Operations Research Letters 45 (2017) 604–609

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

A PTAS for a resource scheduling problem with arbitrary number of parallel machines

Péter Györgyi *

Department of Operations Research, Loránd Eötvös University, H1117 Budapest, Pázmány Péter sétány 1/C, Hungary
Institute for Computer Science and Control, H1111 Budapest, Kende str. 13–17, Hungary

ARTICLE INFO

Article history:
Received 22 June 2016
Received in revised form 19 September 2017
Accepted 21 September 2017
Available online 4 October 2017

Keywords:
Parallel machine scheduling
Non-renewable resource
Approximation scheme

ABSTRACT

In this paper we study a parallel machine scheduling problem with non-renewable resource constraints. That is, besides the jobs and machines, there is a common non-renewable resource consumed by the jobs, which has an initial stock and some additional supplies over time. Unlike in most previous results, the number of machines is part of the input. We describe a polynomial time approximation scheme for minimizing the makespan.

1. Introduction

In this paper we study a parallel machine scheduling problem and describe a polynomial time approximation scheme (PTAS) for it. In our problem, the jobs have an additional resource requirement: there is a non-renewable resource (like raw material, energy, or money) consumed by the jobs. The resource has an initial stock, which is replenished at some a-priori known moments of time. As usual, each job can be scheduled on any machine, the job processing times do not depend on the machines assigned, machines can perform only one job at a time, and preemption of jobs is not allowed. The objective is to minimize the maximal job-completion time, or, in other words, the *makespan* of the schedule.

More formally, there are m parallel machines, $\mathcal{M} = \{M_1, \dots, M_m\}$, a finite set of n jobs $\mathcal{J} = \{J_1, \dots, J_n\}$, and a common resource consumed by some, or possibly all of the jobs. Each job J_j has a processing time $p_j \in \mathbb{Z}_+$ and a resource requirement $a_j \in \mathbb{Z}_{\geq 0}$ from the common resource, noting that $a_j = 0$ is possible. The resource is supplied in q different time moments, $0 = u_1 < u_2 < \cdots < u_q$; the number $\tilde{b}_\ell \in \mathbb{Z}_+$ represents the quantity supplied at u_ℓ, $\ell = 1, 2, \dots, q$. A *schedule* σ specifies a machine and the starting time S_j for each job, and it is *feasible* if (i) on every machine the jobs do not overlap in time, and if (ii) at any time point t the total material supply from the resource is at least the total request of those jobs starting not later than t, i.e., $\sum_{(\ell\,:\,u_\ell \leq t)} \tilde{b}_\ell \geq \sum_{(j\,:\,S_j \leq t)} a_j$. The objective is to minimize the makespan, i.e., the completion time of the job finished last.

This problem is a sub-problem of a more general resource scheduling problem: in the general case there are r resources, the requirements a_j and the supplies b_ℓ are r-dimensional vectors. We denote our problem by $P|rm = 1|C_{\max}$, where $rm = 1$ indicates that there is only one single non-renewable resource. Since the makespan minimization problem with resource consuming jobs on a single machine is NP-hard even if there are only two supply dates [2], the studied problem is NP-hard.

The combination of scheduling and logistic, that is, considering e.g., raw material supplies in the course of scheduling, has a great practical potential, as this problem frequently occurs in real-world applications (e.g. [1,4]).

1.1. Main result and structure of the paper

Section 2 summarizes the previous results, while Section 3 simplifies the resource scheduling problem with some observations and gives an integer programming model of the problem. In Section 4, we prove the following:

Theorem 1. *There is a PTAS for* $P|rm = 1|C_{\max}$.

