Bottleneck Station Scheduling in Semiconductor Assembly and Test Manufacturing Using Ant Colony Optimization

Yang Song, Mike Tao Zhang, *Senior Member, IEEE*, Jingang Yi, *Member, IEEE*, Lawrence Zhang, and Li Zheng

***Abstract*—In semiconductor assembly and test manufacturing (ATM), a station normally consists of multiple machines (maybe of different types) for a certain operation step. It is critical to optimize the utilization of ATM stations for productivity improvement. In this paper, we first formulate the bottleneck station scheduling problem, and then apply ant colony optimization (ACO) to solve it metaheuristically. The ACO is a biological-inspired optimization mechanism. It incorporates each ant agent's feedback information to collaboratively search for the good solutions. We develop the ACO-based scheduling framework and provide the system parameter tuning strategy. The system implementation at an Intel chipset factory demonstrates a significant machine conversion reduction comparing to a traditional scheduling approach.**

***Note to Practitioners*—Scheduling a bottleneck station in ATM is challenging due to expensive equipment, high product mixture, and fluctuant market demands. It is desirable to maximize the equipment utilization and minimize machine conversion time of the bottleneck station. In practice, a manual scheduling method was utilized at an Intel chipset factory and the performance was not satisfactory. To this end, we develop a production scheduling system for a bottleneck station using the ACO technique. ACO is a metaheuristic optimization algorithm that adapts the biological metaphor of a colony of cooperating ants. The system takes into account business rules and prioritized objectives in ATM. The scheduling system was successfully implemented and operated in an Intel chipset factory in 2005. By leveraging the system in a production environment, we design and conduct numerical experiments to optimize the system parameter tuning strategy. Comparing to a previously existing manual approach, the ACO-based scheduling system reduced the machine conversion time of a bottleneck station by 20% and saved Intel millions of dollars.**

***Index Terms*—Ant colony optimization (ACO), bottleneck, heuristics, scheduling, semiconductor manufacturing.**

Manuscript received July 22, 2006; revised January 13, 2007. This paper was recommended for publication by Associate Editor J. Fowler and Editor N. Viswanadham upon evaluation of the reviewers' comments. This work was supported in part by Intel Higher Education Program and the National Science Foundation of China under Grant 50375082. This paper has been presented at the 2005 IEEE/SEMI International Symposium on Semiconductor Manufacturing and in the Intel Assembly and Test Technology Journal in 2005. Given the sensitive and proprietary nature of the semiconductor industry, only normalized performance data are presented in this paper.

Y. Song is with the PD2 Industrial Engineering, Intel Products (Shanghai) Company, Ltd., Pudong, Shanghai 200131, China.

M. T. Zhang was with the AzFSM (Fab 12/22/32) Industrial Engineering, Intel Corporation, Chandler, AZ 85248 USA. He is now with the Submicon Development Center, Spansion Inc., Sunnyvale, CA 94088 USA (e-mail: mike.zhang@spansion.com).

J. Yi is with the Department of Mechanical Engineering, San Diego State University, San Diego, CA 92182 USA.

L. Zhang is with the Comair Rotron (Shanghai) Fan Company, Shanghai, 201107, China.

L. Zheng is with the Department of Industrial Engineering, Tsinghua University, Beijing 100084, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2007.906341

I. Introduction

THERE ARE MAINLY two types of manufacturing processes in semiconductor manufacturing: wafer fabrication (front end) and assembly and test (back end). Assembly and testing manufacturing (ATM) process is different and difficult compared with wafer fabrication in the following aspects. 1) ATM is highly impacted by market demand fluctuation because the finished goods of ATM are shipped directly to customers. 2) Since one ATM factory normally is fed by several wafer fabrication factories, the ATM factories have a higher product-mix and technology-mix. 3) There could be multiple (legacy) machine types for each operation in ATM due to long machine life cycle.

Fig. 1 shows a typical ATM process flow for chipsets in a current manufacturing facility. In the "Wafer Reflow" operation, solder bumps on wafers are reflowed by exposing the wafers to a specified temperature in reflow ovens in order to melt the solder bumps to a rounded shape and create a protective thermal oxide layer. The wafers are cut into individual dies during "SAW" the process. The dies are then shot and stick to substrates in the "SCAM" operation. Residual flux is removed during the "DE-FLUX" process and epoxy materials are then filled into the tiny space between the die and the substrate during "EPOXY." The "PEVI" process is a visual inspection operation to screen out rejects. The "Burn-In" operation forces infant mortality failures in a high temperature environment. In the "TEST" operation, packaged units are 100% electrically tested and categorized into different bins based upon their performances. The "Laser Mark" operation is to mark the units with traceable information. In the "Ball Attach" process, solder balls are attached to the bottom of the units, the balls will serve as connection channels to external devices, such as mother boards. "FVI" and "RVSI" are also visual inspection operations to control unit damage and defects.

Many ATM factories are designed based on theory of constraints (TOC), in which the throughput of the production line is limited by the bottleneck station capacity. Therefore, we concentrate on production scheduling of the bottleneck stations, such as the SCAM process in Fig. 1. If the bottleneck shifts from one station to another due to product-mix change, we will apply the algorithm to the new bottleneck station.

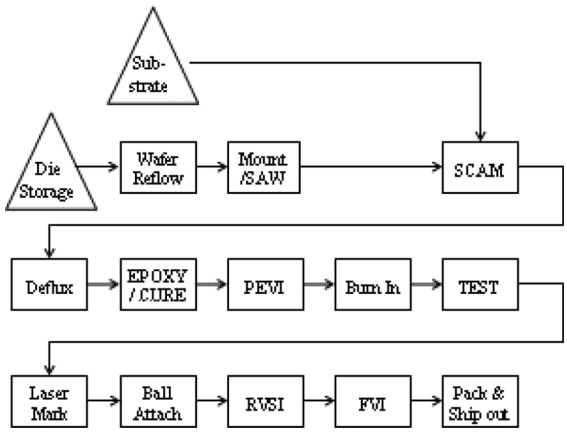


Fig. 1. Semiconductor ATM process flow.

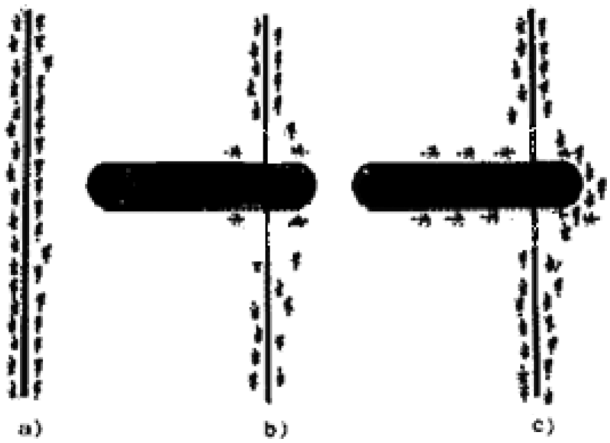


Fig. 2. Ant colony searching paths. (a) No obstacle. (b) An obstacle is interposed; ants go around it by two paths with equal probability. (c) More pheromone is laid down on the shorter path and ants are more likely to follow the shorter path with more pheromone. [1].

ACO is a metaheuristic optimization algorithm inspired by the behaviors of ants. As illustrated in Fig. 2, when ants search for the shortest path from their nest to food source, they communicate information to each other and work cooperatively. When an ant walks on a path, it deposits pheromone on the ground. The following ants smell pheromone and they tend to choose, in probability, the path with strong pheromone concentrations. Due to evaporation, the pheromone intensity reduces proportionally to the lapse of time. Thus, the longer the path, the less the pheromone, and the lower probability that the ants will follow. In this way, ants are likely to quickly find the shortest path [2].

Many heuristic algorithms have been applied to tackle the scheduling problem of bottleneck stations, but the results are not satisfactory. Comparing with those heuristics, ACO has two unique advantages. (1) ACO can effectively integrate problem-specific information and business rules during solution searching. This characteristic is vital for production scheduling problems. Classic heuristics may not be able to handle both business rules and operational constraints in real manufacturing environments. (2) ACO is a model-based algorithm, which is different from an instance-based GA algorithm [3]. The learning process of ACO is distributed in each element of the construction graph. This characteristic of ACO leads to high-quality results with high probability.

The major contribution of this paper is the documentation of a successful application of ACO in real-world semiconductor production scheduling. The success of this application yielded a 20% reduction of bottleneck machine conversion time with all customer demand supported, compared with previous heuristic scheduling method at an Intel chipset factory. Conversion time is defined as the time duration to switch products running on the machine. In this paper, we describe the modeling, development, and implementation of an ACO-based scheduling system at an Intel factory.

This paper is organized as follows. In Section II, we review the related literature on ACO and its application to production scheduling. In Section III, we describe and formulate the ATM bottleneck scheduling problem and define prioritized objectives. We present the ACO-based optimization model in Section IV, and verify our approach by numerical experiments and industrial implementation in Section V. Finally, we provide some concluding remarks in Section VI.

II. Related Work

ACO is a metaheuristic optimization method, and it was first introduced in [4]. Many variants of the ACO algorithm could be used for different combinatorial optimization problems, such as traveling salesman [4], [5], quadratic assignment [6], vehicle routing [7], job shop [8], telecommunication network, and graph coloring.

Other than the ACO algorithm, there are some classic heuristics for production scheduling problem, such as simulated annealing (SA) [9]–[11], genetic algorithm (GA) [12], [13], tabu search (TS) [14]–[16], and reinforcement learning (RL) [17], [18].

The ACO algorithm also has some successful applications in production scheduling. Colorni *et al.* [19] first introduced the mapping method of the job shop problem (JSP) to generate a construction graph, and then applied the ACO algorithm to solve JSP. Stutzle [20] improved the local search procedure and extended the ACO algorithm with max–min boundary to minimize makespan of the flow-shop scheduling problem (FSP). However, his approach only utilized pheromone information, not the problem-specific heuristic value. Bauer *et al.* [21] applied ACO to solve the single machine total tardiness problem, and used a modified due date method to calculate the problem-specific heuristic value. They also performed a pairwise swap technique for local move (path) search. Comparing with classic heuristic algorithms (such as SA), their ACO approach is more efficient in solving benchmark problems. van der Zwaan and Marques [8] introduced a statistic analysis for parameter tuning to improve the performance of ACO when solving JSP. Merkle and Middendorf [22] developed a novel pheromone summation rule of the ACO algorithm for the single machine total weighted tardiness problem. They utilized the global pheromone information instead of the local pheromone information for the transition probability calculation. Blum [23] applied ACO to the group shop scheduling problem, which is a generalization of the classic job shop and open shop scheduling problems. Rajendran and Ziegler [24] proposed and analyzed two ACO algorithms for

the permutation flow shop problem to minimize the makespan of jobs. One of them was extended from the max-min ant system in [20], and incorporated with the pheromone summation rule and a job-index-based local search technique. The other algorithm in [24] is a new development called PACO. When applying to the benchmark problems, the proposed ant algorithms are clearly superior to other heuristics.

Due to the high complexity of semiconductor manufacturing, heuristic algorithms are commonly used for production scheduling. Some of the latest research efforts in the field are as follows. Balasubramanian *et al.* [25] gave two versions of a GA to schedule batching operations in wafer fabrication to minimize the total weighted tardiness on parallel batch machines with incompatible job families. The first GA was applied to assign batches to the machines, while the second GA assigns lots to the parallel machines and form and sequence batches for each single machine. They demonstrated through simulation that the GA outperformed traditional dispatching rules. Choi and Reveliotis [26] provided an approximation framework to schedule a capacitated reentrant production line using neuron-dynamic programming (a systematic approximation schema of dynamic programming). The numeric experiments indicated that the framework was superior to traditional heuristics in term of optimality. Qu and Mason [27] studied the single machine scheduling problem considering product-mix. A meta-GA approach was presented to minimize total weighted completion time and maximize on-time delivery. Upasani *et al.* [28] proposed a heuristic to schedule a wafer fab. The algorithm decomposes the fab scheduling problem into a sequence of scheduling problems for stations by using a shifting bottleneck type algorithm. The experiments demonstrated that this station-based global scheduling heuristic outperformed typically used dispatching rules with an acceptable amount of computation time. In [29], a modified shifting bottleneck heuristic was developed for minimizing the total weighted tardiness in a fab. Most recently, Bixby *et al.* [30] demonstrated promising results in fab scheduling using mixed-integer and constraint programming, which is an optimization approach that differentiates from heuristics.

Specifically in ATM production scheduling, heuristic algorithms are also widely used. Xiong and Zhou [31] presented two hybrid heuristic algorithms based on Petri nets. The algorithms combined the heuristic best-first strategy and the controlled backtracking strategy in order to reduce the machine setup time. Freed and Leachman [32] considered the scheduling problem of the testers with multiple test heads and proposed an enumeration framework. Ellis *et al.* [33] considered the sequence dependency of machine setup times. They proposed four heuristics to solve the test scheduling problem that aims to minimize the makespan. Lin *et al.* [34] applied the theory of constraint to develop the heuristic algorithm with capacity constrains. They treated ATM facility as unrelated parallel machines with multiple constraints.

Several heuristics have been tested at the Intel ATM factory for bottleneck station scheduling. None of them showed promising results until an ACO-based system was successfully developed and implemented in 2005 [35]. In this paper, we describe in detail the mathematical modeling and implementation of the system.

III. Problem Statement

In this section, we describe the scheduling problem we consider and formulate the problem with assumptions, input, and output.

Production scheduling aims to optimally allocate the customer demands to available machines while satisfying business rules. Due to highly fluctuant market demands, frequent machine conversions are required in ATM. It may take different amount of times to convert different machines for different products. If not well scheduled, the time for machine conversions will greatly increase and, thus, idle time of the bottleneck station will be longer. According to the TOC theory, the capacity loss of the bottleneck is the capacity loss of the whole factory. Therefore, a good bottleneck station scheduling is very important for ATM to support all the customer demands and minimize the machine conversion frequency and the total conversion time.

We define the machine throughput (or runrate) as the volume that a machine could produce in one week, the machine conversion time as the time duration to switch products processed by the machine, and "time interval" as the smallest granularity of time we investigate. Without loss of generality, we assume the following.

