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 key 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. Monotone approximation schemes have an important role in the emerging area of algorithmic mechanism design. 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, where its private information 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.

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 [33]. Specifically, the following research agenda was suggested: *"to what extent is incentive compatible efficient computation fundamentally less powerful than "classic" efficient computation?"* (as formulated in [18]). 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 [33, 5, 31, 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 a negative answer to the question raised in [33] for scheduling problems on uniformly related machines, by designing $(1+\varepsilon)$-approximation mechanisms for 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. 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. 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. 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, $L_1, L_2, \ldots, L_m$.

We consider a variety of objective functions (social goals). A well-known objective function is the *makespan*, which is the maximum load. The optimiza-

[*]Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

[†]Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel. levinas@ie.technion.ac.il.

[‡]Max Planck Institute for Informatics, Saarbrücken, Germany. vanstee@mpi-inf.mpg.de

tion problem of finding a schedule which minimizes the makespan is a basic one [25, 24, 26, 27, 15]. The problem of finding a schedule which maximizes the minimum load, also known as the *cover*, is the famous *Santa Claus* problem on uniformly related machines (see e.g. [23, 35, 2, 8, 21, 11, 22]). 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 $f(x) \leq (1+O(1)\varepsilon)f(y)$. 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 $p = 1$ an optimal solution which will satisfy our requirements is 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), 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 for truthfulness is that the allocation algorithm is *monotone* [32, 34, 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.

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, 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 approximating over this set, 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, where given subsets of jobs 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). A number of difficulties arise when trying to modify such

[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.

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 which are small for the machine which receives them are much more difficult; such jobs usually do not affect the approximation ratio but which nevertheless need to 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.

Dhangwatnotai et al. [18] used randomization to construct a monotone PTAS for the three main objective functions listed above (makespan, cover, 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 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 it sees it as a profit of 1, while a human agent would very much be interested in the value of M, and if it is large, it would see it as earning nothing at all (rather than earning 1 in expectation). Their approach of dealing with 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 approach does not seem to work even for the similar problem of maximizing the cover, 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 [22]).

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, a highly structured schedule exists, where the ratio of job sizes assigned to a machine is *unbounded* but 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 overcomes the difficulty that it does not seem to be possible to actually bound the size ratio of jobs assigned to a machine, but still we would like to use dynamic programming *without* introducing a notion of small jobs or inexact calculations. 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 given schedule, which keeps the approximation ratio close to 1. This allows us to deal with *all* of the objective functions mentioned above at once using a dynamic programming formulation implemented by a layered graph, having one layer for each machine. Unlike previous approximation schemes which 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, and even simplifies the proof of the approximation ratio, and the presentation of the algorithm, compared to previous (non-monotone) PTAS's. 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 [28, 9, 19], and even a (deterministic) monotone one for makespan minimization and for maximizing the minimum load [3, 22]. 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 part of the input. It was shown by Hochbaum and Shmoys that the makespan minimization problem has a PTAS for identical (equal speed) machines [26] and for uniformly related machines [27]. 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 [35, 1, 2], and for uniformly related machines [8, 21]. 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 [29, 30].

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 which has such properties. As stated above, 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. There are several ways to define a magnitude vector for a given schedule. A possible solution to the dynamic programming can be viewed as a process where we create the magnitude vector 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 that a fractional schedule exists where the total size of very small jobs which are (partially) assigned to machines with high work is small, and the work on each machine is the same as the work in the integral schedule. We then 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 unsorted, then the difference between their works is very small). Searching for unsorted schedules causes technical difficulties for the algorithm which should find a structured schedule, while a postprocessing step of sorting may harm monotonicity. We therefore do one extra step 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 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 schedule where works are very close to the works in an optimal schedule, and the structured schedule has an objective value which 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 which 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.

The full version of this paper is available as [20].

2 Preliminaries

For our results, we let ε be a small constant such that $0 < \varepsilon \le \frac{1}{32}$ and $\frac{1}{\varepsilon}$ is an integer power of 2 denoted by $r \ge 5$ (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 $0 < p_1 \le p_2 \le \cdots \le p_n$.

An integral schedule is a function $S : J \to M$. 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). 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 $p_j \le \varepsilon \tilde{W}_i^X$.

Note that the partition in (F2) is not necessarily uniquely defined. 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 $J_{\mathbb{R}}(X) = \{j \in J : p_j \le \varepsilon \tilde{W}_{S(j)}^S\}$ and $J_{\mathbb{Z}}(X) = J \setminus J_{\mathbb{R}}(X)$. Note that $\tilde{W}_i^S = \tilde{W}_i^X$ for $i = 1, \ldots, m$. X is 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)). S is 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. The first part of the following claim follows from an observation in [21], and it is easy to show the second part. 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).

Claim. *Assume that $s_1 \le s_2 \le \cdots \le s_m$. 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 $W_1^S \le W_2^S \le \cdots \le W_m^S$. There exists an optimal schedule S_1 for the makespan minimization problem which satisfies $W_1^{S_1} \le W_2^{S_1} \le \cdots \le W_m^{S_1}$. There exists an optimal schedule S_2 for the machine covering problem which satisfies $W_1^{S_2} \le W_2^{S_2} \le \cdots \le W_m^{S_2}$.*

3 The existence of near-optimal highly structured solutions

We define a partition of J into mega-classes. For $k \in \mathbb{Z}$, let $\mathcal{I}_k = (2^k, 2^{k+1}]$, and let mega-class k be $\{j \in J : p_j \in \mathcal{I}_k\}$. 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 $p_j > 2^k \ge 2^{k'+r+1} = \frac{1}{\varepsilon} \cdot 2^{k'+1} \ge \frac{1}{\varepsilon} \cdot p_{j'}$, since $k' + 1 \le k - r$ and $\varepsilon = 2^{-r}$. We refine this partition 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 $|K| \le n$). Let $\lambda = \lceil \log_{1+\varepsilon} 2 \rceil$. For $k \in K$ and $0 \le \ell \le \lambda - 1$, 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 $(1+\varepsilon)^{\lceil \log_{1+\varepsilon} 2 \rceil} \ge (1+\varepsilon)^{\log_{1+\varepsilon} 2} = 2$ 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 $k_1, \ldots, k_2$ where $k_2 \ge k_1$, with the job set $\hat{J}$ consisting of all jobs of J with size 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 $\hat{J}$. These alternative jobs have size in the interval $(\varrho, 2\varrho]$ (except perhaps for one alternative job that may be smaller). To create these alternative jobs we 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. The 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. The alternative mini-class (k, ℓ) contains all the alternative jobs with size in $I_{k,\ell}$, resulting in at most $\lambda + 1$ alternative mini-classes. 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 which 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 possibly the algorithm replaces with alternative jobs. Thus creating all the sets of alternative jobs takes $O(n^3)$. Note that one job can be contained in multiple alternative jobs, but at most one of these alternative jobs will be used.

Definition. An integral schedule *respects* the alternative jobs of mega-classes $k_1, \dots, k_2$, where $k_2 \geq k_1$, if every pair of jobs j, j' with size in the interval $(2^{k_1}, 2^{k_2+1}]$ which are within a common subset (that is, should be combined into one alternative job with possibly other jobs), 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.

Definition. A vector $\bar{a} = (a_0, a_1, \dots, a_m)$ (of length $m+1$) whose components belong to $\mathbb{Z} \cup \{-\infty\}$ is called a *magnitude vector* if $a_0 = -\infty$, for $i = 0, 1, \dots, m-1$, $a_i \leq a_{i+1}$ 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, denoted by $\tau(\bar{a})$. Each component $t = 1, 2, \dots, \tau(\bar{a})$ 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, the value ξ_t is always the first machine which has a larger component of $\bar{a}$ than the previous machine and this component is ν_t. For every $t = 1, 2, \dots, \tau(\bar{a}) - 1$, we let $J^t(\bar{a}) = \{j \in J : 2^{\nu_t + r + 1} < p_j \leq 2^{\nu_{t+1} - r}\}$.

Observation. *For every job j and every magnitude vector $\bar{a}$ with its signature vector $\bar{b}$, there are at most two values of $t \in \{1, \dots, \tau(\bar{a})\}$ 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 $j \notin \cup_\theta J^\theta(\bar{a})$.*

Definition. 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 $a_i = \alpha_i^X (= \lceil \log_2 W_i^X \rceil)$.

Lemma 3.1. *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 $a_i \leq \alpha_i^X$.*

Definition. 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 $t = 1, 2, \dots, \tau(\bar{a}) - 3$, we have

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

This 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 machine of index at least ξ_{t+3}, is relatively small compared to the work of that machine. This holds since $2^{\nu_{t+1} + r + 1} \leq 2^{\nu_{t+2}} < 2^{\nu_{t+3} - r} = \varepsilon \cdot 2^{\nu_{t+3}} = \varepsilon \cdot \tilde{W}_{\xi_{t+3}}^X$.

We define several processes in which a valid fractional schedule is modified into a different valid fractional schedule. These processes are defined algorithmically but they are not a part of the final algorithm, but only of the proof that a highly structured integral schedule must exist.

FNFI. For a subset of jobs $J' \subseteq J$ and a set of bounds $U_1, \dots, U_m$ (for the m machines) such that $\sum_{j \in J'} p_j = \sum_{i=1}^{m} U_i$, the Fractional Next-Fit Increasing (FNFI) algorithm creates a fractional allocation of these jobs in the following way. Let $i = 1$ be the first active machine, and for every $j \in J'$ let $q_j = p_j$. In every step, FNFI picks the minimum index job $j \in J'$. It allocates $\beta = \min\{q_j, U_i\}$ processing time of this job to machine i. It decreases both U_i and q_j by β. If $U_i = 0$, then it increases i by 1, and if $q_j = 0$, then it removes j from J'. FNFI repeats this step until $i = m + 1$ (and $J' = \emptyset$ must hold, these two events happen simultaneously since $\sum_{j \in J'} p_j = \sum_{i=1}^{m} U_i$). FNFI is sometimes used to reassign a subset of jobs in a valid fractional schedule, so that the total sizes of jobs of this subset assigned to each machine is unchanged (i.e., the bounds U_i are given by 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. 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 $U_1, \dots, U_m$ 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, keeps the valid fractional schedule unchanged.

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. 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 which are not completely assigned to it, the one of smallest index and the one 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) each fractional job $j \in J_{\mathbb{R}}(X)$ on machine i (that is, every $j \in J_{\mathbb{R}}(X)$ such that $X(j,i) > 0$) has size $p_j \le \varepsilon \tilde{W}_i^X \le 2\varepsilon W_i^X$, we conclude that for every $i \in M$ we have $(1-2\varepsilon)W_i^X \le W_i^S \le (1+2\varepsilon)W_i^X$. We say that the integral schedule S is created by applying Round-FNFI on X.

Lemma 3.2. *Given a schedule $S : J \to M$ such that $W_1^S \le W_2^S \le \cdots \le W_m^S$, there exists a favorable pair $(X, \bar{a})$ where $W_i^X = W_i^S$ for $i = 1,2,\dots,m$, 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. For $i = 1,2,\dots,m$, let $Q_i = \alpha_i^S$. Since $W_1^S \le W_2^S \le \cdots \le W_m^S$, we have $Q_i \le Q_{i+1}$ for all $i = 1,2,\dots,m-1$. We define a magnitude vector $\bar{a}^S = (a_0^S, a_1^S, \dots, a_m^S)$ as follows. We let $a_0^S = -\infty$, for $i = 1,2,\dots,m$, if $Q_i \le a_{i-1}^S + r$, then $a_i^S = a_{i-1}^S$, and otherwise $a_i^S = Q_i$. The valid fractional schedule X_S is consistent with $\bar{a}^S$ since for every $i \in M$ either $a_i^S = Q_i$ or $Q_i \le a_{i-1}^S + r = a_i^S + r$. In both cases, the size of any job assigned (completely) to machine i cannot exceed $W_i^X \le \tilde{W}_i^X = 2^{Q_i} \le 2^{a_i^S + r}$.

