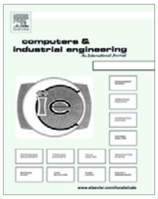
Computers & Industrial Engineering 62 (2012) 141–151

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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



Decomposition-based classified ant colony optimization algorithm for scheduling semiconductor wafer fabrication system ☆

Chengtao Guo [1], Zhibin Jiang *, Huai Zhang [1], Na Li [1]

Department of Industrial Engineering & Logistics Management, School of Mechanical Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai 200240, China

ARTICLE INFO

Article history:
Received 20 March 2010
Received in revised form 26 August 2011
Accepted 1 September 2011
Available online 14 September 2011

Keywords:
Semiconductor wafer fabrication system (SWFS)
Scheduling
Decomposition
Decomposition-based classified ACO (D-CACO)
Ant colony optimization

ABSTRACT

Due to its typical features, such as large-scale, multiple re-entrant flows, and hybrid machine types, the semiconductor wafer fabrication system (SWFS) is extremely difficult to schedule. In order to cope with this difficulty, the decomposition-based classified ant colony optimization (D-CACO) method is proposed and analyzed in this paper. The D-CACO method comprises decomposition procedure and classified ant colony optimization algorithm. In the decomposition procedure, a large and complicate scheduling problem is decomposed into several subproblems and these subproblems are scheduled in sequence. The classified ACO algorithm then groups all of the operations of the subproblems and schedules them according to machine type. To test the effect of the method, a set of simulations are conducted on a virtual fab simulation platform. The test results show that the proposed D-CACO algorithm works efficiently in scheduling SWFS.

1. Introduction

Semiconductor manufacturing process comprises four phases: wafer fabrication, wafer probe, assembly, and final testing (Uzsoy, Church, Ovacik, & Hinchman, 1992), while wafer fabrication is the most complicated, expensive and time consuming part. In semiconductor wafer fabrication system (SWFS), there are hundreds of machines working together under various constraints, and following numerous processing steps, to build multiple layers of chemical patterns on a silicon wafer (Kumar, 1994; Mason, Fowler, & Carlyle, 2002). Every layer needs to be processed in a similar manner, so wafers have to visit a certain machine for several times, each time for a layer of circuitry, and this is known as re-entrant product flow (Toktay & Uzsoy, 1998). In addition, wafer fabrication is also characterized by hybrid machine types. Several types of equipment (e.g. single wafer processing machine, single lot processing machine, and batch-type processing machine) work simultaneously in the SWFS. Due to these features, the scheduling problem of SWFS is a generalization of job shop problem that is strongly NP-hard (Garey & Johnson, 1979). This implies that the SWFS problem with the C_{max} criterion considered in this paper is strongly NP-hard as well.

In view of the complexity of SWFS, dispatching policies are commonly used for the production scheduling problems because these policies can provide approximate solutions for large problems within reasonable computation time (Pinedo, 2000). Dispatching policies usually used in SWFS were reviewed in detail by Lee, Tang, and Chan (2001). Wein (1988) studied the influence of scheduling rules on the performance of SWFS. Under these policies, the priority of jobs waiting for processing on a machine is evaluated. Once a machine is free, the job with the highest priority is picked from currently available jobs that are queuing for processing. It has been shown that a good dispatching policy may significantly improve the performance of SWFS (Kumar, 1994). However, there is no single dispatching rule that can work perfectly for all measures of performance (Holthausa & Rajendran, 1997; Uzsoy et al., 1992). Furthermore, these dispatching policies are myopic and usually cannot lead to optimal solutions as they work well on one machine but may ignore influences among different machines, thus result in poor system performance (Hung & Chang, 2002).

In order to use global shop information, some non-myopic algorithms were developed. One class of such algorithms is based on decomposition, which is used to divide large scheduling problems into smaller ones. There are two major decomposition methods, which are either time-sequence based or machine based. These methods are commonly adopted in decomposing large-scale scheduling problems into smaller ones. Rolling horizon procedures (RHPs), a kind of time-sequence decomposition method, was developed for dynamic scheduling problems (Ovacik & Uzsoy,

---

☆ This manuscript was processed by Area Editor Maged M. Dessouky.

* Corresponding author. Tel.: +86 21 34206065; fax: +86 021 34206477.

E-mail addresses: chengtao526@gmail.com (C. Guo), zbjiang@sjtu.edu.cn (Z. Jiang), zhhuai79@sjtu.edu.cn (H. Zhang), na-li03@sjtu.edu.cn (N. Li).

[1] Tel.: +86 21 34206065.

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2011.09.002

1994, 1995). According to RHPs, a scheduling problem is divided into several subproblems along time-axis. Each subproblem corresponds to a time window of the whole schedule, which is solved by branch-and-bound algorithm or other mathematical programming methods. This approach was improved and applied to schedule SWFS (Sourirajan & Uzsoy, 2007). Among the machine-based decomposition methods, one of the most successful approaches is the Shifting Bottleneck (SB) proposed by Adams, Balas, and Zawack (1988). In SB procedure, one critical unscheduled machine is identified at every iteration and scheduled by a branch-and-bound approach. This process is repeated until all machines will have been scheduled (Singer, 2001). Based on the SB heuristic, Revised Shifting Bottleneck procedure was developed for the scheduling of wafer fabrication environment (Wang, 2000). Later, another modification of the SB heuristic has been proposed in (Mason et al., 2002) for the minimization of the total weighted tardiness in a semiconductor wafer fabrication facility. Both RHPs and SB combine decomposition procedures with exact methods, such as branch-and-bound algorithm, which can obtain high-quality schedules but requires large computer memory and long computation time.

In recent years, some intelligent heuristic algorithms, called metaheuristics, were applied to scheduling problems in SWFS. Compared with traditional heuristics, these algorithms may provide better solutions using a local search procedure. Popular metaheuristics used in SWFS include tabu search method (Geiger, Kempf, & Uzsoy, 1997; Mazumdar, Mathirajan, Gopinath, & Sivakumar, 2008), simulated annealing (Chou, Wang, & Chang, 2008; Yim & Lee, 1999), and genetic algorithm (Chien & Chen, 2007; Mönch, Schabacker, Pabst, & Fowler, 2007).

The Ant Colony Optimization (ACO) algorithm (Dorigo, 1992; Dorigo & Blum, 2005; Dorigo & Stützle, 2004) is a metaheuristic algorithm and it was originally inspired by the behavior of ants. Ants are capable of finding the shortest route from a food source to their nest. They communicate via pheromone that they use in variable quantities to mark their trails. An ant's tendency to choose a certain route is positively correlated to the pheromone intensity of that trail. Since pheromone evaporates over time, its intensity decreases if no more pheromone is laid down. If many ants choose a certain route and all lay down pheromone, the pheromone intensity of this trail increases and will attract more ants. Compared with other methods used for the SWFS scheduling problem, ACO has two advantages. Firstly, ACO can effectively combine some problem-specific information with business rules during solution search. Secondly, ACO is a model-based algorithm, which is different from an instance-based genetic algorithm. The learning process of ACO is distributed in each element of the model, and normally this can lead to high-quality results.

The ACO algorithm has been used successfully in solving production scheduling problems (Dorigo & Stützle, 2004), such as job shop problem (Colorni, Dorigo, Maniezzo, & Trubian, 1994; Huang & Liao, 2008), flow shop problem (Benbouzid-Sitayeb, Ammi, Varnier, & Zerhouni, 2008; Stützle, 1998), group shop problem (Blum, 2002), and open shop problem (Blum, 2005). Recently, ACO algorithm was used to solve scheduling problems in semiconductor manufacturing system. It was used to solve scheduling problem of single batch processing machine (Kashan & Karimi, 2008) and parallel batch processing machines (Li, Qiao, & Wu, 2009) in SWFS, and was also used to solve bottleneck station scheduling problem in semiconductor assembly and test manufacturing system (Song, Zhang, Yi, Zhang, & Zheng, 2007). However, to our best knowledge, there is no previous work on the application of ACO algorithm for scheduling large-scale problem with hybrid machine types.

The metaheuristics have some advantages. They can obtain near-optimal solutions that are much better than those gained by dispatching rules, and need less computation time compared with mathematical programming. However, it is not appropriate to apply a metaheuristic method alone to the SWFS scheduling because of two reasons. First, although the computation time for the metaheuristics is not a significant factor as in mathematical programming techniques, it is still impossible to schedule large-scale complex problems within a reasonable time. Second, traditional metaheuristic approaches usually cope with one specific type of equipment well but are difficult to deal with hybrid machine types in SWFS.

In this paper, we introduce a new heuristic – decomposition-based classified ACO algorithm (D-CACO), an improved ACO algorithm that shows efficacy on SWFS scheduling. The D-CACO divides large-scale scheduling problem into several small subproblems, and then solves these subproblems in sequence using classified ACO algorithm (CACO). Different from traditional ACO algorithm, CACO algorithm can schedule various types of equipment under a single scheduling framework. On a virtual fab simulation platform, the application of D-CACO method can lead to 10–20% reduction of makespan compared to traditional dispatching rule.

The remaining sections of this paper are organized as follows. Section 2 discusses the scheduling model and disjunctive graph of SWFS. Section 3 describes the proposed decomposition-based classified ant colony optimization (D-CACO) algorithm. Two simulation models are introduced in Section 4, and D-CACO algorithm is tested by numerical experiments on simulation models in Section 5. Finally, some concluding remarks and a discussion of potential directions for future research are presented in Section 6.

2. Problem formulation

The problem under consideration can be formulated as follows. There are L jobs that need to be processed on K machines. These jobs belong to the same job family and every job is composed of J operations. Symbols i, j, k and t represent the index of a job, the sequential number of operation in a job, the index of a machine, and the time epoch, respectively. Let $C_{max} = \max\{C_1, C_2, \ldots, C_L\}$ denote the makespan of a schedule, where C_i denotes the completion time of the last operation of job i for $1 \leqslant i \leqslant L$. Semiconductor manufacturing can be described as a job shop model with re-entrant flow to find the solution that minimizes the makespan.

The above problem may be formulated as a mathematical programming problem as follows.

Parameters:

p_{ijk}: processing time of the jth operation of job i, which is processed on machine k

b_k: capacity of machine k, if k is a single processing machine, b_k is equal to 1; if k is a batch processing machine, b_k is equal to the maximum number of operations that could be processed in one batch on machine k

M: a large positive number

Decision variables:

r_{ijk}: the starting time of the jth operation of job i, which is processed on machine k

σ_{ikt}: binary variable indicating whether job i is processed on machine k at time t, and the value of σ_{ikt} is defined as follows:

$$\sigma_{ikt} = \begin{cases} 1, & \text{if job } i \text{ is processed on machine } k \text{ at time } t \\ 0, & \text{otherwise} \end{cases}$$

$\delta^k_{ij,mn}$: binary variable for the order of two operations, and its value is defined as follows: