Computers & Industrial Engineering 61 (2011) 739–743

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

An effective neighborhood search algorithm for scheduling a flow shop of batch processing machines ☆

Deming Lei *, Tao Wang

School of Automation, Wuhan University of Technology, China

ARTICLE INFO

Article history:
Received 5 February 2010
Received in revised form 10 May 2011
Accepted 11 May 2011
Available online 27 May 2011

Keywords:
Neighborhood search
Flow shop scheduling
Batch processing machines
Permutation

ABSTRACT

This paper considers scheduling problem of flow shop with many batch processing machines and objective of maximum lateness. An effective neighborhood search algorithm (NSA) is proposed for the problem, in which a job permutation and a batch permutation are used to indicate the solution of two sub-problems, respectively. Each job permutation consists of several family-permutations for the representation of jobs from the same family. Two swaps are applied to two permutations to produce new solutions. NSA is applied to a number of instances and compared with some methods, and computational results validate the good performance of NSA.

1. Introduction

Batch processing machines (BPM) are an integral part of many practical applications such as semiconductor burn-in operations, environmental stress screening chambers, process industries, wafer fabrication process, testing electrical circuits and metal working. The scheduling problem of BPM consists of two sub-problems. The first one called batch formation is to form batches such that the machine capacity is not violated, and the second called batch scheduling is to determine a schedule of batches to optimize some performance indices.

In flow shop with BPM, the processing routes are same for all jobs, more than one incompatible job family may exist and BPM has a capacity restriction. Scheduling in a flow shop with BPM has attracted much attention and some results have been obtained in the past decade. Sung, Kim, and Yoon (2000) proposed two efficient heuristics incorporating the problem reduction procedure to minimize makespan and total completion times in a multi-stage flow shop with BPMs. Sung and Min (2001) considered two-BPM flow shop scheduling with earliness/tardiness measure under common due-date and proposed a polynomial time algorithm. Damodaran and Srihari (2004) proposed two mixed integer models to schedule batches on two BPMs in a flow shop. Oulamara (2007) discussed scheduling in a no-wait flowshop with two different batching machines and provided a heuristic algorithm. Kashan and Karimi (2009) improved the mixed integer linear formulation of the problem. Amin-Naseri and Beheshti-Nia (2009) presented three heuristics for scheduling in hybrid flow shop with batching machine in some stages.

Mirsanei, Karimi, and Jolai (2009) developed two different simulated annealing (SA) based on two constructive heuristics in a two-stage flow shop with two BPMs. Yimer and Demirli (2009) suggested a genetic algorithm (GA) based solution approach to fuzzy scheduling in a two-stage flow shop with BPM. Manjeshwar, Damodaran, and Srihari (2009) proposed a SA for minimizing the makespan on two-BPM flow shop. Lei, Wang, Chen, Zhang, and Guo (2010) presented an effective GA to optimize makespan and total tardiness simultaneously, a rank and the weighted objective based binary tournament selection and an external archive updating strategy are also adopted to obtain a set of non-dominated solutions. Lei and Guo (2011) proposed a variable neighborhood search (VNS) for many-BPM scheduling with tardiness objectives, in VNS, job permutation is the only optimization object and the solutions of batch formation and batch scheduling can be directly obtained by using the permutation. Liao and Huang (in press) presented a heuristic based tabu search for flow shop with two-BPM such that makespan is minimized.

Most of literature on scheduling in flow shop with BPM adopts makespan as objective and few papers are related to maximum tardiness; scheduling problem of flow shop with more than two BPMs is not fully considered and only GA and VNS have been applied to solve the problem. In this paper, a novel NSA is proposed for the problem, in which the solution of two sub-problems is represented by using a job permutation and a batch permutation, respectively. Job permutation consists of several family-permutations used for jobs from the same family. Some empty batches are allowed in

☆ This manuscript was processed by Area Editor John W. Fowler.
* Corresponding author. Tel.: +86 2786534910.
E-mail address: deminglei11@163.com (D. Lei).

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2011.05.005

the permutation of batch. Two swaps based neighborhood search are applied sequentially to job permutation and batch permutation to produce new solutions. NSA is finally tested and compared with VNS (Lei & Guo, 2011) and a GA.

The remainder of this paper is organized as follows. We describe the problems in the next section. In Section 3, we present an effective NSA for the problem under study. Computational experiments and results discussion are done in Section 4. A summary and discussions on future research conclude the paper in the final section.

2. Problem description

Many-BPM flow shop scheduling problem is composed of n jobs $J_i \in J = \{J_1, J_2, \ldots, J_n\}$ and m BPMs $M_k \in M = \{M_1, M_2, \ldots, M_m\}$. Each job J_i has a size s_i and due-date d_i and its processing time on machine M_k is denoted by p_{ik}. All jobs have the same operation sequence and can be partitioned into F incompatible families. The machine capacity C is known in advance. A batch $b \in B$ is feasible as long as the total size of all jobs in the batch *cannot* exceed C. B is the set of batches. The processing time of batch $b \in B$ on machine M_k is determined by the longest processing job in the batch.

The following assumptions are often considered.
All jobs are available at time 0.
All jobs in a batch are processed simultaneously.
Once the processing of a batch is started, it cannot be interrupted.
No additional jobs can be introduced or removed from a batch while the batch is being processed.
Buffers between any two machines are infinite.
No batch may be processed on more than one machine at a time et al.
The following objective is treated.

$$Minimize\ T_{max} = max_{J_i \in J}\{T_i\} \tag{1}$$

For each job J_i, it must be allocated in a batch, if it is in a batch b, then $T_i = \max_{J_i \in b}\{0, C_b - d_i\}$, C_b indicates the completion time of batch b.

By adopting the three-field notation of Graham, Lawler, Lenstra, and Rinnooy Kan (1979), the above problem can be denoted as $F|C,F, d_i|T_{max}$.

3. NSA for many-BPM flow shop scheduling

In this section, we first describe the coding and decoding procedure for the problem and then depict the main steps of NSA.

3.1. Coding and decoding

The applications of meta-heuristics to production scheduling are always based on the representation of the problem. *For scheduling in a shop with BPM*, job permutation and random key string are often used to denote the solution of the problem. With respect to job permutation, Damodaran, Manjeshwar, and Srihari (2006) used it to represent the solution of batch formation sub-problem, Lei and Guo (2011) and Lei et al. (2010) applied it to indicate the solution of the whole problem and job permutation is the unique optimization object of scheduling algorithm.

In this study, job permutation $(q_{11}, q_{12}, \ldots, q_{1h_1}, q_{21}, q_{22}, \ldots, q_{2h_2}, \ldots, q_{F1}, \ldots q_{Fh_F})$ and batch permutation $(p_1, p_2, \ldots, p_{bnum})$ are used to represent a solution of scheduling problem in flow shop with n jobs and m BPMs. Job permutation is composed of F different family-permutations $(q_{i1}, q_{i2}, \ldots, q_{ih_i})$, $i = 1, 2, \ldots, F$, its first family-permutation P_1 meets $\{q_{11}, q_{12}, \ldots, q_{1h_1}\} = \{1, 2, \ldots, h_1\}$, the second family-permutation P_2 meets $\{q_{21}, q_{22}, \ldots, q_{2h_2}\} = \{h_1 + 1, h_1 + 2, \ldots, h_1 + h_2\}$ and so on, h_i is the number of jobs of the ith family, $\sum_{i=1}^{F} h_i = n$. All integers of batch permutation are in set $\{1, 2, \ldots, b_{num}\}$.

Integer g_i indicates a big enough number of batches formed by jobs in the ith family, in this study, $g_i = \alpha_i + \varepsilon_i$, α_i is the possible maximum number of batches formed by using family-permutation $P_i = (q_{i1}, q_{i2}, \ldots, q_{ih_i})$, $\alpha_i \leqslant h_i$, ε_i represent the increment of g_i, $b_{num} = \sum_{i=1}^{F} g_i$. The following procedure is used to build the schedule of the problem using two permutations.

```
(1) Let $t = 1$, $b_{num} = 1$;
(2) For family-permutation $(q_{t1}, q_{t2}, \ldots, q_{th_t})$, let $c_{bnum} = 0$,
    (a) Begin with $q_{tl}$, $l = 1$, if $c_{b_{num}} + s_{q_{tl}} \leqslant C$, then
      $c_{b_{num}} = c_{b_{num}} + s_{q_{tl}}$ and $l = l + 1$;
      if $C_{numb} + S_{q_{tl}} > C$ and $l < h_t$, then
      $b_{num} = b_{num} + 1$ and $c_{b_{num}} = 0$;
      if $C_{numb} + S_{q_{tl}} > C$ and $l = h_t$, then $b_{num} = b_{num} + 1$ and
      add $q_{tl}$ into $b_{b_{num}}$.
    (b) repeat (a) until all jobs of the $t$th family are grouped
      into batches.
    (c) if $b_{num} < \sum_{i=1}^{t} g_i$, then $b_{num} = \sum_{i=1}^{t} g_i$.
(3) $t = t + 1$, if $t \leqslant u$, go to (2); else stop the iteration and
    batch $b_1, b_2, \ldots, b_{bnum}$ is formed.
Construct the final schedule using batch permutation
    $(p_1, p_2, \ldots, p_{bnum})$. For permutation $(p_1, p_2, \ldots, p_{bnum})$, the
    first batch $b_{p_1}$ is allocated first, and then he second batch
    and so on. Each nonempty batch is allocated in the best
    available beginning time on each machine.
  In (2), if $b_{num} < \sum_{i=1}^{t} g_i$, there are some empty batches in
    $b_l, b_{l+1}, \ldots, b_{bnum}$.
```

For the problem with 20 jobs and 3 machines shown in Table 1, $s_i = 1, 2, 1, 3, 2, 1, 3, 2, 1, 3, 2, 2, 2, 2, 3, 3, 3, 1, 2, 3, 3$, $C = 8$, jobs $1, 2, \ldots, 10$ belong to the first job family and jobs $10, 11, \ldots, 20$ are from the second family, $g_1 = g_2 = 5$. Fig. 1 shows two possible permutations of the problem. Job permutation $(5, 10, 2, 3, 1, 7, 8, 4, 9, 6, 11, 12, 19, 20, 14, 13, 16, 15, 17, 18)$ is composed of two family-permutations Its first family-permutation is

Table 1
An example of the problem under study.

Job	Processing time			Job	Processing time		
	M_1	M_2	M_3		M_1	M_2	M_3
1	2	3	2	11	6	2	3
2	5	4	3	12	5	3	6
3	1	4	5	13	4	4	5
4	5	5	4	14	3	1	7
5	7	3	2	15	2	3	2
6	3	3	1	16	6	5	4
7	4	5	3	17	7	7	4
8	5	6	4	18	5	6	5
9	2	4	2	19	3	4	6
10	6	3	2	20	5	3	3

job permutation

5 10 2 3 1 7 8 4 9 6 11 12 19 20 14 13 16 15 17 18

batch permutation

1 6 10 2 7 8 5 3 9 4

Fig. 1. An illustration for coding.