsilon 2^{\nu_{t+1}}$, and $a_i \geq a_{\xi_{t+1}} = \nu_{t+1}$. The claim holds because for every i, we have $|\alpha_i^X - \alpha_i^{S'}| \leq 1$ since $1 + 12\varepsilon < 2$ and $\frac{1}{1-12\varepsilon} < 2$. ■

Structured schedules and schedules that are quasi-consistent with magnitude vectors. We now turn to discuss the kind of schedules that we are interested in. The following relaxation of consistency allows to schedule a job j on a machine subject to the constraint that j is not too large with respect to the next value of the signature vector.

Definition 13 *A schedule S is* quasi-consistent *with a magnitude vector $\bar{a}$ if for every $i = 1, 2, \ldots, m$ such that $\xi_t \leq i < \xi_{t+1}$, the set of jobs assigned to machine i does not contain any job of a mega-class higher than $\nu_{t+1} + r$, and if $a_i \neq a_{i-1}$ (for $i \in M$) then $|a_i - \alpha_i^S| \leq 1$.*

Definition 14 *A schedule $S : J \to M$ is* structured *if the following properties hold.*

1. *There exists a magnitude vector $\bar{a}$ such that S is quasi-consistent with $\bar{a}$, and furthermore for every $t = 1, 2, \ldots, \tau(\bar{a}) - 5$ there is no j and $i \geq \xi_{t+5}$ such that $p_j \leq 2^{\nu_t - r}$ and $S(j) = i$.*
2. *For every $t = 1, 2, \ldots, \tau(\bar{a}) - 1$, if $J^t(\bar{a}) \neq \emptyset$, then S respects the alternative jobs of mega-classes $\nu_t + r + 1, \ldots, \nu_{t+1} - r - 1$.*
3. *$W_1^S \leq W_2^S \leq \cdots \leq W_m^S$.*
4. *For each pair of jobs $j, j' \notin \cup_t J^t(\bar{a})$ belonging to a common mini-class, if $j < j'$, then $S(j) \leq S(j')$.*

5. *For each pair of alternative jobs j, j' resulting from the set $J^t(\bar{a})$ belonging to a common alternative mini-class such that the index of j is smaller than the index of j', the following holds. If S schedules the original jobs in j and j' on machines i and i', respectively, then $i \leq i'$.*

Theorem 15 *Given a schedule $S : J \to M$ such that $W_1^S \leq W_2^S \leq \cdots \leq W_m^S$, there exists a structured schedule $S^* : J \to M$ such that for $i = 1, 2, \ldots, m$, we have*

$$(1 - 14\varepsilon) \cdot W_i^S \leq W_i^{S^*} \leq (1 + 14\varepsilon) \cdot W_i^S \ . \tag{3}$$

Proof. Let S' be the good schedule that is based on S as established in Lemma 12. We apply a careful sorting procedure of the works of the machines similarly to the one of [15]. In this procedure we are given as an input a partition of the jobs into subsets $\mathcal{J}_1, \ldots, \mathcal{J}_m$, and we create a new partition of the jobs. The new partition is created within $m - 1$ steps, such that in step i only the set of jobs for machine i is selected (and the set for machine m will consist of the remaining jobs which were not selected to be in the set of any machine out of $1, 2, \ldots, m - 1$). We will define how to select the set for machine i without modifying the sets of machines $1, 2, \ldots, i - 1$, and prove that the resulting work of machine i is no smaller than that of machine $i - 1$ (if $i \geq 2$). The selection is done as follows. For every mini-class (including the alternative mini-classes), consider the sorted list of the jobs of this mini-class. The jobs of each mini-class are sorted in a fixed order by non-decreasing size, which for actual mini-classes means sorting according to indices. Given a mini-class, we would like to assign smaller jobs of this mini-class to machines of smaller indices. For alternative mini-classes, the original jobs will not necessarily have to be assigned in this way (sorted by indices), but they will always be considered in one specific (fixed) sorted order (implied by the fixed sorted order by the total size of the alternative jobs). For each set of jobs $\mathcal{J}_i, \ldots, \mathcal{J}_m$ (these sets may have been modified since they were given as an input, when the sets of machines $1, \ldots, i - 1$ were determined), for every mini-class, temporarily replace the sizes of its jobs by the sizes of the smallest jobs of this mini-class (that is, if the set has n_i jobs of a given mini-class, replace them with the n_i first jobs of this mini-class in the sorted list). This results in $m - i + 1$ values of possible work. Select the smallest of these values, and assume that this value was achieved for the set $\mathcal{J}_{i'}$ (breaking ties arbitrarily). This value (of the work) is denoted by Q_i. By swapping jobs with other sets of jobs out of $\{\mathcal{J}_i, \ldots, \mathcal{J}_m\} \setminus \{\mathcal{J}_{i'}\}$, create a subset of jobs of total size Q_i (that is, $\mathcal{J}_{i'}$ will receive the smallest jobs for each mini-class or alternative mini-class). Swap the set $\mathcal{J}_i$ and $\mathcal{J}_{i'}$ (if $i \neq i'$). The set $\mathcal{J}_i$ has now been finalized. Note that if $i \geq 2$, then when $\mathcal{J}_{i-1}$ was determined, the potential work of the set $\mathcal{J}_i$ (with its final subset of jobs) was at least its final work (as the jobs of all mini-classes that were available when $\mathcal{J}_i$ was determined were also available when $\mathcal{J}_{i-1}$ was determined). Thus, as $\mathcal{J}_{i-1}$ was preferred to $\mathcal{J}_i$, $Q_{i-1} \leq Q_i$.

We apply the sorting procedure on the partition defined by S'. The output of this procedure is an integral schedule denoted by S^*. Clearly, $W_1^{S^*} \leq W_2^{S^*} \leq \cdots \leq W_m^{S^*}$. Moreover, properties 2, 4, and 5 in the definition of structured schedules are satisfied. We next prove the bounds on the work of i given by (3) for every machine i. Every machine i receives a subset of jobs which is based on a subset of jobs allocated to some machine i' in S', after swapping pairs of jobs within a common mini-class. Therefore,

$$\frac{W_{i'}^{S'}}{1 + \varepsilon} \leq W_i^{S^*} \leq (1 + \varepsilon) \cdot W_{i'}^{S'} \ . \tag{4}$$

Fix a machine index i. When we choose the set $\mathcal{J}_i$ for machine i, at least one of the original i subsets $\mathcal{J}_1, \ldots \mathcal{J}_i$ (up to swapping some locations of pairs of jobs within common mini-classes and alternative mini-classes) remains available. Let i'' denote such an index. The total size of the jobs in this subset is at most $W_{i''}^{S'} \cdot (1+\varepsilon) \leq W_{i''}^{S} \cdot (1+12\varepsilon)(1+\varepsilon) \leq W_i^S \cdot (1+12\varepsilon)(1+\varepsilon)$ where the first inequality holds by Lemma 12, and the second inequality holds by the monotonicity of works in S. Therefore, machine i receives in S^*

a total work of at most $W_i^S \cdot (1+14\varepsilon)$. We next prove the other inequality, that is, $(1-14\varepsilon) \cdot W_i^S \leq W_i^{S^*}$. The schedule S has at most $i-1$ machines which receive work strictly below W_i^S. Therefore, in S' there are at most $i-1$ machines which receive work strictly below $W_i^S \cdot (1-12\varepsilon)$. In S^* the number of machines with work strictly smaller than $\frac{W_i^S \cdot (1-12\varepsilon)}{1+\varepsilon}$ cannot exceed $i-1$, and due to the monotonicity of the works in S^*, the claim holds.

