Scheduling Integrated Circuit Assembly Operations on Die Bonder

W. L. Pearn, S. H. Chung, and C. M. Lai

***Abstract*—Solving the integrated circuit (IC) assembly scheduling problem (ICASP) is a very challenging task in the IC manufacturing industry. In the IC assembly factories, the jobs are assigned processing priorities and are clustered by their product types, which must be processed on groups of identical parallel machines. Furthermore, the job processing time depends on the product type, and the machine setup time is sequentially dependent on the orders of jobs processed. Therefore, the ICASP is more difficult to solve than the classical parallel machine scheduling problem. In this paper, we describe the ICASP in detail and formulate the ICASP as in integer programing problem to minimize the total machine workload. An efficient heuristic algorithm is also proposed for solving large-scale problems.**

***Index Terms*—Integer programing, integrated circuit (IC) assembly and packaging (ICASP), parallel machine scheduling, thin small outline package (TSOP).**

I. Introduction

In ORDER to increase a company's competition edge and profitability, an integrated circuit (IC) manufacturer needs to utilize its existing capacity efficiently and effectively. In this paper, we consider the IC assembly scheduling problem (ICASP), which has many real-world applications, particularly, in the IC manufacturing industry. This paper studies the scheduling problem for the IC assembly factory mainly producing memory products. The die bonders are considered the most critical resources. Therefore, developing efficient scheduling methods to minimize the total die bonder workload and enhance the utilization of the die bonder is essential. For the ICASP investigated in this paper, the jobs are assigned processing priorities and are clustered by their product families with each family containing several product types, which must be processed on a group of identical parallel machines. Furthermore, the job processing time may vary, depending on the product type (job cluster) of the job process on. Setup times for two consecutive jobs of different product types (job clusters) on the same machine are sequence dependent.

The major four process stages of the IC manufacturing include wafer fabrication, wafer probe, IC assembly, and final testing, as shown in Fig. 1. Wafer fabrication and wafer probe are referred to as the "front-end," while IC assembly and final testing are referred as the "back-end." In the back-end operations, there is a great proliferation of product types, and lots may vary in size from several individual circuits to several thousand [1]. In addition, because of different product profit rates and the varied importance level of customers, there often exists more than one priority level of orders [2], [3].

There are two types of IC packaging, namely ceramic and plastic. Most of the commercial IC chips use plastic packaging. For the IC assembly factory mainly producing memory products, the conventional packages and thin small outline package, type 2 (TSOP2) package dominant the production lines. Fig. 2 shows the process flows of conventional packages and TSOP2 packages. As we can see in Fig. 2, the process flows of conventional packages and TSOP2 packages are the same. Actually, in the floor shop, the machines at each stage can process the operations for these two packages, except for at the die bonding stage. For conventional packages, the die bonding process is to position the good dies on the paddle of the leadframe (using epoxy). While, for TSOP2 packages, the die bonding process is lead on chip (LOC), which the device is fixed with an LOC tape underneath the leadframe, and no curing is needed. Therefore, due to the machine difference and cost consideration, the capacity expansion of the die bonder is usually carefully evaluated. Though the critical resources in most IC assembly factories are die bonders and wire bonders [4]–[6], for memory products, the package lead count of each die is relatively small, and the throughput of the wire bonders is satisfied. Therefore, in the assembly facility mainly producing memory products, the die bonders are treated as the bottleneck.

In contrast to the front-end processes that are highly re-entrant, the back-end process follows a more linear, assembly-line type of flow [6], [7]. In the ICASP, the bottleneck (the die-bonders) is scheduled to be utilized as efficiently as possible, and this implies the reduction of number of setups is crucial. After completing the scheduling on the bottleneck, the lot release time and the scheduling on all the non-bottlenecks facilitate the feeding of the bottleneck.

Since the IC assembly scheduling problem involves constraints on processing priority, job cluster, job-cluster dependent processing time, machine capacity, and sequentially dependent setup times, it is more difficult to solve than the classical parallel-machine scheduling problem. In this paper, we consider a more general version of ICASP and formulate the ICASP as an integer programing problem. The programing model considers the processing priority constraint, and the processing time and the setup time in the capacity constraints. An efficient heuristic is also proposed to obtain the near-optimal solution for large-scale problems.

Manuscript received July 5, 2005; revised June 15, 2006 and November 21, 2006.

The authors are with the Department of Industrial Engineering and Management, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: wlpearn@mail.nctu.edu.tw; shchung@mail.nctu.edu.tw; chunmei.iem88g@nctu.edu.tw)

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/TEPM.2007.899091

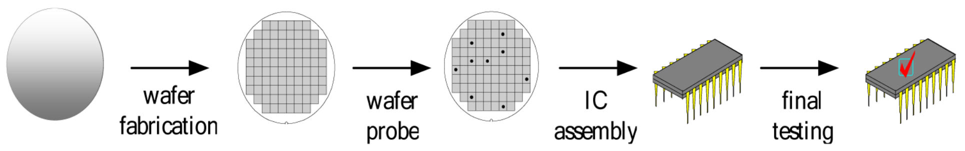

Fig. 1. Four stages of the IC manufacturing.