- The machine throughput (or runrate) considers the average of rework, downtime, and facility maintenances.
- The length of the machine conversion (setup) time depends only on the machine types and the product types.
- Each lot needs only to be processed on one machine of the station and each machine should only process at most one lot at a time. We do not consider specific lot size, lot split, and lot merge.
- Once a process on a machine is started, it cannot be interrupted until the process is finished.
- We assume the time interval t_q for scheduling is 2 h, which is based upon computing time requirement, operations practice, and historical machine conversion performance. Machines must produce the same type of products within the interval and conversion could only happen at the beginning/end of the intervals.

The inputs of the bottleneck station scheduling problem are:

P_i	product i;
M_k	machine k;
$P = \{P_i, 1 \le i \le l\}$	set of all products;
$M = \{M_k, 1 \le k \le m\}$	set of all machines;
W_1	current week;
W_2	next week;
D_{1i}	demand of P_i in W_1;
D_{2i}	demand of P_i in W_2;
$R_{i,k}$	runrate of P_i on M_k;
$CT_{i,j,k}$	conversion time of M_k from P_i to P_j.

TABLE I
A GANTT CHART OF THE SCHEDULING RESULTS

	t_0	t_1	t_2	t_3	t_4
M_1	P_1	P_1	P_1	P_2	P_2
M_2	P_2	P_2	P_2	P_2	P_2
M_3	P_3	P_3	P_4	P_4	P_4

Other possible inputs also include product priority, yield, machine initial condition, setup time, Work-In-Process (WIP) inventory level, manufacturing starting time and due date, and machine scheduled downtime plans.

The optimization problem normally consists of an objective function and constraint conditions. We present the objective function below and the constraint conditions (i.e., the business rules) in Section IV.

Let $A(t,k)$ denote the type of product on machine M_k at time interval t. The scheduling result of decision variable $A(t,k)$'s is showed in a Gantt chart (see Table I), which represents the product type allocated on every machine in different time intervals. A machine conversion happens if the product type switches between two sequential time intervals.

The objective function is to minimize the weighted summation of the three objectives

$$\min \sum_i f_i \cdot O_i (i = 1, 2, 3)$$

where f_i is the weight of objective O_i.

Objective O_1 aims to minimize total unsupported customer demands for the current and next weeks, where

$$O_1 = \sum_{i \in P} \left\{ D_{1i} - \sum_{t \in W_1} \sum_{k \in M} R_{i,k} \cdot X(i,k,t) \right\} + \sum_{i \in P} \left\{ D_{2i} - \sum_{t \in W_2} \sum_{k \in M} R_{i,k} \cdot X(i,k,t) \right\} \text{ and}$$

$$X(i,k,t) = \begin{cases} 1, & A(t,k) = i \\ 0, & A(t,k) \neq i \end{cases}.$$

Objective O_2 aims to minimize the total number of conversions, where

$$O_2 = \sum_{k \in M} \sum_t O(A(t,k), A(t-1,k)) \text{ and}$$

$$O(x,y) = \begin{cases} 1, & x \neq y \\ 0, & x = y \end{cases}.$$

The objective O_3 aims to minimize the total conversion time, where

$$O_3 = \sum_{k \in M} \sum_t O(A(t,k), A(t-1,k)) \cdot CT_{A(t,k),A(t-1,k),k}.$$

The larger f_i, the more important O_i. Normally, we should keep at least the ratio of 10 between f_i's. In this objective function, we set $f_1 = 1000$, $f_2 = 100$, and $f_3 = 10$. Many feasible solutions (in the format of Table I) may exist for a bottleneck station scheduling problem. We intend to identify the best solution measured by the objective function.

IV. ACO MODELING

In this section, we explain how we apply the ACO algorithm to solve the bottleneck scheduling problem with the objective function and the business rules.

We first create a construction graph for the scheduling problem defined in Section III. As illustrated in Fig. 3, the graph is a unidirected multipartite network. Nodes represent machine statuses in different time intervals. Edges are weighted based upon the constraints. In this paper, the construction graph is used to solve an unrelated parallel machines problem for the bottleneck scheduling. If we consider the flow shop problem including up- and downstream stations, the graph can be extended by adding one more index of stations in each node.

Every artificial ant of a colony performs probabilistic walks one by one in the graph. Every ant creates a solution through its walking path from the origin (Node O) to the destination (Node D). One "cycle" is completed if all ants finish their search. For each cycle, one solution (as illustrated in Table I) is generated. We measure the solution by the objective function described in Section III. If the value of the objective function for this solution is smaller than that for any existing solution, we keep this best-so-far solution. If not, we remove this solution. We then update the construction graph by resetting the transition probability of all the edges proportional to the goodness of the solution. We repeat the ant search for the next cycle until we satisfy with the best-so-far solution or the time limit is reached.

A. Construction Graph

The most important step of the ACO algorithm is to create a construction graph and build mapping relationships between the graph and the scheduling problem. These relationships include the definition of the nodes in terms of the variables of the scheduling problem.

Fig. 3 illustrates a unidirected multipartite network clustered by time intervals. The scheduling time horizon starts at time interval t_0 and ends at t_n. In the graph, node $\{P_i, M_k, t_q\}$ represents machine M_k processes product P_i in time interval t_q, where $1 \leq q \leq n$. For example, node $\{P_1, M_2, t_0\}$ represents machine M_2 processes product P_1 in time interval t_0. The nodes belonging to the same machines are connected by the unidirectional edges, which imply a machine can be converted between time intervals to product different products. Notice that there are no connecting edges between different machines because a machine cannot be changed to a different machine any time. Each edge is weighted by a positive real number between zero and one. The weight determines the probability ants walk from one node to another.

When the scheduling process starts, first we have to make decision on what to produce on each machine at time interval t_0. In the graph, that is equivalent to the scenario where one ant leaves node O and travels to node $\{P_i, M_k, t_0\}$. The choice is based on the transition probability and the transition policy that we will elaborate in Sections IV-B and C. For example, in Fig. 3, we assume machine M_1 can only produce product P_1 and P_2, and M_2 can only produce P_2 and P_3. In time interval t_0, there are two options for M_1 and two options for M_2, and thus the total combination of $2 \times 2 = 4$ options. Based upon the transition probability and the transition policy, the ant selects

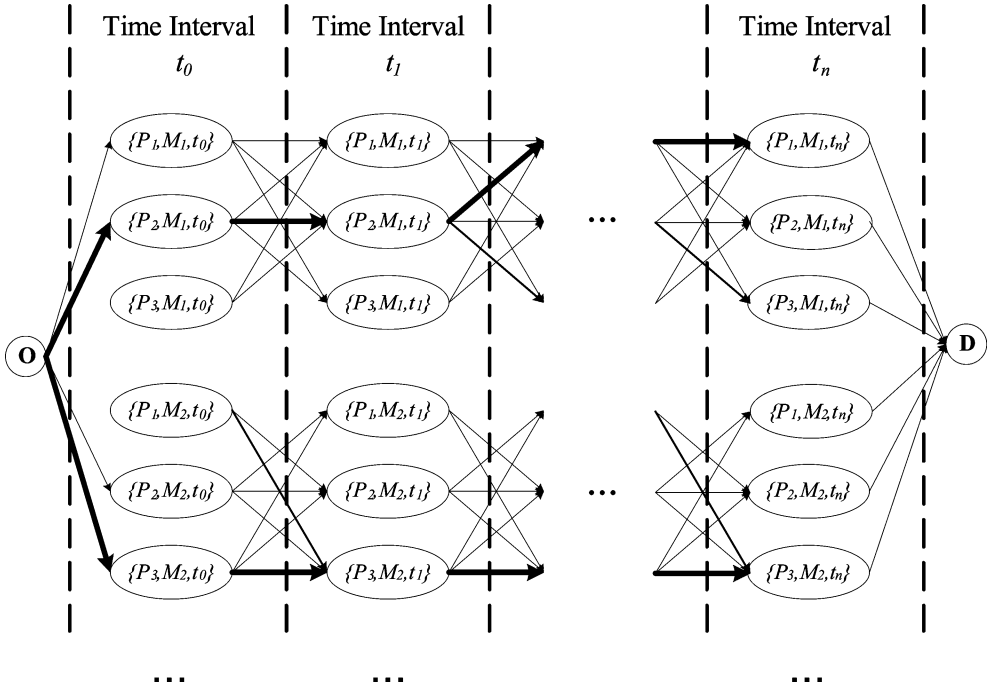


Fig. 3. Construction graph for the scheduling problem.

