Approximation Schemes for Job Shop Scheduling Problems with Controllable Processing Times*

Klaus Jansen[1]†, Monaldo Mastrolilli[2]‡, and Roberto Solis-Oba[3]§
[1]Universität zu Kiel, Germany, kj@informatik.uni-kiel.de
[2]IDSIA Lugano, Switzerland, monaldo@idsia.ch
[3]The University of Western Ontario, Canada, solis@csd.uwo.ca

Abstract

In this paper we study the job shop scheduling problem under the assumption that the jobs have controllable processing times. The fact that the jobs have controllable processing times means that it is possible to reduce the processing time of the jobs by paying a certain cost. We consider two models of controllable processing times: continuous and discrete. For both models we present polynomial time approximation schemes when the number of machines and the number of operations per job are fixed.

Keywords: scheduling, job shop, controllable processing times, approximation schemes.

1 Introduction

Most scheduling models assume that jobs have *fixed* processing times. However, in real-life applications the processing time of a job often depends on the amount of resources such as facilities, manpower, funds, etc. allocated to it, and so its processing time can be reduced when additional resources are assigned to the job. A scheduling problem in which the processing times of the jobs can be reduced at some expense is called a scheduling problem with *controllable* processing times. Scheduling problems with controllable processing times have gained importance in scheduling research since the pioneering works of Vickson [23, 24]. For a survey of this area until 1990, the reader is referred to [17]. Recent results include [2, 3, 4, 10, 14, 15, 20, 22, 26].

*A preliminary version of this paper appeared in the Seventh Italian Conference on Theoretical Computer Science (ICTCS'01).

†Partially supported by EU project APPOL, "Approximation and Online Algorithms", IST-1999-14084.

‡Supported by Swiss National Science Foundation project 200021-104017/1, "Power Aware Computing", and by the "Metaheuristics Network", grant HPRN-CT-1999-00106.

§Partially supported by Natural Sciences and Engineering Research Council of Canada grant R3050A01.

1.1 Job Shop Scheduling with Controllable Processing Times

The job shop scheduling problem is a fundamental problem in Operations Research. In this problem there is a set $\mathcal{J} = \{J_1, ..., J_n\}$ of jobs that must be processed by a group of m machines. Every job J_j consists of an ordered sequence of at most μ operations $O_{1j}, O_{2j}, \ldots, O_{k_j j}$, $k_j \leq \mu$. For every operation O_{ij} there is a specific machine m_{ij} that must process it during p_{ij} units of time. A machine can process only one operation at a time, and for any job at most one of its operations can be processed at any moment. The problem is to schedule the jobs so that the maximum completion time $C_{\max}$ is minimized. Time $C_{\max}$ is called the *length* or *makespan* of the schedule.

In the *non-preemptive* version of the job shop scheduling problem, operations must be processed without interruption. The *preemptive* version allows an operation to be interrupted and continued at a later time. In the job shop scheduling problem with controllable processing times a feasible solution is specified by a schedule σ that indicates the starting times for the operations and a vector δ that gives their processing times and costs. Let us denote by $T(\sigma, \delta)$ the makespan of schedule σ with processing times according to δ, and let $C(\delta)$ be the *total cost* of δ. We define, and study, the following three optimization problems.

P1. Minimize $T(\sigma, \delta)$, subject to $C(\delta) \leq \kappa$, for some given value $\kappa > 0$.

P2. Minimize $C(\delta)$, while ensuring that $T(\sigma, \delta) \leq \tau$, for some given value $\tau > 0$.

P3. Minimize $T(\sigma, \delta) + \alpha C(\delta)$, for some given value $\alpha > 0$.

We consider two variants of each one of the three above problems. The first variant allows *continuous* changes to the processing times of the operations. In this case, we assume that the cost of reducing the processing time of an operation is an affine function of the processing time. This is a common assumption made when studying problems with controllable processing times [20, 22]. The second variant allows only discrete changes to the processing times of the operations, and there is a finite set of possible processing times and costs for every operation O_{ij}.

1.2 Known Results

The job shop scheduling problem is considered to be one of the most difficult to solve problems in combinatorial optimization, both, from the theoretical and the practical points of view. The problem is NP-hard even if each job has at most three operations, there are only two machines, and processing times are fixed [8]. The job shop scheduling problem is also NP-hard if there are only 3 jobs and 3 machines [21]. Moreover, the problem is NP-hard in the strong sense if each job has at most three operations and there are only three machines [6]. The same result holds if preemptive schedules are allowed [8]. The problems addressed in this paper are all generalizations of the job shop scheduling problem with fixed

processing times, and therefore, they are strongly NP-hard. Moreover, Nowicki and Zdrzalka [16] show that the version of problem P3 for the less general flow shop problem with continuously controllable processing times is NP-hard even when there are only two machines.

The practical importance of NP–hard problems necessitates efficient ways of dealing with them. A very fruitful approach has been to relax the notion of optimality and settle for *near–optimal* solutions. A near–optimal solution is one whose objective function value is within some small multiplicative factor of the optimal. *Approximation algorithms* are heuristics that in polynomial time provide provably near-optimal solutions. A *Polynomial Time Approximation Scheme* (PTAS for short) is an approximation algorithm which produces solutions of value within a factor $(1+\varepsilon)$ of the optimum, for any value $\varepsilon > 0$.

Williamson et al. [25] proved that the non-preemptive job shop scheduling problem does not have a polynomial time approximation algorithm with worst case bound smaller than $\frac{5}{4}$ unless $P = NP$. The best known approximation algorithm [7] has worst case bound $O((\log(m\mu)\log(min(m\mu, p_{max}))/\log\log(m\mu))^2)$, where p_{max} is the largest processing time among all operations. For those instances where m and μ are fixed (the restricted case we are focusing on in this paper), Shmoys et al. [19] gave a $(2+\varepsilon)$-approximation algorithm for any fixed $\varepsilon > 0$. This result has recently been improved by Jansen et al. [12, 5] who designed a PTAS that runs in linear time. On the other hand the preemptive version of the job shop scheduling problem is NP-complete in the strong sense even when $m = 3$ and $\mu = 3$ [8]. When processing times are controllable, to the best of our knowledge, the only known closely related result is due to Nowicki [15]. In [15] the flow shop scheduling problem with controllable processing times is addressed, and a 4/3-approximation algorithm for the flow shop version of problem P3 is described.

1.3 New Results

We present the first known polynomial time approximation schemes for problems P1, P2, and P3, when the number m of machines and the number μ of operations per job are fixed.

For problem P1 we present an algorithm which finds a solution (σ, δ) of cost at most κ and makespan no larger than $(1+O(\varepsilon))$ times the optimum makespan, for any given value $\varepsilon > 0$. We observe that, since for problem P2 deciding whether there is a solution of length $T(\sigma, \delta) \leq \tau$ is already NP-complete, then unless P=NP, we might only expect to obtain a solution with cost at most the optimal cost and makespan not greater than $\tau(1+\varepsilon)$, for a given value $\varepsilon > 0$. Our algorithm for problem P3 finds a solution of value at most $(1+O(\varepsilon))$ times the optimum, for any $\epsilon > 0$.

Our algorithms can handle both, continuously and discretely controllable processing times, and they can be extended to the case of convex piecewise linear cost functions. Our algorithms have a worst case performance ratio that is better than the 4/3-approximation algorithm for problem P3 described in

Nowicki [15]. Moreover, the linear time complexity of our PTAS for problem P3 is the best possible with respect to the number of jobs.

Our algorithms are based on a paradigm that has been successfully applied to solve other scheduling problems. First, partition the set of jobs into "large", "medium" and "small" jobs. The sets of large and medium jobs have a constant number of jobs each. We compute all possible schedules for the large jobs. Then, for each one of them, schedule the remaining jobs inside the empty gaps that the large jobs leave by first using a linear program to assign jobs to gaps, and then computing a feasible schedule for the jobs assigned to each gap.

A major difficulty with using this approach for our problems is that the processing times and costs of the operations are not fixed, so we must determine these values before we can use the above approach. One possibility is to use a linear program to assign jobs to gaps and to determine the processing times and costs of the operations. But, we must be careful since, for example, a natural extension of the linear program described in [12] defines a polytope with an exponential number of extreme points, and it does not seem to be possible to solve such linear program in polynomial time. We show how to construct a small polytope with only a polynomial number of extreme points that contains all the optimum solutions of the above linear program. This polytope is defined by a linear program that can be solved exactly in polynomial time and approximately, to within any pre-specified precision, in fully polynomial time (i.e. polynomial also in the reverse of the precision).

Our approach is general enough that it can be used to design polynomial time approximation schemes for both the discrete and the continuous versions of problems P1-P3 with and without preemptions. In this paper we present polynomial time approximation schemes for the continuous version of problems P1-P3, and a PTAS for the discrete version of P3. The polynomial time approximation schemes for the discrete version of P1 and P2 can be easily derived from the results described here. We present a series of transformations that simplify any instance of the above problems. Some transformations may potentially increase the value of the objective function by a factor of $1 + O(\varepsilon)$, for a given value $\varepsilon > 0$, so we can perform a constant number of them while still staying within $1 + O(\varepsilon)$ of the optimum. We say that this kind of transformations produce $1 + O(\varepsilon)$ *loss*. A transformation that does not modify the value of the optimum solution is said to produce *no loss*. In the following, for simplicity of notation, we assume that $1/\varepsilon$ is an integral value.

1.4 Organization

The rest of the paper is organized in the following way. We first address problems with continuous processing times. In Sections 2 and 3, we present polynomial time approximation schemes for problem P1 with and without preemptions. In Sections 4 and 5 we present polynomial time approximation schemes for problems P2 and P3. In Section 6 we study problem P3 with discrete processing times, and show how to design a linear time PTAS for it.

2 Non-Preemptive Problem P1 with Continuous Processing Times

Problem P1 is to compute a schedule with minimum makespan and cost at most κ for some given value κ. In the case of continuously controllable processing times, we assume that for each operation O_{ij} there is an interval $[\ell_{ij}, u_{ij}]$, $0 \leq \ell_{ij} \leq u_{ij}$, specifying its possible processing times. The cost for processing an operation O_{ij} in time ℓ_{ij} is denoted as $c_{ij}^{\ell} \geq 0$ and for processing it in time u_{ij} the cost is $c_{ij}^{u} \geq 0$. For any value $\delta_{ij} \in [0,1]$ the cost for processing operation O_{ij} in time

$$p_{ij}^{\delta_{ij}} = \delta_{ij}\ell_{ij} + (1-\delta_{ij})u_{ij}$$

is

$$c_{ij}^{\delta_{ij}} = \delta_{ij}c_{ij}^{\ell} + (1-\delta_{ij})c_{ij}^{u}.$$

We assume that ℓ_{ij}, u_{ij}, c_{ij}^{ℓ}, c_{ij}^{u} and δ_{ij} are rational numbers. Moreover, without loss of generality, we assume that for every operation O_{ij}, $c_{ij}^{u} \leq c_{ij}^{\ell}$, and if $c_{ij}^{u} = c_{ij}^{\ell}$ then $u_{ij} = \ell_{ij}$.