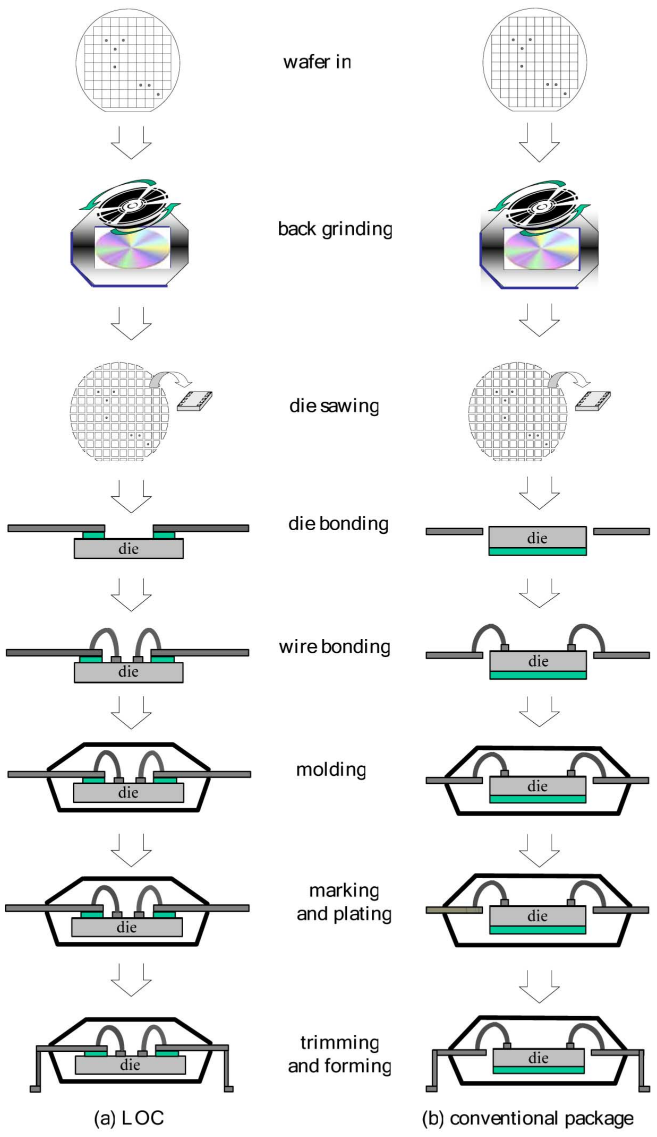

Fig. 2. Process flows of plastic packaging products.

II. Literature Review

The recent increase in device complexity has led to the development of complex capital-intensive assembly systems. The developments in the area of planning and scheduling IC assembly operations, therefore, have seized more attention from the academic world than before. Liu *et al.* [4] developed a computer-aided scheduling system. Potoradi *et al.* [5] developed a simulation-based scheduling to maximize demand fulfillment. Liu *et al.* [8] developed a lot release methodology for minimizing machine conversion. Yin *et al.* [9] developed a rule-based finite-capacity daily scheduling system. Tovia *et al.* [6] considered a simple version of the ICASP for the high production volume and high production mix environment and provided a mathematical model. Tovia *et al.* [6] also presented a rule-based heuristic approach to solve the problem approximately. Unfortunately, their models do not consider sequence-dependent setup times and job processing priority simultaneously, which may not reflect the real situation accurately.

A number of papers addressed the machine scheduling involving sequence-dependent setup times. Lee and Pinedo [10] considered the identical parallel machine problem with sequence-dependent setup times. Schutten and Leussink [11] presented a branch-and-bound algorithm for solving the identical parallel machine problem with release date, due dates, and family setup times, with the objective of minimizing the maximum lateness of any job. Webster and Azizoglu [12] presented two dynamic programing algorithms for solving the identical parallel machine problem with family setup times to minimize total weighted flowtime. Dunstall and Wirth [13] proposed a branching scheme to the identical parallel-machine problem with family setups to minimize the weighted sum of completion times. Chern and Liu [14] constructed five family-based scheduling rules, and built a simulation model to examine these five rules. In those research works, the constraint of multiple processing priorities has never been considered; therefore, their models may not be applicable to the ICASP.

The survey papers [15], [16] provide a wide range of parallel-machine scheduling with precedence constraint. However, the precedence relations studied in these papers can be presented by graphs in tree-types: in-tree type precedence (each job has at most one successor), out-tree type precedence (each job has at most one predecessor), and chain-type precedence (each job has at most one predecessor and at most one successor). In the ICASP, the jobs are completely partitioned and a precedence relation defined by the processing priority. Therefore, the algorithms for solving scheduling problems with tree-type precedence constraint may not be applied to the ICASP. Cheng and Diamond [17] considered the parallel machine scheduling problem for minimizing total flowtime, and provided a dynamic programing algorithm. Unfortunately, their model is developed only for two-class priority and does not consider the sequentially dependent setup times.

The parallel-machine scheduling problem with precedence constraints and sequence-dependent setup times is a combination of a partitioning (assigning jobs to machines) and a sequencing (sequencing the jobs on each machine) problem. Hurink and Knust [18] considered this parallel machine problem of the objective of minimizing the makespan. Hurink and Knust [18] concluded that no efficient list scheduling algorithm can lead to a dominant set of schedules, and recommended that the solution approach needs to consider partitioning and sequencing simultaneously.

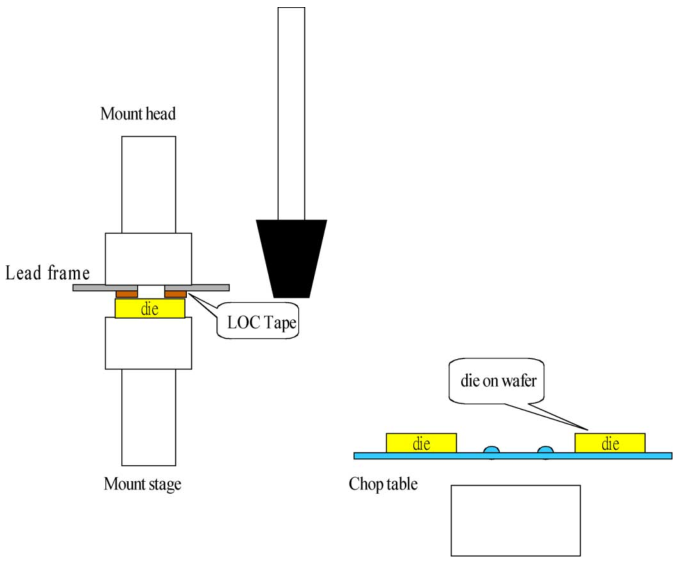

Fig. 3. Die bonding on the LOC die bonder.

III. DESCRIPTION OF ICASP

The IC assembly scheduling problem is to seek a schedule for the jobs to be processed in the time horizon, which minimizes the total die bonder workload, satisfying the job processing priority without violating the machine capacity constraint. In ICASP, the jobs are processed on groups of identical die bonders, and the total processing time is constant. Thus, reducing the total setup time is essential to the minimization of the total machine workload. Process characteristics modeled include sequence dependent setup times, processing priority consideration, and machine capacity constraint.

These integrated circuits, or dies, are formed on wafers that are typically grouped into lot sizes of 25. The size of each lot may vary which depends on the design of dies and die yield. Note that, at the die bonding stage, a lot flowing into the die bonding area is in the form of a complete wafer, and it flows out of this area in the form of a die on leadframe.

A. Sequence-Dependent Setup Times

Since different types of dies must be operated on the LOC die bonder with some specific size of chop table, mount head, and mount stage, and some parameter setting on the machines, some setup operations may be required. Fig. 3 shows the die bonding on the LOC die bonder. The requirement for parameter settings can be regarded as a fixed constant. In the situation, where the current job is formed on 12-in wafers, and the next job is formed on 8-in wafers, and *vice versa*, the next job would have to put on hold until the chop table is changed. Furthermore, in the situation, where the current job is performed with a small mount head and mount stage and the die size of next job is large, the next job would have to be put on hold until the mount stage and mount head is changed.

