Computers & Industrial Engineering 99 (2016) 372–381

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Minimizing total flow time on a batch processing machine using a hybrid max–min ant system

N. Rafiee Parsa, B. Karimi *, S.M. Moattar Husseini

Department of Industrial Engineering and Management Systems, Amirkabir University of Technology, P.O. Box 1591634311, Tehran, Iran

ARTICLE INFO

Article history:
Available online 10 June 2016

Keywords:
Batch processing machines
Semiconductor manufacturing
Max–min ant system
Total flow time
Dynamic programming

ABSTRACT

A single batch processing machine scheduling problem with non-identical job sizes to minimize the total flow time is investigated. The problem is formulated as a binary mixed integer programming model. Since the research problem is shown to be NP-hard, a hybrid metaheuristic algorithm based on the max–min ant system (MMAS) is proposed. MMAS is an ant colony optimization algorithm derived from ant system. In the proposed algorithm, first, a sequence of jobs is constructed based on the MMAS algorithm. Then, a dynamic programming algorithm is applied to obtain the optimal batching for the given job sequence. At last, an effective local search procedure is embedded in the algorithm for finding higher quality solutions. The performance of the proposed algorithm is compared with CPLEX and available heuristics in the literature. Computational results demonstrate the efficacy of the proposed algorithm in terms of the solution quality.

1. Introduction

Batch processing machine (BPM) scheduling is a well-known scheduling problem with many applications, such as burn-in operations in semiconductor industrials, heat-treating ovens, chemical processes performed in tanks or kilns, testing process of electrical circuits, and wafer fabrication process. In batch scheduling environment, several jobs can be processed as a batch on a machine simultaneously. The goal is to group the jobs into batches and then schedule the batches on the machine in order to optimize some measure of effectiveness.

In semiconductor manufacturing, integrated circuits are processed through several stages. In the last step, integrated circuits proceed to the final testing. This research is motivated by burn-in operations occurring during the final testing. Since the burn-in operation has the longest processing time among the other testing operations, it is generally considered as the bottleneck process in the final testing step. Thus, the efficient scheduling of these operations is essential and can greatly affect the production efficiency. A detailed description of the semiconductor manufacturing and burn-in operations could be found in Mönch, Fowler, Dauzère-Pérès, Mason, and Rose (2011).

In this research, we consider the single BPM problem with non-identical job sizes to minimize the total flow time. There is a set of jobs which are available for processing on a batch processing machine. Each job is characterized by its processing time and size. The machine can process a set of jobs as a batch as long as the total size of the batch is less than or equal to the machine capacity. Jobs cannot be split across the batches. The processing time of a batch is determined by the longest processing time among the jobs in the batch. In this paper, from now on, the term SUMCJ refers to the problem of minimizing total flow time on a batch processing machine considering the above assumptions. Description of the research problem and its formulation will be given in Section 2.

In high-tech manufacturing such as semiconductor manufacturing where the product life cycles are short, developing appropriate strategies to reduce the inventory holding costs is essential. Minimizing the total flow time is a measure of reducing work-in-process inventory holding costs. Thus, an efficient scheduling to minimize the total flow time is managerially important.

In recent years, scheduling problems related to the batch processing machines have been investigated extensively by many researchers. We concentrate on reviewing the studies having the most similarities with our assumptions in this research; especially the papers investigating the single batch processing machine problem with non-identical job sizes.

Considering a single batch processing machine with non-identical job sizes, Uzsoy (1994) gave the complexity results for both makespan minimization and total flow time minimization

* Corresponding author.
E-mail addresses: n.rafieeparsa@aut.ac.ir (N. Rafiee Parsa), B.Karimi@aut.ac.ir (B. Karimi), moattarh@aut.ac.ir (S.M. Moattar Husseini).

http://dx.doi.org/10.1016/j.cie.2016.06.008
0360-8352/

for the first time. He provided some heuristics and a branch and bound algorithm and showed that these procedures provide high quality solutions. Several heuristics were proposed by Jolai and Dupont (1998) for minimizing the total flow time. Dupont and Jolai (1998) considered the same problem and developed various heuristics to minimize makespan. They showed that the performance of these heuristics is considerable. Azizoglu and Webster (2000) developed a branch and bound algorithm to minimize the total weighted flow time on a batch processing machine. Their algorithm can handle less than 25 jobs in a reasonable amount of time.

Dupont and Dhaenens-Flipo (2002) considered the problem of makespan minimization on a single BPM and proposed a branch and bound procedure. Li, Li, Wang, and Liu (2005) provided an approximation algorithm with worst-case ratio $(2+\varepsilon)$ for minimizing makespan, where jobs have different release times. Rafiee Parsa, Karimi, and Husseinzadeh Kashan (2010) presented a tight lower bound based on a column generation method and a branch and price algorithm to solve the research problem proposed by Dupont and Dhaenens-Flipo (2002). They showed that their proposed algorithm has a better performance than the branch and bound algorithm proposed by Dupont and Dhaenens-Flipo (2002).

Chen, Du, and Huang (2011) proposed a clustering algorithm based on the definition of waste ratio of batch to minimize makespan. They compared their proposed algorithm with the previous algorithms and showed that their algorithm outperforms the others. Wang (2011) addressed the problem of minimizing total weighted tardiness and proposed a two-phase heuristic. Malapert, Gueret, and Rousseau (2012) presented a constraint programming approach to minimize the maximum lateness. They decomposed the problem into finding an assignment of the jobs to the batches, and then minimizing the lateness of the batches. Lee and Lee (2013) developed construction-based heuristics and improvement-based heuristics to minimize the makespan. They showed that their suggested heuristics produce high quality solutions in most cases. Zhou, Chen, Xu, and Li (2014) considered the problem of minimizing makespan on a single batch processing machine in the presence of dynamic job arrivals. They proposed a number of constructive heuristics for this problem. Cabo, Possani, Potts, and Song (2015) introduced a new neighborhood search to minimize the maximum lateness of the jobs. Their computational experiments show the effectiveness of the proposed neighborhood search algorithm.

There are several studies focused on developing various metaheuristic algorithms for the single batch processing machine. Melouk, Damodaran, and Chang (2004) proposed a mixed integer linear programing model and a simulated annealing algorithm for a single BPM problem with non-identical job sizes to minimize the makespan. Damodaran, Kumar Manjeshwar, and Srihari (2006) developed a genetic algorithm for the same problem and compared the results with the simulated annealing approach. Kashan, Karimi, and Jolai (2006) provided two different genetic algorithms based on different encoding schemes. They showed that the performance of the proposed algorithms have a superior performance than the simulated annealing approach proposed by Melouk et al. (2004). The extension of scheduling a single BPM with non-identical job sizes to fuzzy environment is considered by Cheng, Li, and Chen (2010). They developed an improved ant colony optimization method to minimize the fuzzy makespan. Kashan, Karimi, and Jolai (2010) addressed the bi-criteria scheduling problem of minimizing the makespan and maximum tardiness. They proposed two multi-objective genetic algorithms based on different representation schemes. Xu, Chen, and Li (2012) introduced a definition of waste and idle space for the BPM problem with dynamic job arrivals to minimize the makespan. They proposed a heuristic and an ant colony optimization algorithm to solve the problem. Damodaran, Ghrayeb, and Guttikonda (2013) developed a Greedy Randomized Adaptive Search Procedure (GRASP) to minimize the makespan. They compared their proposed approach with the algorithms proposed by Melouk et al. (2004) and Damodaran et al. (2006). Jia and Leung (2014) formulated makespan minimization as a problem of minimizing the wasted space and presented an improved ant system algorithm. Al-Salamah (2015) proposed an artificial bee colony approach to minimize the makespan. Li, Chen, Xu, and Li (2015) considered the problem of minimizing earliness–tardiness on a single batch processing machine with a common due date. They proposed a hybrid genetic algorithm for this problem.

Despite the SUMCJ problem is shown to be NP-hard by Uzsoy (1994), to the best of our knowledge, no metaheuristic or approximation algorithms is developed for this problem in the literature. Thus, there are rooms for proposing different metaheuristic algorithms for the SUMCJ problem. This motivates us to focus on developing a hybrid metaheuristic algorithm based on Max–Min Ant System (MMAS) for finding near optimal solutions for the first time.

Ant colony optimization (ACO) is a population based constructive approach which has been successfully employed for several hard combinatorial optimization problems. It takes inspiration from the behavior of real ant colonies. MMAS is one of the improved variant of the ACO algorithms. The intention of this approach is to exploit the search history more effectively and prevent stagnation of the search (Stutzle & Hoos, 2000).

Stutzle and Hoos (2000) introduced MMAS to solve the traveling salesman problem and quadratic assignment problem for the first time. Later, Stutzle and Dorigo (2002) proved the convergence of MMAS algorithm theoretically. The performance of this approach has been proven in the literature for many combinatorial optimization problems, e.g., Pitakaso, Almeder, Doerner, and Hartl (2007), Wong and See (2009), Ning, Lam, and Lam (2010), Jia and Leung (2014).

In this paper, a mixed integer linear programming formulation is developed for the research problem. Then, a hybrid metaheuristic algorithm, called HMMAS, is proposed. In the proposed algorithm, first, a sequence of jobs is constructed based on the MMAS algorithm. Then, the optimal batching for the predetermined sequence of jobs is obtained through a dynamic programming algorithm. An effective local search procedure is also proposed for finding the higher quality solutions.

The rest of the paper is organized as follows: The characteristics of the problem and a mathematical formulation are described in Section 2. The details of HMMAS algorithm for obtaining near optimal solutions to the SUMCJ problem is explained in Section 3. The results of the computational experiments to evaluate the performance of the proposed metaheuristic algorithm are reported in Section 4. Finally, the conclusions and directions of future research are given in Section 5.

2. Problem description

In this section, after describing the characteristics of the problem, a binary mixed integer linear programming model is developed for the research problem. Consider a single batch processing machine with n jobs to be processed. Each job j is available at time 0 and has a processing time (p_j) and a corresponding size (s_j). The capacity of batch processing machine is C and the sum of the size of jobs in each batch cannot exceed C. Processing of a batch cannot be interrupted after it is started, and other jobs cannot be inserted into the machine until processing is completed. The processing time of a batch is given by the longest processing time among the jobs in the batch. The completion time of the jobs