Finally, we prove property 1 of structured schedules. We first show that the integral schedule S^* is quasi-consistent with the magnitude vector $\bar{a}$. For every i, we have $|\alpha_i^S - \alpha_i^{S^*}| \leq 1$ since $1+14\varepsilon < 2$ and $\frac{1}{1-14\varepsilon} < 2$. Intuitively, we will show that a job set is not exchanged with another job set which is two magnitudes (or more) away. To see this claim, consider a machine i such that $\xi_t \leq i < \xi_{t+1}$ and denote by j the maximum sized job on machine i according to S^*. We need to prove that $p_j \leq 2^{\nu_{t+1}+r+1}$. The set of jobs $\mathcal{J}_i$ which the sorting procedure allocated to machine i was scheduled on a machine i' in S' (possibly swapping pairs of jobs in common mini-classes). A job j' of the same mini-class as j was allocated to machine i' in S'. Recall that $\frac{W_{i'}^{S'}}{1+\varepsilon} \leq W_i^{S^*} \leq (1+\varepsilon) \cdot W_{i'}^{S'}$. By Lemma 12, S' is almost consistent with $\bar{a}$, and therefore $p_{j'} \leq 2^{a_{i'}+r+1}$. Since j and j' belong to a common mini-class, they also belong to a common mega-class, and thus we also have $p_j \leq 2^{a_{i'}+r+1}$. In order to prove that $p_j \leq 2^{\nu_{t+1}+r+1}$ it suffices to show that $i' < \xi_{t+2}$. Assume by contradiction that $i' \geq \xi_{t+2}$. We have $W_{i'}^S \geq W_{\xi_{t+2}}^S > 2^{\nu_{t+2}-1}$ which holds since the works in S are monotonically non-decreasing and $2W_{\xi_{t+2}}^S > \tilde{W}_{\xi_{t+2}}^S = 2^{\alpha_{\xi_{t+2}}^S} = 2^{\nu_{t+2}}$. On the other hand, $W_i^S \leq W_{\xi_{t+1}}^S \leq 2^{\nu_{t+1}}$. By inequality (4) and Lemma 12, we have $W_i^{S^*} \geq \frac{W_{i'}^{S'}}{1+\varepsilon} \geq \frac{W_{i'}^S(1-12\varepsilon)}{1+\varepsilon} \geq W_{i'}^S(1-14\varepsilon)$. Therefore, we get

$$
\begin{aligned}
2^{\nu_{t+1}} &\geq W_i^S \\
&\geq \frac{W_i^{S^*}}{1+14\varepsilon} && \text{by (3)} \\
&\geq \frac{W_{i'}^S(1-14\varepsilon)}{1+14\varepsilon} \\
&> 2^{\nu_{t+2}-1} \cdot \frac{1-14\varepsilon}{1+14\varepsilon} \\
&> 2^{\nu_{t+2}-3} \geq 2^{\nu_{t+1}+r-2}\,, && \text{since } \varepsilon \leq \frac{1}{32}
\end{aligned}
$$

contradicting $r \geq 5$. Therefore, S^* is quasi-consistent with the magnitude vector $\bar{a}$.

The last part of the proof is symmetric to the above proof that S^* is quasi-consistent with $\bar{a}$. Fix a value of $t = 1, 2, \ldots, \tau(\bar{a}) - 5$, it remains to prove that there is no j and $i \geq \xi_{t+5}$ such that $p_j \leq 2^{\nu_t - r}$ and S^* schedules job j to machine i.

Consider a machine i such that $\xi_{t+5} \leq i < \xi_{t+6}$. The set of jobs $\mathcal{J}_i$ which the sorting procedure allocated to machine i, was scheduled on a machine i' in S' (possibly swapping pairs of jobs in common mini-classes). Let j be a job such that $S^*(j) = i$. In order to prove that $p_j > 2^{\nu_t - r}$ it suffices to show that $i' \geq \xi_{t+4}$. Assume by contradiction that $i' < \xi_{t+4}$. We have $W_{i'}^S \leq W_{\xi_{t+4}}^S \leq 2^{\nu_{t+4}}$ which holds since the works in S are monotonically non-decreasing and $W_{\xi_{t+4}}^S \leq \tilde{W}_{\xi_{t+4}}^S = 2^{\alpha_{\xi_{t+4}}^S} = 2^{\nu_{t+4}}$. On the other hand, $W_i^S \geq W_{\xi_{t+5}}^S > 2^{\nu_{t+5}-1}$. By (4) and Lemma 12, we have $W_i^{S^*} \leq W_{i'}^{S'} \cdot (1+\varepsilon) \leq W_{i'}^S(1+12\varepsilon)(1+\varepsilon) \leq$

$W_{i'}^S(1+14\varepsilon)$. Therefore, using (3) we get

$$\begin{aligned}
2^{\nu_{t+5}} &\leq 2W_i^S \\
&\leq \frac{2W_i^{S^*}}{1-14\varepsilon} && \text{by (3)} \\
&\leq \frac{2W_{i'}^S(1+14\varepsilon)}{1-14\varepsilon} \\
&\leq 2^{\nu_{t+4}+1} \cdot \frac{1+14\varepsilon}{1-14\varepsilon} \\
&< 2^{\nu_{t+4}+3} \leq 2^{\nu_{t+5}-r+2}\,, && \text{since } \varepsilon \leq \frac{1}{32}
\end{aligned}$$