Thus, the setup time required for switching one product type to another depends on the size of wafer and die. As the jobs come from several product families with each family including a few product types, switching a job to another among different product types within the same product family only requires the parameter setting operations on the machine. In other cases, switching a job from one to another among different product types from different product families must consider the total corresponding setup time occurring due to changing the chop table, changing mount stage and mount head, and the parameter setting operations on the machine.

B. Processing Priority Consideration

Due to different product profit rates and the varied importance level of customers, there often exist more than one priority level of orders in most semiconductor companies. Based on the processing priority, for any two jobs scheduled on the same machine, job A with higher processing priority must be completed before job B with lower processing priority can be begun. Throughout this paper, we assume that each lot is assigned a value of processing priority, which is known at the beginning of the planning horizon. The assignment of processing priority method is beyond the scope of this work, and we refer the interested reader to [5] and [8] for approaches to assign priority value.

C. Machine Capacity Constraint

Normally, the lead time for the total assembly portion is 4 to 6 days. By deducting the setup times and processing times on the non-bottleneck machines, the time horizon of the bottleneck is set to 2 days. Due to the variety of lot size and the importance of the process lot in the initial status, a rolling horizon approach is used in the ICASP. In addition, the preventive maintenance (PM) may be frequent and time consuming. When PM of machine m_k is scheduled to perform in the planning horizon, we need to deduct the times from the available machine capacity and update the initial status. In real-life applications, the capacity for each machine can be set based on the available capacity in the time horizon, and the processing time unit can be “minute” or “hour.” Throughout this paper, we have set the “minute” as the unit of the processing time, setup time, machine workload, and machine capacity in our investigation.

Problem Complexity: The ICASP is NP-hard. Even without the multiple processing priority constraint, the ICASP special case which minimizes makespan on a single machine in the presence of sequence-dependent setup times is equivalent to the Traveling Salesman Problem, and it has been shown to be NP-hard [3], [19].

IV. INTEGER PROGRAMMING FORMULATION

A mathematical programing formulation is a natural way to solve machine scheduling problems [20]. The IP formulations for ICASP have been investigated, but our IP formulation includes both sequentially dependent setup times and processing priority conditions at the same time; therefore, is considerably more complicated than those in [6].

We first define $R = \{R_0, R_1, R_2, \ldots, R_I\}$ containing $I + 1$ clusters of jobs, each job cluster $R_i = \{r_{ij} \,|j = 0, 1, 2, \ldots, J_i\}$ containing J_i $(j = 1, 2, \ldots, J_i)$ jobs to be processed and one pseudojob r_{i0} $(j = 0)$ which is used as the initial status of a machine. Thus, job cluster $R_0 = \{r_{00}\}$ contains one pseudojob, job cluster $R_1 = \{r_{10}, r_{11}, r_{12}, \ldots, r_{1J_1}\}$ contains $J_1 + 1$ jobs, job cluster $R_i = \{r_{i0}, r_{i1}, r_{i2}, \ldots, r_{iJ_i}\}$ contains $J_i + 1$ jobs, and job cluster $R_I = \{r_{I0}, r_{I1}, r_{I2}, \ldots, r_{IJ_I}\}$ contains $J_I + 1$ jobs. We also define $A = \{0, 1, 2, \ldots, H\}$ as the set of processing priority code containing $H + 1$ priority levels. Let h_{ij} ($h_{ij} \in A$) be the processing priority code of job r_{ij}. This code is in the form of a non-negative integer, in such a way, a smaller priority code of job indicates that this job has a higher processing priority. Thus, set $h_{ij} < h_{i'j'}(h_{ij}, h_{i'j'} \in A)$ if job r_{ij} has a higher processing priority than job $r_{i'j'}$.

We also define $M = \{m_1, m_2, \ldots, m_K\}$ as the group of machines containing a set of K identical machines. Let W_k be the predetermined machine capacity expressed in terms of processing time unit. Further, let n_{ij} be the lot size of job r_{ij}, and p_{ik} be the unit processing time for each job r_{ij} in cluster R_i ($r_{ij} \in R_i$) on machine m_k. Therefore, the job processing time for job r_{ij} is $n_{ij}p_{ik}$. Let $s_{ii'}$ be the sequence-dependent setup time between any two consecutive jobs $r_{ij}(\in R_i)$ and $r_{i'j'}(\in R_{i'})$ from different job clusters ($i \neq i'$). Note that the priority codes and lot size for the job r_{i0} should be set to zero so that these pseudojobs should be scheduled as the first jobs on each machine, which indicates the initial status of each machine.

Let x_{ijk} be the variable indicating whether the job r_{ij} is scheduled on machine m_k, with $x_{ijk} = 1$ if job r_{ij} is scheduled on machine m_k, and $x_{ijk} = 0$ otherwise. Let $y_{iji'j'k}$ be the precedence variable defined on two jobs r_{ij} and $r_{i'j'}$ scheduled on machine m_k, with $y_{iji'j'k} = 1$ if job r_{ij} precede job $r_{i'j'}$ (not necessarily directly), and $y_{iji'j'k} = 0$ otherwise. Let $z_{iji'j'k}$ be the direct-precedence variable defined on two jobs r_{ij} and $r_{i'j'}$ scheduled on machine m_k, with $z_{iji'j'k} = 1$ if job r_{ij} direct precede job $r_{i'j'}$, and $z_{iji'j'k} = 0$ otherwise.

To find a schedule for the jobs which minimize the total machine workload without violating the machine capacity and processing priority constraints, we consider the following integer programing model. Although the first summation term (the total processing time) in the objective function of the integer programing model is constant, it is necessary to be used to provide the information of total machine workload in the solutions because managers prefer to know the total machine workload instead of only the total setup time. In addition, with the processing time included in the objective function, the integer programing model can be used to solve a more general ICASP problem, where the machines are unrelated.

