Computers & Operations Research 39 (2012) 939–951

Contents lists available at ScienceDirect

Computers & Operations Research

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

A branch and bound algorithm for minimizing total completion time on a single batch machine with incompatible job families and dynamic arrivals

Shiqing Yao, Zhibin Jiang*, Na Li

Department of Industrial Engineering & Logistics Management, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Available online 1 July 2011

Keywords:
Branch and bound algorithm
Dynamic arrivals
Batch scheduling
Incompatible job families

ABSTRACT

In this paper, we consider a single batch machine scheduling problem with incompatible job families and dynamic job arrivals. The objective is to minimize the total completion time. This problem is known to be strongly NP-hard. We present several dominance properties and two types of lower bounds, which are incorporated to construct a basic branch and bound algorithm. Furthermore, according to the characteristics of dynamic job arrivals, a decomposed branch and bound algorithm is proposed to improve the efficiency. The proposed algorithms are tested on a large set of randomly generated problem instances.

1. Introduction

This paper addresses a batch processor scheduling problem. Jobs arrive dynamically and thus have unequal ready times. Jobs in the same family have the same process time and could be processed simultaneously in batches. The batch size may vary by jobs while the maximum size is given. The objective is to minimize the total completion time. This problem arises from the batch scheduling problem in wafer fabrications, where typical furnace machines serve as batch tools. In general, the capacity of furnace machines is six lots with each containing 25 wafers. Lots belonging to different process recipes cannot be batched together, since their process parameters (i.e., pressure and temperature) are quite different. This leads to incompatible job families.

Referring to [1], we denote the problem of interest as

$$1|p\text{-}batch, r_j, b<n, incompat|\sum C_j$$

This problem belongs to NP-hard in strong sense based on reductions to the known NP-hard problem in strong sense: minimizing the total completion time on a single machine with unequal ready times [2].The problem we are interested in involves two interrelated decision making problems: (1) when to group jobs into batches; and (2) how to sequence the batches after they form. Due to dynamic arrivals, when to batch jobs indirectly affects sequencing decisions. On the other hand, the sequencing results also influence batching decisions.

In this paper, we study the structural properties of the problem, develop two types of lower bounds, and propose a branch and bound (*B&B*) algorithm and its improved version.

2. Literature review

Batch scheduling problems can be categorized from different aspects. Readers could refer to [3,4] for the details. In this section, we review the highly related works.

As for identical job family, many relevant problems are solvable. Glassey and Weng [5] proposed a dynamic scheduling algorithm for minimizing the average waiting time. Webster and Baker [6] developed a dynamic programming algorithm for $1|p\text{-}batch, b<n, r_j, p_j=p|\sum C_j$ with $O(n^3)$ overall worst-case time complexity. It is worthwhile to note that this problem is a special case of the problem of interest. Furthermore, Baptiste [7] showed that $1|p\text{-}batch, b<n, r_j, p_j=p|\sum w_jC_j$ also belongs to *P*.

As for multiple job families, there exist two batch types, i.e., compatible job families and incompatible job families. For compatible job families, jobs from different families can be processed together. Jobs in a batch start and complete at the same time, and the process time of a batch is equal to the largest process time among the jobs in its batch. Chandru et al. [8] studied the optimality structure of $1|p-batch, b<n|\sum C_j$ and proposed a *B&B* algorithm. For the case of m jobs families, Chandru et al. [8] presented an $O(m^3b^{m+1})$ time dynamic programming algorithm, and Brucker et al. [9] designed a dynamic programming algorithm with a further gain in efficiency, which only requires $O(b^2m^22^m)$ time. In consideration of weighted jobs, Uzsoy and Yang [10] developed a *B&B* algorithm and several heuristics for $1|p-batch, b<n|\sum w_jC_j$. In recent years,

* Corresponding author. Tel.: +86 13918891152; fax: +86 2134206065.
E-mail address: zbjiang@sjtu.edu.cn (Z. Jiang).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.06.003

the research emphasis has been shifted to batch scheduling with dynamic job arrivals. Lee et al. [11] considered two special cases of $1|p-batch,\ r_j,\ b<n|C_{max}$ and proposed some efficient heuristics for the general case. Wang and Uzsoy [12] developed a meta-heuristic algorithm using random keys genetic algorithm for $1|p\text{-}batch,\ r_j,\ b<n|L_{max}$ and improved not only the approximate ratio but the running time as well.