contradicting $r \geq 5$. ■

4 The scheme and its analysis: an overview

Our scheme is based on computing the optimal structured schedule S^*. Our algorithm will use a dynamic programming procedure which is based on a shortest path (or an optimal bottleneck path) in a directed layered graph $G = (V, E)$ with weights on its vertices. Each layer of an index $1, 2, \ldots, m$ corresponds to a machine, and each vertex in one of these layers encodes a set of jobs which were scheduled prior to the current machine (if the current vertex is reached, then the jobs were assigned to machines of smaller indices), and a set of jobs that were scheduled up to and including the current machine. The difference between these sets easily reveals the work of the current machine, and allows us to restrict the paths in the graph to schedules in which the works are monotonically non-decreasing. To prioritize the possible outputs, we number all vertices of each layer with distinct integers, and we always search for paths whose reverse sequence of numbers along the path is minimal (lexicographically) out of paths which have an optimal cost with respect to our goal function. As we are interested in finding a structured schedule, the dynamic programming formulation will basically test the possible magnitude vectors (but for each vertex it will only remember a constant number of components of the signature vector). As the component of the magnitude vector that corresponds to a specific machine determines which jobs can be assigned to this machine, the smaller jobs must be assigned to machines of smaller indices. The fourth condition of structured schedules implies a very specific assignment for similar jobs. Moreover, the magnitude vector determines the alternative jobs. See Section 5 for further details.

The claim that the resulting solution is an approximation scheme (for each of the problems which we consider) is a trivial consequence of Theorem 15. The monotonicity proof is based on analyzing a scenario where a machine changes its speed. There are two basic changes that we analyze. In the first one, a machine increases or decreases its speed but remains in the same position in the sorted list of machines. In the second case, a pair of machines with equal speeds change their relative position in the sorted list of machines (without changing their speed). The 'concatenation' of a finite number of basic changes resulted in a scenario in which the machine changes its speed. The proof of monotonicity is based on the property that the dynamic programming finds an optimal solution subject to some constraints, and it is also heavily based on the fact that the works of the machines are monotonically non-decreasing and the details of the tie-breaking rule. We refer to Section 6 for the complete proofs of these results which we summarize as follows.

Theorem 16 *There are monotone PTAS's for the problems of minimizing* $\sum_{i=1}^{m} f(L_i)$ *where* f *is a well-behaved function, maximizing* $\min_{i \in M} L_i$*, and minimizing* $\max_{i \in M} L_i$*.*

We further note that we can use the payment scheme due to Archer and Tardos [5] to create a truthful mechanism. To do so we compute the payment for each agent by calculating (exactly) the integral of the work function. We refer to Section 7 for details.

5 A dynamic programming formulation for computing the best highly structured solution

In this section we show how to compute the optimal structured schedule S^*. Our algorithm will use a dynamic programming procedure which is based on a shortest path (or an optimal bottleneck path) in a directed layered graph $G = (V, E)$ with weights on its vertices.

We define a layered graph, in which the algorithm computes a path corresponding to an optimal solution with respect to a given goal function. Each layer of an index $1, 2, \ldots, m$ corresponds to a machine, and each vertex in one of these layers encodes a set of jobs which were scheduled to machines of indices smaller than the index of the current machine, and a set of jobs which were scheduled up to and including the current machine. The difference between the total sizes of jobs of these sets is the work of the current machine, and the property that the values of the work are completely defined by the properties of the vertices allows us to restrict the paths in the graph to schedules in which the works are monotonically non-decreasing. Given the work W_i of the current machine i, the weight of the vertex is the load of this machine L_i, or $f(L_i)$ for a well-behaved function f. The edges between layers correspond to compatibility conditions which in particular enforce the condition that the works of machines are monotonically non-decreasing. The order of the layers is according to the speeds of the machines, that is, machines with higher speeds have a higher index of their layers, and any subset of machines with equal speeds are ordered according to a fixed ordering of the machines. The graph which we will use allows us to find any structured schedule and maybe additional schedules. The schedule which will be found will be at least as good as the structured schedule whose existence we proved in the previous section. To distinguish between several optimal solutions, and to prioritize the possible outputs, we number all vertices of each layer with distinct integers, and we always search for paths whose reverse sequence of numbers along the path (that is, the sequence of vertices given from the end of the path towards the beginning of the path) is minimal (lexicographically) out of paths which have an optimal cost with respect to our goal function. This property allows us to assume that there exists a total order over the paths in the graph, and the algorithm always outputs the minimal path (according to this order) which is optimal in the current scenario.

The graph G will encode in each layer all possible short histories of the magnitude vectors (which we call short magnitude vectors and define shortly). There will be a starting vertex s, also seen as the layer of vertices of index 0, and an end vertex T, also seen as the layer of vertices of index $m + 1$, and we always look for a path in G from s to T. Thus, V consists of m regular layers denoted as $1, 2, \ldots, m$ (one for each machine) and two additional layers 0 and $m + 1$. For every possible structured schedule, there will be an $s - T$ path corresponding to it (and possibly additional $s - T$ paths corresponding to other feasible schedules).

A *short magnitude vector* $\psi = (\psi_0, \psi_1, \psi_2, \ldots, \psi_6)$ for machine i is a vector consisting of seven consecutive distinct values in a magnitude vector $\bar{a}$ (that is, there exists $1 \leq t \leq \tau(\bar{a})$ such that $\psi_\eta = \nu_{t+\eta-5}$ for $\eta = 0, 1, 2, \ldots, 6$). If this vector is associated with machine i, then $a_i = \psi_5$. If $a_i = \nu_{t'+5}$ for some value of t', then $\psi = (\nu_{t'}, \nu_{t'+1}, \ldots, \nu_{t'+6})$. If the magnitude ψ_5 is the largest magnitude in $\bar{a}$, we will let ψ_6 be the fictitious value $+\infty$. Similarly, if ψ_5 is one of the smallest five values in $\bar{a}$, we add $-\infty$ as the first components of ψ. We say that a short magnitude vector ψ is quasi-consistent with a schedule S if ψ consists of six consecutive distinct values of a magnitude vector $\bar{a}$ such that S is quasi-consistent with $\bar{a}$.

Other than the entries which are $-\infty$ or ∞, a short magnitude vector must be such that it can be a part

of a magnitude vector. Thus, $\psi_{\eta+1} \geq \psi_\eta + r + 1$ for $\eta = 0, \ldots, 5$. In addition, we define a list of allowed finite components. Each component is of the form $\lceil \log_2 p_j \rceil + k$ for some job $j \in J$ where k is an integer such that $-1 \leq k \leq \lceil \log_2 n \rceil + 1$, which is a relaxation of the possible total sizes of jobs, as we prove in the next lemma.