Minimize

$$\sum_{k=1}^{K}\left\{\sum_{i=0}^{I}\sum_{j=0}^{J_i} x_{ijk}n_{ij}p_{ik} + \sum_{i=0}^{I}\sum_{j=0}^{J_i}\left(\sum_{i'=0}^{I}\sum_{j'=0}^{J_{i'}} z_{iji'j'k}s_{ii'}\right)\right\}$$

subject to

$$x_{i_k 0k} = 1, \qquad \text{for all } k \tag{1}$$

$$\sum_{i=0}^{I} x_{i0k} = 1, \qquad \text{for all } k \tag{2}$$

$$\sum_{k=1}^{K} x_{ijk} = 1, \qquad \text{for all } i, j > 0 \tag{3}$$

$$\sum_{i=0}^{I}\sum_{j=0}^{J_i} x_{ijk}n_{ij}p_{ik} + \sum_{i=0}^{I}\sum_{j=0}^{J_i}\left(\sum_{i'=0}^{I}\sum_{j'=0}^{J_{i'}} z_{iji'j'k}s_{ii'}\right) \le W_k, \quad \text{for all } k \tag{4}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - \mathrm{Q}(x_{ijk} + x_{i'j'k}) \le 0, \quad \text{for all } i, j, k \tag{5}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - \mathrm{Q}(x_{i'j'k} - x_{ijk} + 1) \le 0, \quad \text{for all } i, j, k \tag{6}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - \mathrm{Q}(x_{ijk} - x_{i'j'k} + 1) \le 0, \quad \text{for all } i, j, k \tag{7}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - [(h_{ij} - h_{i'j'}) \times (y_{iji'j'k} - y_{i'j'ijk})] - \mathrm{Q}(x_{ijk} + x_{i'j'k} - 2) \ge 1 + |h_{ij} - h_{i'j'}| \quad \text{for all } i, j, k \tag{8}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) + [(h_{ij} - h_{i'j'}) \times (y_{iji'j'k} - y_{i'j'ijk})] + \mathrm{Q}(x_{ijk} + x_{i'j'k} - 2) \le 1 - |h_{ij} - h_{i'j'}| \quad \text{for all } i, j, k \tag{9}$$

$$y_{iji*j*k} - \mathrm{Q}(y_{iji'j'k} + y_{i'j'i*j*k} - 2) \ge 1 \quad \text{for all } i, j, k \tag{10}$$

$$y_{iji'j'k} \ge z_{iji'j'k} \qquad \text{for all } i, j, k \tag{11}$$

$$\sum_{i=0}^{I}\sum_{j=0}^{J_i} x_{ijk} - \sum_{r_{ij} \neq r_{i'j'}} z_{iji'j'k} = 1, \qquad \text{for all } k \tag{12}$$

$$\sum_{r_{ij} \neq r_{i'j'}} z_{iji'j'k} \le 1, \qquad \text{for all } i, j, k \tag{13}$$

$$\sum_{r_{ij} \neq r_{i'j'}} z_{i'j'ijk} \le 1, \qquad \text{for all } i, j, k \tag{14}$$

$$x_{ijk} \in \{0, 1\}, \qquad \text{for all } i, j, k \tag{15}$$

$$y_{iji'j'k} \in \{0, 1\}, \qquad \text{for all } i, j, k \tag{16}$$

$$z_{iji'j'k} \in \{0, 1\}, \qquad \text{for all } i, j, k. \tag{17}$$

The objective function seeks to minimize the sum of the total processing time $\sum_{i=0}^{I}\sum_{j=0}^{J_i} x_{ijk}n_{ij}p_{ik}$ and the total setup time $\sum_{i=0}^{I}\sum_{j=0}^{J_i}\left(\sum_{i'=0}^{I}\sum_{j'=0}^{J_{i'}} z_{iji'j'k}s_{ii'}\right)$ over the K identical machines. The constraints in (1) assigns the initial status of each machine m_k. For example, in the situation, where the initial status of machine m_1 is pseudojob r_{30}, we will assign $i_1 = 3$ and $x_{301} = 1$. The constraints in (2) guarantee that only one pseudojob r_{i0} is scheduled on a machine. Constraints in (3) guarantee that job r_{ij} is processed by one machine exactly once. The constraints in (4) state that each machine workload does not exceed the machine capacity.

The constraints in (5) ensure that the precedence variables $y_{iji'j'k}$ and $y_{i'j'ijk}$ should be set to zero ($y_{iji'j'k} + y_{i'j'ijk} \le 0$) if any two jobs r_{ij} and $r_{i'j'}$ are not scheduled on the machine m_k ($x_{ijk} + x_{i'j'k} = 0$). The number Q is a constant, which is chosen to be sufficiently large so that the constraints in (5) are satisfied for ($x_{ijk} + x_{i'j'k} = 0$). The constraints in (6) and (7) ensure that the precedence variables $y_{iji'j'k}$ and $y_{i'j'ijk}$ should be set to zero ($y_{iji'j'k} + y_{i'j'ijk} \le 0$) if any two jobs r_{ij} and $r_{i'j'}$ are not scheduled on the same machine m_k. The constraints in (6) indicates the case that job r_{ij} is scheduled on machine m_k and the job $r_{i'j'}$ is scheduled on another machine

$(x_{i'j'k} - x_{ijk} + 1 = 0)$, and the constraints in (7) indicates the case that job $r_{i'j'}$ is scheduled on machine m_k, and the job r_{ij} is scheduled on another machine $(x_{ijk} - x_{i'j'k} + 1 = 0)$. The constraints in (8) and (9) ensure that one job should precede another if two jobs r_{ij} and $r_{i'j'}$ are scheduled on the same machine $(x_{ijk} + x_{i'j'k} - 2 = 0)$. In the situation where these two jobs have the same priority code $(h_{ij} = h_{i'j'})$, the constraints in (8) and (9) ensure that one job should precede another $(y_{iji'j'k} + y_{i'j'ijk} = 1)$. In the situation where these two jobs have different priority codes $(h_{ij} \neq h_{i'j'})$, the constraints in (8) and (9) ensure that job with smaller priority code (higher processing priority) should precede the other job with larger priority code (lower processing priority) $(y_{iji'j'k} = 1$ and $y_{i'j'ijk} = 0$ if $h_{ij} < h_{i'j'})$ or $(y_{iji'j'k} = 0$ and $y_{i'j'ijk} = 1$ if $h_{ij} > h_{i'j'})$. The constraints in (10) ensure that the job r_{ij} precedes job r_{i*j*} ($y_{iji*j*k} = 1$) when the job r_{ij} precedes job $r_{i'j'}$ $(y_{iji'j'k} = 1)$ and the job $r_{i'j'}$ precedes job r_{i*j*} $(y_{i'j'i*j*k} = 1)$.