Remark 1 *For simplicity, we assume that the total cost of the solution is at most 1. To see why, let us consider an instance of problem P1. Divide all c_{ij}^{u} and c_{ij}^{ℓ} values by κ to get an equivalent instance in which the bound on the total cost of the solution is* 1*, i.e.*

$$C(\delta) = \sum_{j=1}^{n}\sum_{i=1}^{\mu} c_{ij}^{\delta_{ij}} \leq 1.$$

Moreover, since the total cost is at most 1*, without loss of generality we can assume that the maximum cost for each operation is at most 1. More precisely, if $c_{ij}^{\ell} > 1$ for some operation O_{ij}, we set $c_{ij}^{\ell} = 1$ and make*

$$\ell_{ij} = \frac{u_{ij}(c_{ij}^{\ell}-1) - \ell_{ij}(c_{ij}^{u}-1)}{c_{ij}^{\ell} - c_{ij}^{u}}$$

to get an equivalent instance of the problem in which $c_{ij}^{\ell} \leq 1$ for all operations O_{ij}.

2.1 Lower and Upper Bounds for the Problem

In the following we compute lower (LB) and upper (UB) bounds for the optimum makespan. Let $U = \sum_{j=1}^{n}\sum_{i=1}^{\mu} u_{ij}$. Consider an optimal solution (σ^*, δ^*) for problem P1 and let us use OPT to denote the optimum makespan. Let $P^* = \sum_{j=1}^{n}\sum_{i=1}^{\mu} p_{ij}^{\delta_{ij}^*} = \sum_{j=1}^{n}\sum_{i=1}^{\mu}(\ell_{ij}-u_{ij})\delta_{ij}^* + U$ be the sum of the processing times of all jobs in this optimum solution. Define $c_{ij} = c_{ij}^{\ell} - c_{ij}^{u}$, so

$C(\delta^*) = \sum_{j=1}^{n}\sum_{i=1}^{\mu}(c_{ij}\delta^*_{ij} + c^u_{ij}) \leq 1$. Note that δ^* is a feasible solution for the following linear program.

$$\begin{array}{lll} \min & P = \sum_{j=1}^{n}\sum_{i=1}^{\mu}(\ell_{ij} - u_{ij})x_{ij} + U & \\ \text{s.t.} & \sum_{j=1}^{n}\sum_{i=1}^{\mu} x_{ij}c_{ij} \leq 1 - \sum_{j=1}^{n}\sum_{i=1}^{\mu} c^u_{ij}. & \\ & 0 \leq x_{ij} \leq 1 & \forall j = 1, ..., n \text{ and } i = 1, ..., \mu. \end{array}$$

Let $\tilde{x}$ be an optimal solution for this linear program and let $P(\tilde{x})$ be its value. Observe that $P(\tilde{x}) \leq P^*$ and $C(\tilde{x}) \leq 1$. Moreover, note that

- if $1 - \sum_{j=1}^{n}\sum_{i=1}^{\mu} c^u_{ij} < 0$, then there exists no solution with cost at most 1.
- If $1 - \sum_{j=1}^{n}\sum_{i=1}^{\mu} c^u_{ij} = 0$ then in any feasible solution with cost at most 1, the processing time of each operation O_{ij} must be u_{ij}. In this case we know that $U/m \leq OPT \leq U$, since the total sum of the processing times of all the operations is U . By dividing all processing times by U, we get the bounds $LB = 1/m \leq OPT \leq UB = 1$.
- If $1 - \sum_{j=1}^{n}\sum_{i=1}^{\mu} c^u_{ij} > 0$, we simplify the instance by dividing all costs by $1 - \sum_{j=1}^{n}\sum_{i=1}^{\mu} c^u_{ij}$. Then, the above linear program can be simplified as follows,

$$\begin{array}{lll} \max & \sum_{j=1}^{n}\sum_{i=1}^{\mu} t_{ij}x_{ij} & \\ \text{s.t.} & \sum_{j=1}^{n}\sum_{i=1}^{\mu} x_{ij}w_{ij} \leq 1. & \\ & 0 \leq x_{ij} \leq 1 & j = 1, ..., n \text{ and } i = 1, ..., \mu. \end{array}$$

  where $t_{ij} = u_{ij} - \ell_{ij}$ and $w_{ij} = c_{ij}/(1 - \sum_{j=1}^{n}\sum_{i=1}^{\mu} c^u_{ij})$.

This linear program is a relaxation of the classical knapsack problem, which can be solved as follows [13]. Sort the operations in non-increasing order of ratio value t_{ij}/w_{ij}. Consider, one by one, the operations O_{ij} in this order, assigning to each one of them value $x_{ij} = 1$ as long as their total weight $\sum_{j=1}^{n}\sum_{i=1}^{\mu} x_{ij}w_{ij}$ is at most 1. If necessary, assign to the next operation a fractional value to ensure that the total weight of the operations is exactly 1. Set $x_{ij} = 0$ for all the remaining operations. This algorithm runs in $O(n\mu)$ time [13]. If we schedule all the jobs one after another with processing times as defined by the optimal solution $\tilde{x}$ of the knapsack problem, we obtain a feasible schedule for the jobs $\mathcal{J}$ with makespan $P(\tilde{x})$. Since $P(\tilde{x}) \geq P^* \geq OPT \geq P^*/m \geq P(\tilde{x})/m$, then by dividing all ℓ_{ij} and u_{ij} values by $P(\tilde{x})$, we get the same bounds as above, i.e., $LB = 1/m$ and $UB = 1$.

2.2 Overview of the Algorithm

We might assume without loss of generality that each job has exactly μ operations. To see this, consider a job J_i with $\mu_i < \mu$ operations $O_{1i}, O_{2i}, \ldots O_{\mu_i i}$. we create $\mu - \mu_i$ new operations O_{ij} with zero processing time and cost, i.e., we set

$\ell_{ij} = u_{ij} = 0$ and $c^u_{ij} = c^\ell_{ij} = 0$. Each one of these operations must be processed in the same machine which must process operation $O_{\mu_i i}$. Clearly, the addition of these operations does not change the makespan of any feasible schedule for $\mathcal{J}$.

We briefly sketch the algorithm here and give the details in the following sections. Let $\kappa > 0$ be the upper bound on the cost of the solution and $\varepsilon > 0$ be a positive value. Consider an optimum solution (σ^*, δ^*) for the problem; let τ^* be the makespan of this optimum solution. Our algorithm finds a solution (σ, δ) of cost $C(\delta) \leq \kappa$ and makespan at most $(1 + O(\varepsilon))\tau^*$.

1. Partition the jobs into 3 groups: $\mathcal{L}$ (large), $\mathcal{M}$ (medium), and $\mathcal{S}$ (small). Set $\mathcal{L}$ contains the largest $k \leq m^2/\varepsilon$ jobs according to the optimum solution (σ^*, δ^*). Set $\mathcal{M}$ contains the $q\mu m^2/\varepsilon - 1$ next largest jobs, where $q = 6\mu^4 m^3/\varepsilon$. The value for k is chosen so that the total processing time of the medium jobs in solution (σ^*, δ^*) is at most $\epsilon\tau^*$ (the exact value for k is given in Section 2.6). The set $\mathcal{S}$ of small jobs is formed by the remaining jobs. Note that $|\mathcal{L}|$ and $|\mathcal{M}|$ are constant. It is somewhat surprising to know that we can determine the sets $\mathcal{L}, \mathcal{M}$, and $\mathcal{S}$, even when an optimum solution $\sigma^*, \delta^*)$ is not known. We show in Section 2.6 how to do this. From now on we assume that we know sets $\mathcal{L}$, $\mathcal{M}$ and $\mathcal{S}$. An operation that belongs to a large, medium, or small job is called a *large, medium, or small operation*, respectively, regardless of its processing time.

2. Determine the processing times and costs for the medium operations.

3. Try all possible ways of scheduling the large jobs on the machines. Each feasible schedule for $\mathcal{L}$ leaves a set of empty gaps where the small and medium jobs can be placed. For each feasible schedule for the large jobs use a linear program to determine the processing times and costs for the small and large operations, and to determine the gaps where each small and medium operation must be scheduled. then, transform this assignment into a feasible schedule for the entire set of jobs.

4. . Output the schedule with minimum length found in Step 3.

2.3 Simplifying the Input

We can simplify the input by reducing the number of possible processing times for the operations. We begin by showing that it is possible to upper bound the processing times of small operations by $\frac{\varepsilon}{qk\mu m}$.

Lemma 2 *With no loss, for each small operation O_{ij} we can set $u_{ij} \leftarrow \bar{u}_{ij}$ and $c^u_{ij} \leftarrow \frac{c^\ell_{ij} - c^u_{ij}}{u_{ij} - \ell_{ij}}(u_{ij} - \bar{u}_{ij}) + c^u_{ij}$, where $\bar{u}_{ij} = \min\{u_{ij}, \frac{\varepsilon}{qk\mu m}\}$.*

Proof. As it was shown in Section 2.1, the optimal makespan is at most 1 and, therefore, the sum of the processing times of all jobs cannot be larger than

m. Let $P_j^* = \sum_{i=1}^{\mu} p_{ij}^{\delta_{ij}^*}$ be the sum of the processing times of the operations of job J_j according to an optimum solution (σ^*, δ^*). Let p be the length of the longest small job according to this optimum solution. By definition of $\mathcal{L}$ and $\mathcal{M}$,

$$|\mathcal{L} \cup \mathcal{M}| \cdot p = \frac{k\mu m^2 q}{\varepsilon} p \leq \sum_{J_j \in \mathcal{M} \cup \mathcal{L}} P_j^* \leq m,$$

and so $p \leq \frac{\varepsilon}{qk\mu m}$. Hence, the length of an optimum schedule is not increased if the largest processing time u_{ij} of any small operation O_{ij} is set to $\bar{u}_{ij} = \min\{u_{ij}, \frac{\varepsilon}{qk\mu m}\}$. If we do this, we also need to set $c_{ij}^u \leftarrow \frac{c_{ij}^\ell - c_{ij}^u}{u_{ij} - \ell_{ij}}(u_{ij} - \bar{u}_{ij}) + c_{ij}^u$ (this is the cost to process operation O_{ij} in time $\bar{u}_{ij}$). ■

In order to compute a $1 + O(\varepsilon)$-approximate solution for P1 we show that it is sufficient to consider only a constant number of different processing times and costs for the medium jobs.

Lemma 3 *There exists a $(1 + 2\varepsilon)$-optimal schedule where each medium operation has processing time of the form $\frac{\varepsilon}{m|\mathcal{M}|\mu}(1 + \varepsilon)^i$, for $i \in \mathbb{N}$.*

Proof. Let A be the set of medium operations for which $p_{ij}^{\delta_{ij}^*} \leq \frac{\varepsilon}{m|\mathcal{M}|\mu}$. Since $c_{ij}^u \leq c_{ij}^\ell$, by increasing the processing time of any operation, its corresponding cost cannot increase. Therefore, if we increase the processing times for the operations in A to $\frac{\varepsilon}{m|\mathcal{M}|\mu}$, the makespan increases by at most $|A|\frac{\varepsilon}{m|\mathcal{M}|\mu} \leq \varepsilon/m$, and so the length of an optimum solution would increase by at most a factor of $1 + \varepsilon$. For the remaining medium operations, round up their processing times $p_{ij}^{\delta_{ij}^*}$ to the nearest value of the form $\frac{\varepsilon}{m|\mathcal{M}|\mu}(1 + \varepsilon)^i$, for some $i \in \mathbb{N}$. Since this rounding increases the processing time by at most a factor $1 + \varepsilon$, the value of an optimum solution increases by at most the same factor, $1 + \varepsilon$. ■

By the discussion in Section 2.1, the processing times of the operations are at most 1. By the previous lemma, the number of different processing times for medium operations is $O(\log(m|\mathcal{M}|\mu)/\varepsilon)$ (clearly, the same bound applies to the number of different costs). Since there is a constant number of medium operations after the above rounding, there is also a constant number of choices for the values of their processing times and costs.