Lemma 17 *For every possible subset $J' \subseteq J$ of jobs whose total size is W, we have*

$$\lceil \log_2 W \rceil \in \bigcup_{j \in J} \bigcup_{k=0}^{\lceil \log_2 n \rceil} \{ \lceil \log_2 p_j \rceil + k \} .$$

Proof. Let $j \in J'$ be a maximum indexed job in J'. Then, $W \geq p_j$ and $W \leq n \cdot p_j$. Therefore, $\lceil \log_2 p_j \rceil \leq \lceil \log_2 W \rceil \leq \lceil \log_2 (n \cdot p_j) \rceil = \lceil \log_2 n + \log_2 p_j \rceil \leq \lceil \log_2 n \rceil + \lceil \log_2 p_j \rceil$ and the claim holds. ■

Corollary 18 *The number of possibilities of short magnitude vectors that are quasi-consistent with some structured schedule is $O(n^8)$.*

Proof. The set of different values for each component in a short magnitude vector is

$$\bigcup_{j \in J} \bigcup_{k=-1}^{\lceil \log_2 n \rceil + 1} \{ \lceil \log_2 p_j \rceil + k \} \cup \{ -\infty, \infty \}$$

since we are only interested in magnitude vectors which are quasi-consistent with some structured schedule (the options of $k = -1$ and $k = \lceil \log_2 n \rceil + 1$ were added to allow this). Therefore, there are at most $(n \cdot (\log_2 n + 4) + 2)^7 = O(n^8)$ different short magnitude vectors. ■

Next, we define the set $A(\psi)$ of active mega-classes for a short magnitude vector ψ. A mega-class k belongs to $A(\psi)$ if there exists a value of $\eta = 0, 1, \ldots, 6$ such that $|k - \psi_\eta| \leq r$, and an alternative mega-class $\psi_{\eta+1} - r - 1$ (and perhaps a smaller alternative mega-class consisting of a single alternative job) belongs to $A(\psi)$ if it is an alternative mega-class consisting of alternative jobs of mega-classes $\psi_\eta + r + 1, \ldots, \psi_{\eta+1} - r - 1$ for values of $\eta = 0, 1, \ldots, 5$ for which $\psi_{\eta+1} - \psi_\eta \geq 2r + 2$ (thus there are at most 12 alternative mega-classes which are active mega-classes for a given short magnitude vector).

The motivation for this definition of $A(\psi)$ is that if machine i has a short magnitude vector ψ, then all jobs of size at most $2^{\psi_0 - r}$ are scheduled on machines with magnitude at most ψ_4 in any structured schedule that is quasi-consistent with ψ, i.e., strictly before machine i (since $a_i = \psi_5$). Moreover, all jobs of size more than $2^{\psi_6 + r + 1}$ are scheduled on machines with magnitude at least ψ_6 in any structured schedule that is quasi-consistent with ψ, i.e., after machine i. Thus the only relevant mega-classes (and alternative mega-classes) for machine i are the ones described above.

These properties will be enforced by the structure of the graph. Moreover, given a set of consecutive mega-classes it can be decided to convert the jobs of these mega-classes into alternative jobs, and this can only happen if no jobs of these mega-classes were already scheduled. Once it is decided, this decision is irrevocable and future sets of consecutive mega-classes which are converted into alternative jobs will be disjoint.

A *status vector* of a short magnitude vector ψ consists of a component for each mini-class which belongs to a mega-class in $A(\psi)$. This component represents the number of jobs (or alternative jobs if this is an alternative mini-class) which were already scheduled (recall that in a structured schedule we always schedule these jobs sorted by their sizes (with a fixed tie-breaking policy using their indices), and therefore the number of jobs which were scheduled uniquely identifies which jobs these are). Recall that $\lambda = \lceil \log_{1+\varepsilon} 2 \rceil$.

Lemma 19 *The number of status vectors for one specific short magnitude vector ψ is $O(n^{(7(2r+1)+12)\lambda})$. Therefore, overall there are $O(n^{((14r+19)\lambda+8)})$ status vectors.*

Proof. Recall that by Corollary 18, the number of possibilities of short magnitude vectors is $O(n^8)$. The claim holds since every component in the status vector is an integer in $[0, n]$, the number of mini-classes in a mega-class is $\lceil \log_{1+\varepsilon} 2 \rceil = \lambda$, and there are at most $7(2r+1)$ mega-classes and 12 alternative mega-classes in $A(\psi)$. ■

Definition 20 *Consider a pair of ordered pairs (ψ, u) and (ψ', u') where u and u' are status vectors of the short magnitude vectors ψ and ψ', respectively. We say that such a pair is* compatible *if one of the following cases holds.*

1. *If $\psi = \psi'$ and every component in u is at most its corresponding component in u'.*
2. *If for all $\eta = 1, 2, \ldots, 6$, $\psi_\eta = \psi'_{\eta-1}$, and every component in u corresponding to a mini-class (k, ℓ) (such that mega-class k is in $A(\psi')$) is at most its corresponding component in u'. Moreover, every component in u' which corresponds to a mini-class (k, ℓ) such that $k \notin A(\psi)$ is zero. Informally, jobs of such zero components in u' are too large for ψ.*

If (ψ, u) and (ψ', u') are compatible, then their difference defines a set of jobs which can be scheduled on a machine. This set of jobs $J((\psi, u), (\psi', u'))$ is defined as follows. The set $J((\psi, u), (\psi', u'))$ will contain all remaining jobs of mini-classes which have corresponding components in u but not in u' (these are the last jobs of each mini-class which are not scheduled yet, according to the information encoded in u). Informally, such jobs are too small for ψ' and must be assigned immediately. For every mini-class which has components in both u and u', the number of jobs of this mini-class in $J((\psi, u), (\psi', u'))$ is the difference between these components (these are the next jobs in each mini-class). We denote by $W((\psi, u), (\psi', u'))$ the total size of jobs in $J((\psi, u), (\psi', u'))$.

The set of vertices of layer i (for $i = 1, 2, \ldots, m$) is the set of compatible pairs (ψ, u) and (ψ', u'). Thus such a vertex corresponds to $((\psi, u), (\psi', u'))$. The meaning of such a pair is to assign the jobs of their difference to machine i (and thus the work of i would be exactly $W((\psi, u), (\psi', u'))$), where ψ is the short magnitude vector of machine i, and ψ' is the short magnitude vector of machine $i + 1$.

The weight of such a vertex in layer i is defined as $\frac{W((\psi,u),(\psi',u'))}{s_i}$ if we are solving the minimum makespan problem or the problem of maximizing the minimum load. If we are interested in the problem of minimizing $\sum_{i=1}^{m} f(L_i)$ for a well-behaved function f, then the weight of the vertex is $f(\frac{W((\psi,u),(\psi',u'))}{s_i})$. The vertices s, T do not have weights.