The constraints in (11) ensure that job r_{ij} could precede job $r_{i'j'}$ directly $(z_{iji'j'k} = 1)$ only when $y_{iji'j'k} = 1$ and job r_{ij} could not precede job $r_{i'j'}$ directly $(z_{iji'j'k} = 0)$ if job r_{ij} is scheduled after $r_{i'j'}$ $(y_{iji'j'k} = 0)$. The constraints in (12) state that there should exist $n - 1$ direct-precedence variables, which are set to one, on the schedule with n jobs. The constraints in (13) guarantee that at most one job $r_{i'j'}$ could be scheduled after job r_{ij} directly for all the jobs scheduled on the same machine m_k. The constraints in (14) guarantee that at most one job $r_{i'j'}$ could be scheduled precede job r_{ij} directly for all the jobs scheduled on the same machine m_k.

In the aforementioned integer programing formulation, the total number of variables and equations increase as the number of machines or the number of jobs increase. The computational complexity of the integer programing model is as follows. For a parallel-machine problem with I job clusters and K machines, containing a total of $N_I = 1 + (J_1 + 1) + (J_2 + 1) + \cdots + (J_I + 1)$ jobs, the integer programing model contains $N_I K$ variables of x_{ijk}, $N_I K(N_I - 1)$ variables of $y_{iji'j'k}$, and $N_I K(N_I - 1)$ variables of $z_{iji'j'k}$. Further, the constraint sets in (1), (2), (4), and (12) each contains K equations, the constraint set in (3) contains $N_I - (I + 1)$ equations, constraint sets in (5)-(9) and (11) each contains $N_I K(N_I - 1)$ equations, the constraint sets in (10) contains $N_I K(N_I - 1)(N_I - 2)$ equations, and the constraint sets in (13) and (14) contains $N_I K$. Thus, the total number of variables is $2N_I^2 K - N_I K$, and the total number of equations is $N_I^3 K + 3N_I^2 K - 2N_I K + N_I + 4K - (I + 1)$.

To accelerate the execution in solving the integer programing problem, we use both a depth-first search strategy by choosing the most recently created node, and a strong branching rule causing variable selection based on partially solving a number of subproblems with tentative branches to find the most promising branch. By using the depth-first search strategy, when the tree size or the number of fully developed branches exceeds limitations induced by computation time or memory requirements, the program terminates and returns the best solution achieved. The implementation thus allows us to set various limits on the number of memory nodes so that feasible solutions may be obtained efficiently within reasonable amount of computer time. The node limit is set to determine the maximum number of nodes solved before the program terminates, without reaching optimality [21].

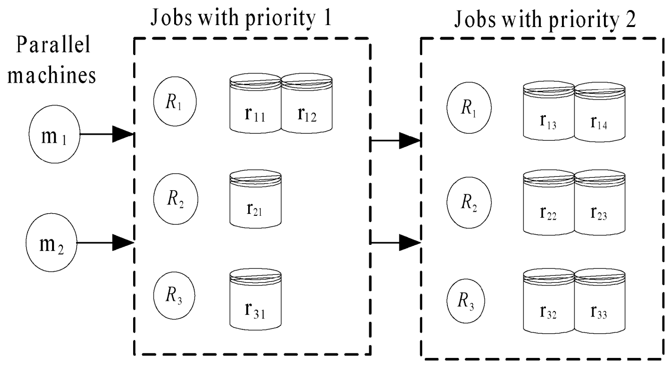

Fig. 4. ICASP example with two parallel machines, two priority levels, and three job clusters.

TABLE I
SETUP TIMES REQUIRED FOR SWITCHING ONE PRODUCT TYPE TO ANOTHER FOR R_1, R_2, AND R_3

	To			
From	U	R_1	R_2	R_3
U	–	10	10	10
R_1	0	0	6	10
R_2	0	3	0	10
R_3	0	3	10	0

TABLE II
JOB PROCESSING TIMES FOR R_1, R_2, AND R_3 ON THE MACHINES m_1 AND m_2

	R_1	R_2	R_3
m_1	25	15	10
m_2	25	15	10

V. SOLUTIONS FOR THE ICASP

Consider the following ICASP example with two parallel machines (m_1 and m_2), two processing priority levels (1 and 2, in which a job with priority 1 has a higher processing priority than the jobs with priority 2), and three clusters of jobs (R_1, R_2, and R_3) ready for processing initially, as shown in Fig. 4. Job cluster R_1 contains four jobs, both r_{11} and r_{12} with priority 1 and both r_{13} and r_{14} with priority 2. Job cluster R_2 contains three jobs, r_{21} with priority 1 and both r_{22} and r_{23} with priority 2. Job cluster R_3 contains three jobs, r_{31} with priority 1 and both r_{32} and r_{33} with priority 2. Table I displays the setup times required for switching one product type to another for the three types 1, 2, and 3. In Table I, the label U denotes that the machine is in idle status. Table II displays the job processing times for job clusters R_1, R_2, and R_3 on the machines m_1 and m_2. Note that the setup times and the processing times are associated with the product types, regardless of processing priority levels. The capacity of each machine is set to 140 min in this example. The initial status of machine m_1 is r_{30}, and that of machine m_2 is r_{10}.

To solve the integer programing problem for the ICASP example, we adopt ILOG OPL [21] to generate the constraints and variables of the model. For the ICASP example with two machines, two processing priority levels, three job clusters, and ten jobs, the model contains 756 variables and 6626 equations. We run the integer programing model using the IP software ILOG

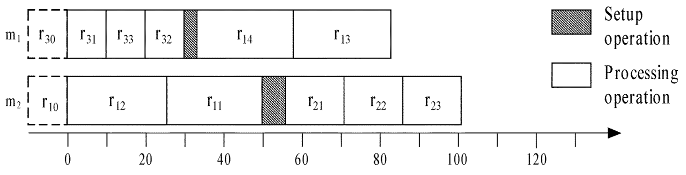

Fig. 5. Optimal solution on the ICASP example.

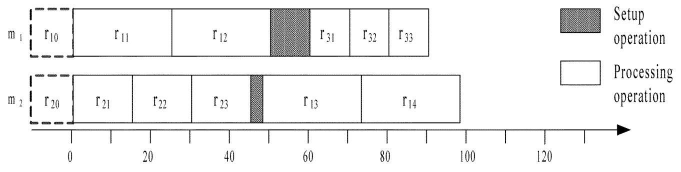

Fig. 6. Optimal solution on the ICASP example with initial status of r_{10} and r_{20}.

CPLEX on a Pentium IV 3.0-GHz PC. The optimal solution for this example is shown in Fig. 5. The total machine workload is 184. For machine m_1, the total machine workload is 83 with setup time 3 and processing time 80. For machine m_2, the total machine workload is 101 with setup time 6 and processing time 95.

