Computers & Operations Research 46 (2014) 49–58

Contents lists available at ScienceDirect

Computers & Operations Research

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

An improved meta-heuristic for makespan minimization of a single batch machine with non-identical job sizes

Zhao-hong Jia [a], Joseph Y.-T. Leung [b,*]

[a] Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University, Hefei, Anhui 230039, PR China
[b] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, United States

ARTICLE INFO

Available online 18 January 2014

Keywords:
Meta-heuristic
Batch machines
Max–min ant system
NP-hard

ABSTRACT

We consider the problem of minimizing the makespan on a single batch machine with non-identical job sizes, where several jobs can be simultaneously processed as a batch. We formulate makespan minimization as a problem of minimizing the wasted space. Applying a candidate set strategy to narrow the search space, combined with a wasted-space-based heuristic to update the pheromone information, an improved max–min ant system algorithm is presented. A specific local search method is incorporated to gain better performance. Appropriate parameter settings in the proposed algorithm are determined by extensive experiments. The experimental results show that the proposed algorithm outperforms several previously studied algorithms.

1. Introduction

A parallel batch machine can process several jobs simultaneously as a batch. Such machines are widely used in many manufacturing industries such as the semiconductor industry, casting industry, metal industry, aeronautical industry, pharmaceutical industry, and logistics freight [1]. In the semiconductor industry, there are four main steps in the processing of very large-scale integrated circuits: wafer fabrication, wafer probe, assembly and final testing. Integrated circuits are subjected to burn-in operations that take place in the final testing stage. The purpose of the burn-in operations is to subject the integrated circuit to thermal stress for a certain period of time in order to bring out the latent defects (if there is any). The burn-in time of an integrated circuit is typically specified by the customers and is quite lengthy. A typical burn-in time is 120 h, as opposed to a few hours in the other operations in the manufacturing process. Since the burn-in process is time-consuming, it emerges as a major bottleneck of the entire manufacturing process. Thus, efficient execution of the burn-in operation is critical for the success of the company.

The burn-in operation can be described as follows. Each integrated circuit constitutes a job. The jobs need to be loaded onto boards which are then put into an oven for an extended period of time. Each job has a prespecified burn-in time and a certain size (which is the number of boards the job occupies). The oven has a finite capacity; i.e., the number of boards that can be put into the oven. Several jobs can be put into the oven at the same time, provided that the capacity of the oven is not exceeded. A job must be put into the oven for its prespecified burn-in time, but it can be longer. Therefore, the processing time of a batch is simply the longest burn-in time of all the jobs in the batch. The problem is to determine how the jobs should be batched together so as to minimize the makespan. We call this problem the Single Batch Machine with Non-identical Job Sizes, in short, the SBMN problem.

The SBMN problem has received tremendous attention in the last two decades. Many research papers have been written on this subject, see the literature review in the next section. Its popularity is due largely to the following facts. First, since the burn-in operation is a lengthy process, a reduction in the makespan represents a significant increase in the throughput. Second, a reduction in the makespan represents a significant saving of energy. The oven typically maintains a temperature of 120 °C, and hence consumes a lot of energy. Third, the SBMN problem is an intriguing problem because it involves two dimensions – time and space.

The SBMN problem is known to be NP-hard. Motivated by the computational complexity of the problem, we propose an improved max–min ant system (MMAS) algorithm to solve it. Utilizing the equivalence between the minimization of the wasted space and the minimization of makespan, we construct a candidate set, comprising the unscheduled jobs that can decrease the wasted space of the current batch, to narrow the search space. We also provide a novel definition of heuristic information to make the ants move faster towards better solutions. In addition, to further improve the solution quality, a local optimization strategy is used in the algorithm. Experimental results show that our algorithm outperforms several previously studied algorithms.

Our approach differs from previous approaches to the SBMN problem in several aspects. First, although some meta-heuristics have been developed for the SBMN problem, little effort has been made to

* Corresponding author. Tel.:+1 973 596 3387; fax: +1 973 596 5777.
E-mail address: leung@oak.njit.edu (J.-T. Leung).

0305-0548/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2014.01.001

apply the MMAS algorithm to the problem and compare comprehensively its performance with other ant colony optimization (ACO) algorithms. Second, once a solution is obtained, we apply a local optimization algorithm to further enhance the solution quality.

The rest of the paper is organized as follows. In Section 2, we review previous work related to the SBMN problem and MMAS. Section 3 describes the SBMN problem. In Section 4, an improved MMAS algorithm is presented in detail. The experimental framework and the computational results are provided in Section 5. Finally, we draw some concluding remarks and suggest some future lines of research in Section 6.

2. Literature review

Recently, considerable research has been devoted to the SBMN problem and MMAS. Literature germane to SBMN and MMAS will be introduced in the following two subsections.

2.1. SBMN

Batching generally denotes a grouping of jobs with the aim to process them either sequentially (serial-batching) or simultaneously (parallel-batching) on the same machine [1]. In this paper we study the parallel-batching scheduling on a single machine. Thus, we pay our attention to those works that have commonalities in their assumptions with ours, especially those investigating the case of arbitrary job sizes.

Uzsoy [2] proposed the pioneer work on the SBMN problem. He proved that both makespan minimization and total completion time minimization are strongly NP-hard, and presented several heuristics and a branch-and-bound algorithm to solve the SBMN problem. The most well-known of them are the FFLPT and FFDECR algorithms. In the FFLPT, jobs are first arranged in decreasing order of their processing times and then the First-Fit (FF) algorithm is applied to the resulting list of jobs. FFDECR differs from FFLPT only in that jobs are sorted in decreasing order of the job sizes. Chen et al. [3] proposed a clustering algorithm CACB (Constrained Agglomerative Clustering of Batches). They compared the CACB algorithm with the BFLPT (Best Fit Longest Processing Time) and the GA (Genetic Algorithm) heuristics, and showed that the CACB outperformed the other two heuristics. Sung and Choung [4] considered the SBMN problem with job release times and presented some better heuristics. Dupont and Dhaenens-Flipo [5] developed a branch-and-bound procedure for minimizing makespan.

Zhang et al. [6] provided the first theoretical results in worst-case ratios. They proved that the worst-case ratio of FFLPT is not greater than 2, while the worst-case ratio of FFDECR can be arbitrarily large. Besides, they also proposed heuristics under the proportional assumption which means that large jobs have processing times no less than those of small jobs. Under such an assumption, they showed that the worst-case ratio is 3/2. For the general case, they proposed a more complicated heuristic with a worst-case ratio of 7/4. Li et al. [7] presented an approximation algorithm with the worst-case ratio $2+\varepsilon$ for the general problem with arbitrary release times and job sizes, where $\varepsilon > 0$ can be made arbitrarily small. Kashan et al. [8] generalized the proportional assumption made by Zhang et al. [6] and obtained an $O(n \log n)$ algorithm with an asymptotic worst-case ratio of 4/3 for the special case in which the job sizes and job processing times are agreeable.

Some researchers considered objectives other than the makespan. Ghazivini and Dupont [9] provided various heuristics to minimize the mean flow time of jobs, where heuristic DYNA achieved the best worst-case performance. DYNA, based on the SPT rule, is an iterative and parametric heuristic. Azizoglu and Webster [10] presented a branch-and-bound procedure to minimize the total weighted completion time on a batch machine with arbitrary job processing times, job weights and job sizes; however, their algorithm can handle less than 25 jobs in a reasonable amount of time.

Since exhaustive enumerations are computationally expensive, researchers have fallen back on meta-heuristic algorithms. Meta-heuristic algorithms can be categorized as one based on neighborhood search and one based on constructive method [11]. The neighborhood search meta-heuristics begin with some initial solutions and then a local search strategy is used to iteratively improve them. For SBMN, its initial solution consists of a batching of the jobs and a sequence of the batches. Since the numbers of batches vary with different solutions, it is difficult to encode the sequence of batches directly at the initial stage. Thus, most neighborhood meta-heuristics encode the job sequence first and then apply some heuristics to group the jobs together [12]. Kashan et al. [13] proposed a typical neighborhood-based meta-heuristic, where random sequences of jobs are generated by using GA operators, and then the First-Fit (FF) method is employed to batch the jobs. By using some heuristics to form the batches, Damodaran et al. [14] employed genetic algorithms to minimize makespan. Melouk et al. [15] proposed a simulated annealing method to minimize the makespan on a single machine. It should be emphasized that the optimization abilities of meta-heuristics are restrained in that their performance partially relies on the heuristics to construct the batches, which may lead to performance degradation. Moreover, extra heuristics would increase the computational time. For these reasons, constructive meta-heuristics are more efficient than the neighborhood ones for solving SBMN.

The constructive-based meta-heuristics construct each solution from scratch by iteratively adding the components into an initially empty solution until a feasible solution is produced. Recently, ant colony optimization (ACO), one of the most representative constructive meta-heuristics, has become a hot spot in meta-heuristics research. ACO refers to a class of distributed algorithms that share some properties with the self-organized interaction between several simple agents (ants) [16]. Ant system (AS) is the original version of ACO. Recently, a number of its variants have been proposed, such as rank-based AS [17], ant colony system [18] and MMAS [19]. ACO has found success in addressing a wide range of NP-hard combinatorial optimization problems, such as the travelling salesman problem [20], mobile agent routing problem [21], and vehicle routing problem [22]. Recently, ACO has been used to solve batch scheduling problems with non-identical job sizes. Cheng et al. [23] introduced an improved ACO method, combined with the Metropolis criterion, to solve the SBMN problem in a fuzzy manufacturing system. Xu et al. [12] proposed an ACO algorithm to minimize the makespan of the SBMN problem with different release times, and its performance was empirically validated to be superior to that of CPLEX and genetic algorithms.

We refer the readers to the article by Mathirajan and Sivakumar [1] for an extensive review of the SBMN problem.

2.2. MMAS

ACO stems from the foraging behavior of ants, particularly in the collaborative communication among them in a search for food [24]. MMAS, a special variant of ACO algorithms, aims to achieve better search performance than AS by increasing the exploitation of the best solution found during the search process, and to avoid early search stagnation more effectively.

Specifically, MMAS [19] differs from the other ACOs in three aspects:

(i) Only one ant, either the best solution in the current iteration (iteration-best ant) or the best solution from the beginning of the search (global-best ant), adds pheromone after each iteration to exploit the best solution found so far.