Let the rounded processing times and costs for the medium operations be denoted as $\bar{p}_{ij}$ and $\bar{c}_{ij}$. Below we show that when the medium operations are processed according to these $(\bar{p}_{ij}, \bar{c}_{ij})$-values, it is possible to compute a $1+O(\varepsilon)$-approximate solution for problem P1 in polynomial time. Let us consider all $O(\log(m|\mathcal{M}|\mu)/\varepsilon)$ possible choices for processing times for the medium operations. Clearly, for one of such choices the processing times and costs are as described in Lemma 3. Hence, from now on, we assume that we know these $(\bar{p}_{ij}, \bar{c}_{ij})$-values for the medium operations. In order to simplify the following discussion, for each medium operation O_{ij} we set $\ell_{ij} = u_{ij} = \bar{p}_{ij}$ and $c_{ij}^\ell = c_{ij}^u = \bar{c}_{ij}$, thus, fixing their processing times and costs.

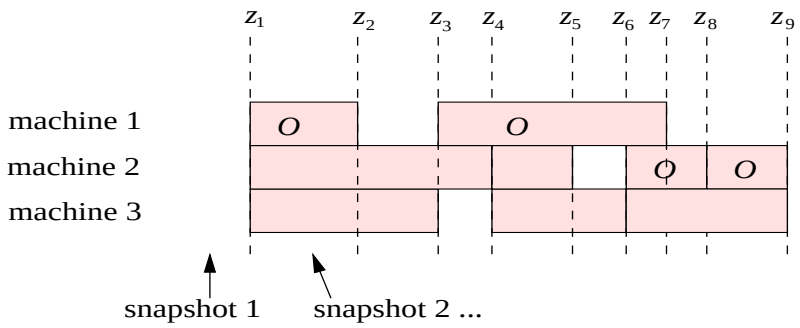

Figure 1: A feasible schedule for the set of jobs $\{J_1, J_2, J_3\}$, each consisting of 3 large operations, that respects the following relative schedule: $R = (s_{11}, s_{13}, s_{12}, f_{11}, f_{13}, s_{23}, f_{12}, s_{21}, s_{22}, f_{21}, f_{22}, s_{32}, s_{31}, f_{23}, f_{32}, s_{33}, f_{31}, f_{33})$.

2.4 Relative Schedules

A *relative schedule* for the large operations is an ordering of the starting and ending times of the operations. We say that a feasible schedule S for the large operations *respects* a relative schedule R if the starting and ending times of the operations as defined by S are ordered as indicated in R (breaking ties in an appropriate way), see Figure 1. Fix a relative schedule R for the large operations. The starting, s_{ij}, and finishing, f_{ij}, times of an operation O_{ij} define an interval $[s_{ij}, f_{ij}]$ where each operation O_{ij} must be processed (see Figure 1). Let

$$z_1 < z_2 < ... < z_{g-1}$$

be the ordered sequence of all different s_{ij} and f_{ij} values, for $j = 1, ..., n$ and $i = 1, ..., \mu$. Let us introduce two additional values $z_g \geq z_{g-1}$ and $z_0 \leq z_1$ to bound intervals without large operations. The intervals

$$M(v) := [z_{v-1}, z_v] \text{ for } v = 1, ..., g$$

are called *snapshots* (see Figure 1). Let $M(1), M(2), \ldots, M(g)$, be the snapshots defined by R. Note that snapshots $M(1)$ and $M(g)$ are empty. The number of snapshots g is at most $g \leq 2k\mu + 1$.

Lemma 4 *The number of different relative schedules for the large jobs is at most* $(2ek)^{2k\mu}$*, where* e *is the Euler number.*

Proof. The number of possible starting times for the operations of a large job J_j is at most the number of subsets of size μ that can be chosen from a set of $(2k\mu - 1)$ positions (there are $2\mu k - 1$ choices for the starting times of each operation of J_j). Since each large operation can end in the same snapshot in which it starts, the number of ways of choosing the starting and ending times

of the operations of a large job is at most the number of subsets of size 2μ that can be chosen from a set of $2(2k\mu - 1)$ positions (we consider two positions associated with each snapshot, one to start and one to end an operation, but both positions denote the same snapshot). For each large job J_j there are at most $\binom{4\mu k-2}{2\mu}$ different choices of snapshots where operations of J_j can start and end. Since $\binom{4\mu k-2}{2\mu} = \frac{(4\mu k-2)(4\mu k-3)\ldots(4\mu k-2\mu-1)}{(2\mu)!} \leq \frac{(4\mu k)^{2\mu}}{(2\mu/e)^{2\mu}} = (2ek)^{2\mu}$ and the number of large jobs is k, then there are at most $(2ek)^{2k\mu}$ different relative schedules. ■

2.5 Assigning Small and Medium Operations to Snapshots

By Lemma 4 the number of different relative schedules is bounded by a constant. Our algorithm considers all relative schedules for the large jobs, one of which must be equal to the relative schedule R^* defined by some optimum solution (σ^*, δ^*). We show that when relative schedule R^* is used, we can find in polynomial time a $1 + O(\varepsilon)$-approximate solution for problem P1. Given relative schedule R^*, to obtain a solution for problem P1 that respects R^* we must select the processing times for the large and small operations and we must schedule the medium and small operations within the snapshots defined by R^*. We use a linear program to compute the processing times and costs for the small and large operations, and to decide the snapshots where the small and medium operations must be placed. Then, we find a feasible schedule for the operations in every snapshot.

Let us first describe the linear program. We use a variable x_{ij}^{ℓ} for each large operation O_{ij}; this variable defines the processing time and cost of operation O_{ij}. For convenience we define another variable x_{ij}^{u} with value $1 - x_{ij}^{\ell}$. The processing time of operation O_{ij} is then

$$x_{ij}^{\ell}\ell_{ij} + x_{ij}^{u}u_{ij},$$

and its cost is

$$x_{ij}^{\ell}c_{ij}^{\ell} + x_{ij}^{u}c_{ij}^{u}.$$

Let α_{ij} be the snapshot where the large operation O_{ij} starts processing in the relative schedule R^* and let β_{ij} be the snapshot where it finishes processing. Let $Free(R^*)$ be the set of (snapshot, machine) pairs such that no large operation is scheduled by R^* in snapshot $M(\ell)$ on machine h. For every medium and small job J_j, let Λ_j be the set of tuples of the form $(s_1, s_2, \ldots, s_\mu)$ such that $1 \leq s_1 \leq s_2 \leq \ldots \leq s_\mu \leq g$, and $(s_i, m_{ij}) \in Free(R^*)$, for all $i = 1, \ldots, \mu$. This set Λ_j determines the free snapshots where it is possible to place the operations of job J_j.

Let $\Delta = \{(\delta_1, \delta_2, \ldots, \delta_\mu) \mid \delta_k \in \{0, 1\} \text{ for all } k = 1, \ldots, \mu\}$, be the set of all μ-dimensional binary vectors. For each medium and small job J_j we define a set of at most $(2g)^\mu$ variables $x_{j,(s,\delta)}$, where $s \in \Lambda_j$ and $\delta \in \Delta$. These variables will indicate the snapshots where small and medium jobs will be scheduled. To

understand the meaning of these variables, let us define

$$x_{ij}(w,1) = \sum_{(s,\delta)\in\Lambda_j\times\Delta,\ s_i=w,\ \delta_i=1} x_{j,(s,\delta)}$$

and

$$x_{ij}(w,0) = \sum_{(s,\delta)\in\Lambda_j\times\Delta,\ s_i=w,\ \delta_i=0} x_{j,(s,\delta)},$$

for each operation i, job J_j, and snapshot $M(w)$. Given a set of values for the variables $x_{j,(s,\delta)}$, they define the processing times for the jobs and they also give an assignment of medium and small operations to snapshots: the amount of time that an operation O_{ij} is processed within snapshot $M(w)$ is $x_{ij}(w,1)\cdot\ell_{ij} + x_{ij}(w,0)\cdot u_{ij}$, and the fraction of O_{ij} that is assigned to this snapshot is $x_{ij}(w,0) + x_{ij}(w,1)$.

Example 1 *Consider an instance with $\mu = 2$ having a job J_j with the following processing time functions:*

$$\begin{array}{ll} O_{1j} & p_{1j}^{\delta_{1j}} = 0.5\cdot\delta_{1j} + (1-\delta_{1j}) \\ O_{2j} & p_{2j}^{\delta_{2j}} = 0.1\cdot\delta_{2j} + 0.5\cdot(1-\delta_{2j}) \end{array}$$

Furthermore, assume that in some feasible solution $p_{1j} = 0.65$ and $p_{2j} = 0.34$, and operation O_{1j} is placed on the third snapshot, while operation O_{2j} is in the seventh snapshot. By setting $x_{j,((3,7),(0,0))} = 0.3$, $x_{j,((3,7),(0,1))} = 0$, $x_{j,((3,7),(1,0))} = 0.3$ and $x_{j,((3,7),(1,1))} = 0.4$, we see that $p_{1j} = (x_{j,((3,7),(0,0))} + x_{j,((3,7),(0,1))}) + (x_{j,((3,7),(1,0))} + x_{j,((3,7),(1,1))}) \times 0.5 = 0.65$, and $p_{2j} = (x_{j,((3,7),(0,0))} + x_{j,((3,7),(1,0))}) \times 0.5 + (x_{j,((3,7),(0,1))} + x_{j,((3,7),(1,1))}) \times 0.1 = 0.34$.