Note that the solution will be different when the initial statuses of machines are different. When the initial status of machine m_1 is r_{10} and that of machine m_2 is r_{20}, the optimal solution will become 188, as shown in Fig. 6. For machine m_1, the total machine workload is 90 with setup time 10 and processing time 80. For machine m_2, the total machine workload is 98 with setup time 3 and processing time 95.

As we can see from the solutions of these two examples, in the situation where the initial status of machine m_1 is r_{30} and that of machine m_2 is r_{10}, the first job scheduled on m_1 is r_{31} and the first job scheduled on m_2 is r_{12}. Total machine workload is 184. In the situation where the initial status of machine m_1 is r_{10} and that of machine m_2 is r_{20}, the first job scheduled on machine m_1 becomes r_{11}, and the first job scheduled on m_2 becomes r_{11}. The total machine workload becomes 188.

VI. Heuristic Algorithm

For large-scale problems, the depth-first strategy can solve the problem with more computation effort. However, if the computational run time is primary concern, the following heuristic algorithm may be considered.

The proposed algorithm essentially consists of two phases. Phase I creates a multiple of K machine schedules simultaneously by finding the feasible job with the smallest setup times to add it to the end of partial schedule PS_k. Note that a job is feasible and is added to the machine schedule only if the capacity constraint and the processing priority restrictions are not violated. After Phase I, partial schedules like $\mathrm{PS}_k = (r_{i_k0}, u_1, \ldots, u_{g-1}, u_g, \ldots, u_{G_k})_k$ should be generated, in which r_{i_k0} represents the initial status of the machine m_k, u_g represents the job be scheduled at position g on machine m_k, and G_k represents the total number of jobs in the schedule PS_k.

For the jobs left unscheduled in Phase I due to the processing priority constraint, in Phase II, we calculate the insertion cost of every unscheduled job r_{ij} at every possible position of each partial schedule PS_k to insert the job to the lowest insertion cost position. Note that a job is inserted into the machine schedule only if the capacity constraint and the processing priority restrictions are not violated. Let $\lambda_k(u_{g-1}, r_{ij}, u_g)$ be the additional setup cost when job r_{ij} is inserted between position $g-1$ and g in schedule PS_k. Note that the setup time $S_{I(u_{g-1})I(u_g)}$ is determined by the product types, $I(u_{g-1})$ and $I(u_g)$, of job u_{g-1} and u_g.

The procedures of the proposed heuristic algorithm are described as follows.

Phase I- Schedule construction

Step 1) For each machine m_k, let partial schedule $\mathrm{PS}_k = (r_{i_k0})_k$ initially.
Step 2) Sort the setup times for all pairs of job type i and i' and create a list in ascending order of magnitude.
Step 3) Starting from the top of the setup time list, find the first feasible link in the list, which can be used to extend the end of PS_{k*} without violating the machine capacity constraints and the priority restrictions. If there is a tie for the feasible jobs, choose the job with the highest priority.
Step 4) Repeat Step 3 until no feasible job can be added to extend the end of any PS_k. If there are jobs left unscheduled, proceed to Phase II. Otherwise, stop.

Phase II- Job Insertion

Step 1) For each unscheduled job r_{ij}, first compute its best feasible insertion position, by $\lambda^*_k(u_{g-1}, r_{ij}, u_g)$ in each machine's partial schedule PS_k

$$\lambda_k(u_{g-1}, r_{ij}, u_g) = S_{I(u_{g-1})i} + S_{iI(u_g)} - S_{I(u_{g-1})I(u_g)}.$$

TABLE III
Data Set

Factor	Values considered	Total values
Number of job clusters (I)	3, 6	2
Levels of job priority (H)	3, 5	2
Number of machines (K)	3, 4, 5	3
Number of jobs ($\sum J_i$)	12	1

TABLE IV
Summary Results

I	H	K	$\bar{S}_{opt}$	$\bar{S}_h$	e (%)
3	3	3	93	93	0.00
6	3	3	240	240	0.00
3	5	3	81	81	0.00
6	5	3	267	273	2.25
3	3	4	93	93	0.00
6	3	4	186	186	0.00
3	5	4	120	120	0.00
6	5	4	201	204	1.49
3	3	5	96	96	0.00
6	3	5	168	171	1.79
3	5	5	117	120	2.56
6	5	5	177	177	0.00

TABLE V
PRODUCT TYPES, PROCESSING TIME, AND PROCESSING PRIORITY IN THE REAL-WORLD EXAMPLE

Job ID	*Product type*	*Lot size*	*Unit processin g time*	*Processing priority*	*Job ID*	*Product type*	*Lot size*	*Unit processin g time*	*Processi ng priority*	*Job ID*	*Product type*	*Lot size*	*Unit processin g time*	*Processi ng priority*
1	1	8	42	2	36	7	19	42	3	71	12	20	40	1
2	1	8	42	2	37	7	18	42	3	72	12	20	40	2
3	1	7	42	3	38	7	18	42	3	73	12	20	40	2
4	2	7	42	2	39	8	20	42	2	74	12	20	40	3
5	2	7	42	3	40	8	20	42	2	75	12	20	40	3
6	3	22	40	1	41	8	19	42	2	76	12	20	40	3
7	3	23	40	1	42	8	19	42	3	77	12	19	40	4
8	3	21	40	2	43	8	19	42	3	78	12	19	40	4
9	3	20	40	2	44	8	18	42	3	79	13	23	45	2
10	3	20	40	3	45	8	18	42	3	80	13	22	45	2
11	3	20	40	3	46	8	18	42	4	81	13	22	45	2
12	3	18	40	3	47	8	18	42	4	82	13	20	45	3
13	3	18	40	4	48	9	19	45	2	83	13	20	45	3
14	4	20	40	2	49	9	19	45	2	84	13	20	45	3
15	4	22	40	2	50	9	18	45	3	85	14	20	50	1
16	4	22	40	2	51	9	18	45	3	86	14	20	50	1
17	4	20	40	3	52	9	18	45	3	87	14	19	50	2
18	4	20	40	3	53	9	18	45	3	88	14	19	50	3
19	4	18	40	3	54	10	20	45	2	89	14	19	50	3
20	4	18	40	4	55	10	20	45	3	90	15	20	50	2
21	4	18	40	4	56	10	20	45	3	91	15	20	50	2
22	5	16	40	2	57	10	20	45	3	92	15	20	50	3
23	5	18	40	2	58	10	19	45	3	93	15	20	50	4
24	5	16	40	3	59	10	19	45	3	94	15	20	50	4
25	5	16	40	3	60	10	19	45	3	95	16	18	42	4
26	5	17	40	3	61	10	18	45	4	96	16	18	42	4
27	5	16	40	3	62	10	18	45	4	97	16	17	42	5
28	6	9	42	2	63	11	23	40	1	98	17	15	50	3
29	6	9	42	2	64	11	23	40	1	99	17	15	50	4
30	6	8	42	2	65	11	23	40	2	100	18	12	50	4
31	6	8	42	3	66	11	22	40	2	101	18	12	50	4
32	7	20	42	1	67	11	22	40	3	102	19	16	45	3
33	7	20	42	1	68	11	22	40	3	103	19	16	45	4
34	7	19	42	2	69	11	21	40	3	104	20	7	45	5
35	7	19	42	2	70	11	21	40	3	105	20	7	45	5