$\{P_2, M_1, t_0\}$ and $\{P_3, M_2, t_0\}$ for time interval t_0, and then move to time interval t_1. Once the ant arrives at node D, its walking path from O to D generates a scheduling solution as shown in Table I.

B. Transition Probability

When the ant moves from one node of time interval t_q to the next node cluster of time interval t_{q+1} in the construction graph, there may exist multiple connecting edges (which lead to the feasible nodes). Which edge to select among all these options?

The transition probability determinates the probability of each edge that the ant will walk on. It can be calculated by

$$\Pr_{i,k,t} = \frac{\tau_{i,k,t}^{\alpha} \cdot \eta_{i,k,t}^{\beta}}{\sum_{j \in P} \tau_{i,k,t}^{\alpha} \cdot \eta_{i,k,t}^{\beta}} \tag{1}$$

where $\tau_{i,k,t}$ is the pheromone value that corresponds to machine k producing product i in time interval t (see Section IV-D for details). $\eta_{i,k,t}$ is the corresponding heuristic value. Determined by the business rules, the heuristic value has significant impacts on the feasibility of solutions.

The exponential parameters α and β in (1) tune the relative importance in the pheromone value and the heuristic value. If $\beta = 0$, the ants do not utilize the heuristic value in solution generating, which will lead to rapid emergence of a local minimum. If $\alpha = 0$, on the other hand, the ants do not communicate through pheromone and they perform a classical stochastic greedy algorithm only based upon the business rules. Therefore, a tradeoff between the pheromone intensity and the heuristic value appears to be necessary and can be achieved by tuning α and β. We will discuss the parameter tuning strategy in Section V.

In the rest of this section, we will discuss in detail the business rules that determine the heuristic value.

We categorize all the business rules into "hard constraint rules" and "soft constraint rules." A hard rule is the rule that must be satisfied; otherwise, the solution will be infeasible. If a move of the ant in the construction graph violates any hard rule, the heuristic value of the associated edge will be set to zero. Hence, the transition probability for that edge will be zero. However, a soft rule does not need to be strictly enforced. It can be violated with a penalty. If the ant move only violates a soft rule, the ACO algorithm will decrease the heuristic value to a certain degree that is predefined empirically.

Let us use an example to explain how the business rules determine the heuristic value. Fig. 4 presents a detailed flow chart of a soft constraint rule called "loading balance for different products."

As the production goes on, some customer demands have already been fulfilled, while some others still have not. Thus, we may have to convert machines to ensure on time completion of all the products.

First, the ideal number of machines needed for each product *IdealNum* is calculated based on the ratio of the customer demands and the machine runrates. Next, we compare *IdealNum* to the actual allocated machine *ActualNum* and generate set variables *DevOverMachine* (a set of products with excessive machine capacity and high probability of demand fulfillment) and *DevLackMachine* (a set of products with insufficient machine capacity and low probability of demand fulfillment). If both *DevOverMachine* and *DevLackMachine* are not empty, we can generate *the releasable machine list* and reallocate the machine resources that produce the products in *DevOverMachine* to the products in *DevLackMachine* by updating the heuristic values of the associated edges.

For example, in a certain time interval t, we have product P_1 and P_2 in *DevOverMachine* and P_3 in *DevLackMachine*. P_1 can release one machine M_1, P_2 can release two machines M_2 and

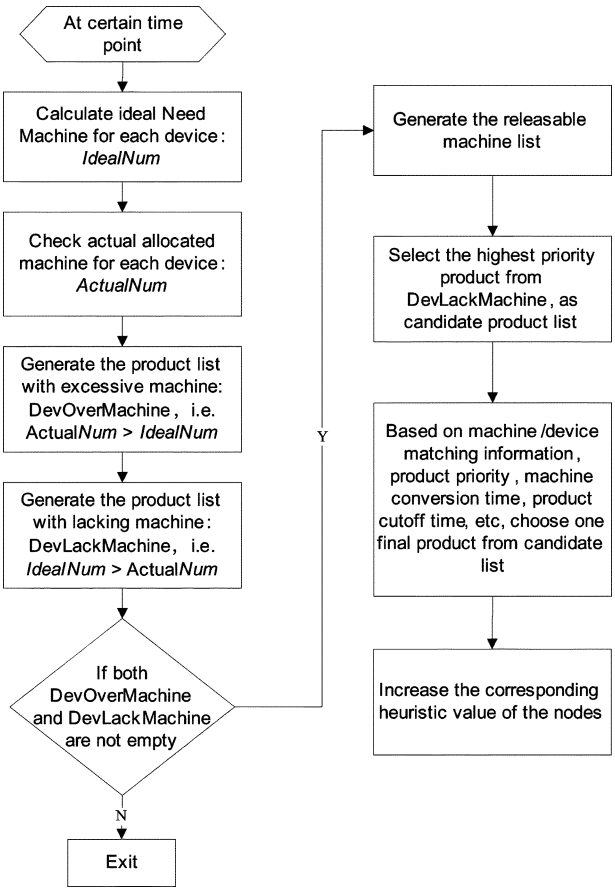


Fig. 4. A business rule that impacts the transition probability: "loading balance for different products."

M_3, and P_3 requires one more machine to fulfill the demand. So, *the releasable machine list* is $\{M_1, M_2, M_3\}$. The preference rate is used to describe the preference level for a certain product to be processed on a certain machine. It is predetermined empirically. The larger the rate, the higher the preference level. Assume the preference rates for P_3 are: 0.5, 1, and 0.3 for M_1, M_2, and M_3, respectively. Also assume the machine conversion times are 8 h from P_1 to P_3 on M_1, 2 h from P_2 to P_3 on M_2, and 4 h from P_2 to P_3 on M_3. We update the heuristic values as follows:

$$\eta_{3,1,t+1} = \left(1 + \mathrm{PreferenceRate} + \frac{1}{\mathrm{ConversionTime}}\right) \cdot \eta_{3,1,t} = \left(1 + 0.5 + \frac{1}{8}\right) \cdot \eta_{3,1,t} = 1.625 \cdot \eta_{3,1,t}$$

$$\eta_{3,2,t+1} = \left(1 + 1 + \frac{1}{2}\right) \cdot \eta_{3,2,t} = 2.5 \cdot \eta_{3,2,t}$$

$$\eta_{3,3,t+1} = \left(1 + 0.3 + \frac{1}{4}\right) \cdot \eta_{3,3,t} = 1.55 \cdot \eta_{3,3,t}.$$

If other parameters remain unchanged, the transition probability for machine M_2, $\mathrm{Pr}_{3,2,t+1}$, will be larger than those of other two machines M_1 and M_3. Therefore, machine M_2 will be most likely have to be converted to product P_3 in time interval $t+1$.

For another example, we consider a hard constraint rule called "machine downtime restriction rule." If the machine M_2 is down to production in time interval t_2, we will set the heuristic value $\eta_{i,2,2}$ to zero for any product ($\eta_{i,2,2} = 0, \forall P_i \in P$). Therefore, the transition probability $\mathrm{Pr}_{i,2,2}$ is zero, which implies that no product can be processed on M_2 in t_2.

We also consider the following ATM business rules as hard or soft constraints in our math model and system.

- Product cutoff time hard constraint: The production tasks must be finished before the corresponding cutoff time.
- Machine-product matching hard constraint: Some types of products can only be processed on certain types of machines.
- Machine-product matching soft constraint: Some types of products are preferred to process on certain types of machines.
- Optimized machine conversion time soft constraint: A conversion with a shorter conversion time is preferred if a conversion occur.
- WIP balance soft constraint: The WIP of each station has a preferred level.
- Machine experiment time hard constraint: If a machine is in use for experiments, it cannot be used for production.
- Week-to-week machine loading smoothness soft constraint: The machine loading is balanced among different weeks of the scheduling time horizon.
- Week-to-week product loading smoothness soft constraint: The product loading is balanced among different weeks of the scheduling time horizon.

C. Transition Policy

A transition policy, which has been proposed by Dorigo *et al.* [4], is also applied to balance the decision between keeping the best-known solution and exploring additional region of the solution space. Let q_0 be a configurable parameter within $[0, 1]$. When an ant makes a move, it will generate a random number q (which follows a uniform distribution over $[0, 1]$) and compare q to q_0. If $q \leq q_0$, the ant will move though the edge with the best-known value of $\tau_{i,k,t}^{\alpha} \cdot \eta_{i,k,t}^{\beta}$; otherwise, the ant will obey the transition probability to select the next node. Tuning q_0 allows the ACO algorithm to choose whether to concentrate on the local area around the best-known solution or to explore new search space. The use of the transition policy increases the algorithm convergence speed significantly.

D. Pheromone Update

Due to evaporation, the pheromone intensity would decrease. Normally, the evaporation quantity is proportional to the lapse of time. Therefore, older pheromone (information) has less influence on the future transitions. When all ants have completed constructing a solution, the pheromone values of all the edges should be modified for the next solution construction cycle based upon the following equation:

$$\tau_{i,k,t+1} = (1 - \rho)\tau_{i,k,t} + \rho \cdot \tau'_{i,k,t+1} \tag{2}$$

where $\rho \in (0, 1)$ is the evaporation coefficient and $\tau'_{i,k,t+1}$ is the pheromone value that corresponds to the best solution so far.

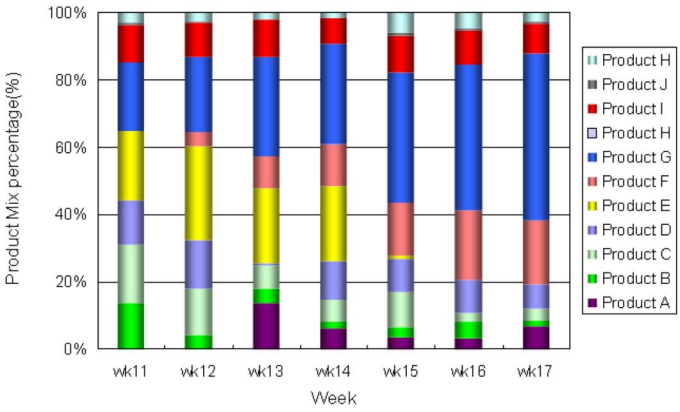


Fig. 5. Product-mix weekly changes of the Intel chipset factory.

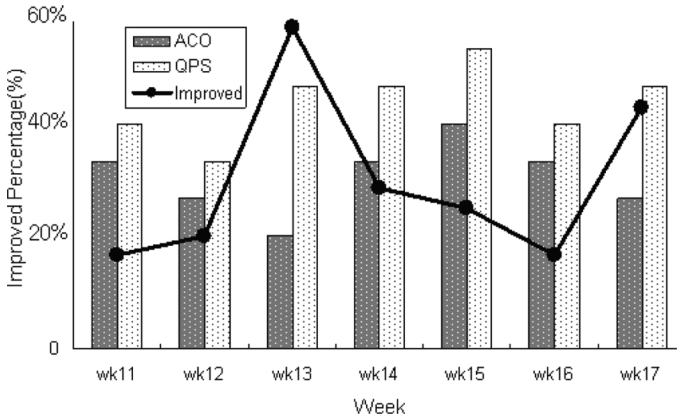


Fig. 6. Reduction of the numbers of machine conversions. The line shows the percentage improvement from the QPS method to the ACO system.

The pheromone update consists of two actions [8]. First, a fraction of pheromone on each edge is evaporated so that old pheromone would not have too strong influence on the future ant moves. This action prevents searching in the neighborhood of a local minimum only. Second, an increment of pheromone is given to those edges belonging to the best-so-far solution found by the ant colony. The increment is proportional to the solution quality. For example, if we found the best-known production schedule, we will largely increase the pheromone values of the edges on this path. The global pheromone update will speed up the searching process for the best solution.

V. Implementation and Results

We successfully implemented the ACO-based production scheduling system at an Intel chipset ATM factory. We only focus on the bottleneck station scheduling in this paper.

In the past, the bottleneck station production schedule was carried out manually by quantity per shift (QPS) method: first, we assume a fixed batch size and calculate the QPS for every product according to their weekly demands; second, we allocate capacity for products with large demands based on their QPS, and then allocate the rest of capacity for other products. Unfortunately, the QPS method has serious disadvantages including: 1) high risk of human errors and inconsistency; 2) short time horizon coverage; and 3) nonoptimal solution.

We successfully implemented the ACO-based bottleneck station scheduling system in Intel chipset ATM factory in 2005. Comparing with the QPS method, the ACO-based system demonstrated remarkable improvement. We studied the following case from the Intel chipset factory during week 10–17 in 2005. Fig. 5 shows the product-mix of the factory during that period of time. There were six different products with weekly volume change over 50%.

We utilized the ACO system and the QPS method to generate the production schedule for SCAM (which is a bottleneck station) separately to compare their results. There were no unsupported customer demands in both results, which clearly minimize the value of objective 1. Fig. 6 illustrates an over 20% reduction (most of the weeks) of the number of machine conversions for the bottleneck station. Similar reduction in machine conversion time was also achieved. As the results, the capacity loss decreased and the product unit cost reduced significantly.

The parameter values in the ACO algorithm are critical for computing time and solution quality. Therefore, we also designed and conducted numerical experiments to study system parameter tuning strategy using the production system we developed.

We considered the parameter space by two independent subspaces [8]: 1) the transition probability (α and β) and 2) the pheromone update (the total number of ants *AntNum* and the evaporation coefficient ρ). We studied these two independent subspaces separately.

