ARTICLE IN PRESS

JID: EOR [m5G;December 12, 2016;10:41]

European Journal of Operational Research 000 (2016) 1–9

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Parallel batch scheduling with inclusive processing set restrictions and non-identical capacities to minimize makespan

Shuguang Li[a,b,*]

[a] Key Laboratory of Intelligent Information Processing in Universities of Shandong (Shandong Institute of Business and Technology), Yantai 264005, China
[b] College of Computer Science and Technology, Shandong Institute of Business and Technology, Yantai 264005, China

ARTICLE INFO

Article history:
Received 1 March 2016
Accepted 24 November 2016
Available online xxx

Keywords:
Scheduling
Batching machines
Inclusive processing set restrictions
Non-identical capacities
Makespan

ABSTRACT

We consider the problem of scheduling n jobs on m parallel batching machines with inclusive processing set restrictions and non-identical capacities. The machines differ in their functionality but have the same processing speed. The inclusive processing set restriction has the following property: the machines can be linearly ordered such that a higher-indexed machine can process all those jobs that a lower-indexed machine can process. Each job is characterized by a processing time that specifies the minimum time needed to process the job, a release date before which it cannot be processed, and a machine index which is the smallest index among the machines that can process it. Each batching machine has a limited capacity and can process a batch of jobs simultaneously as long as its capacity is not violated. The capacities of the machines are non-identical. The processing time of a batch is the maximum of the processing times of the jobs belonging to it. Jobs in the same batch have a common start time and a common completion time. The goal is to find a non-preemptive schedule so as to minimize makespan (the maximum completion time). When all jobs are released at the same time, we present two fast algorithms with approximation ratios 3 and 9/4, respectively. For the general case with unequal release dates, we develop a polynomial time approximation scheme (PTAS), which is also the first PTAS even for the case with equal release dates and without processing set restrictions.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Scheduling of jobs on parallel machines is one of the classical problems in combinatorial optimization. It is well known that these problems are usually NP-hard for standard objective functions like minimizing makespan (the maximum completion time), even for two identical machines. There is a rich literature on this class of problems (see, e.g., Chen, Potts, and Woeginger (1998), Leung (2004), Brucker (2007)).

In practice, the machines running in parallel are often non-identical. They may differ in their functionality as well as their processing speeds. Such machines are called *unrelated machines*. In between identical and unrelated, there is a class of machines that differ in their functionality but have the same processing speed. In such settings, jobs have a restricted set of machines to which they may be assigned, called its *processing set*, while the processing time of a job is independent of the machines. Scheduling problems with processing set restrictions have been studied extensively under different names. These include "scheduling typed task systems" (Jaffe, 1980; Jansen, 1994), "multi-purpose machine scheduling" (Brucker, Jurisch, & Krämer, 1997; Vairaktarakis & Cai, 2003), "scheduling with eligibility constraints" (Centeno & Armacost, 2004; Hwang, Chang, & Hong, 2004a; Lee, Leung, & Pinedo, 2011; Li, 2006), "scheduling with processing set restrictions" (Epstein and Levin, 2011; Glass and Kellerer, 2007; Huo and Leung, 2010; Li and Wang (2010); Ou, Leung, & Li, 2008), and "scheduling with assignment restriction" (Bar-Noy, Freund, & Naor, 2001; Lam, Ting, To, & Wong, 2002). See the survey papers by Leung and Li (2008); 2016).

There are two special cases of processing set restrictions that have received increasing attention recently: (i) processing sets that do not partially overlap and are said to be *nested*; (ii) processing sets that are not only nested but also include one another, and are called *inclusive processing sets*. It is the latter case that this paper focuses on.

In the classic scheduling theory (Drozdowski, 2009), each machine can process at most one job at a time. In the past few decades, along another line of research there has been significant interest in scheduling problems concerning batching machines. A *batching machine* (or batch processing machine, BPM for short in

* Correspondence to: College of Computer Science and Technology, Shandong Institute of Business and Technology, Yantai 264005, China
E-mail address: sgliytu@hotmail.com

http://dx.doi.org/10.1016/j.ejor.2016.11.044
0377-2217/© 2016 Elsevier B.V. All rights reserved.

Please cite this article as: S. Li, Parallel batch scheduling with inclusive processing set restrictions and non-identical capacities to minimize makespan, European Journal of Operational Research (2016), http://dx.doi.org/10.1016/j.ejor.2016.11.044

the literature) is a machine that can process a group of jobs as a batch simultaneously. Webster and Baker (1995) distinguished three types of models for scheduling batching machines: the *serial batch* model, in which the processing time of a batch is equal to the sum of processing times of jobs belonging to it (see also Albers & Brucker (1993)); the *parallel batch* model, in which the processing time of a batch is the maximum of the processing times of jobs belonging to it (see also Brucker et al. (1998)); and the *fixed batch* model, in which the processing time of a batch is a constant, independent of the jobs it contains (see also Ahmadi, Ahmadi, Dasu, & Tang (1992)). We refer the readers to Lee, Uzsoy, and Martin-Vega (1992) for the motivation, and to Brucker et al. (1998), Potts and Kovalyov (2000), Mathirajan and Sivakumar (2006), Mönch, Fowler, Dauzère-Pérès, Mason, and Rose (2011) for surveys of recent results. We focus on the parallel batch model in this paper.

We note, with some surprise, that despite the extensive research on scheduling with either processing set restrictions or batching, there is essentially no work which takes both of them into consideration. In this paper we initiate the study of parallel batch scheduling with inclusive processing set restrictions. The objective function we consider is minimizing makespan. This is perhaps the most popular objective considered in scheduling theory.

The problem we consider can be formally described as follows. Given a set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ and a set of m batching machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. The machines differ in their functionality but have the same processing speed. They can be linearly ordered such that a higher-indexed machine can process all those jobs that a lower-indexed machine can process. Each job j is characterized by a *processing time* p_j that specifies the minimum time needed to process the job, a *release date* r_j before which it cannot be processed, and a *machine index* a_j which is the smallest index among the machines that can process it. Job j can be processed by machine M_i if and only if $i \geq a_j$. The machines in $\{M_{a_j}, M_{a_j+1}, \ldots, M_m\}$ are called *eligible machines* for job j. Machine M_i has a limited *capacity* K_i and can process a batch of jobs simultaneously as long as the total number of the jobs in the batch does not exceed K_i, $i = 1, 2, \ldots, m$. The capacities of the machines are non-identical. The processing time of a batch is the maximum of the processing times of the jobs belonging to it. Jobs in the same batch have a common start time and a common completion time. Thus, scheduling involves grouping the jobs into batches and processing the batches on the machines. The goal is to find a non-preemptive schedule so as to minimize *makespan*, $C_{\max} = \max_j C_j$, where C_j denotes the completion time of job j in the schedule. Following Brucker (2007), Graham, Lawler, Lenstra, and Kan (1979), we denote this problem as $P|r_j, a_j, p-batch, K_i|C_{\max}$.

The problem contains many fundamental scheduling problems as special cases which are strongly NP-hard, hence it is also strongly NP-hard. Therefore, we will design approximation algorithms for this problem. An approximation algorithm can be evaluated by its approximation ratio, which is defined as the worst-case ratio between the value of the solution obtained by the algorithm and the optimal solution value (for minimization problems) on any input instance of the problem. An algorithm with approximation ratio ρ is called a *ρ-approximation algorithm*. A family of algorithms $\{A_\varepsilon\}$ is called a *polynomial time approximation scheme* (PTAS) if, for any arbitrarily small positive constant ε, A_ε is a $(1+\varepsilon)$-approximation algorithm running in time that is polynomial in the input size of the problem instance. If the running time is polynomial in $1/\varepsilon$ as well, then we have a *fully polynomial time approximation scheme* (FPTAS) (Papadimitriou & Steiglitz, 1998).

Below, we will briefly survey the existing results on the related scheduling problems with the objective of minimizing makespan.

When all $K_i = 1$ and all $a_j = 1$, $P|r_j, a_j, p-batch, K_i|C_{\max}$ reduces to the well-known classical problem $P|r_j|C_{\max}$, which is NP-hard even if all $r_j = 0$ and $m = 2$ Garey and Johnson (1979). The special case of all $r_j = 0$ is denoted as $P||C_{\max}$, which is strongly NP-hard Lawler, Lenstra, Kan, and Shmoys (1993). For $P||C_{\max}$, Graham proposed algorithms called *list scheduling* (LS) and *largest processing time first* (LPT) in his seminal works (Graham, 1966; 1969). The approximation ratios of LS and LPT are $2 - 1/m$ and $4/3 - 1/(3m)$, respectively. The problem $P||C_{\max}$ also admits a PTAS when m is part of the input Hochbaum and Shmoys (1987) and an FPTAS when m is a fixed number Sahni (1976). For problem $P|r_j|C_{\max}$, Hall and Shmoys (1989) obtained the first PTAS.