Batch scheduling with incompatible job families refers that jobs from different families cannot be processed together. This type of batch processing commonly exists in front-end operations. The earliest studies mainly focused on dynamic scheduling problems. A number of heuristics for minimizing total waiting time have been developed by Fowler et al. [13,14], Weng and Leachman [15], Robinson et al. [16], Van der Zee et al. [17], and Cigolini et al. [18]. As for static scheduling problems, they can also be divided into problems with equal ready times and problems with dynamic job arrivals [19]. With respect to equal ready times, most problems seem to be easy. For example, the Greedy Batching (*GRB*) algorithm can obtain an optimal solution of $1|p\text{-}batch,\ b<n,\ incompat|C_{max}$, the *GRWC* (Greedy Weighted Completion Time) algorithm can solve $1|p\text{-}batch,\ b<n,\ incompat|\sum w_jC_j$, and the *GREDD* (Greedy Earliest Due Date) algorithm can solve $1|p\text{-}batch,\ b<n,\ incompat|L_{max}$. All of the above problems belong to *P* because their relevant algorithms are polynomially solvable. As the problem of minimizing $\sum w_jT_j$ on a single machine is NP-hard, many batch scheduling problems related to the tardiness-related objectives are difficult to solve. Mehta and Uzsoy [20] proposed a dynamic programming algorithm and some heuristics for $1|p\text{-}batch,\ b<n,\ incompat|\sum T_j$. Perez et al. [21] developed a two-stage combined heuristic for $1|p\text{-}batch,\ b<n,\ incompat|\sum w_jT_j$. In consideration of unequal ready times, $1|p\text{-}batch,\ r_j,\ b<n,\ incompat|C_{max}$ is reducible to the batch problem of minimizing L_{max} with equal ready times. Hence, it belongs to *P*. Except for that, nearly all the problems with dynamic job arrivals belong to NP-hard in strong sense. Hence, the techniques for solving NP-hard problem were used. For example, Uzsoy [19] developed several heuristics for $1|p\text{-}batch,\ r_j,\ b<n,\ incompat|L_{max}$. Kruz and Masonz [22] developed a heuristic algorithm for $1|p\text{-}batch,\ r_j,\ b<n,\ incompat|\sum w_jT_j$. Based on this heuristic, Tangudu and Kurz [1] proposed a *B&B* algorithm. As for parallel machines problems, there are also some gratifying results. Venkataramana and Srinivasa Raghavan [23] built up a mix integer programing and provided two heuristics for minimizing $\sum w_jC_j$, and Chiang et al. [24] developed a memetic algorithm to minimize $\sum w_jT_j$.

We observe that the problem focused in this paper is a general case of $1|p\text{-}batch,\ b<n,\ r_j,\ p_j=p|\sum C_j$, while it is a special case of $1|p\text{-}batch,\ r_j,\ b<n,\ incompat|\sum w_jT_j$. Since the problem of interest is closely related to the former one, we can adopt the existing results achieved by Webster and Baker [6] to construct the optimality properties. On the other hand, we can design its structural properties, dominance properties, and lower bounds in better ways for it is easier than the latter one.

3. Basic definitions and structural properties

The problem focused in this paper is to schedule a job set N on a single batch machine with maximum batch size b. Let K be the family set and N_k the job set of family k. J_{ik} denotes the ith job of family k, and r_{ik} is its ready time or arrival time. p_k stands for the process time of family k. W.l.o.g, we assumes that $r_{ik}\geq r_{i'k}$ if $i\geq i'$.

In the *B&B* algorithm, a node of the search tree represents a partial schedule which contains the scheduled jobs. The following properties derived from [6] help to make clear the optimal structure of a partial schedule. For completeness, they are briefly repeated as below.

Property 1. (WB) *There exists an optimal schedule, where each time that a group of consecutive batches begins processing is equal to a job arrival time.*

Property 2. (WB) *There exists an optimal schedule, where the lots (of each family) are batched as fully as possible at each batch, and the batches (of each family) are sequenced in non-decreasing order of arrival time.*

Remark 1. Although Properties 1 and 2 are obtained according to a batch scheduling problem with single job family, they are also suitable for the problem with incompatible job families. Property 1 helps us to confine the optimal batch start times to a discrete set. Property 2 indicates that each batch size of the optimal schedule can be easily determined once the start time of the batch as well as its job family is assigned.