For each snapshot $M(\ell)$ we use a variable t_ℓ to denote its length. For any $(\ell,h) \in Free(R^*)$, we define the load $L_{\ell,h}$ on machine h in snapshot $M(\ell)$ as the total processing time of the small and medium operations that are assigned to h in $M(\ell)$, i.e.,

$$L_{\ell,h} = \sum_{J_j\in\mathcal{S}\cup\mathcal{M}} \sum_{\substack{i=1 \\ m_{ij}=h}}^{\mu} \left(x_{ij}(\ell,1)\ell_{ij} + x_{ij}(\ell,0)u_{ij}\right). \tag{1}$$

The total cost of a schedule is given by

$$C = \sum_{J_j\in\mathcal{S}\cup\mathcal{M}} \sum_{\ell=1}^{g} \sum_{i=1}^{\mu} \left(x_{ij}(\ell,1)c_{ij}^{\ell} + x_{ij}(\ell,0)c_{ij}^{u}\right) + \sum_{J_j\in\mathcal{L}} \sum_{i=1}^{\mu} (x_{ij}^{\ell}c_{ij}^{\ell} + x_{ij}^{u}c_{ij}^{u}).$$

We use the following linear program $LP(R^*)$ to determine processing times and costs of large and small operations, and to allocate small and medium operations

to snapshots.

$$
\begin{array}{lll}
\min & T = \sum_{\ell=1}^{g} t_\ell & \\
\text{s.t.} & C \leq 1, & \text{(c1)} \\
& \sum_{\ell=\alpha_{ij}}^{\beta_{ij}} t_\ell = x_{ij}^{\ell}\ell_{ij} + x_{ij}^{u}u_{ij}, \quad J_j \in \mathcal{L},\ i = 1,\ldots,\mu, & \text{(c2)} \\
& x_{ij}^{\ell} + x_{ij}^{u} = 1, \quad J_j \in \mathcal{L},\ i = 1,\ldots,\mu, & \text{(c3)} \\
& \sum_{(s,\delta)\in\Lambda_j\times\Delta} x_{j,(s,\delta)} = 1, \quad J_j \in \mathcal{S}\cup\mathcal{M}, & \text{(c4)} \\
& L_{\ell,h} \leq t_\ell, \quad (\ell,h) \in Free(R^*), & \text{(c5)} \\
& x_{ij}^{\ell}, x_{ij}^{u} \geq 0, \quad J_j \in \mathcal{L},\ i = 1,\ldots,\mu, & \text{(c6)} \\
& x_{j,(s,\delta)} \geq 0, \quad J_j \in \mathcal{S}\cup\mathcal{M},\ (s,\delta)\in\Lambda_j\times\Delta, & \text{(c7)} \\
& t_\ell \geq 0, \quad \ell = 1,\ldots,g. & \text{(c8)}
\end{array}
$$

In this linear program the value of the objective function T is the length of the schedule, which we want to minimize. Constraint (c1) ensures that the total cost of the solution is at most one. Condition (c2) requires that the total length of the snapshots where a large operation is scheduled is exactly equal to the length of the operation. Constraint (c4) assures that every small and medium operation is completely assigned to snapshots, while constraint (c5) checks that the total load of every machine h during each snapshot ℓ does not exceed the length of the snapshot. Let (σ^*, δ^*) denote an optimal schedule where the processing times and costs of medium jobs are fixed as described in the previous section.

Lemma 5 *The optimal solution of $LP(R^*)$ has value no larger than the makespan of (σ^*, δ^*).*

Proof. We only need to show that (σ^*, δ^*) defines a feasible solution for $LP(R^*)$. For any operation O_{ij}, let $p_{ij}^{\delta_{ij}^*}(w)$ be the amount of time that O_{ij} is processed during snapshot $M(w)$ in the optimum schedule (σ^*, δ^*). We determine now the values for the variables $t_\ell^*, x_{ij}^{\ell*}, x_{ij}^{u*}$, and $x_{j,(s,\delta)}^*$ defined by (σ^*, δ^*). Set $x_{ij}^{\ell*} = \delta_{ij}^*$ and $x_{ij}^{u*} = 1 - \delta_{ij}^*$ for all large operations O_{ij}. The values for the variables t_ℓ^* can be easily obtained from the snapshots defined by the large operations. Let

$$x_{ij}^*(w,1) = \delta_{ij}^* \frac{p_{ij}^{\delta_{ij}^*}(w)}{p_{ij}^{\delta_{ij}^*}}$$

and

$$x_{ij}^*(w,0) = (1-\delta_{ij}^*) \frac{p_{ij}^{\delta_{ij}^*}(w)}{p_{ij}^{\delta_{ij}^*}}.$$

The processing time $p_{ij}^{\delta_{ij}^*}(w)$ and cost $c_{ij}^{\delta_{ij}^*}(w)$ of O_{ij} can be written as $x_{ij}^*(w,1)\ell_{ij} + x_{ij}^*(w,0)u_{ij}$ and $x_{ij}^*(w,1)c_{ij}^{\ell} + x_{ij}^*(w,0)c_{ij}^{u}$, respectively. Now we show that there is a feasible solution $x_{j,(s,\delta)}^*$ for $LP(R^*)$ such that

(*i*) $x_{ij}^*(w,1) = \sum_{(s,\delta)\in\Lambda_j\times\Delta,\ s_i=w,\ \delta_i=1} x_{j,(s,\delta)}^*$, and

(*ii*) $x^*_{ij}(w,0) = \sum_{(s,\delta)\in\Lambda_j\times\Delta,\ s_i=w,\ \delta_i=0} x^*_{j,(s,\delta)}$.

Therefore, for this solution, $p_{ij}^{\delta^*_{ij}}(w)$ and $c_{ij}^{\delta^*_{ij}}(w)$ are linear combinations of the variables $x^*_{j,(s,\delta)}$. We determine the values for the variables $x^*_{j,(s,\delta)}$ as follows.

1. For each job $J_j \in \mathcal{S}\cup\mathcal{M}$ do
2. (a) Compute $S_j = \{(i,w,d) \mid x^*_{ij}(w,d) > 0, i = 1,\ldots,\mu,\ w = 1,\ldots,g,\ d = 0,1\}$.
3. (a) If $S_j = \emptyset$ then exit.
4. (a) Let $f = \min\{x^*_{ij}(w,d) \mid (i,w,d) \in S_j\}$ and let I, W, and D be such that $x^*_{Ij}(W,D) = f$.
5. (a) Let $s = (s_1, s_2, \ldots, s_\mu) \in \Lambda_j$ and $\delta = (d_1, d_2, \ldots, d_\mu) \in \Delta$ be such that $s_I = W$, $d_I = D$ and $x^*_{ij}(s_i,d_i) > 0$, for all $i = 1,\ldots,\mu$.
6. (a) $x^*_{j,(s,\delta)} \leftarrow f$
7. (a) $x^*_{ij}(s_i,d_i) \leftarrow x^*_{ij}(s_i,d_i) - f$ for all $i = 1,2,\ldots,\mu$.
8. (a) Go back to step 2.

With this assignment of values to the variables $x^*_{j,(s,\delta)}$, equations (*i*) and (*ii*) above hold for all jobs $J_j \in \mathcal{S}\cup\mathcal{M}$, all operations, and all snapshots w. Therefore, the above solution for $LP(R^*)$ schedules the jobs in the same positions and with the same processing times as the optimum schedule (σ^*, δ^*). ■

2.6 Finding a Feasible Schedule

The linear program $LP(R^*)$ has at most $1 + \mu k + n - k + mg$ constraints. By condition (c3) each one of the μk large operations O_{ij} must have at least one of its variables x^ℓ_{ij} or x^u_{ij} set to a positive value. By condition (c4) every one of the $n-k$ small and medium jobs J_j must have at least one of its variables $x_{j,(s,\delta)}$ set to a positive value. Furthermore, there has to be at least one snapshot ℓ for which $t_\ell > 0$. Since in any basic feasible solution of $LP(R^*)$ the number of variables that receive positive values is at most equal to the number of rows of the constraint matrix, then in a basic feasible solution there are at most mg variables with fractional values.

This means that in the schedule defined by a basic feasible solution of $LP(R^*)$ at most mg medium and small jobs receive fractional assignments, and therefore, there are at most that many jobs from $\mathcal{M}\cup\mathcal{S}$ for which at least one operation is split into two or more different snapshots. Let $\mathcal{F}$ be the set of jobs that received fractional assignments. We show later how to schedule those jobs. For the moment, let us remove them from our solution.

Even without fractional assignments, the solution for the linear program might still not define a feasible schedule because there may be ordering conflicts

among the small and medium operations assigned to the same snapshot. To eliminate these conflicts, we first remove the set $\mathcal{V} \subseteq \mathcal{M} \cup \mathcal{S}$ of jobs which have at least one operation with processing time larger than $\varepsilon/(\mu^3 m^2 g)$. Since the sum of the processing times of the jobs as defined by the solution of the linear program is at most m, then $|\mathcal{V}| \leq \mu^3 m^3 g/\varepsilon$, so in this step we remove only a constant number of jobs.

Let $\mathcal{O}(\ell)$ be the set of operations from $\mathcal{M} \cup \mathcal{S}$ that remain in snapshot $M(\ell)$. Let $p_{max}(\ell)$ be the maximum processing time among the operations in $\mathcal{O}(\ell)$. Note that $p_{max}(\ell) \leq \varepsilon/(\mu^3 m^2 g)$. Every snapshot $M(\ell)$ defines an instance of the classical job shop scheduling problem, since the solution of $LP(R^*)$ determines the processing time of every operation. Hence, we can use Sevastianov's algorithm [18] to find in $O(n^2\mu^2 m^2)$ time a feasible schedule for the operations in $\mathcal{O}(\ell)$; this schedule has length at most $\bar{t}_\ell = t_\ell + \mu^3 m p_{max}(\ell)$. Hence, we must increase the length of every snapshot $M(\ell)$ to $\bar{t}_\ell$ to accommodate the schedule produced by Sevastianov's algorithm. Summing up all these snapshot enlargements, we get a solution of length at most $T^* + \mu^3 m p_{max}(\ell) g \leq T^*(1+\varepsilon)$, where T^* is the value of an optimum solution for $LP(R^*)$.

It remains to show how to schedule the set of jobs $\mathcal{V} \cup \mathcal{F}$ that we removed. Recall that the value for parameter q is $q = 6\mu^4 m^3/\epsilon$. since, as we showed in Section 2.4 $g \leq 2k\mu + 1$, then the number of jobs in $\mathcal{V} \cup \mathcal{F}$ is

$$|\mathcal{V} \cup \mathcal{F}| \leq \mu^3 m^3 g/\epsilon + mg \leq qk. \tag{2}$$

Lemma 6 *Consider an optimum solution (σ^*, δ^*) for problem P1. Let $P_j^* = \sum_{i=1}^{\mu} p_{ij}^{\delta^*_{ij}}$ denote the length of job J_j according to δ^*. There exists a positive constant k such that if the set of large jobs contains the k jobs J_j with the largest P_j^* value, then $\sum_{J_j \in \mathcal{V} \cup \mathcal{F}} P_j^* \leq \varepsilon/m$.*

Proof. Sort the jobs J_j non-increasingly by P_j^* value, and assume for convenience that $P_1^* \geq P_2^* \geq ... \geq P_n^*$. Partition the jobs into groups $G_1, G_2, ..., G_d$ as follows: $G_i = \{J_{(1+q)^{i-1}+1}, ..., J_{(1+q)^i}\}$. Let $P(G_j) = \sum_{J_i \in G_j} P_j^*$ and let $G_{\rho+1}$ be the first group for which $P(G_{\rho+1}) \leq \varepsilon/m$. Since $\sum_{J_j \in \mathcal{J}} P_j^* \leq m$ and $\sum_{i=1}^{\rho} P(G_i) > \rho\varepsilon/m$ then $\rho < \frac{m^2}{\varepsilon}$. Choose $\mathcal{L}$ to contain all jobs in groups G_1 to G_ρ, and so $k = (1+q)^\rho$. Note that $G_{\rho+1}$ has $(1+q)^{\rho+1} - (1+q)^\rho = qk$ jobs. Since $|\mathcal{V} \cup \mathcal{F}| \leq qk$, then $|G_{\rho+1}| = qk \geq |\mathcal{V} \cup \mathcal{F}|$ and, so,

$$\sum_{J_j \in \mathcal{V} \cup \mathcal{F}} P_j^* \leq \sum_{J_j \in G_\rho} P_j^* \leq \varepsilon/m.$$

■

We select the set of large jobs by considering all subsets of k jobs, for all integer values k of the form $(1+q)^\rho$ and $0 \leq \rho \leq m^2/\varepsilon$. For each choice of k the set of medium jobs is obtained by considering all possible subsets of qk jobs. Since there is only a polynomial number of choices for large and medium jobs, the algorithm runs in polynomial time.

The processing time of every small operation O_{ij} in $\mathcal{V} \cup \mathcal{F}$, is set to $p_{ij} = u_{ij}$, and its cost is $c_{ij} = c^u_{ij}$. Furthermore, recall that we are assuming that each medium operation O_{ij} is processed in time $p_{ij} = \bar{p}_{ij}$ and cost $c_{ij} = \bar{c}_{ij}$ (see Section 2.1). Note that we have, thus, determined the processing time and cost for each operation O_{ij} of the jobs in $\mathcal{V} \cup \mathcal{F}$. Let $P_j = \sum_{i=1}^{\mu} p_{ij}$. Then

$$\sum_{J_j \in \mathcal{V} \cup \mathcal{F}} P_j = \sum_{J_j \in \mathcal{M} \cap (\mathcal{V} \cup \mathcal{F})} P_j + \sum_{J_j \in \mathcal{S} \cap (\mathcal{V} \cup \mathcal{F})} P_j.$$

By Lemma 2 and inequality (2),

$$\sum_{J_j \in \mathcal{S} \cap (\mathcal{V} \cup \mathcal{F})} P_j \leq \frac{qk\mu\varepsilon}{qk\mu m} = \frac{\varepsilon}{m}.$$

By the arguments in Section 2.1, $\bar{p}_{ij} \leq \max\{ p^{\delta^*_{ij}}_{ij}(1+\varepsilon), \varepsilon/(m|\mathcal{M}|\mu) \}$ and, therefore,

$$\begin{aligned} \sum_{J_j \in \mathcal{M} \cap (\mathcal{V} \cup \mathcal{F})} P_j &\leq \sum_{J_j \in \mathcal{M} \cap (\mathcal{V} \cup \mathcal{F})} P^*_j(1+\varepsilon) + \frac{\varepsilon}{m} \\ &\leq \frac{\varepsilon}{m}(2+\varepsilon), \end{aligned}$$

by Lemma 6. Therefore, we can schedule the jobs from $\mathcal{V} \cup \mathcal{F}$ one after the other at the end of the schedule without increasing too much the length of the schedule.

Theorem 7 *For any fixed m and μ, there exists a polynomial-time approximation scheme for problem P1.*

3 Preemptive Problem P1 with Continuous Processing Times

In this section we consider problem P1 when preemptions are allowed. Recall that in the preemptive problem any operation may be interrupted and resumed later without penalty. The preemptive version of problem P1 is NP-complete in the strong sense, since the special case of preemptive flow shop with three machines and fixed processing times is strongly NP-complete [8]. We show that our approximation scheme for the non-preemptive version of problem P1 can be extended to the preemptive P1 problem. The approach is similar to the non-preemptive case, but we have to handle carefully the set of large jobs $\mathcal{L}$ to ensure that we find a feasible solution of value "close" to the optimal.

3.1 Selection of Processing Times and Costs for Large Jobs

As in the non-preemptive case we divide the set of jobs $\mathcal{J}$ into *large*, *medium* and *small* jobs, denoted as $\mathcal{L}$, $\mathcal{M}$ and $\mathcal{S}$, respectively. Sets $\mathcal{L}$ and $\mathcal{M}$ have a

constant number of jobs. Again, let k denote the number of large jobs. We begin by transforming the given instance into a more structured one in which every large job can only have a constant number of distinct processing times. We prove that by using this restricted set of choices it is still possible to get a solution with cost not larger than 1 and makespan within a factor $1 + O(\varepsilon)$ of the optimal value. The restricted selection of processing times is done as follows.

1. Let V be the following set of values

$$\left\{ \frac{\varepsilon}{mk\mu}, \frac{\varepsilon}{mk\mu}(1+\varepsilon), \frac{\varepsilon}{mk\mu}(1+\varepsilon)^2, ..., \frac{\varepsilon}{mk\mu}(1+\varepsilon)^{b-2}, 1 \right\},$$

   where b is the smallest integer such that $\frac{\varepsilon}{mk\mu}(1+\varepsilon)^{b-1} \geq 1$.

2. For each operation O_{ij} of a large job J_j consider as its possible processing times the set V_{ij} of values from V that fall in the interval $[\ell_{ij}, u_{ij}]$.

This selection of restricted processing times is motivated by the following lemma.

Lemma 8 *By using the restricted set V of processing times, the following holds:*

- *There are $O(\frac{1}{\varepsilon}\log\frac{mk\mu}{\varepsilon})$ different processing times for each operation of a large job.*
- *By using the restricted set of processing times, there is a solution with cost at most 1 and makespan within a factor $1+2\varepsilon$ of the optimum.*

Proof. Let b the cardinality of set V, then $\frac{\varepsilon}{mk\mu}(1+\varepsilon)^{b-2} < 1$ and $\frac{\varepsilon}{mk\mu}(1+\varepsilon)^{b-1} \geq 1$. Therefore

$$\begin{aligned} b-2 &< \frac{\log_2 \frac{mk\mu}{\varepsilon}}{\log_2(1+\varepsilon)} \\ &\leq \frac{1}{\varepsilon}\log_2\frac{mk\mu}{\varepsilon}, \end{aligned}$$

for every $\varepsilon \leq 1$. By using similar ideas as in the proof of Lemma 3 we can prove that our transformation may potentially increase the makespan value by a factor of $1+2\varepsilon$, while the corresponding cost does not increase. Note that the proof of Lemma 3 does not rely on the fact that the jobs cannot be preempted. ■

The above lemma allows us to work with instances with a constant number of distinct processing times and costs for the large jobs. Since there are $k\mu$ large operations and each can take $O(\frac{1}{\varepsilon}\log\frac{mk\mu}{\varepsilon})$ distinct processing times and costs, the number of possible choices is bounded by $(k\mu)^{O(\frac{1}{\varepsilon}\log\frac{km\mu}{\varepsilon})}$. Note that for each distinct processing time from the restricted set, there is an associated cost that

can be computed by using the linear relationship between costs and processing times. Our algorithm considers all possible selections (from the restricted set) of processing times and costs for large jobs. By Lemma 8 at least one of them corresponds to a near optimal solution, i.e. a solution with cost at most 1 and makespan within $1 + 2\varepsilon$ times the optimal makespan. In the following we assume, without loss of generality, that the processing times and costs of large operations are chosen according to this near optimal solution. We denote these processing times and costs of large operations O_{ij} by p_{ij} and c_{ij}, respectively.

3.2 Computing a Partial Schedule

Consider any given preemptive schedule for the jobs $\mathcal{J}$. Look at the time at which any operation from a large job starts or ends. These times define a set of time intervals. Again, we call these time intervals *snapshots*. Observe that the number of snapshots g is still bounded by $g \leq 2k\mu + 1$. Since $\mathcal{L}$ has a constant number of jobs (and, hence, there is a constant number of snapshots), we can consider all relative orderings of the large jobs in the snapshots. For each relative schedule $R = (M(1), \ldots, M(g))$ of $\mathcal{L}$, we formulate a linear program $LP'(R)$ as described below.

Let α_{ij} be the snapshot where the large operation O_{ij} starts processing in the relative schedule R and let β_{ij} be the snapshot where it finishes processing. An operation O_{ij} of a large job is scheduled in consecutive snapshots $\alpha_{ij}, \alpha_{ij} + 1, \ldots, \beta_{ij}$, *but* only a fraction (possible equal to zero) of the operation might be scheduled in any one of these snapshots. However, in every snapshot there is at most one operation from any given large job. For each job $J_j \in \mathcal{L}$ we use a set of decision variables $x_{j,(i_1,\ldots,i_\mu)} \in [0,1]$ for those tuples $(i_1, \ldots, i_\mu)$ corresponding to snapshots where the operations of J_j might be scheduled, as described above. More precisely, for each job $J_j \in \mathcal{L}$ we consider tuples $(i_1, \ldots, i_\mu) \in A_j$, where

$$A_j = \{(i_1, \ldots, i_\mu) | \alpha_{wj} \leq i_w \leq \beta_{wj} \text{ and } 1 \leq w \leq \mu\}.$$

The meaning of these variables is that $x_{j,(i_1,\ldots,i_\mu)} = 1$ if and only if each operation O_{wj} of job J_j is scheduled in snapshot i_w for each $w = 1, ..., \mu$. Note that any tuple $(i_1, \ldots, i_\mu) \in A_j$ represents a valid ordering for the operations of job J_j. These variables $x_{j,(i_1,\ldots,i_\mu)}$ indicate which fraction of each operation is scheduled in every snapshot. Let the total load $L^{total}_{\ell,h}$ on machine h in snapshot $M(\ell)$ be defined as the total processing time of operations that are executed by machine h during snapshot ℓ. This value $L^{total}_{\ell,h}$ is defined as the sum of the load $L_{\ell,h}$ due to medium and small operations (see Equation (1) in Section 2.5) plus the contribution $L^{large}_{\ell,h}$ of large operations, i.e.,

$$L^{large}_{\ell,h} = \sum_{J_j \in \mathcal{L}} \sum_{(i_1,\ldots,i_\mu) \in A_j} \sum_{k=1,\ldots,\mu, i_k = \ell, m_{kj} = h} x_{j,(i_1,\ldots,i_\mu)} p_{kj}.$$

The total cost is defined as in the non-preemptive case, with the difference that the contribution of the large jobs is now equal to $\sum_{J_j \in \mathcal{L}} \sum_{i=1}^{\mu} c_{ij}$. The new

linear program $LP'(R)$ is the following.

$$
\begin{array}{lll}
\min & T = \sum_{\ell=1}^{g} t_\ell & \\
\text{s.t.} & C \leq 1, & \text{(c1')} \\
& \sum_{(i_1,\ldots,i_\mu)\in A_j} x_{j,(i_1,\ldots,i_\mu)} = 1 \quad J_j \in \mathcal{L} & \text{(c2')} \\
& \sum_{(s,\delta)\in\Lambda_j\times\Delta} x_{j,(s,\delta)} = 1, \quad J_j \in \mathcal{S}\cup\mathcal{M}, & \text{(c3')} \\
& L_{\ell,h}^{total} \leq t_\ell, \quad 1 \leq \ell \leq g,\ 1 \leq h \leq m, & \text{(c4')} \\
& x_{j,(i_1,\ldots,i_\mu)} \geq 0, \quad J_j \in \mathcal{L}, \text{ and } (i_1,\ldots,i_\mu) \in A_j, & \text{(c5')} \\
& x_{j,(s,\delta)} \geq 0, \quad J_j \in \mathcal{S}\cup\mathcal{M},\ (s,\delta)\in\Lambda_j\times\Delta, & \text{(c6')} \\
& t_\ell \geq 0, \quad \ell = 1,\ldots,g. & \text{(c7')}
\end{array}
$$

where Λ_j and Δ are as in the non-preemptive case. Linear program $LP'(R)$ has at most $1 + n + mg$ constraints. By conditions (c2') and (c3') every job J_j must have at least one of its variables ($x_{j,(s,\delta)}$ for medium and small and $x_{j,(i_1,\ldots,i_\mu)}$ for large jobs) set to a positive value. Furthermore, there is at least one snapshot ℓ for which $t_\ell > 0$. Since in any basic feasible solution of $LP'(R)$ the number of variables that receive positive values is at most the number of rows of the constraint matrix, then in a basic feasible solution there are at most mg variables with fractional values. Note that in any solution of this linear program the schedule for the large jobs is always feasible, since there is at most one operation of a given job in any snapshot. However, this schedule might not be feasible because of possible ordering conflicts among the small and medium operations assigned to a snapshot.

3.3 Computing a Feasible Solution

We find a feasible schedule for every snapshot as follows. Let us consider a snapshot $M(\ell)$.

1. Remove from the snapshot the operations belonging to large jobs. These operations will be reintroduced to the schedule later.
2. Use Sevastianov's algorithm to find a feasible schedule for the (fractions of) small jobs in the snapshot.
3. Put back the operations from the large jobs, scheduling them in the empty gaps left by the medium and small jobs. Note that it might be necessary to split an operation of a large job in order to make it fit in the empty gaps. At the end we have a feasible schedule because there is at most one operation of each large job in the snapshot.

Finally we observe that the computed solution has at most $mg + n\mu + mg$ preemptions. The first mg preemptions come from the basic feasible solution of the linear program and the remaining $n\mu + mg$ preemptions are created by introducing the operations of the large jobs in the gaps left by the medium and small jobs. Hence, our solution has $O(n)$ preemptions. Choosing the size of $\mathcal{L}$ as we did for the non-preemptive case we ensure that the length of the schedule is at most $1 + O(\varepsilon)$ times the length of an optimum schedule.

Theorem 9 *For any fixed m, μ, and $\varepsilon > 0$, there exists a polynomial-time approximation scheme for the preemptive version of problem P1, that produces a solution with at most $O(n)$ preemptions.*

4 Problem P2 with Continuous Processing Times

Problem P2 requires the computation of a solution with minimum cost and makespan at most τ. Note that for some values of τ problem P2 might not have a solution, and furthermore, deciding whether there is a solution with makespan at most τ is NP-complete. Therefore, the best that we can expect, unless P=NP, is to find a solution with cost at most the optimal cost and makespan not greater than $\tau(1+\varepsilon)$. Given a value $\tau \geq 0$, let (σ^*, δ^*) be an optimum solution for problem P2, if such a solution exists. For any value $\varepsilon > 0$, we present an algorithm that either finds a schedule for $\mathcal{J}$ of length at most $(1+3\varepsilon)\tau$ and cost at most $C(\delta^*)$, or it decides that a schedule of length at most τ does not exist.

We embed the PTAS for problem P1 described in the previous section, within a binary search procedure as follows. Let $C_\ell = \sum_{j=1}^{n} \sum_{i=1}^{\mu} c_{ij}^{\ell}$ and $C_u = \sum_{j=1}^{n} \sum_{i=1}^{\mu} c_{ij}^{u}$. Clearly, the value of the optimum solution for problem P2 lies in the interval $[C_u, C_\ell]$. Let $\rho = \min\{c_{ij}^{\ell} - c_{ij}^{u} \mid i = 1, \ldots, n, j = 1, \ldots, \mu\}$. Divide the interval $[C_u, C_\ell]$ into sub-intervals of size $\rho\varepsilon$. The number of sub-intervals is $N = \lceil (C_\ell - C_u)/(\rho\varepsilon) \rceil$. We use $UB := C_\ell$ and $LB := C_u$ as initial upper and lower bounds for the binary search. In each iteration the algorithm performs the following steps:

a) Use the PTAS for problem P1 with cost bound $\kappa = LB + \lceil N/2 \rceil \rho\varepsilon$ to find a schedule of length T_κ at most $1+\varepsilon$ times the optimum length of a schedule with this cost κ;

b) If $T_\kappa \leq (1+\varepsilon)\tau$ then update the upper bound UB to κ, otherwise update the lower bound LB to κ.

1. Set $N = \lceil (UB - LB)/(\rho\varepsilon) \rceil$.

The algorithm terminates when $LB = UB$, and outputs $LB - \rho\varepsilon$ if $T_{LB} \leq (1+\varepsilon)\tau$, otherwise the algorithm reports that there is no schedule of length at most τ. We note that if at the end the algorithm does not find a schedule of length at most $(1+\varepsilon)\tau$ it is because even with the smallest processing times for all the jobs no schedule of length at most τ exists. On the other hand, every time that the lower bound is updated, the algorithm finds a schedule of length at most $(1+\varepsilon)\tau$. At the end, the algorithm finds a value LB for which the PTAS for problem P1 finds a schedule of length $T_{LB} \leq (1+\varepsilon)\tau$. The optimal cost $C(\delta^*)$ could be smaller than LB, but it is larger than $LB - \rho\varepsilon$. Hence, this latter value is the one that the algorithm outputs. Observe that for cost $\kappa = LB - \rho\varepsilon$, the length of an optimum schedule is at most $(1+\varepsilon)\tau$ and, thus,

our PTAS finds a solution of length at most $(1+\varepsilon)^2\tau \leq (1+3\varepsilon)\tau$, for $\varepsilon < 1$. The number of iterations that we need to perform in the binary search is at most $\log_2((C_\ell - C_u)/(\rho\varepsilon))$, which is polynomial in the binary encoding of the input size.

Theorem 10 *There is a PTAS for problem P2 which finds a solution with minimum cost and makespan at most* $(1+3\varepsilon)\tau$*, for any value* $\varepsilon > 0$*, if a solution with makespan at most* τ *exists.*

The preemptive version of problem P2 can be solved by using the same algorithm that we described above.

Theorem 11 *There is a PTAS for the preemptive version of problem P2 which computes a solution with minimum cost and makespan at most* $(1+3\varepsilon)\tau$*, for any value* $\varepsilon > 0$*, if a solution with makespan at most* τ *exists.*

4.1 A Fast Approximation Algorithm for Problem P2

In this section we show how to compute a solution for problem P2 with minimum cost and makespan at most $m\tau$. This algorithm has worst performance ratio than the one that we have just described, but its running time is only $O(n)$, and the constant factor hidden in the order notation is fairly small. In Section 2, we described a linear program which computes job processing times and costs such that the sum of processing times is minimized and the total cost is at most κ. Similarly, we can formulate a linear program which determines job processing times by defining the vector (δ_{ij}) which minimizes the total cost while keeping the sum of processing times to at most τm. By scheduling the jobs one after another, in any given order, and with processing times according to (δ_{ij}), we get a solution with minimum cost and makespan at most $m\tau$. The linear program is the following.

$$\begin{array}{lll} \min & \sum_{j=1}^{n}\sum_{i=1}^{\mu}\delta_{ij}(c_{ij}^{\ell}-c_{ij}^{u})+\sum_{j=1}^{n}\sum_{i=1}^{\mu}c_{ij}^{u} & \\ \text{s.t.} & \sum_{j=1}^{n}\sum_{i=1}^{\mu}\delta_{ij}(u_{ij}-\ell_{ij}) \leq \sum_{j=1}^{n}\sum_{i=1}^{\mu}u_{ij}-\tau m. & \\ & 0 \leq \delta_{ij} \leq 1 & j=1,...,n \text{ and} \\ & & i=1,...,\mu. \end{array}$$

This linear program is a relaxation of the classical knapsack problem in minimization form. The minimization form of the problem can easily be transformed into an equivalent maximization form and solved as described in Section 2.1. Therefore, for problem P2 we can find a solution with minimum cost and makespan at most $m\tau$ in $O(n)$ time.

5 Problem P3 with Continuous Processing Times

Problem P3 is to compute a schedule that minimizes $T + \alpha C$, where T is the makespan, C is the total cost of the schedule, and $\alpha > 0$ is a given parameter.

Using modified cost values $c_{ij}^{\delta_{ij}} := \alpha c_{ij}^{\delta_{ij}}$, we can restrict the problem to the case $\alpha = 1$, without loss of generality. Our PTAS for problem P1 can be modified to work also for P3. It is enough to give the following observations. For each operation O_{ij}, let

$$d_{ij} = \min_{0 \le \delta_{ij} \le 1} \{p_{ij}^{\delta_{ij}} + c_{ij}^{\delta_{ij}}\} = \min\{\ell_{ij} + c_{ij}^{\ell}, u_{ij} + c_{ij}^{u}\}.$$

For every job J_j let $d_j = \sum_{i=1}^{\mu} d_{ij}$. We partition the set of jobs into *large* $\mathcal{L}$ and *small* $\mathcal{S}$ jobs. So, this time there is no set of medium jobs, and sets $\mathcal{L}$ and $\mathcal{S}$ can be computed in linear time. The set $\mathcal{L}$ of large jobs includes the k jobs with largest value d_j, where k is a constant chosen as in Lemma 6 to ensure that $\sum_{J_i \in \mathcal{L}} d_i \le \varepsilon/m$, and it is computed similarly as described for problem P1. Let $T^* + C^*$ be the optimum objective function value. It is easy to see that $T^* + C^* \le D$, where $D = \sum_j d_j$. Furthermore, $T^* + C^* \ge D/m$ since

$$\begin{aligned} T^* + C^* &\ge \frac{1}{m} \sum_{ij} \left[\delta_{ij}^* \ell_{ij} + (1 - \delta_{ij}^*) u_{ij}\right] + \sum_{ij} \left[\delta_{ij}^* c_{ij}^{\ell} + (1 - \delta^*) c_{ij}^{u}\right] \\ &\ge \frac{1}{m} \sum_{ij} \left[\delta_{ij}^* (\ell_{ij} + c_{ij}^{\ell}) + (1 - \delta_{ij}^*)(u_{ij} + c_{ij}^{u})\right] \\ &\ge \frac{1}{m} \sum_{ij} \left[\delta_{ij}^* d_{ij} + (1 - \delta_{ij}^*) d_{ij}\right] = \frac{D}{m}. \end{aligned}$$

By dividing all execution times and costs by D, we may assume that $D = 1$ and

$$\frac{1}{m} \le T^* + C^* \le 1.$$

The linear program $LP(R)$ has to be modified as follows. The objective function is changed to $\min \sum_{\ell=1}^{g} t_\ell + C$ and we eliminate constraint (c1). Again, the number of fractional values can be bounded by a constant and the algorithm is as before. We observe that the most time consuming part of this approach is solving the linear program. However, since we want to get an approximate solution, it is not necessary to find an optimum solution for the modified linear program, an approximate solution would be enough. To speed up our algorithm to run in linear time we can use the ideas described in Section 6.