When all $K_i = 1$, $P|r_j, a_j, p-batch, K_i|C_{\max}$ becomes the problem of scheduling with inclusive processing set restrictions, denoted as $P|r_j, a_j|C_{\max}$. The special case of it where all jobs are released at the same time is denoted as $P|a_j|C_{\max}$. Since $P|a_j|C_{\max}$ is a special case of the classical unrelated machines scheduling problem $R||C_{\max}$, the 2-approximation algorithm Lenstra, Shmoys, and Tardos (1990) and $(2 - 1/m)$-approximation algorithm Shchepin and Vakhania (2005) developed for $R||C_{\max}$ are applicable to $P|a_j|C_{\max}$. There also exist algorithms for $P|a_j|C_{\max}$ with approximation ratios better than $2 - 1/m$: a $(2 - 1/(m-1))$-approximation algorithm (Hwang, Chang, & Lee, 2004b; Kafura & Shen, 1977), a 3/2-approximation algorithm (Glass & Kellerer, 2007), a 4/3-approximation algorithm and a PTAS (Ou et al., 2008), an FPTAS when m is a fixed number (Ji & Cheng, 2008; Li, Li, & Zhang, 2009). The problem $P|r_j, a_j|C_{\max}$ also admits a PTAS when m is part of the input and an FPTAS when m is a fixed number Li and Wang (2010)).

When all $K_i = B$ $(1 < B < n)$ and all $a_j = 1$, $P|r_j, a_j, p-batch, K_i|C_{\max}$ is the parallel batch scheduling problem $P|r_j, p-batch, B|C_{\max}$. This problem is strongly NP-hard even for the single machine case $1|r_j, p-batch, B|C_{\max}$ (Brucker et al., 1998). Lee and Uzsoy (1999) initiated the study of $1|r_j, p-batch, B|C_{\max}$ and proposed a number of heuristics, one of which was proved to be a 2-approximation algorithm by Liu and Yu (2000). Deng, Poon, and Zhang (2003) obtained the first PTAS for $1|r_j, p-batch, B|C_{\max}$. Lee et al. (1992) proposed a $(4/3 - 1/(3m))$-approximation algorithm for $P|p-batch, B|C_{\max}$ (all jobs are released at the same time). For $P|r_j, p-batch, B|C_{\max}$, there is a fast $(7/3 - 1/(3m))$-approximation algorithm (Liu, Ng, & Cheng, 2014) and a PTAS (Li, Li, & Zhang, 2005).

To the best of our knowledge, the general $P|r_j, a_j, p-batch, K_i|C_{\max}$ has not been studied to date. In this paper, we present two fast algorithms for $P|a_j, p-batch, K_i|C_{\max}$ (all jobs are released at the same time) with approximation ratios 3 and 9/4, respectively. We also develop a PTAS for the general $P|r_j, a_j, p-batch, K_i|C_{\max}$ problem, which is also the first PTAS even for $P|p-batch, K_i|C_{\max}$ (all $r_j = 0$ and all $a_j = 1$). We draw upon several ideas from Hall and Shmoys (1989), Li et al. (2005), Li and Wang (2010)) but the combination of parallel batch scheduling and inclusive processing set restrictions makes the analysis quite involved and non-trivial.

Before we proceed, we introduce some frequently used terminologies and notations. Job j is *available* for machine M_i if M_i is an eligible machine for it and it has been released but not yet assigned to any machine. For machine M_i $(i = 1, 2, \ldots, m)$, a batch is called a *full batch* if it contains exactly K_i jobs, otherwise it is called a *partial batch*. Let $p(B_g)$, $S(B_g)$ and $C(B_g)$ denote the processing time, the start time and the completion time of batch B_g, respectively. Let $q(B_g)$ be the processing time of the smallest job in batch B_g. A batch can be naturally regarded as a set. Therefore, if job j is contained in batch B_g, we simply say $j \in B_g$. Let $r(B_g)$ denote the release date of B_g, which is defined to be $r(B_g) = max\{r_j | j \in B_g\}$. Let $a(B_g)$ denote the machine index associated with B_g, which is defined to be $a(B_g) = max\{a_j | j \in B_g\}$. The machines in $\{M_{a(B_g)}, M_{a(B_g)+1}, \ldots, M_m\}$ are called *eligible machines* for batch B_g. Let $\mathcal{J}_i = \{j \in \mathcal{J} | a_j = i\}$, $i = 1, 2, \ldots, m$. Then, $\mathcal{J} = \cup_{i=1}^m \mathcal{J}_i$. The jobs in $\mathcal{J}_i$ can be processed on any of the machines $M_i, M_{i+1}, \ldots, M_m$.

Download English Version:

https://daneshyari.com/en/article/4959874

Download Persian Version:

https://daneshyari.com/article/4959874

Daneshyari.com