There are several approximation schemes for similar scheduling problems with non-renewable resource constraints (see Section 2), however, to our best knowledge, this is the first time, that an arbitrary number of parallel machine is considered in an approximation algorithm for scheduling with non-renewable resources. Note that the latter problem is already APX-hard in case of two

* Correspondence to: Department of Operations Research, Loránd Eötvös University, H1117 Budapest, Pázmány Péter sétány 1/C, Hungary.
E-mail address: gyorgyi.peter@sztaki.mta.hu.

https://doi.org/10.1016/j.orl.2017.09.007
0167-6377/

resources [11], so limiting the number of resources to one is necessary to have a PTAS unless P = NP. The problem $P|rm = 1|C_{\max}$ was the only problem with unknown approximability status in the class $P|rm|C_{\max}$ [11].

Our PTAS reuses ideas from known PTAS-es designed for $P \parallel C_{\max}$ (e.g. [13,12]). Actually, we invoke a variant of the latter. However, there are no resource constraints in those PTAS-es, therefore the jobs differ only in their processing times. Rounding techniques are useful in a PTAS to simplify the instances (e.g. Lemmas 1 and 2), because they introduce only small errors, but rounding the resource supplies or resource requirements does not seem a viable approach. Instead, we will sort the jobs into different categories, and use enumeration to find suboptimal schedules for the problem with rounded processing times.

1.2. Terminology

An *optimization problem* Π consists of a set of instances, where each instance has a set of *feasible solutions*, and each solution has an (objective function) value. In a *minimization problem* a feasible solution of minimum value is sought, while in a maximization problem one of maximum value. An *ε-approximation algorithm* for an optimization problem Π delivers in polynomial time for each instance of Π a solution whose objective function value is at most $(1+\varepsilon)$ times the optimum value in case of minimization problems, and at least $(1-\varepsilon)$ times the optimum in case of maximization problems. For an optimization problem Π, a family of approximation algorithms $\{A_\varepsilon\}_{\varepsilon>0}$, where each A_ε is an ε-approximation algorithm for Π is called a *Polynomial Time Approximation Scheme (PTAS) for* Π.

2. Previous work

Makespan minimization on parallel machines is one of the oldest problem of scheduling theory. The problem is strongly NP-hard [6], but there is a PTAS for it [13].

Scheduling problems with resource consuming jobs were introduced by [2,3], and [15]. In [2], the computational complexity of several variants with a single machine was established, while in [3] activity networks requiring only non-renewable resources were considered. In [15] a parallel machine problem with preemptive jobs was studied with a single non-renewable resource. This resource had an initial stock and some additional supplies, like in the model presented above, and it was assumed that the rate of consuming the non-renewable resource was constant during the execution of the jobs. These assumptions led to a polynomial time algorithm for minimizing the makespan, which is in a strong contrast to the NP-hardness of the scheduling problem analyzed in this paper. Further results can be found in e.g., [16,17,7,5,8–10,14,11].

In [8,9] and [10] there are several approximability results for the single machine variant of the problem. [11] provided PTAS-es for some parallel machine variant of the problem and showed that the problem with two resources and two supplies is APX-hard. See also [11] for further previous results of the topic.

3. Preliminaries

Note that the following assumption holds without loss of generality and it has two easy corollaries:

Assumption 1. $\sum_{\ell=1}^{q} \tilde{b}_\ell = \sum_{j\in\mathcal{J}} a_j$.

Corollary 1. *$C^*_{\max} > u_q$ and we have enough resource for each job that starts after u_q.*

Observation 1. *For a PTAS, it is sufficient to provide a schedule with a makespan of $(1+c\varepsilon)$ times the optimum value, where c is a constant i.e. it does not depend on the input. Hence, to reach a desired performance ratio δ, we let $\varepsilon := \delta/c$, and perform the computations with the choice of ε.*

The observation above shows the meaning of the next lemmas.