To facilitate the numerical experiments, we defined a benchmarking problem based upon the real production information of the Intel chipset factory at week 34 in 2005. The best solution of the problem is with the values of three objectives in Section III at 0, 17, and 88, respectively.

In order to study the influence of the parameter α and β, a two-factor analysis was carried out to the benchmark scheduling problem. For each pair of (α, β), we ran the ACO algorithm five times and recorded the number of cycles it took to converge, i.e., reach the best solution. Given the fixed parameter $\rho = 0.1$ and $AntNum = 10$, the experimental results are shown in Fig. 7 and Table II. Normally, the system computing time is proportional to the number of the cycles to converge. Hence, the number of the cycles is a good indicator to measure the system performance.

In the case of $\beta = 0$, the ant colony does not utilize the problem-specific heuristic value during the solution searching, and every ant is only guided by the pheromone intensity. So, the algorithm is expected to take a long time to find the best solution (> 1000 cycles). If $\alpha = 0$, the ants do not communicate through pheromone and only perform a simple stochastic greedy algorithm. In this case, the algorithm may converge to a local (not global) optimal solution. For most of the other (α, β) pairs, the best solution can be found in less than 1000 cycles. Among the (α, β) pairs we tested, the ACO algorithm reaches the fastest convergence of 206 cycles when $\alpha = 4$ and $\beta = 5$. Since ACO is a heuristic algorithm, we cannot claim this pair of (α, β) is the optimal but it should be a near-optimal setting.

In order to understand the influence of the pheromone update, we used the benchmarking problem to study the total number of ants *AntNum* and the evaporation coefficient ρ separately.

The convergence performance of the ACO algorithm can be roughly defined by the total computing cycles, which is the product of *AntNum* and the average number of cycles. Obvi-

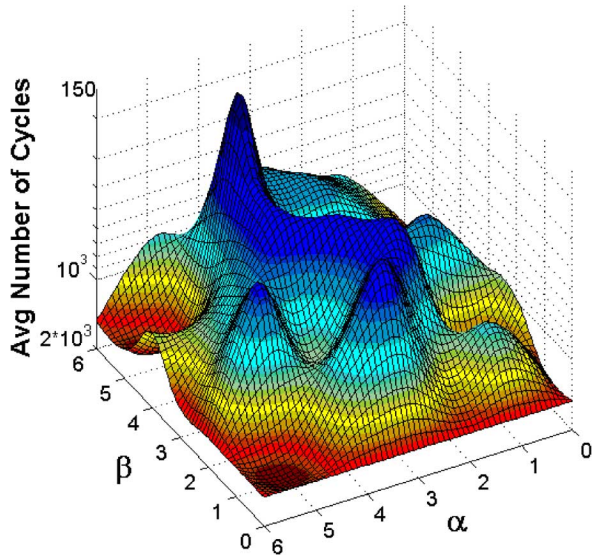


Fig. 7. Convergence performance: average number of cycles in need for the ACO algorithm to obtain the best solution for different pairs of (α, β).

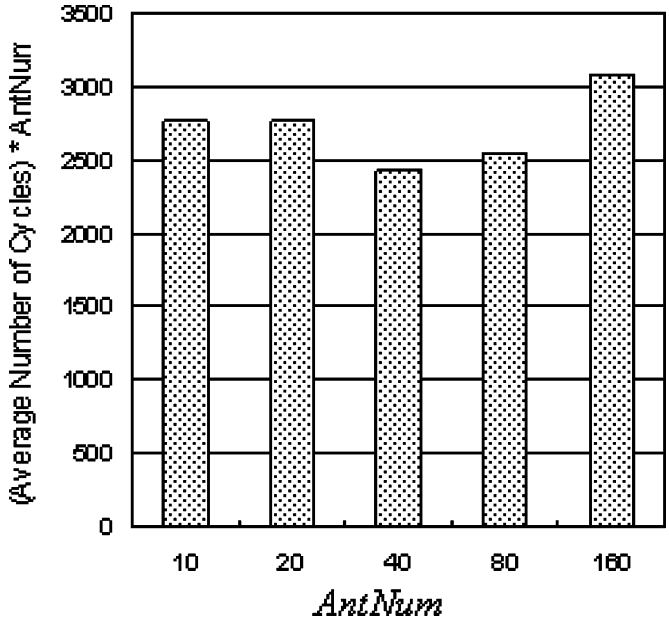


Fig. 8. The number of ants on ACO algorithm convergence performance.

TABLE II
CONVERGENCE PERFORMANCE

β	α=0	α=1	α=2	α=3	α=4	α=5	α=6
0	>1000	>1000	>1000	>1000	>1000	>1000	>1000
1	>1000	785	998	390	732	>1000	>1000
2	991	>1000	423	458	931	450	>1000
3	980	541	536	351	401	690	>1000
4	>1000	804	873	831	297	922	942
5	>1000	897	574	570	206	>1000	>1000
6	>1000	832	688	546	862	803	>1000

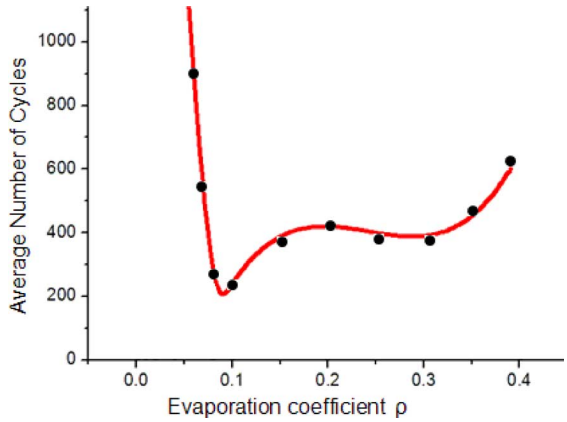


Fig. 9. Impacts of evaporation coefficient on ACO algorithm convergence performance.

ously, more ants may find the best solution in fewer cycles. Given a fixed set of α, β, and ρ values, we studied the influence of *AntNum* on the ACO algorithm convergence performance. Fig. 8 shows the impacts of *AntNum* with $\alpha = 4$, $\beta = 5$, and $\rho = 0.1$ over five system runs. We can clearly see that *AntNum* has a small impact on the algorithm performance and the best value is $AntNum = 40$.

Fig. 9 shows the performance of the ACO algorithm for different values of the evaporation coefficient with the parameter set $\alpha = 4$, $\beta = 5$, and $AntNum = 40$. If $\rho < 0.05$, the algorithm cannot converge to the best solution within 1000 cycles; if $0.1 < \rho < 0.4$, the number of the cycles to converge increase rapidly; and the best evaporation coefficient value is around $\rho = 0.1$.

Beside the numerical experiments above, we also study the relationship between the system convergence performance and the product-mix, the number of machines, and the duration of the machine conversion time. We summarize the findings as follows.