6 Problem P3 with Discrete Processing Times

For the case of discretely controllable processing times, the possible processing times and costs of an operation O_{ij} are specified by a discrete set Δ_{ij} of values

$$\Delta_{ij} = \left\{\delta_1, \delta_2, \ldots, \delta_{w(i,j)}\right\},$$

where $0 \le \delta_\ell \le 1$ for all $\ell = 1, 2, \ldots, w(i,j)$. When the processing time of operation O_{ij} is $p_{ij}^{\delta_k} = \delta_k \ell_{ij} + (1 - \delta_k) u_{ij}$, the cost is equal to $c_{ij}^{\delta_k} = \delta_k c_{ij}^{\ell} +$

$(1-\delta_k)c_{ij}^u$. For each operation O_{ij}, let $d_{ij} = \min_{\delta_{ij} \in \Delta_{ij}} \{p_{ij}^{\delta_{ij}} + c_{ij}^{\delta_{ij}}\}$. For every job J_j let $d_j = \sum_{i=1}^{\mu} d_{ij}$, and let $D = \sum_j d_j$. We partition the set of jobs into large $\mathcal{L}$ and small $\mathcal{S}$ jobs, where the set $\mathcal{L}$ includes the k jobs with the largest d_j values, and k is a constant computed as in Lemma 6 so that the set T containing the qk jobs with the next largest d_j values has $\sum_{J_j \in T} d_j \leq \varepsilon/m$. The set of large jobs can be computed in $O(n\mu|\Delta_{\max}|)$ time, where $|\Delta_{\max}| = \max_{ij} |\Delta_{ij}|$. By multiplying all $c_{ij}^{\delta_{ij}}$ values by the parameter α, we can assume without loss of generality that the objective function for problem P3 is: $\min T(\sigma, \delta) + C(\delta)$. Let $p_{ij}^{\delta_{ij}^*}$ and $c_{ij}^{\delta_{ij}^*}$ be the processing time and cost of operation O_{ij} in an optimal solution. Let F^* be the value of an optimal solution for P3. It is easy to see that $F^* \leq D$ and

$$F^* \geq \frac{1}{m} \sum_{ij} p_{ij}^{\delta_{ij}^*} + \sum_{ij} c_{ij}^{\delta_{ij}^*} \geq \frac{D}{m}.$$

By dividing all execution times and costs by D, we may assume that $D = 1$ and

$$\frac{1}{m} \leq F^* \leq 1. \tag{3}$$

The following lemma shows that with $1 + 2\varepsilon$ loss we can reduce to $O(\log n)$ the number of different costs and processing times for each operation.

Lemma 12 *With $1 + 2\varepsilon$ loss, we assume that $|\Delta_{ij}| = O(\log n)$ for every operation O_{ij}.*

Proof. To prove this claim, divide the interval $[0, 1]$ into b subintervals as follows,

$$I_1 = [0, \frac{\varepsilon}{\mu nm}], I_2 = (\frac{\varepsilon}{\mu nm}, \frac{\varepsilon}{\mu nm}(1+\varepsilon)], ..., I_b = (\frac{\varepsilon}{\mu nm}(1+\varepsilon)^{b-1}, 1],$$

where b is the largest integer such that $\frac{\varepsilon}{\mu nm}(1+\varepsilon)^{b-1} < 1$. Clearly $b = O(\log n)$. We say that d is a *choice* for operation O_{ij} if $d \in \Delta_{ij}$. For each operation O_{ij}, partition the set of choices Δ_{ij} into b groups $g_1, g_2, ..., g_b$, such that $d \in \Delta_{ij}$ belongs to group g_h iff c_{ij}^d falls in interval I_h, $h \in \{1, ..., b\}$. For each group take the choice (if any) with the lowest processing time and delete the others. The new set of choices has at most $O(\min\{|\Delta_{ij}|, \log n\})$ elements and by using arguments similar to those used in the proof of Lemma 3 we can prove that with this transformation the cost of an optimum solution can be at most $1+2\varepsilon$ times the optimum value for the original problem. The transformed instance can be computed in $O(n\mu|\Delta_{\max}|)$ time. ∎

By using arguments similar to those in Lemma 12 we can obtain, with $1+2\varepsilon$ loss, a new instance with $O(\log k)$ different costs and processing times for each large operation. Since there is a constant number of large operations, there is only a constant number of possible assignments of costs and processing times for them. By trying all possible assignments of cost and processing times, we can find for each large operation O_{ij} a processing time $\bar{p}_{ij}$ and cost $\bar{c}_{ij}$ such that

$\bar{p}_{ij} \leq \max\{\, p_{ij}^{\delta^*_{ij}}(1+\varepsilon), \varepsilon/(mk\mu)\,\}$ and $\bar{c}_{ij} \leq c_{ij}^{\delta^*_{ij}}$. Let us use the same definition of relative schedule given for the continuous case. Let R denote a relative schedule that respects the ordering of the large operations in some optimal schedule.

For each small job J_j we define a set of at most $O((g \log n)^\mu)$ variables $x_{j,(s,\delta)}$, where $s \in \Lambda_j$ and $\delta \in \Delta_j = \{(\delta_{1j}, \delta_{2j}, \ldots, \delta_{\mu j}) \mid \delta_{ij} \in \Delta_{ij} \text{ for all } i = 1, \ldots, \mu\}$. As in the continuous case for problem P3, we define a linear program $LP''(R)$ to compute the processing times and snapshots for the small jobs. $LP''(R)$ is obtained from the linear program $LP(R)$ of Section 2 by deleting constraints (c1), (c3), and (c6), and making the following additional changes. Variable $x_{j,(s,\delta)}$ takes value $0 \leq f \leq 1$ to indicate that a fraction f of operation O_{ij}, $i = 1, \ldots, \mu$ is scheduled in snapshot s_i with processing time $p_{ij}^{\delta_i}$. Let C be the cost function, i.e.,

$$C = \sum_{J_j \in \mathcal{S}} \sum_{(s,\delta) \in \Lambda_j \times \Delta} \sum_{i=1}^{\mu} x_{j,(s,\delta)} c_{ij}^{\delta_k} + \sum_{J_j \in \mathcal{L}} \sum_{i=1}^{\mu} \bar{c}_{ij}.$$

The objective function is now to minimize $\sum_{\ell=1}^{g} t_\ell \; + C$. Constraint (c2) is replaced with

$$\sum_{\ell=\alpha_{ij}}^{\beta_{ij}} t_\ell = \bar{p}_{ij}, \quad \text{for all } J_j \in \mathcal{L}, i = 1, \ldots, \mu.$$

As in Lemma 5, we can prove that an optimum solution of problem P3 is a feasible solution for $LP''(R)$. The rest of the algorithm is as that described in Section 2.6. By using interior point methods to solve the linear program, we get a total running time for the above algorithm that is polynomial in the input size [1]. It is easy to check that similar results can be obtained if, instead of finding the optimum solution for the linear program, we solve it with a given accuracy $\varepsilon > 0$. In the next section we show that we can solve approximately the linear program in $O(n|\Delta_{\max}|)$ time. Therefore, for every fixed m, μ and ϵ, all computations (including Sevastianov's algorithm [18]) can be carried out in time $O(n|\Delta_{\max}| + n \min\{\log n, |\Delta_{\max}|\} \cdot f(\varepsilon, \mu, m))$, where $f(\varepsilon, \mu, m)$ is a function that depends on ε, μ and m. This running time is linear in the size of the input.

Theorem 13 *For any fixed m and μ, there exists a linear time approximation scheme for P3 with discretely controllable processing times.*

6.1 Approximate Solution of the Linear Program

In this section we show how to find efficiently a solution for $LP''(R)$ of value no more than $1 + O(\epsilon)$ times the value of the optimum solution for problem P3. To find an approximate solution for the linear program we first rewrite it as a convex block-angular resource-sharing problem, and then use the algorithm of [9] to solve it with a given accuracy. A *convex block-angular resource sharing problem* has the form:

$$\lambda^* = \min\left\{ \lambda \left| \sum_{j=1}^{K} f_i^j(x^j) \le \lambda, \text{for all } i = 1, \ldots, N, \text{and } x^j \in \mathcal{B}^j, j = 1, \ldots, K \right. \right\},$$

where $f_i^j : \mathcal{B}^j \to \mathbb{R}^+$ are N nonnegative continuous convex functions, and $\mathcal{B}^j$ are disjoint convex compact nonempty sets called *blocks*. The algorithm in [9] finds a $(1+\rho)$-approximate solution for this problem for any $\rho > 0$ in $O(N(\rho^{-2}\ln\rho^{-1} + \ln N)(N\ln\ln(N/\rho) + KF))$ time, where F is the time needed to find a ρ-approximate solution to the problem:

$$\min\left\{ \sum_{i=1}^{N} p_i f_i^j(x^j) \,\middle|\, x^j \in \mathcal{B}^j \right\},$$

for some vector $(p_1, \ldots, p_N) \in \mathbb{R}^N$. We can write $LP''(R)$ as a convex block-angular resource sharing problem as follows. First we compute an estimate V for the value of an optimum solution of problem P3, and add the constraint

$$\sum_{\ell=1}^{g} t_\ell + C + 1 - V \le \lambda,$$

to the linear program, where λ is a nonnegative value. Since $1/m \le V \le 1$, we can use binary search on the interval $[1/m, 1]$ to guess V with a given accuracy $\varepsilon > 0$. This search can be completed in $O(\log(\frac{1}{\varepsilon}\log m))$ iterations by doing the binary search over the values:

$$\frac{1}{m}(1+\varepsilon), \frac{1}{m}(1+\varepsilon)^2, ..., \frac{1}{m}(1+\varepsilon)^b, 1,$$

where b is the largest integer for which $\frac{1}{m}(1+\varepsilon)^b < 1$. We replace constraint (c5) of $LP''(R)$ by

(5') $\quad L_{\ell,h} + 1 - t_\ell \le \lambda, \quad$ for all $(\ell, h) \in Free(R)$

where $Free(R)$ is as defined in Section 2.5 and

$$L_{\ell,h} = \sum_{J_j \in \mathcal{S}} \sum_{(s,\delta) \in \Sigma_j \times \Delta} \sum_{\substack{q=1 \\ s_q = \ell, m_{qj} = h}}^{\mu} x_{j,(s,\delta)} p_{qj}^{\delta_q}.$$

This new linear program, that we denote as $LP''(R, V, \lambda)$, has the above block-angular structure. To see this, let us define the blocks $\mathcal{B}^j$ and convex functions f_i^j. The blocks $\mathcal{B}^j$ are the sets $\{x_{j,(s,\delta)} \mid (s,\delta) \in \Sigma_j \times \Delta$, and constraints (c4) and (c8) hold$\}$, for every small job J_j. Note that these blocks are $(g|\Delta_{\max}|)^\mu$-dimensional *simplicies*. The block $\mathcal{B}^0 = \{< t_1, t_2, .., t_g > |$constraints (c2) and

(c9) hold} has constant dimension. Let $f_{\ell,h} = L_{\ell,h} + 1 - t_\ell$. Since $t_\ell \leq V \leq 1$, these functions are nonnegative. For every small job J_j, let

$$f_0^j(x^j) = \sum_{(s,\delta)\in\Lambda_j\times\Delta} \sum_{i=1}^{\mu} x_{j,(s,\delta)} c_{ij}^{\delta_i}.$$

For every $(\ell, h) \in Free(R)$, let

$$f_{\ell h}^j(x^j) = \sum_{(s,\delta)\in\Lambda_j\times\Delta} \sum_{\substack{q=1 \\ s_q=\ell, m_{qj}=h}}^{\mu} x_{j,(s,\delta)} p_{qj}^{\delta_q}.$$