We next consider the nonempty set of pairs $(X', \bar{a})$ such that X' is consistent with $\bar{a}$, and such that for $i = 1,2,\dots,m$, $W_i^{X'} = W_i^S$ and $a_i \le \alpha_i^{X'}$ (the set is indeed 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. Based on X' we will define 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 $U_1, \dots, U_m$ 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. We argue that X satisfies (F2). We define $J_{\mathbb{R}}(X) = J_{\mathbb{R}}(X')$ and show that if $j \in J_{\mathbb{R}}(X)$ and $i \in M$ satisfy that $X(j,i) > 0$, then $p_j \le \varepsilon \tilde{W}_i^X$. Since the works of the machines are sorted in a non-decreasing order, it suffices to show that for $j \in J_{\mathbb{R}}(X)$ and i such that $X(j,i) > 0$, there exists $j' \ge j$, $j' \in J_{\mathbb{R}}(X')$ and $i' \le i$ such that $X'(j',i') > 0$, since in such a case $p_j \le p_{j'} \le \varepsilon \tilde{W}_{i'}^{X'} \le \varepsilon \tilde{W}_i^X$. Assume by contradiction that this claim does not hold for j and i. Then, since FNFI assigns job j (possibly partially) to machine i, $\sum_{j' \in J_{\mathbb{R}}(X'): j' < j} p_{j'} < \sum_{\gamma=1}^{i} U_\gamma$, however $\sum_{j' \in J_{\mathbb{R}}(X'): j' < j} p_{j'} \ge \sum_{\gamma=1}^{i} U_\gamma$ since no other jobs of $J_{\mathbb{R}}(X')$ are assigned by X' to the first i machines. 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 $1,2,\dots,i$, the maximum size of a job j such that $X(j,\gamma) > 0$ for some $1 \le \gamma \le i$ 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 $\gamma \in \{1,2,\dots,i\}$. 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 $i' \le \gamma$ such that $X'(j',i') > 0$ and $j' \ge j$ 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 $1 \le t \le \tau(\bar{a}) - 3$. Let $j \in J$ be such that there is $i \in [\xi_{t+3}, m]$ with $X(j,i) > 0$ and $p_j \le 2^{\nu_t - r}$. If there is no such job, then we are done. We have $j \in J_{\mathbb{R}}(X) = J_{\mathbb{R}}(X')$ because $\tilde{W}_i^X \ge 2^{a_i} \ge 2^{\nu_{t+3}} > 2^{\nu_t} \cdot 2^{3r} = \frac{1}{\varepsilon^3} \cdot 2^{\nu_t} \ge \frac{1}{\varepsilon^3} p_j$ where the first inequality holds by Lemma 3.1, 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 $A_{t+1} = \{\xi_{t+1}, \dots, \xi_{t+2} - 1\}$. If all jobs assigned (possibly fractionally) by X to these machines have size of at most $2^{\nu_t + r + 1}$, then we can redefine $a_{i'}$ for $i' \in A_{t+1}$ to be ν_t, contradicting the minimality of length 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 $p_{j'} \le 2^{\nu_{t+1} + r + 1} \le 2^{\nu_{t+3} - r - 1} < \varepsilon \cdot \tilde{W}_{\xi_{t+3}}^X$. If $\sum_{\gamma=\xi_{t+3}}^{m} \sum_{j'': p_{j''} \le 2^{\nu_t - r}} p_{j''} \cdot X(j'', \gamma) > 2^{\nu_{t+1} + r + 1}$, then $\sum_{\gamma=\xi_{t+3}}^{m} \sum_{j'': p_{j''} \le 2^{\nu_t - r}} p_{j''} \cdot X(j'', \gamma) > p_{j'}$. 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 fractions of jobs (where each such job has size at most $2^{\nu_t - r} = \varepsilon 2^{\nu_t} \le \varepsilon^2 2^{\nu_{t+1} - 1} \le \varepsilon^2 \frac{\tilde{W}_\gamma^X}{2} < \varepsilon^2 W_\gamma^X$ 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). This completes the proof.

Definition. 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 $|a_i - \alpha^S_i| \le 1$.

Definition. 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 $i \ge \xi_{t+4}$ such that $p_j \le 2^{\nu_t - r}$ and $S(j) = i$.
2. For every $t = 1, 2, \ldots, \tau(\bar{a}) - 1$ if $J^t(\bar{a}) = \{j \in J : 2^{\nu_t + r + 1} < p_j \le 2^{\nu_{t+1} - r}\} \neq \emptyset$, then X respects the alternative jobs of mega-classes $\nu_t + r + 1, \ldots, \nu_{t+1} - r - 1$.

Lemma 3.3. *Given a schedule $S : J \to M$ such that $W^S_1 \le W^S_2 \le \cdots \le W^S_m$, there exists a good schedule $S' : J \to M$ such that for $i = 1, 2, \ldots, m$, we have*

$$(3.1) \quad (1 - 12\varepsilon) \cdot W^S_i \le W^{S'}_i \le (1 + 12\varepsilon) \cdot W^S_i .$$

Proof. By Lemma 3.2, 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. First, for every $t = 4, 5, \ldots, \tau(\bar{a})$, we reschedule all parts of jobs j such that $p_j \le 2^{\nu_{t-3} - r}$ and for which there exists $i > \xi_t$ such that $X(j, i) > 0$ by moving them to machine ξ_t. We denote by $\tilde{X}$ the resulting fractional schedule. 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 $2 \le i \le m$). 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 $2^{\nu_{t-2} + r + 1} \le \frac{\varepsilon}{2} \cdot 2^{\nu_t}$ from machine i (and move these parts of jobs to machine ξ_t). Thus $W^X_i - W^{\tilde{X}}_i \le \frac{\varepsilon}{2} \cdot \sum_{t=4}^{t'} 2^{\nu_t} \le \varepsilon \cdot 2^{\nu_{t'}} \le 2\varepsilon \cdot W^X_i$. We conclude that for every i, we have $(1 - 2\varepsilon) W^X_i \le W^{\tilde{X}}_i \le (1 + 2\varepsilon) W^X_i$.

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 3.2, it can be shown that if a job moved to machine i, then its size is below $\varepsilon W^{\tilde{X}}_i$, since $1 - 2\varepsilon > \varepsilon$). 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 $p_j \le \varepsilon \tilde{W}^X_i \le 2\varepsilon W^X_i$, so for every $i \in M$ we have

$$(3.2) \qquad (1 - 4\varepsilon) W^X_i \le W^{\tilde{S}}_i \le (1 + 4\varepsilon) W^X_i .$$

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 $2^{\nu_{t-3}} < \varepsilon 2^{\nu_t} = \varepsilon 2^{a_{\xi_t}}$.

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 $i \ge \xi_{t+1}$ that $p_j \le 2^{\nu_{t+1} - r} = \varepsilon \cdot \tilde{W}^X_{\xi_{t+1}} \le \varepsilon \cdot \tilde{W}^X_i$. We remove the jobs in $J^t(\bar{a})$ from their positions in $\tilde{S}$, and we will schedule the alternative jobs instead (which gives a schedule of the original jobs which respects the alternative jobs of mega-classes $\nu_t + r + 1, \ldots, \nu_{t+1} - r - 1$). 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 jobs and less parts of at most four jobs, all of which have size of at most $\varepsilon \tilde{W}^X_i \le 2\varepsilon W^X_i$. Thus, $W^{\tilde{S}}_i - 8\varepsilon W^X_i \le W^{S'}_i \le W^{\tilde{S}}_i + 8\varepsilon W^X_i$. Using (3.2), we get (3.1).

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 $p_j \le 2^{\nu_{t+1} - r} = \varepsilon 2^{\nu_{t+1}}$, and $a_i \ge a_{\xi_{t+1}} = \nu_{t+1}$. The claim holds because for every i, we have $|\alpha^X_i - \alpha^{S'}_i| \le 1$ since $1 + 12\varepsilon < 2$ and $\frac{1}{1 - 12\varepsilon} < 2$. This completes the proof.

Definition. A schedule S is *quasi-consistent* with a magnitude vector $\bar{a}$ if for every $i = 1, 2, \ldots, m$ such that $\xi_t \le i < \xi_{t+1}$, 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 $|a_i - \alpha^S_i| \le 1$.

Definition.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 size of j is smaller than the size 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 3.1. *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*

$$(1 - 14\varepsilon) \cdot W_i^S \leq W_i^{S^*} \leq (1 + 14\varepsilon) \cdot W_i^S \ . \qquad (3.3)$$

4 The scheme and its analysis

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, and a set of jobs which 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.

The claim that the resulting solution is a PTAS (for each of the problems which we consider) is a trivial consequence of Theorem 3.1. 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 speed 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 machine changes its speed. The proof of monotonicity is based on the fact that the dynamic programming finds an optimal solution subject to some constraints, however it is heavily based on the fact that the works of the machines are monotonically non-decreasing and the details of the tie-breaking rule. We summarize as follows.

Theorem 4.1. *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 weight function.

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. In *Proc. of the 21st Symp. on Discrete Algorithms (SODA)*, pages 1005–1016, 2010.

[16] G. Christodoulou, A. Kovács, and R. van Stee. A truthful constant approximation for maximizing the minimum load on related machines. In *Proc. of the 6th International Workshop on Internet and Network Economics (WINE)*, pages 182–193, 2010.

[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. In *Proc. of the 48th Symp. on Foundations of Computer Science (FOCS)*, pages 15–24, 2008.

[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, A. Levin, and R. van Stee. A unified approach to truthful scheduling on related machines. *CoRR*, abs/1207.3523, 2012.

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

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

[23] 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.

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

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

[26] 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.

[27] 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.

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

[29] 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.

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

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

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

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

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

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