Lemma 1. *With $1+\varepsilon$ loss, we can assume that all processing times are integer powers of $1+\varepsilon$(trivial).*

Lemma 2 ([11])**.** *In order to have a PTAS for $P|rm|C_{\max}$, it suffices to provide a family of algorithms $\{A_\varepsilon\}_{\varepsilon>0}$ such that A_ε is an ε-approximation algorithm for the restricted problem where the supply dates before u_q are from the set $\{\ell\varepsilon u_q : \ell = 0, 1, 2, \ldots, \lfloor 1/\varepsilon \rfloor\}$.*

We can model $P|rm = 1|C_{\max}$ with a mathematical program with integer variables in a way similar to that of [11]. We define the values $b_\ell := \sum_{\nu\,:\,u_\nu \le u_\ell} \tilde{b}_\nu$, that is, b_ℓ equals the total amount supplied from the resource up to u_ℓ and let $\mathcal{T} := \{u_1, u_2, \ldots, u_q\}$. We introduce $q\cdot|\mathcal{J}|\,|\mathcal{M}|$ binary decision variables $x_{j\ell k}$, $(j\in\mathcal{J}, \ell = 1,\ldots,q, k\in\mathcal{M})$ such that $x_{j\ell k} = 1$ if and only if job j is assigned to machine k and to the time point u_ℓ, which means that the requirements of job j must be satisfied by the resource supplies up to time point u_ℓ. The mathematical program is

$$C^*_{\max} = \min \max_{k\in\mathcal{M}} \max_{u_\ell\in\mathcal{T}} \left(u_\ell + \sum_{j\in\mathcal{J}} \sum_{\nu=\ell}^{q} p_j x_{j\nu k} \right) \tag{1}$$

s.t.

$$\sum_{k\in\mathcal{M}} \sum_{j\in\mathcal{J}} \sum_{\nu=1}^{\ell} a_j x_{j\nu k} \le b_\ell, \quad u_\ell \in \mathcal{T} \tag{2}$$

$$\sum_{k\in\mathcal{M}} \sum_{\ell=1}^{q} x_{j\ell k} = 1, \quad j\in\mathcal{J} \tag{3}$$

$$x_{j\ell k} \in \{0,1\}, \quad j\in\mathcal{J},\ u_\ell\in\mathcal{T},\ k\in\mathcal{M}. \tag{4}$$

The objective function expresses the completion time of the job finished last using the observation that for every machine there is a time point from which the machine processes the jobs without idle times. Constraints (2) ensure that the jobs assigned to time points u_1 through u_ℓ use only the resources supplied up to time u_ℓ. Eqs. (3) ensure that all jobs are assigned to some machine and time point. Any feasible job assignment $\bar{x}$ gives rise to a set of schedules which differ only in the ordering of jobs assigned to the same machine k, and time point u_ℓ.

$\bar{x}$ is a *partial* assignment, if it satisfies (4) and $\sum_{k\in\mathcal{M}}\sum_{\ell=1}^{q} x_{j\ell k} \le 1$, $j\in\mathcal{J}$. If it satisfies also (2), then it is a *feasible* partial assignment.

Subroutine Sch describes how we create a (partial) schedule from a (partial) assignment.

Subroutine Sch [11]

Input: $\bar{\mathcal{J}} \subseteq \mathcal{J}$ and $\bar{x}$ such that for each $j\in\bar{\mathcal{J}}$ there exists a unique (ℓ, k) with $\bar{x}_{j\ell k} = 1$, and $\bar{x}_{j\ell k} = 0$ otherwise.

Output: partial schedule S^{part} of the jobs in $\bar{\mathcal{J}}$.

1. S^{part} is initially empty, then we schedule the jobs on each machine in increasing u_ℓ order (first we schedule those jobs assigned to u_1, and then those assigned to u_2, etc.):
2. When scheduling the next job with $\bar{x}_{j\ell k} = 1$, then it is scheduled at time $\max\{u_\ell, C_{last}(k)\}$, where $C_{last}(k)$ is the completion time of the last job scheduled on machine M_k, or 0 if no job has been scheduled yet on M_k.

Remark 1. Note that if $\bar{x}$ is feasible partial assignment, then S^{part} is a feasible partial schedule, since: