Applied Soft Computing 38 (2016) 395–404

Contents lists available at ScienceDirect

Applied Soft Computing

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

An ACO algorithm for makespan minimization in parallel batch machines with non-identical job sizes and incompatible job families

Zhao-hong Jia [a,b], Chao Wang [b], Joseph Y.-T. Leung [c,d,*]

[a] Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, PR China
[b] School of Computer Science and Technology, Anhui University, Hefei, Anhui 230039, PR China
[c] School of Management, Hefei University of Technology, Hefei, Anhui 230009, PR China
[d] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

ARTICLE INFO

Article history:
Received 16 January 2015
Received in revised form 11 June 2015
Accepted 29 September 2015
Available online 19 October 2015

Keywords:
Parallel batch machines
Max–Min Ant System
NP-hard
Incompatible job families
Makespan

ABSTRACT

We study the problem of scheduling a set of N jobs with non-identical job sizes from F different families on a set of M parallel batch machines; the objective is to minimize the makespan. The problem is known to be NP-hard. A meta-heuristic based on Max–Min Ant System (MMAS) is presented. The performance of the algorithm is compared with several previously studied algorithms by computational experiments. According to our results, the average distance between the solutions found by our proposed algorithm and the lower bounds is about 4% less than that of the best of all the compared algorithms, demonstrating that our algorithm outperforms the previously studied algorithms.

1. Introduction

Batch scheduling problems occur in many industries, such as semiconductor manufacturing, cargo handling in port, transportation, storage systems, and so on. The problem is motivated by the diffusion operation in the wafer fabrication of semiconductor manufacturing. Due to the chemical nature of the process, only jobs with the same recipe can be processed together [1]. All jobs requiring the same recipe can be viewed as a job family, and jobs in the same family have the same processing times. Effective scheduling of these operations is particularly important because of their long processing time requirements, generally 10 h as opposed to 1 or 2 h for most other processes.

In contrast to the classical machine scheduling, a batch processing machine (BPM) can process several jobs in a batch simultaneously. There are two types of batch scheduling problems: s-batch and p-batch. In s-batch, the jobs in a batch are processed in serial and the processing time of a batch is the sum of the processing times of all the jobs in the batch, while in p-batch, the jobs in a batch are processed in parallel and the processing time of a batch is the longest processing time of the jobs in the batch. P-batch scheduling is more important than s-batch scheduling in semiconductor manufacturing [2]. Additionally, p-batch scheduling is often encountered in many modern manufacturing industries such as food, chemical and mineral processing, pharmaceutical and metalworking industries as well as environmental stress screening chamber fabrication [3].

In this paper, we consider p-batch scheduling on parallel BPMs with identical machine capacity, non-identical job sizes and incompatible job families. Jobs with non-identical sizes come from different job families. All jobs in the same family have the same processing times, while jobs from different families have different processing times. The jobs have to be grouped into batches such that the total size of the jobs in the batch does not exceed the machine capacity. Moreover, jobs from different families cannot be grouped together in the same batch. The jobs are assumed to be ready at time zero. The processing time of a batch is equal to the longest processing time of all the jobs in the batch [4]; in our case the processing times of all the jobs in a batch are the same. The batches are then scheduled on the machines to minimize the makespan. The problem of minimizing the makespan on a single BPM with non-identical job sizes has been shown to be NP-hard [5]. Thus, our problem is also NP-hard.

Like other batch scheduling problems, the problem in this paper can be addressed by solving two independent subproblems: group the jobs into batches and schedule the batches on the parallel BPMs. We propose a Max–Min Ant System (MMAS) algorithm to group the jobs into batches, and then apply the Multi-Fit (MF) algorithm [6] to schedule the batches on the machines.

In the literature, there are several articles that deal with incompatible job families [1,30,31,35,36]. The article in [1] assumes that the job sizes are identical, while we deal with arbitrary job sizes. The articles in [30,31] deal with the total weighted tardiness objective, while we study the makespan objective. The articles [35,36] study exactly the same problem as ours. Article [35] studies parallel BPMs while article [36] studies a single BPM. We will be comparing the performance of our heuristic against the heuristics of [35]. As we shall see later, our heuristic outperforms all of the heuristics in [35].

The paper is organized as follows. In the following section we review previous related work on BPM scheduling problems as well as the MMAS algorithm. Section 3 formally defines the problem, and a lower bound is provided. The proposed algorithm and its implementation are described in Section 4, and the results of the computational experiment are reported in Section 5. Finally, in Section 6, we conclude the paper with a summary and some directions for future research.

* Corresponding author at: Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA. Tel.: +1 9735963387; fax: +1 9735965777.
E-mail addresses: zhjia@mail.ustc.edu.cn (Z.-h. Jia), chaowang_ahu@163.com (C. Wang), leung@njit.edu (J.Y.-T. Leung).

http://dx.doi.org/10.1016/j.asoc.2015.09.056
1568-4946/

2. Previous related work

Recently, a lot of research has been done on scheduling on BPMs as well as the MMAS algorithm. Previous work related to these two areas will be presented in the next two subsections.

2.1. BPM problem

We focus on reviewing the literature that have commonalities with the problem assumptions as ours.

Ikura and Gimple [7] probably are the first to introduce the batch scheduling problem. They provided an $O(n^2)$ algorithm to minimize the makespan on a single BPM with identical job processing times and job sizes, and different job release times. Lee et al. [8] proposed efficient algorithms for minimizing the number of tardy jobs and maximum tardiness under a set of assumptions. They also gave a heuristic with a tight worst-case bound to minimize the makespan on parallel BPMs. Chandru et al. [9] proposed a branch-and-bound algorithm for scheduling on a single BPM to minimize the total completion time. However, their algorithm is only effective on small-scale instances. Chandru et al. [10] demonstrated that the problem of scheduling on a single BPM with a fixed number of job families can be solved in polynomial time. For a fixed number of incompatible job families, Leung [11] provided a polynomial-time algorithm to minimize the makespan on parallel BPMs, assuming unit job sizes. Uzsoy [1] examined a number of problems related to the scheduling on BPMs with incompatible job families and unit job sizes. He developed efficient optimal algorithms to minimize the makespan, maximum lateness and total weighted completion time. He further applied some of the results to parallel BPMs. He also investigated the problem with dynamic job arrivals with the objectives of minimizing the makespan and maximum lateness. Under the assumptions of job number and machine capacity being fixed, Mehta and Uzsoy [12] gave a polynomial-time dynamic programming algorithm for scheduling on a single BPM to minimize total tardiness. They also examined the heuristic which can find near-optimal solutions in reasonable computational time. Lee and Uzsoy [13] studied the problem of scheduling on a single BPM to minimize the makespan, when the jobs have release times. They proposed polynomial and pseudo-polynomial time algorithms for several special cases. They also provided efficient heuristics for the general problem and conduct extensive computational experiments to evaluate the performances of the algorithms.

One assumption of the above research is that each job has unit size. However, job sizes are often different in practice. Hence, some researchers turned to study scheduling problems with non-identical job sizes on a single BPM. Uzsoy [5] proved that minimizing the makespan on a single BPM with non-identical job sizes is strongly NP-hard. He provided a number of heuristics and a branch-and-bound algorithm that can solve small-scale problems. Dupont and Jolai Ghazvini [14] provided two heuristics, named the Best-Fit-Longest-Processing-Time (BFLPT) and the Successive Knapsack (SK). BFLPT is based on the Best-Fit algorithm for the bin-packing problem while SK builds a schedule, batch by batch, where the jobs are grouped to minimize the unoccupied space in the batch. Uzsoy and Yang [15] presented a number of heuristics and a branch-and-bound algorithm to minimize the total weighted completion time.

Under the assumptions that only jobs from the same family can be batched together and jobs in the same family have the same processing times, Dobson and Nambimadom [16] proposed several heuristics to minimize the total weighted completion time on a single BPM. Azizoglu and Webster [17] developed a branch-and-bound algorithm that can solve problems up to about 25 jobs in reasonable time. To minimize the makespan on a single BPM with non-identical job sizes, Dupont and Dhaenens-Flipo [18] provided a branch-and-bound method. Based on the Longest-Processing-Time (LPT) and the First-Fit (FF) rules, Cheng et al. [19] gave two polynomial-time heuristics to schedule jobs with arbitrary sizes and incompatible families on a single BPM to minimize the makespan and the total batch completion time, respectively. Jolai [20] presented a polynomial-time dynamic programming algorithm to minimize the number of tardy jobs for a fixed number of job families and limited machine capacity. Li et al. [21] provided an approximation algorithm for the general problem with arbitrary release times and job sizes. Yao et al. [22] solved the problem of scheduling on a single BPM with non-identical job sizes and dynamic job arrivals through an improved branch-and-bound algorithm.

Recently, meta-heuristics have been applied to solve scheduling problems on a single BPM. Damodaran et al. [23] presented a Simulated Annealing (SA) algorithm to minimize the makespan on a single BPM. Kashan et al. [24] provided two Genetic Algorithms (GAs) for the same problem. Jia and Leung [25] proposed a MMAS algorithm for minimizing the makespan on a single BPM. Based on an Ant Colony Optimization (ACO) algorithm, Li et al. [26] addressed the bi-criteria problem of scheduling jobs with different arrival times, incompatible families, sequence-dependent setup times and the qual-run requirements of advanced process control on parallel BPMs to minimize the Total Weighted Tardiness (TWT) and the makespan simultaneously. To schedule a single BPM with jobs of unequal sizes, dynamical arrivals, and due dates with the aim of minimizing the makespan and the TWT simultaneously. Wang and Chou [27] developed a SA-based Pareto multi-objective algorithm.

With the development of new production environments, the study of scheduling on parallel BPMs with identical job sizes has emerged. Lee et al. [8] proposed the LPT heuristic algorithm. To minimize the makespan, maximum lateness, and total weighted completion time, Uzsoy [1] applied a number of algorithms to schedule on single and parallel BPMs with incompatible job families and job release times. Brucker et al. [28] developed a dynamic programming algorithm for scheduling on two identical and parallel BPMs with identical deadline, unit processing time and unit setup time. For the minimization of total weighted tardiness on parallel BPMs with incompatible families, identical job sizes and arbitrary job weights, Mönch and Almeder [29] presented an Ant Colony System (ACS) and a MMAS algorithms. Venkataramana and Srinivasa Raghavan [30] gave an ACO-based algorithm by using the structural properties of the problem. Almeder and Mönch [31] proposed an ACO algorithm and a Variable Neighborhood Search (VNS) approach hybridized with a decomposition heuristic and a local search scheme for the same problem. According to the experimental results, the VNS method shows better performance than the ACO and the GA algorithms in both the running time and the solution quality.

Since job sizes are generally non-identical in practice, researchers began to study the scheduling problems with non-identical job sizes on parallel BPMs. Ozturk et al. [32] addressed the makespan minimization problem on parallel BPMs with non-identical job sizes and release dates; they gave a 2-approximation algorithm as well as a MILP model. Chang et al. [33] provided a SA algorithm to minimize the makespan on parallel BPMs with non-identical job sizes. The SA exploits a better neighboring solution after the jobs are grouped into batches and then the batches are assigned to the machines by the LPT rule. To minimize the makespan on parallel BPMs, Damodaran and Chang [34] presented several heuristics and compared their performances with that of the SA (Chang et al. [33]) and CPLEX. Koh et al. [35,36] addressed the problem of scheduling on single and parallel BPMs with arbitrary job sizes and incompatible job families to minimize the makespan, total completion time and total weighted completion time. For each problem, they designed a number of heuristics and compared the