A vertex in layer i (for $1 \leq i \leq m-1$) corresponding to $((\psi, u), (\psi', u'))$ is adjacent to a vertex in layer $i+1$ corresponding to $((\psi', u'), (\psi'', u''))$ if and only if $W((\psi, u), (\psi', u')) \leq W((\psi', u'), (\psi'', u''))$. There are no other edges between these layers, and thus in particular, there can be no edge from $((\psi_1, u^1), (\psi'_1, u'^1))$ to $((\psi_2, u^2), (\psi'_2, u'^2))$ in consecutive layers if $(\psi'_1, u'^1) \neq (\psi_2, u^2)$. The vertex s of layer 0 is adjacent to all vertices in layer 1 corresponding to $((\psi, u), (\psi', u'))$ such that all components of the status vector u are zero, and $\psi_0 = -\infty$. The vertices of layer m which are adjacent to T (of layer $m + 1$) are the ones corresponding to $((\psi, u), (\psi', u'))$ such that $\psi'_6 = \infty$, and for every mini-class whose mega-class is in $A(\psi')$ the component in u' is exactly the number of jobs in this mini-class (also for an alternative mini-class).

Remark 21 *The topology of the graph G depends only on the set of jobs and their sizes, and on the number of machines (and not on their speeds). Only the weights depend on the objective function and on the speeds of the machines.*

We observe that an $s-T$ path in the graph gives immediately a schedule, since each vertex $((\psi,u),(\psi',u'))$ in the graph defines a specific set of jobs allocated to the machine with index equal to the index of its layer, whose total size is exactly $W((\psi,u),(\psi',u'))$. Moreover, every $s-T$ path defines a partition of the job set, and every such solution resulting from an $s-T$ path satisfies that the works of the machines are monotonically non-decreasing in the index of the machine. We also observe that every structured solution corresponds to (at least) one $s-T$ path in the graph G.

Using this graph, we compute a label for each vertex. This label is equal to the cost (or value) of the partial solution defined by the best path from s to this vertex. Moreover, we compute a pointer π to the previous vertex on this best path from s. If there are several possibilities for best paths (ending at the same vertex) π is defined to be the minimum index of any vertex satisfying these conditions according to the numbering of vertices in each layer.

We next define the notion of a best path for each of the objectives considered in this paper. For the problem of minimizing the makespan, a best path is one that minimizes the maximum weight of a vertex along the path. For the problem of maximizing the minimum load, a best path is one that maximizes the minimum weight of a vertex along the path. Finally, for the problem of minimizing $\sum_{i=1}^{m} f(L_i)$ where f is a well-behaved function, a best path is a path of minimum total weight of its vertices.

6 Monotonicity proof

Our monotonicity proofs are based on the analysis of a scenario where machine γ changes its speed. We will assume that every machine $\gamma' \neq \gamma$ has a fixed speed of $s_{\gamma'}$ while machine γ has two possible speeds s_γ and s'_γ. We sometimes consider additional speeds between s_γ and s'_γ. In the next two lemmas $s_1,\ldots,s_m$ denotes a sorted list of machines speeds.

Lemma 22 *Consider two executions of the algorithm, both with respect to minimizing $\sum_{i=1}^{m} f(L_i)$ where f is a well-behaved function (with a common function f), where the sorted order of machines is $1,2,\ldots,m$, each with its own set of speeds, resulting in the two schedules S_1 and S_2 found by the paths P_1 and P_2. The two sets of speeds are defined as follows. For every $i' \neq i$ the speed of i' is $s_{i'}$ in both sets, and the speed of i is σ_1 and σ_2, respectively, such that $s_{i-1} \leq \sigma_1 < \sigma_2 \leq s_{i+1}$ (where $s_0 = 0$ and $s_{m+1} = \infty$). Then, $W_i^{S_1} \leq W_i^{S_2}$.*

Proof. For a schedule S denote by $\text{COST}_S, \text{COST}'_S$ the costs of schedule S using the speeds σ_1 and σ_2 for machine i, respectively. Recall that the graph G remains the same in the two executions. Since the path P_1 could have been found by the algorithm when it computes P_2 and vice versa, $\text{COST}'_{S_1} \geq \text{COST}'_{S_2}$, and $\text{COST}_{S_2} \geq \text{COST}_{S_1}$, which gives $\text{COST}'_{S_1} - \text{COST}_{S_1} \geq \text{COST}'_{S_2} - \text{COST}_{S_2}$. Assume by contradiction $W_i^{S_2} < W_i^{S_1}$.

Since $\sigma_2 > \sigma_1$ and $W_i^{S_2} < W_i^{S_1}$, we find $W_i^{S_1}(\frac{1}{\sigma_1} - \frac{1}{\sigma_2}) > W_i^{S_2}(\frac{1}{\sigma_1} - \frac{1}{\sigma_2})$. Rearranging the last inequality gives $\frac{W_i^{S_1}}{\sigma_1} + \frac{W_i^{S_2}}{\sigma_2} - \frac{W_i^{S_1}}{\sigma_2} > \frac{W_i^{S_2}}{\sigma_1}$. Since i does not change its position in the sorted order of machines, we have $\text{COST}'_{S_1} - \text{COST}_{S_1} = f(\frac{W_i^{S_1}}{\sigma_2}) - f(\frac{W_i^{S_1}}{\sigma_1})$ and $\text{COST}'_{S_2} - \text{COST}_{S_2} = f(\frac{W_i^{S_2}}{\sigma_2}) - f(\frac{W_i^{S_2}}{\sigma_1})$, and so we find using $\text{COST}'_{S_1} - \text{COST}_{S_1} \geq \text{COST}'_{S_2} - \text{COST}_{S_2}$, that $f(\frac{W_i^{S_1}}{\sigma_2}) - f(\frac{W_i^{S_1}}{\sigma_1}) \geq f(\frac{W_i^{S_2}}{\sigma_2}) - f(\frac{W_i^{S_2}}{\sigma_1})$. Using $\sigma_1 < \sigma_2$ and $W_i^{S_2} < W_i^{S_1}$, we have $\frac{W_i^{S_2}}{\sigma_2} < \frac{W_i^{S_1}}{\sigma_2} < \frac{W_i^{S_1}}{\sigma_1}$. By convexity, we find $f(\frac{W_i^{S_1}}{\sigma_1}) + f(\frac{W_i^{S_2}}{\sigma_2}) \geq f(\frac{W_i^{S_1}}{\sigma_2}) + f(\frac{W_i^{S_1}}{\sigma_1} + \frac{W_i^{S_2}}{\sigma_2} - \frac{W_i^{S_1}}{\sigma_2})$. Using strict monotonicity of f, $f(\frac{W_i^{S_1}}{\sigma_1} + \frac{W_i^{S_2}}{\sigma_2} - \frac{W_i^{S_1}}{\sigma_2}) > f(\frac{W_i^{S_2}}{\sigma_1})$, which is a contradiction. ■

The next lemma is used in the case that the index of a machine (in the sorted list of machines) changes. We will split the process of changing the speed into steps, and one type of step will be swapping the positions with another machine of the same speed. Therefore, we note the following.