According to the properties, the primary notations and definitions are given as follows. Additional ones will be introduced when necessary.

A partial schedule σ_y is defined as follows: $\sigma_y:=s_{[1]}\bigcirc s_{[2]}\cdots s_{[l]}\cdots s_{[y]}$, where $s_{[l]}:=(t_{[l]},k_{[l]})$. In details, $s_{[l]}$ is the batch in position l in the sequence of batches. $t_{[l]}$ is the start time of $s_{[l]}$, and $k_{[l]}$ is the job family of $s_{[l]}$. $C(\sigma_y):=t_{[l]}+p_{k_{[l]}}$, where $C(\sigma_y)$ is the completion time of the last batch in σ_y. Note that $C(\sigma_l)\leq t_{[l+1]}$ is satisfied for $l\in\{0,1,2,\ldots,y-1\}$. $J(\sigma_y)$ and $N\backslash J(\sigma_y)$ are, respectively, a set of the jobs in σ_y and a set of the jobs not included in σ_y. A partial schedule σ_y is referred to as a schedule when $N\backslash J(\sigma_y)=\emptyset$. We denote by $\Omega(\sigma_y)$ the schedule obtained by appending to σ_y the optimal schedule of the unscheduled jobs. $RF(\sigma_y)$ is the remaining family set of the jobs in $N\backslash J(\sigma_y)$. $J_k(\sigma_y)$ and $N_k\backslash J_k(\sigma_y)$ are the job set of family k in σ_y and the job set of family k not included in σ_y, respectively. $F(\sigma_y)$ stands for the total completion time of the jobs in σ_y. Furthermore, $F(\Delta|\sigma)$ is defined to be the total completion time for the job set Δ in schedule σ. And, $F[x|\sigma]$ and $J[x|\sigma]$ are, respectively, the total completion time of the jobs in the xth batch and its associated job set. $U_k(t_1,t_2)$ is the number of released jobs of family k during $(t_1,t_2]$. As for a partial schedule σ_y, $R_k(t|\sigma_y)$ stands for the released jobs of family k in $N\backslash J(\sigma_y)$ at time t.

According to Property 1, for any partial schedule σ_y, the potential next batch time which contains h jobs of family k, $t_k(h|\sigma_y)$, can be calculated as follows:

$$t_k(h|\sigma_y)=\begin{cases} C(\sigma_y) & \text{if } h\leq R_k(C(\sigma_y)|\sigma_y)\\ \min\{t|U_k(C(\sigma_y),t)=h-R_k(C(\sigma_y)|\sigma_y)\} & \text{if } h>R_k(C(\sigma_y)|\sigma_y)\end{cases}$$

According to Property 2, we denote by $B_k(t|\sigma_y)$ the batch size if family k are chosen at time t. Clearly, $B_k(t|\sigma_y)=\min(R_k(t|\sigma_y),b)$.

For convenience, readers can find the notations in Appendix A. When there is no ambiguity, $C(\sigma_y)$, $J(\sigma_y)$, $N\backslash J(\sigma_y)$, $J_k(\sigma_y)$, $N_k\backslash J_k(\sigma_y)$, $RF(\sigma_y)$, $R_k(t|\sigma_y)$, $t_k(h|\sigma_y)$, and $B_k(t|\sigma_y)$ are simplified into C_y, J, $N\backslash J$, J_k, $N_k\backslash J_k$, RF, $R_k(t)$, $t_k(h)$, and B_k, respectively.

Let σ_y and $\sigma_{y'}$ be the two partial schedules, we say that σ_y dominates $\sigma_{y'}$ or $\sigma_{y'}$ is dominated, if

$$F(\Omega(\sigma_y))\leq F(\Omega(\sigma_{y'}))$$

We denote by batch $(t,\ k)$ the child node where the batch belongs to family k and is processed at time t. In the *B&B* algorithm, the branch procedure is to create child nodes from each active node. Each child node is created by appending the next scheduled batch to the end of the partial schedule σ_y. In this problem, the number of potential child nodes is $|K|b$ in worst case. Clearly, as the number of jobs increases, the total number of nodes increases greatly as well. To stop searching those non-optimal child nodes as early as possible, dominance properties and efficient lower bounds are indispensable. Related techniques are discussed in the following sections.