Step 2) The job r_{ij} is inserted into the lowest insertion cost position of the machine m_{k^*} determined by the lowest insertion cost $\lambda^*_{k^*}(u_{g-1}, r_{ij}, u_g)$.

$$\lambda^*_{k^*}(u_{g-1}, r_{ij}, u_g) = \min_{k=1,\ldots,K} [\lambda^*_k(u_{g-1}, r_{ij}, u_g)].$$

Step 3) Repeat Steps 1 and 2 until all jobs are scheduled.

VII. COMPUTATIONAL TEST

In this section, two computational tests are presented. The purpose of the first test is to show the results for the problems of small or moderate size. The second test is the scheduling problem based on real-world applications.

A. *Computational Test 1*

In this test, computational results were presented by a set of randomly generated test problems, with similar characteristics to industrial data. Twelve jobs are to be completed within two days. Thus, the machine capacity is set to 2880 min. Table III shows the data set used to generate the test problems. We consider two values of number of job clusters ($I = 3,\ 6$), two sets of level of priority ($H = 3,\ 5$), and three values of number of machines ($K = 3,\ 4,\ 5$). The unit processing time for the product types are 40, 45, and 50. The lot size of each job was generated using uniform [10, 15] for three machines, uniform [14, 19] for four machines, and uniform [18, 23] for five machines. Thus, we have a total of 12 combinations of problem parameters. For each combination, we generate ten instances, yielding a total of 120 problems.

The IP model was tested using a computer program coded in ILOG OPL language and solved with ILOG CPLEX on a Pentium IV 3.0-GHz PC. The heuristic algorithm was coded in Compaq Visual Fortran 6.6. For evaluating the solution quality, percentage error $e = [(\overline{S}_h - \overline{S}_{\text{opt}})/\overline{S}_{\text{opt}}] \times 100$ is employed, where $\overline{S}_h$ is the average setup time of the heuristic solution, and $\overline{S}_{\text{opt}}$ is the optimal average setup time obtained from the IP model. Table IV lists the results. The proposed heuristic is effective, and each percentage error is less than 3%.

Using the combination of depth-first search strategy and strong branching rule showed to be powerful. For every test problems of three and four machines in the data set, the optimal solution was reached with the 15 000 nodes created (within 5 min of execution time). For every test problem of five machines in the data set, the optimal solution was reached with the 17 000 nodes created (within 10 min of execution time).

B. *Computational Test 2*

In this section, we consider the following example taken from an IC assembly shop-floor in an IC manufacturing factory located in the Science-Based Industrial Park, Tainan, Taiwan, R.O.C. For the case we investigated, there are 20 product types of TSOP2 packaging being processed on 33 parallel LOC die bonders.

This real example contains 105 wafer lots with processing priority, lot size, and unit processing time, which would be die bonding under certain size of chop table, mount stage, and mount head, as shown in Table V. These jobs are to be completed on the 33 parallel die bonders within two days. Therefore, the machine capacity is set to 2880 min.

The setup time required for switching one product type to another depends on the chop table changes, mount stage and mount head changes, and parameter settings, as shown in Table VI. The time to change chop table is 240 min, the time to change mount

TABLE VI
SETUP TIMES REQUIRED FOR SWITCHING ONE PRODUCT TYPE TO ANOTHER IN THE REAL-WORLD EXAMPLE

	To																				
From	U	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
U	0	270	270	150	150	150	270	150	150	150	150	150	150	150	150	150	270	150	150	150	270
1	0	0	30	270	270	270	30	270	270	270	270	270	270	270	270	270	150	390	270	270	150
2	0	30	0	270	270	270	30	270	270	270	270	270	270	270	270	270	150	390	270	270	150
3	0	390	390	0	30	30	390	30	30	30	30	150	150	150	150	150	390	150	150	150	390
4	0	390	390	30	0	30	390	30	30	30	30	150	150	150	150	150	390	150	150	150	390
5	0	390	390	30	30	0	390	30	30	30	30	150	150	150	150	150	390	150	150	150	390
6	0	150	150	270	270	270	0	270	270	270	270	270	270	270	390	390	150	390	390	390	150
7	0	390	390	30	30	30	390	0	30	30	30	150	150	150	150	150	390	150	150	150	390
8	0	390	390	30	30	30	390	30	0	30	30	150	150	150	150	150	390	150	150	150	390
9	0	390	390	30	30	30	390	30	30	0	30	150	150	150	150	150	390	150	150	150	390
10	0	390	390	30	30	30	390	30	30	30	0	150	150	150	150	150	390	150	150	150	390
11	0	390	390	30	30	30	270	30	30	30	30	0	30	30	150	150	390	150	150	150	390
12	0	390	390	30	30	30	270	30	30	30	30	30	0	30	150	150	390	150	150	150	390
13	0	390	390	30	30	30	270	30	30	30	30	30	30	0	150	150	390	150	150	150	390
14	0	270	270	30	30	30	270	30	30	30	30	30	30	30	0	30	390	150	30	30	390
15	0	270	270	30	30	30	270	30	30	30	30	30	30	30	30	0	390	150	30	30	390
16	0	30	30	270	270	270	30	270	270	270	270	270	270	270	270	270	0	270	270	270	30
17	0	270	270	30	30	30	270	30	30	30	30	30	30	30	30	30	270	0	30	30	270
18	0	270	270	30	30	30	270	30	30	30	30	30	30	30	30	30	390	150	0	30	390
19	0	270	270	30	30	30	270	30	30	30	30	30	30	30	30	30	390	150	30	0	390
20	0	30	30	270	270	270	30	270	270	270	270	270	270	270	270	270	30	270	270	270	0

stage and mount head is 120 min, and parameter settings is 30 min in this case.

Solving the real-world ICASP by our proposed algorithm (the program codes of the algorithm are written in Compaq Visual Fortran 6.6), the sets of machine schedule are generated. The proposed algorithm takes only 0.07 CPU s to obtain the solution with total machine workload 87 602 with setup time 6480 and processing time 81 122 on 33 die bonders.