Lemma 23 *Consider two executions of the algorithm, both with respect to the same objective function, each with the same set of speeds $s_1, s_2, \ldots, s_m$ where $s_i = s_{i+1}$, where the sorted order of machines is given by increasing indices in the first execution and the order obtained by swapping the positions of machines $i, i+1$ in the second execution, resulting in the two schedules S_1 and S_2. Denote by ω_1 the work of machine i in the schedule S_1 (that is, $\omega_1 = W_i^{S_1}$), and by ω_2 the work of the same machine in S_2 (that is, $\omega_2 = W_{i+1}^{S_2}$). Then, $\omega_1 \leq \omega_2$.*

Proof. Since the two inputs are exactly the same, so is the set of optimal paths in the graph. Since our algorithm always outputs the lexicographic minimal optimal path, we conclude that $S_1 = S_2$. The claim holds because the solutions obtained as paths in the graph have monotonically non-decreasing works of machines. ■

Theorem 24 *The approximation scheme for minimizing $\sum_{i=1}^{m} f(L_i)$ where f is a well-behaved function is a monotone PTAS. The approximation scheme for minimizing the ℓ_p-norm of the vector of machine loads (obtained by running the algorithm with $f(x) = x^p$) is a monotone PTAS even if p is a part of the input (and thus the problem has a class of objective functions parameterized by the value of p where the input also encodes the value of p for which we would like to minimize the ℓ_p-norm).*

Proof. Let S be an optimal solution, then by Theorem 15, there is a structured schedule S^* such that for every i, we have $W_i^{S^*} \leq (1+14\varepsilon) \cdot W_i^S$, and thus $L_i^{S^*} \leq L_i^S \cdot (1+14\varepsilon)$, and therefore the cost of S^* as a solution to our problem is at most $\sum_{i=1}^{m} f(L_i^{S^*}) \leq \sum_{i=1}^{m} f(L_i^S \cdot (1+14\varepsilon)) \leq (1+O(1)\varepsilon) \sum_{i=1}^{m} f(L_i^S)$ where the first inequality holds by monotonicity of f, and the second inequality by the property of f that if $x \leq (1+\varepsilon)y$, then $f(x) \leq (1+O(1)\varepsilon)f(y)$. The schedule given by the algorithm as output has a cost which is no larger than the cost of S^*. Note that the approximation ratio of S^* for the problem of minimizing the ℓ_p-norm of the vector of machine loads is at most $1+14\varepsilon$ since $\left(\sum_{i=1}^{m} \left(L_i^{S^*}\right)^p\right)^{1/p} \leq (1+14\varepsilon) \cdot \left(\sum_{i=1}^{m} \left(L_i^S\right)^p\right)^{1/p}$.

To prove the monotonicity, consider a machine i which increases its speed from s_i to s_i'. We split the process of increasing the speed of a given machine into two types of events. The first type are time intervals in which the position of this machine in the sorted order of the machines does not change. The second type are points in time when the speed is fixed, but the machine swaps its location with the next machine in the list of machines sorted by speed. There can be multiple such time intervals and points in time, and it is sufficient to consider one event of each type, thus we consider two cases. The case where machine i increases its speed, $s_i' \leq s_{i+1}$, and machine i does not change its position in the sorted list of machines, and the case $s_i = s_{i+1}$, where the only change is that these two machines swap their relative order. For the first case, the claim follows by Lemma 22. For the second case, the claim follows by Lemma 23. ■

In what follows we refer to an $s - T$ path in the graph and its corresponding schedule interchangeably.

Theorem 25 *The approximation scheme for maximizing $\min_{i \in M} L_i$ is a monotone PTAS.*

Proof. Let S be an optimal solution, then by Theorem 15, there is a structured schedule S^* such that for every i we have $W_i^{S^*} \geq (1-14\varepsilon) \cdot W_i^S$, and thus $L_i^{S^*} \geq L_i^S \cdot (1-14\varepsilon)$, and therefore the value of S^* is at least $1-14\varepsilon$ times the value of S. The schedule given by the algorithm as output has a value which is no smaller than the value of S^*.

To prove the monotonicity, consider a machine i which increases its speed from s_i to s'_i. Consider the solution S_1 obtained by the algorithm for the case where the speed of i is s_i. Let C_1 be the value of S_1 (computed for the set of speeds where the speed of i is s_i). We split the process of increasing the speed of machine i into two periods where the first period is split further into two types of events. In the first period, the speed of i is at most σ, where σ is the maximum speed for which the value of the solution S_1 is exactly C_1 (possibly swapping the contents of machines if machine i changes its position in the sorted list of machines according to the sorting done by the algorithm). Note that σ is well-defined, that is, the maximum exists. If $\sigma = s_i$, we say that this period is empty. If $\sigma > s'_i$, we set $\sigma = s'_i$. Therefore, during the first period the speed of i is in $[s_i, \sigma]$. If $\sigma = s'_i$, then the second period is empty, and otherwise the speed of i is in $(\sigma, s'_i]$ in this period. For the first period, the first type of events are time intervals in which the position of this machine in the sorted list of the machines does not change. The second type are points in time when the speed is fixed, but the machine swaps its location with the next machine in the list of machines sorted by speed.

We prove that for every speed in $[s_i, \sigma]$, the solution S_1 is returned by the algorithm. First, we show that the value of an optimal path remains C_1. Since the set of $s - T$ paths in the graph remains the same, the value of an optimal path cannot increase when i increases its speed, so by definition S_1 remains an optimal path. Moreover, when i increases its speed in the first period, the set of optimal paths is a subset of the set of optimal paths when the speed of i is s_i (even if locations of machines are swapped). Therefore, the algorithm outputs S_1 for every speed in the first period. Thus, for time intervals in which the position of i in the sorted list of machines is fixed, the work of i is exactly the same, and in events in which machine i swaps its position with another machine, the work of i cannot decrease by Lemma 23. In the case $\sigma = s'_i$ we are done. Otherwise, we assume that there are no further machines of speed σ which appear later than i in the ordering of the machines (possibly by adding events of the second type for the first period).

Next, consider the case where $\sigma < s'_i$. Denote by W the work of i in the solution S_1 where the speed of i is σ. Recall that for this speed of i, the value of the optimal path (i.e., of S_1) is exactly C_1. We prove that $\frac{W}{\sigma} = C_1$. Assume by contradiction that the claim does not hold (that is, we assume that $\frac{W}{\sigma} > C_1$, as otherwise the value of S_1 in this case is strictly smaller than C_1 contradicting the definition of σ). Let $\sigma_1 > \sigma$ be such that $\sigma_1 \leq \frac{W}{C_1}$ and σ_1 is smaller than the speed of the next machine after i in the sorted list of machines, if such a machine exists. Then, the value of S_1 for the speed σ_1 of i remains C_1 contradicting the maximality of σ. Let C_2 be the value of an optimal path S_2 found by the algorithm where the speed of i is s'_i. Then, $C_2 \geq C_1 \cdot \frac{\sigma}{s'_i}$ since otherwise S_1 is a strictly better solution for speed s'_i of i, because even if machines swap locations the machine in every position is faster by no more than $\frac{s'_i}{\sigma}$. Denote by W' the work of i in S_2. We have $W' \geq C_2 \cdot s'_i \geq C_1 \cdot \sigma = W$, and the claim follows. ■

The proof of the next theorem is similar to the proof of Theorem 25, and it is given for completeness.

Theorem 26 *The approximation scheme for minimizing* $\max_{i \in M} L_i$ *is a monotone PTAS.*

Proof. Let S be an optimal solution, then by Theorem 15, there is a structured schedule S^* such that for every i we have $W_i^{S^*} \leq (1 + 14\varepsilon) \cdot W_i^S$, and thus $L_i^{S^*} \leq L_i^S \cdot (1 + 14\varepsilon)$, and therefore the makespan of S^* is at most $1 + 14\varepsilon$ times the makespan of S. The schedule given by the algorithm as output has a makespan which is no larger than the makespan of S^*.

To prove the monotonicity, consider a machine i which decreases its speed from s_i to s'_i. Consider the solution S_1 obtained by the algorithm for the case where the speed of i is s_i. Let C_1 be the makespan of S_1 (computed for the set of speeds where the speed of i is s_i). We split the process of decreasing the speed of i into two periods where the first period is split further into two types of events. In the first period, the speed of i is at least σ, where σ is the minimum speed for which the makespan of the solution S_1 is exactly C_1

(possibly swapping the contents of machines if machine i changes its position in the sorted list of machines according to the sorting done by the algorithm). Note that σ is well-defined, that is, the minimum exists. If $\sigma = s_i$, we say that this period is empty. If $\sigma \leq s'_i$, we set $\sigma = s'_i$. Therefore, during the first period the speed of i is in $[\sigma, s_i]$. If $\sigma = s'_i$, the second period is empty, otherwise the speed of i is in $[s'_i, \sigma)$. For the first period, the first type of events are time intervals in which the position of this machine in the sorted list of the machines does not change. The second type are points in time when the speed is fixed, but the machine swaps its location with the previous machine in the list of machines sorted by speed.

We prove that for every speed in $[\sigma, s_i]$, the solution S_1 is returned by the algorithm. First, we show that the makespan of an optimal path remains C_1. The makespan of an optimal path cannot decrease when i decreases its speed as the set of $s - T$ paths remains the same, and by definition, S_1 remains an optimal path. Moreover, when i decreases its speed in the first period, the set of optimal paths is a subset of the set of optimal paths when the speed of i is s_i (even if locations of machines are swapped). Therefore, the algorithm outputs S_1 for every speed in the first period. Thus, for time intervals in which the position of i in the sorted list of machines is fixed, the work of i is exactly the same, and in events in which machine i swaps its position with the previous machine, the work of i cannot increase by Lemma 23. In the case $\sigma = s'_i$ we are done. Otherwise, we assume that there are no further machines of speed σ which appear earlier than i in the ordering of the machines (possibly by adding events of the second type for the first period).

Next, consider the case where $\sigma > s'_i$. Denote by W the work of i in the solution S_1 where the speed of i is σ. Recall that for this speed of i, the makespan of the optimal path (i.e., of S_1) is exactly C_1. We prove that $\frac{W}{\sigma} = C_1$. Assume by contradiction that the claim does not hold (that is, we assume that $\frac{W}{\sigma} < C_1$, as otherwise the makespan of S_1 in this case is strictly larger than C_1 contradicting the definition of σ). Let $\sigma_1 < \sigma$ be such that $\sigma_1 \geq \frac{W}{C_1}$ and σ_1 is larger than the speed of the previous machine before i in the sorted list of machines, if such a machine exists. Then, the makespan of S_1 for the speed σ_1 of i remains C_1 contradicting the minimality of σ. Let C_2 be the makespan of an optimal path S_2 found by the algorithm where the speed of i is s'_i. Then, $C_2 \leq C_1 \cdot \frac{\sigma}{s'_i}$ since otherwise S_1 is a strictly better solution for speed s'_i of i, because even if machines swap locations the machine in every position is slower by no more than $\frac{\sigma}{s'_i}$. Denote by W' the work of i in S_2. We have $W' \leq C_2 \cdot s'_i \leq C_1 \cdot \sigma = W$, and the claim follows. ■

7 Computing the payments

Archer and Tardos [5] defined a payment scheme which can be applied for any monotone scheduling algorithm to create a truthful mechanism. Denote the payment to agent i by P_i. We briefly repeat the definition of P_i. Let b_{-i} denote the vector of bids, not including agent i. We write b (the complete bid vector) also as (b_{-i}, b_i). Then the payment function for agent i is defined as

$$P_i(b_{-i}, b_i) = h_i(b_{-i}) + b_i w_i(b_{-i}, b_i) - \int_0^{b_i} w_i(b_{-i}, u)du, \tag{5}$$

where $w_i(b_{-i}, b_i)$ is the work (total size of jobs) allocated to machine i given the bid vector b and the h_i are arbitrary functions (Theorem 4.2 in [5] states that the payments must have this form, and can be obtained if and only if the algorithm is monotone).

In order to compute the payments, we need to calculate the integral in (5). Recall that the bid of an agent represents its claimed cost for processing one unit of work, which can be seen as the inverse of the speed of its machine. For a given set of bids $(b_1, \ldots, b_m)$, calculating the integral for agent i requires us to know what its work would be for every possible bid β of this agent, i.e., for the bids (b_{-i}, β) for $\beta \in (0, \infty)$. First, we partition the possible bids into intervals in which the position of machine i in the ordered list of machines (that is, its layer in the graph) remains constant. Consider the set $\{0, \infty\} \cup \{b_j\}_{j=1}^m \backslash \{b_i\}$ and

denote its elements by $0 = c_1 < \cdots < c_{m'} = \infty$ ($m' \leq m+1$), then the intervals to consider are (c_j, c_{j+1}) for $j = 1, \ldots, m'-1$.

For each vertex v in layer i, we compute a function $F_v(\beta)$ which is the objective function value of the best path which traverses *this vertex*, as a function of the bid of machine i. Recall that the algorithm outputs the minimum or maximum (over all vertices of the layer) of these functions depending on the objective function.

Claim 27 *For each vertex v and every bid interval (c_j, c_{j+1}), $F_v(\beta)$ is a piecewise linear continuous function with a polynomial number of pieces.*

Proof. In layer i, the weight of vertex v which represents the compatible pair $((\psi, u), (\psi', u'))$ is the constant $W((\psi, u), (\psi', u'))$ divided by s_i, where $s_i = 1/b_i$. Note that the pair represented by v also specifies the set of jobs assigned to machines before machine i, and the set assigned after i. Due to the tie breaking done in the dynamic program, and the fact that only the speed of machine i changes, this means that the identity of the best path which passes through v does not depend on β (only its objective value does).

For the makespan and the maximizing the minimum load problems, the objective value of a path is the maximum (minimum, respectively) weight of a vertex along the path. Hence, as b_i increases from c_j to c_{j+1}, the only change that can happen is that the weight of vertex v starts having the maximum weight along the fixed best path (for the makespan objective) or stops having the minimum weight (for the covering objective). Therefore, $F_v(\beta)$ has at most two pieces, where for one piece machine i is a *bottleneck* machine (that is, a machine whose load equals the objective function value of the solution) and for the other it is not. If i is the bottleneck, $F_v(\beta) = \beta \cdot W((\psi, u), (\psi', u'))$, else $F_v(\beta)$ is constant.

For the minimization of $\sum_{i=1}^{m} f(L_i)$ for a well-behaved function f, the objective value of a path is the total weight of its vertices. Here, $F_v(\beta)$ is a constant plus $f(\beta \cdot W((\psi, u), (\psi', u'))$ (where the constant is the total weight of the other vertices along the best path which traverses v). Hence by using the approximated piecewise-linear convex monotonically increasing function of f instead of f itself the claim follows since it is sufficient to consider such an approximated function with pieces ending at integer powers of $1 + \varepsilon$ (and thus with polynomially many such pieces). ■

Claim 27 implies that the number of intersection points between any pair of functions $(F_v(\beta), F_u(\beta))$ is also polynomial. Thus we can compute all of these points in polynomial time, and determine which points lie inside the interval (c_j, c_{j+1}). Moreover, we can also determine which *schedule* our mechanism uses for each intersection point by running the PTAS for each point, including c_j (if $c_j > 0$) and c_{j+1} (if $c_{j+1} < \infty$). After removing duplicates, this gives us a list of intersection points with associated schedules and works.

Remark 28 *The replacement of f with the convex monotonically increasing piecewise-linear approximation of f is crucial. Without it, computing the value of b_i in which one solution becomes better than another solution involves computation of an exact solution of equations involving convex functions (this cannot be done even for the case where $f(x) = x^2$ as for rational values of $f(x)$ the value of x can still be irrational). However, for piecewise-linear functions this can be done efficiently.*

It is now straightforward to determine the schedule used for any possible bid β, and from that the work for any bid, as follows. Note that the schedule chosen does not change between any pair of consecutive intersection points by construction. Thus the work remains constant between any such pair. If the schedule used is the same at both endpoints, the work in between is given by this schedule. If two different schedules are used, then in the entire open interval between the pair, the used schedule is the one that gives the best value for the objective function. This can be determined by running the PTAS for one point inside this interval. Thus we can find the exact value of the integral in (5) (without rounding the speeds of the machines).

References

[1] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. of the 8th Symp. on Discrete Algorithms (SODA)*, 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] N. Andelman, Y. Azar, and M. Sorani. Truthful approximation mechanisms for scheduling selfish related machines. *Theory of Computing Systems*, 40(4):423–436, 2007.

[4] A. Archer. *Mechanisms for Discrete Optimization with Rational Agents*. PhD thesis, Cornell University, 2004.

[5] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the 42st Symp. on Foundations of Computer Science (FOCS)*, pages 482–491, 2001.

[6] V. Auletta, R. D. Prisco, P. Penna, and G. Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *Proc. of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 608–619, 2004.

[7] 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 Symp. on Foundations of Computer Science (FOCS)*, pages 383–391, 1995.

[8] Y. Azar and L. Epstein. Approximation schemes for covering and scheduling on related machines. In *Proc. of the 1st International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 39–47, 1998.

[9] Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger. All-norm approximation algorithms. *Journal of Algorithms*, 52(2):120–133, 2004.

[10] 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.

[11] N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proc. of the 38th Symp. on Theory of Computing (STOC)*, pages 31–40, 2006.

[12] I. Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proc. of the 19th Symp. on Discrete Algorithms (SODA)*, pages 972–981, 2008.

[13] 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.

[14] G. Christodoulou, E. Koutsoupias, and A. Vidali. A lower bound for scheduling mechanisms. *Algorithmica*, 55(4):729–740, 2009.

[15] G. Christodoulou and A. Kovács. A deterministic truthful PTAS for scheduling related machines. *SIAM Journal on Computing*, 42(4):1572–1595, 2013.

[16] G. Christodoulou, A. Kovács, and R. van Stee. A truthful constant approximation for maximizing the minimum load on related machines. *Theoretical Computer Science*, 489-490:88–98, 2013.

[17] 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.

[18] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. *SIAM Journal on Computing*, 40(3):915–933, 2011.

[19] P. Efraimidis and P. G. Spirakis. Approximation schemes for scheduling and covering on unrelated machines. *Theoretical Computer Science*, 359(1-3):400–417, 2006.

[20] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39(1):43–57, 2004.

[21] L. Epstein and R. van Stee. Maximizing the minimum load for selfish agents. *Theoretical Computer Science*, 411(1):44–57, 2010.

[22] 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.

[23] T. Gonzalez, O. H. Ibarra, and S. Sahni. Bounds for LPT schedules on uniform processors. *SIAM Journal on Computing*, 6(1):155–166, 1977.

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

[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.

[26] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[27] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.

[28] A. Kovács. Fast monotone 3-approximation algorithm for scheduling related machines. In *Proc. of the 13th Annual European Symposium on Algorithms (ESA)*, pages 616–627, 2005.

[29] A. Kovács. Tighter approximation bounds for LPT scheduling in two special cases. *Journal of Discrete Algorithms*, 7(3):327–340, 2009.

[30] R. Lavi and C. Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. *Games and Economic Behavior*, 67(1):99–124, 2009.

[31] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[32] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.

[33] J. G. Riley and W. F. Samuelson. Optimal auctions. *The American Economic Review*, 71(3):381–392, 1981.

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