For every $x^0 \in \mathcal{B}^0$ let

$$f_0^0(x^0) = \sum_{J_j\in\mathcal{L}} \sum_{i=1}^{\mu} c_{ij} + \sum_{\ell=1}^{g} t_\ell \ + 1 - V,$$

and for every $(\ell, h) \in Free(R)$, let

$$f_{\ell h}^0(x^0) = 1 - t_\ell.$$

All these functions are convex and nonnegative. Now, we can define $LP''(R, V, \lambda)$: minimize the value λ such that

$$\begin{array}{ll} \sum_{J_j\in\mathcal{S}} & f_0^j(x^j) + f_0^0(x^0) \leq \lambda, \qquad \text{for all } x^k \in \mathcal{B}^k, \\ \sum_{J_j\in\mathcal{S}} & f_{\ell h}^j(x^j) + f_{\ell h}^0(x^0) \leq \lambda, \quad \text{for all } (\ell, h) \in Free(R) \text{ and } x^k \in \mathcal{B}^k. \end{array}$$

Using the algorithm in [9], a $1+\rho$, $\rho > 0$ approximation for this problem can be obtained by solving on each block $\mathcal{B}^j$ a constant number of block optimization problems of the form:

$$\min\{p^T f^j(x) \mid x \in \mathcal{B}^j\},$$

where p is a $(g|\Delta_{\max}|+1)$-dimensional positive price vector, and f^j is a $(g|\Delta_{\max}|+1)$-dimensional vector whose components are the functions $f_0^j, f_{\ell h}^j$. Note that $\mathcal{B}^0$ has constant dimension, and thus, the corresponding block optimization problem can be solved in constant time. But, the blocks $\mathcal{B}^j$ for $J_j \in \mathcal{S}$ do not have a constant dimension. To solve the block optimization problem on these blocks we must find a snapshot where to place each operation of a small job J_j and determine its processing time, so that the total cost plus processing time of all operations times the price vector is minimized. To choose the snapshots, we select for each operation the snapshot in which the corresponding component of the price vector is minimum. Then, we select for each O_{ij} the value δ_{ij} that minimizes its cost plus processing time. This can be done in $O(|\Delta_{\max}|)$ time for each block, so the algorithm of [9] finds a feasible solution for $LP''(R, V, 1+\rho)$ in $O(nw)$ time. Linear program $LP''(R, V, 1+\rho)$ increases the length of each snapshot by ρ, and, therefore, the total length of the solution is $V + g\rho \leq (1+2\varepsilon)V^*$, for $\rho = \frac{\epsilon}{mg}$, where V^* is the optimal solution value.

6.1.1 Bounding the Number of Fractional Assignments

There is a problem with this method: we cannot guarantee that the solution found by the algorithm is basic feasible. Hence, it might have a large number of fractional assignments. In the following we show that the number of fractional assignments is $O(n)$. Since the number of fractional assignments is $O(n)$, using the rounding technique described in [11], we can obtain in linear time a new feasible solution with only a constant number of fractional assignments. The algorithm in [9] works by choosing a starting solution $x_0 \in \mathcal{B}^j$ and then it repeats the following three steps for at most $O(mg \log(mg))$ times:

Step 1 (Compute prices). Use a deterministic or randomized procedure to compute a price vector p.

Step 2 (Block optimization). Use a block solver to compute an optimal solution of each block problem.

Step 3 (New iterate). Replace the current approximate solution by a convex combination of the previous solutions kept on record.

By starting from a solution x_0 in which every vector $x_0^j \in \mathcal{B}^j$, $j \neq 0$, is integer, we get at the end at most $O(n \cdot mg \log(mg))$ fractional assignments. To achieve the promised running time we additionally need that $\lambda(x_0) \leq c\lambda^*$ [9], where c is a constant and $\lambda(x_0)$ is the value of λ corresponding to x_0. This is accomplished as follows. For convenience, let us rename the jobs so that $J_1, ..., J_{\bar{n}}$ are the small jobs, where $\bar{n} = n - k$. Choose the processing time $p_{ij}^{\delta_{ij}}$ and cost $c_{ij}^{\delta_{ij}}$ for every small operation O_{ij} so that $d_{ij} = p_{ij}^{\delta_{ij}} + c_{ij}^{\delta_{ij}}$. Put the small jobs one after another in the last snapshot. Set $t_g = \sum_{J_j \in \mathcal{S}} \sum_{i=1}^{\mu} p_{ij}^{\delta_{ij}}$. The large operations are scheduled as early as possible, according to the optimal relative schedule R. Assign to each $t_\ell \in \{t_1, t_2, ..., t_{g-1}\}$ a value equal to the maximum load of snapshot ℓ according to the above schedule. By inequality (3), we know that

$$\sum_{J_j \in \mathcal{S}} \sum_{i=1}^{\mu} d_{ij} \leq 1.$$

Furthermore, we have

$$\sum_{\ell=1}^{g-1} t_\ell + \sum_{J_j \in \mathcal{L}} \sum_{i=1}^{\mu} c_{ij} \leq V,$$

since by construction $\sum_{\ell=1}^{g-1} t_\ell$ cannot be greater than the optimal length, and the costs of large operations are chosen according to the optimal solution. Hence, $\sum_{\ell=1}^{g} t_\ell + C \leq 1 + V$, and

$$\sum_{\ell=1}^{g} t_\ell + C + 1 - V \leq 2,$$

$$L_{\ell,h} + 1 - t_\ell \leq 1;$$

so $\lambda(x_0) \leq 2$. Since $\lambda^* = 1$, it follows that $\lambda(x_0) \leq 2\lambda^*$.

7 Extensions

By using similar techniques as above, we can design a PTAS also for the case of piecewise linear cost functions. In this case it is possible to change the processing time of operation O_{ij} to any value in the intervals $[\ell_{ij}(q), u_{ij}(q)]$, where $q = 1, ..., w_{ij}$. Note that it is not assumed that the intervals are adjacent, i.e., it might be the case that $u_{ij}(q) < \ell_{ij}(q+1)$, $q = 1, ..., w_{ij} - 1$. The cost for processing O_{ij} in time $\ell_{ij}(q)$ is $c^{\ell}_{ij}(q)$ and for processing it in time $u_{ij}(q)$ the cost is $c^{u}_{ij}(q)$. For any value $\delta_{ij} \in [0, 1]$ the cost for processing operation O_{ij} in interval q and in time $p^{\delta_{ij}}_{ij} = \delta_{ij}\ell_{ij}(q) + (1 - \delta_{ij})u_{ij}(q)$ is $c^{\delta_{ij}}_{ij} = \delta_{ij}c^{\ell}_{ij}(q) + (1 - \delta_{ij})c^{u}_{ij}(q)$. When m and μ are fixed, we can speed up the running time for problem P3 to $O(nw_{\max} + n \min\{\log n, w_{\max}\} \cdot f(\varepsilon, \mu, m))$, where $w_{\max} = \max_{ij} w_{ij}$.

8 Conclusions

We have studied the job shop scheduling problem with controllable processing times. This problem models the situation when the processing time of a job can be reduced by assigning more resources to it. Of course, adding computational resources to a job incurs a cost, which has to be balanced against the profit earned by completing the job sooner. We have defined several versions of the problem by considering different ways of dealing with the trade-off between cost and completion time of the jobs.

We described several polynomial time approximation schemes for the case when the number of machines and the maximum number of operations per job are fixed. Our algorithms guarantee finding near optimum solutions, but at the expense of high running times. An interesting open question in this area is to design an algorithm with low running time which can find schedules of value within a constant factor of the optimum.

Acknowledgements. We thank the referees for their valuable comments.

References

[1] K. Anstreicher. Linear programming in $o(\frac{n^3 \ln n}{l})$ operations. *SIAM Journal on Optimization*, 9:803–812, 1999.

[2] Z. Chen, Q. Lu, and G. Tang. Single machine scheduling with discretely controllable processing times. *Operation Research Letters*, 21:69–76, 1997.

[3] T. Cheng, Z.-L. Chen, C.-L. Li, and B.-T. Lin. Scheduling to minimize the total compression and late costs. *Naval Research Logistics*, 45:67–82, 1998.

[4] T. Cheng and N. Shakhlevich. Proportionate flow shop with controllable processing times. *Journal of Scheduling*, 2:253–265, 1999.

[5] A. Fishkin, K. Jansen, and M. Mastrolilli. Grouping techniques for scheduling problems: simpler and faster. In *Proceedings of the 9th Annual European Symposium on Algorithms (ESA'01)*, LNCS 2161, 206–217, 2001.

[6] M. Garey, D. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:117–129, 1976.

[7] L. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk. Better approximation guarantees for job-shop scheduling. *SIAM Journal on Discrete Mathematics*, 14(1):67–92, 2001.

[8] T. Gonzalez and S. Sahni. Flowshop and jobshop schedules: complexity and approximation. *Operations Research*, 26:36–52, 1978.

[9] M. D. Grigoriadis and L. G. Khachiyan. Coordination complexity of parallel price-directive decomposition. *Mathematics of Operations Research*, 21:321–340, 1996.

[10] A. Janiak, M.Y. Kovalyov, W. Kubiak, and F. Werner. Approximation Schemes for Scheduling with Controllable Processing Times. *Universit at Magdeburg*, Research Report, 2001.

[11] K. Jansen, R. Solis-Oba, and M. Sviridenko. A linear time approximation scheme for the job shop scheduling problem. In *Proceedings of APPROX'99*, LNCS 1671, 177–188, 1999.

[12] K. Jansen, R. Solis-Oba, and M. Sviridenko. Makespan minimization in job shops: a polynomial time approximation scheme. In *Proceedings of the 31st Annual ACM Symposium on the Theory of Computing (STOC 99)*, 394–399, 1999.

[13] E. Lawler. Fast approximation algorithms for knapsack problems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 77)*, 206–218, 1977.

[14] M. Mastrolilli. A PTAS for the single machine scheduling problem with controllable processing times. In *Algorithm Theory - SWAT 2002, 8th Scandinavian Workshop on Algorithm Theory*, LNCS 2368, 51–59, 2002.

[15] E. Nowicki. An approximation algorithm for the m-machine permutation flow shop scheduling problem with controllable processing time. *European Journal of Operational Research*, 70:342–349, 1993.

[16] E. Nowicki and S. Zdrzalka. A two-machine flow shop scheduling problem with controllable job processing times. *European Journal of Operational Research*, 34:208–220, 1988.

[17] E. Nowicki and S. Zdrzalka. A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, 26:271–287, 1990.

[18] S. Sevastianov. Bounding algorithms for the routing problem with arbitrary paths and alternative servers. *Cybernetics* (in Russian), 22:773–780, 1986.

[19] D. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23:617–632, 1994.

[20] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

[21] Yu.N. Sotskov and N.V. Shakhlevich. NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59:237–266, 1995.

[22] M. Trick. Scheduling multiple variable-speed machines. *Operations Research*, 42:234–248, 1994.

[23] R. Vickson. Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine. *Operations Research*, 28:1155–1167, 1980.

[24] R. Vickson. Two single machine sequencing problems involving controllable job processing times. *AIIE Trans.*, 12:258–262, 1980.

[25] D. Williamson, L. Hall, J. Hoogeveen, C. Hurkens, J. Lenstra, S. Sevastianov, and D. Shmoys. Short shop schedules. *Operations Research*, 45:288–294, 1997.

[26] F. Zhang, G. Tang, and Z.-L. Chen. A 3/2-approximation algorithm for parallel machine scheduling with controllable processing times. *Operations Research Letters*, 29:41–47, 2001.