VIII. CONCLUSION

In this paper, we considered the ICASP, which is a variation of the parallel-machine scheduling problem and is often found in real-world practice. The ICASP is first described in detail to capture the unique structure and characteristics of the IC assembly process. We then formulate the ICASP as an integer programing model to minimize the total machine workload. An efficient heuristic algorithm is also proposed for solving large-scale problems. From the computational tests, the performances of the proposed model and heuristic algorithm are quite satisfactory. A real-world example taken from an IC assembly shop-floor in an IC manufacturing factory, where 105 jobs to be processed on 33 machines, is solved by the proposed algorithm to obtain the near optimal solution within 0.07 CPU s.

REFERENCES

[1] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry, part I: System characteristics, performance evaluation and production planning," *Trans. Inst. Electron. Eng.*, vol. 24, no. 4, pp. 47–60, 1992.

[2] R. Uzsoy, L. A. Martin-Vege, C. Y. Lee, and P. A. Leonard, "Production scheduling algorithms for a semiconductor test facility," *IEEE Trans. Semicond. Manuf.*, vol. 4, no. 4, pp. 270–280, Nov. 1991.

[3] T. Freed and R. C. Leachman, "Scheduling semiconductor device test operations on multihead testers," *IEEE Trans. Semicond. Manuf.*, vol. 12, no. 4, pp. 523–530, Nov. 1999.

[4] T. H. Liu, A. J. C. Trappey, and F. W. Chan, "A scheduling system for IC packaging industry using STEP enabling technology," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 20, no. 4, pp. 256–267, Dec. 1997.

[5] J. Potoradi, O. S. Boon, S. J. Mason, J. W. Fowler, and M. E. Prund, E. Yiicesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, Eds., "Using simulation-based scheduling to maximize demand fulfillment in a semiconductor assembly facility," in *Proc. 2002 Winter Simulation Conf.*, 2002, pp. 1857–1861.

[6] F. Tovia, S. J. Mason, and B. Ramasami, "A scheduling heuristic for maximizing wirebonder throughput," *IEEE Trans. Electron. Packag. Manuf.*, vol. 27, no. 2, pp. 145–150, Apr. 2004.

[7] P. Peter and T. Yang, "Integrated facility layout and material handling system design in semiconductor fabrication facilities," *IEEE Trans. Semicond. Manuf.*, vol. 15, no. 3, pp. 91–101, Aug. 2002.

[8] W. Liu, T. J. Chua, T. X. Cai, F. Y. Wang, and W. J. Yan, "Practical lot release methodology for semiconductor back-end manufacturing," *Production Planning Control*, vol. 16, no. 3, pp. 297–308, 2005.

[9] X. F. Yin, T. J. Chua, F. Y. Wang, M. W. Liu, T. X. Cai, W. J. Yan, C. S. Chong, J. P. Zhu, and M. Y. Lam, "A rule-based heuristic finite capacity scheduling system for semiconductor backend assembly," *Int. J. Comput. Integr. Manuf.*, vol. 17, no. 8, pp. 733–749, 2004.

[10] Y. H. Lee and M. Pinedo, "Scheduling jobs on parallel machines with sequence-dependent setup times," *Eur. J. Oper. Res.*, vol. 100, pp. 464–474, 1997.

[11] J. M. J. Schutten and R. A. M. Leussink, "Parallel machine scheduling with release dates, due dates and family setup times," *Int. J. Prod. Econ.*, vol. 46–47, pp. 119–125, 1996.

[12] S. Webster and M. Azizoglu, "Dynamic programing algorithms for scheduling parallel machines with family setup times," *Comput. Oper. Res.*, vol. 28, pp. 127–137, 2001.

[13] S. Dunstall and A. Wirth, "A comparison of branch-and-bound algorithms for a family scheduling problem with identical parallel machines," *Eur. J. Oper. Res.*, vol. 167, pp. 283–296, 2005.

[14] C. C. Chern and Y. L. Liu, "Family-based scheduling rules of a sequence-dependent wafer fabrication system," *IEEE Trans. Semicond. Manuf.*, vol. 16, no. 1, pp. 15–25, Feb. 2003.

[15] T. C. E. Cheng and C. C. S. Sin, "A state-of-art review of parallel-machine scheduling research," *Eur. J. Operational Res.*, vol. 47, pp. 271–292, 1990.

[16] E. Mokotoff, "Parallel machine scheduling problems: A survey," *Asia–Pacific J. Oper. Res.*, vol. 18, pp. 193–242, 2001.

[17] T. C. E. Cheng and J. E. Diamond, "Scheduling two job classes on parallel machines," *Trans. Inst. Electron. Eng.*, vol. 27, pp. 689–693, 1995.

[18] J. Hurink and S. Knust, "List scheduling in a parallel machine environment with precedence constraints and setup times," *Oper. Res. Lett.*, vol. 29, pp. 231–239, 2001.

[19] C. Y. Liu and S. C. Chang, "Scheduling flexible flow shops with sequence-dependent setup effects," *IEEE Trans. Robot. Autom.*, vol. 16, no. 4, pp. 408–419, Aug. 2000.

[20] A. H. G. R. Kan, *Machine Scheduling Problems: Classification, Complexity and Computations*. The Hague, The Netherlands: Martinus Nijhoff, 1976.

[21] *ILOG OPL Studio 3.6 User's Manual*. Gentilly Cedex, France: ILOG SA, 2002.

W. L. Pearn received the Ph.D. degree in operations research from the University of Maryland, College Park.

He is a Professor of Operations Research and Quality Assurance at National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C. He was with AT&T Bell Laboratories as a member of quality research staff before joining NCTU. His research interests include process capability, network optimization, and production management. His publications have appeared in the *Journal of the Royal Statistical Society, Series C*, *Journal of Quality Technology*, *Journal of Applied Statistics*, *Statistics and Probability Letters*, *Quality and Quantity*, *Metrika*, *Statistics, Journal of the Operational Research Society*, *Operations Research Letters*, *Omega, Networks*, *International Journal of Productions Research*, and others.

S. H. Chung received the Ph.D. degree in industrial engineering from Texas A&M University, College Station.

She is a Professor of the Department of Industrial Engineering and Management, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C. Her research interests include production planning, scheduling, cycle time estimation, and performance evaluation. She has published and presented research papers in the areas of production planning and scheduling for IC manufacturing.

C. M. Lai received the Ph.D. degree in industrial engineering and management from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C, in 2007.

Her research interests include production planning, scheduling, and cycle time estimation.