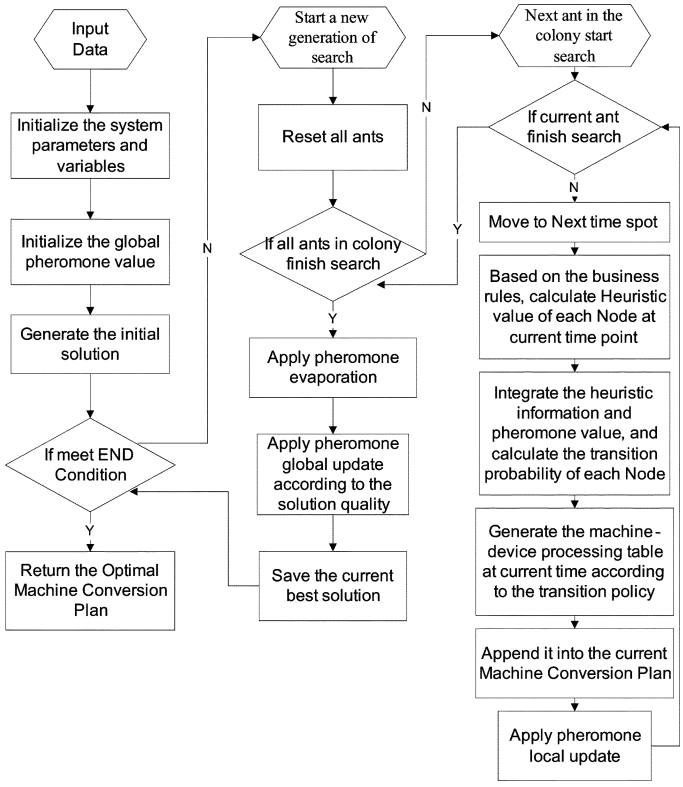
1) The metaheuristic approach demonstrated better convergence performance than single-thread heuristics. From Fig. 7 and Table II, we can see that the system convergence performance degrades whenever α or β is set to zero. The best performance is achieved through the proper balance of the pheromone value and the heuristic value.
2) The algorithm is not very sensitive to the number of ants.
3) There is a lower bound (or threshold) for the evaporation coefficient. That implies that the algorithm may not converge if we do not take advantage of good solutions available. On the other hand, too large evaporation coefficients can slow down the improvement of solution quality for each cycle.
4) When the product-mix or the number of machines increases, the conversion reduction over QPS is continuously improved. However, the system computing time increases exponentially.
5) When the machine conversion time increases, the conversion reduction over QPS is first improved and then downgraded. There is a machine conversion time that maximizes the benefit of the ACO system. Meanwhile, the system computing time keeps flat with small variation.

VI. CONCLUSION

In this paper, we formulated the bottleneck station scheduling in semiconductor ATM into an optimization problem. The ob-

jective is to minimize the total unsupported demands and machine conversion time. The constraints are defined by the ATM business rules. The optimization problem was mapped to a unidirected multipartite network and solved using an ACO technique. ACO is a bioinspired metaheuristic algorithm inspired by the communication behaviors of ants. We also designed and conducted numerical experiments to tune the system parameters using a real-world benchmarking problem. The ACO-based scheduling system was successfully implemented and verified in an Intel ATM factory. Comparing to the previous QPS method and other heuristic approaches, we experienced over 20% of average machine conversion time reduction with the better order-fulfillment performance. The proposed ACO-based scheduling system demonstrated promising results in solving the single-station scheduling problem. In the future, we would like to solve the factory-level scheduling problem by the integration of decomposition and the ACO algorithm.

APPENDIX
FLOW CHART FOR ACO SCHEDULING ALGORITHMS



ACKNOWLEDGMENT

The authors would like to thank the Editors and the anonymous reviewers for their constructive feedback. They also thank Dr. S. Ding at the University of California, Berkeley, R. Akhavan-tabatabaei at North Carolina State University, and K. Bryant at Intel Corporation for their helpful comments. Thanks also go to V. Zhu, Y. Chen, and E. McBride at Intel Products (Shanghai) Company, Ltd., for their support.

REFERENCES

[1] W. Wen and W. Du, "An abstract on the ant colony algorithms," *Automat. in Petro-Chemical Industry*, pp. 1–19, 2002.
[2] M. Dorigo and G. D. Caro, "Ant algorithms for discrete optimization," *Artif. Life*, vol. 5, no. 2, pp. 137–172, 1999.
[3] J. Quinlan, "Combining instance-based and model-based learning," in *Proc. 10th Int. Conf. Mach. Learn.*, San Mateo, CA, 1993, pp. 236–243.
[4] M. Dorigo, V. Maniezzo, and A. Colorni, The Ant System: An Autocatalytic Optimizing Process Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, Tech. Rep. 91–016 Revised, 1991.
[5] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern.—Part B*, vol. 26, no. 1, pp. 29–41, 1996.
[6] V. Maniezzo and A. Colorni, "The ant system applied to the quadratic assignment problem," *IEEE Trans. Knowln Data Eng.*, vol. 11, pp. 769–778, 1999.
[7] B. Bullnheimer, R. F. Hartl, and C. Strauss, "An improved ant system algorithm for the vehicle routing problem," *Annn Oper. Res.*, vol. 89, pp. 319–328, 1999.
[8] S. van der Zwaan and C. Marques, "Ant colony optimisation for job shop scheduling," in *Proc. 3rd Workshop Genetic Algorithms Artif. Life (GAAL'99)*, 1999.
[9] J. Mittenthal, R. Madabhushi, and I. R. Arif, "A hybrid simulated annealing approach of single machine scheduling problems with nonregular penalty functions," *Comput. Oper. Res.*, vol. 20, no. 2, pp. 103–111, 1993.
[10] H. Ishbuchi, S. Misaka, and H. Tanaka, "Modified simulated annealing algorithms for the flow shop sequencing problem," *Eur. J. Oper. Res.*, vol. 81, no. 2, pp. 388–398, 1995.
[11] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 113, no. 1, pp. 123–126, 1998.
[12] C. W. Holsapple, V. S. Jacob, R. Pakath, and J. S. Zaveri, "A genetics based hybrid scheduler for generating static schedules in flexible manufacturing contexts," *IEEE Trans. Syst., Man, Cybern.—Part B*, vol. 23, no. 4, pp. 953–972, 1993.
[13] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 983–997, 1996.
[14] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *Eur. J. Oper. Res.*, vol. 47, no. 1, pp. 65–74, 1990.
[15] M. Laguna, J. W. Barnes, and F. W. Glover, "Intelligent scheduling with tabu search: An application to jobs with linear delay penalties and sequence-dependent setup costs and times," *Appl. Intell.*, vol. 3, no. 2, pp. 159–172, 1993.
[16] M. Laguna, J. W. Barnes, and F. W. Glover, "Tabu search methods for a single machine scheduling problem," *J. Intell. Manufact.*, vol. 2, no. 2, pp. 63–73, 1991.
[17] W. Brauer and G. Weiss, "Multi-machine scheduling: A multi-agent learning approach," in *Proc. Int. Conf. Multi-Agent Syst.*, 1998, pp. 42–48.
[18] M. E. Aydin and E. Öztemel, "Dynamic job-shop scheduling using reinforcement learning agents," *Robot. Autonomous Syst.*, vol. 33, no. 2, pp. 169–178, 2000.
[19] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job-shop scheduling," *Belgian J. Oper. Res., Statist. Comput. Sci. (JORBEL)*, vol. 34, no. 1, pp. 39–53, 1994.
[20] T. Stutzle, "An ant approach to the flow shop problem," in *Proc. Eur. Congress Intelligent Tech. Soft Comput.*, Aachen, Germany, 1998, pp. 1560–1564.
[21] A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss, "An ant colony optimization approach for the single machine total tardiness problem," in *Proc. 1999 Congr. Evol. Comput.*, 1999, pp. 1445–1450.
[22] D. Merkle and M. Middendorf, "Ant colony optimization with global pheromone evaluation for scheduling a single machine," *Appl. Intell.*, vol. 18, pp. 105–111, 2003.
[23] C. Blum, "ACO applied to group shop scheduling: A case study on intensification and diversification," in *ANTS 2002*, M. Dorigo, Ed. *et al.* Berlin, Germany: Springer-Verlag, 2002, pp. 14–27.
[24] C. Rajendran and H. Ziegler, "Ant colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *Eur. J. Oper. Res.*, vol. 155, pp. 426–438, 2004.
[25] H. Balasubramanian, L. Monch, J. W. Fowler, and M. Pfund, "Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness," *Int. J. Prod. Res.*, vol. 42, no. 8, pp. 1621–1638, 2004.
[26] J. Y. Choi and S. Reveliotis, "Relative value function approximation for the capacitated re-entrant line scheduling problem," *IEEE Trans. Automat. Sci. Eng.*, vol. 2, no. 3, pp. 285–299, 2005.

[27] P. Qu and S. Mason, "Metaheuristic scheduling of 300-mm lots containing multiple orders," *IEEE Trans. Semiconduct. Manufact.*, vol. 18, pp. 633–643, 2005.
[28] A. A. Upasani, R. Uzsoy, and K. Sourirajan, "A problem reduction approach for scheduling semiconductor wafer fabrication facilities," *IEEE Trans. Semiconduct. Manufact.*, vol. 19, no. 2, pp. 216–225, May 2006.
[29] S. J. Mason, J. W. Fowler, and W. M. Carlyle, "A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops," *J. Scheduling*, vol. 5, pp. 247–262, 2002.
[30] R. Bixby, R. Burda, and D. Miller, "Short-interval detailed production scheduling in 300 mm semiconductor manufacturing using mixed integer and constraint programming," in *Proc. IEEE/SEMI Adv. Semiconduct. Manufact. Conf.*, Boston, MA, 2006, pp. 148–154.
[31] H. H. Xiong and M. C. Zhou, "Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search," *IEEE Trans. Semiconduct. Manufact.*, vol. 11, pp. 384–393, 1998.
[32] T. Freed and R. C. Leachman, "Scheduling semiconductor device test operations on multihead testers," *IEEE Trans. Semiconduct. Manufact.*, vol. 12, no. 4, pp. 523–530, Nov. 1999.
[33] K. P. Ellis, Y. Lu, and E. K. Bish, "Scheduling of wafer test processes in semiconductor manufacturing," *Int. J. Prod. Res.*, vol. 42, no. 2, pp. 215–242, 2004.
[34] J. T. Lin, F. K. Wang, and W. T. Lee, "Capacity-constrained scheduling for a logic IC final test facility," *Int. J. Prod. Res.*, vol. 42, no. 1, pp. 79–99, 2004.
[35] Y. Song, M. T. Zhang, L. Zhang, and L. Zheng, "ACO algorithm for machine conversion reduction in semiconductor assembly manufacturing," in *Proc. IEEE/SEMI Int. Symp. Semiconduct. Manufact.*, San Jose, CA, Sep. 2005, pp. 339–343.

Yang Song received the M.S. degree from the Department of Industrial Engineering, Tsinghua University, Beijing, China, in 2002.

He is currently a Senior Industrial Engineer with Intel Shanghai. His research interests are manufacturing science, lean, six sigma, and operations research.

Mike Tao Zhang (S'98–M'01–SM'05) received the M.S. and Ph.D. degrees from the Department of Industrial Engineering and Operations Research, in 2000 and 2001, respectively, as well as the Management of Technology Certificate in 2000 from the Haas School of Business and the College of Engineering, University of California, Berkeley.

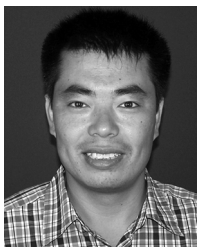
He is currently a Senior Manager of Systems Automation and Industrial Engineering at Spansion Inc., Sunnyvale, CA. He has been a Senior Engineer, a Group Leader, a Department Manager, and a Staff Engineer at various Intel sites. He was awarded three patents and published over 50 papers and four books/book chapters. His research interests are industrial automation, manufacturing systems, operations research/management, and supply chain management.

Dr. Zhang is a Member of the Honor Society of Phi Kappa Phi, and also a Senior Member of the Institute of Industrial Engineers (IIE). He is Co-Chair of the IEEE Robotics and Automation Society Technical Committee on Semiconductor Manufacturing Automation. He is an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING and a Guest Editor of *Assembly Automation* and the IEEE ROBOTICS AND AUTOMATION MAGAZINE. He is Program Chair of the 2007 IEEE Conference on Automation Science and Engineering. He is also the recipient of the Intel ATM Achievement Award and the IIE Outstanding Young Industrial Engineer Award. He is listed in *Marquis Who's Who in the World*.

Jingang Yi (S'99–M'02) received the B.S. degree in electrical engineering from the Zhejiang University, Hangzhou, China, in 1993, the M.Eng. degree in precision instruments from Tsinghua University, Beijing, China, in 1996, the M.A. degree in applied mathematics, and the Ph.D. degree in mechanical engineering from the University of California, Berkeley, in 2001 and 2002, respectively.

He is currently an Assistant Professor in Mechanical Engineering at San Diego State University. From May 2002 to January 2005, he was with Lam Research Corporation, Fremont, CA, as a member of Technical Staff. From January 2005 to December 2006, he was with the Department of Mechanical Engineering, Texas A&M University, as a Visiting Assistant Professor. His research interests include intelligent and autonomous systems, dynamic systems and control, intelligent sensing and actuation systems, mechatronics, automation science and engineering with applications to semiconductor manufacturing and intelligent transportation systems.

Dr. Yi is a member of American Society of Mechanical Engineering (ASME). He was the recipient of the Kayamori Best Paper Award of the 2005 IEEE Conference on Robotics and Automation (ICRA).



Lawrence Zhang received the M.S. degree in industrial engineering at Tsinghua University, Beijing, China, in 2006 and the B.S. degree in mechanical engineering from Tianjin University, Tianjin, China, in 1992.

He is currently a Plant Manager with Comair Rotron (Shanghai) Fan Company. He was a Manufacturing System Manager and a Department Manager with Intel Shanghai and Intel Philippines, as well as a Production Manager and a Department Manager with Motorola Tianjin. He has published over ten papers. His business and research interests are industrial automation, manufacturing systems, operations research/management, and supply chain management.

Li Zheng received the B.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1986 and 1991, respectively.

He is currently a Professor with the Department of Industrial Engineering, Tsinghua University. He was a Visiting Professor at Georgia Tech during 1994–1996. He has authored/coauthored over 200 publications, and five book chapters. His research interests include production and operation planning and scheduling, production system analysis, and information driven manufacturing.

Dr. Zheng is a member of the Institute of Industrial Engineers (IIE) and founding Chair of IIE, China Chapter. He won several important award, such as the National Science and Technology Progress Award in 2005, the National Invention Award in 1990, and the Excellence Young Faculty Award from MOE in 2000.