Production Planning & Control: The Management of Operations

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tppc20

A case study on the wafer probing scheduling problem

W. L. Pearn , S. H. Chung & M. H. Yang

Published online: 15 Nov 2010.

To cite this article: W. L. Pearn , S. H. Chung & M. H. Yang (2002) A case study on the wafer probing scheduling problem, Production Planning & Control: The Management of Operations, 13:1, 66-75

To link to this article: http://dx.doi.org/10.1080/09537280110061593

PRODUCTION PLANNING & CONTROL, 2002, VOL. 13, NO. 1, 66–75

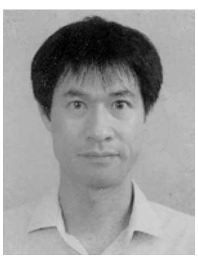
A case study on the wafer probing scheduling problem

W. L. PEARN, S. H. CHUNG and M. H. YANG

Keywords wafer probing, parallel-machine scheduling, sequence-dependent setup time, integer programming, algorithm

Abstract. The wafer probing scheduling problem (WPSP) is a practical generalization of the classical parallel-machine scheduling problem, which has many real-world applications, particularly, in the integrated circuit (IC) manufacturing industry. In this paper, a case study on the WPSP is presented, which is taken from a wafer probing shop floor in an IC manufacturing factory. For the WPSP case investigated, the jobs are clustered by their product types, which are processed on groups of identical parallel machines and must be completed before the due dates. The job processing time depends on the product type, and the machine setup time is sequentially dependent on the orders of jobs processed. The objective in this case is to seek a probing job schedule with minimum total machine workload. Since the WPSP case investigated involves constraints on job clusters, job-cluster dependent processing time, due dates, machine capacity, and sequentially dependent setup time, it is more difficult to solve than the classical parallel-machine sched-

Authors: W. L. Pearn, S. H. Chung and M. H. Yang, Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan, ROC.



W. L. Pearn is Professor of the Department of Industrial Engineering and Management, National Chiao-Tung University, Taiwan, ROC. He received his PhD degree in operations research from the University of Maryland, College Park, MD, USA. He worked for AT&T Bell Laboratories Switch Network Control and Process Quality Centers. He has published numerous papers in the areas of network optimization, machine scheduling, and process capability analysis.

S. H. Chung is Professor of the Department of Industrial Engineering and Management, National Chiao-Tung University, Taiwan, ROC. She received the PhD degree in industrial engineering from Texas A&M University, College Station, TX, USA. Her research interests include production planning, scheduling, system simulation, and production planning of IC manufacturing. She has published and presented research papers in the areas of flexible manufacturing system planning, scheduling, and cell formation.

Ming-Hsien Yang is a PhD candidate at the Department of Industrial Engineering and Management, National Chiao-Tung University, Taiwan, ROC. He received his MS degree in industrial engineering and management from National Chiao-Tung University. He is also a Lecturer with Industrial Engineering and Management Department, National Lien Ho Insitute of Technology, Taiwan, ROC. His research interests include production planning and scheduling.

Production Planning & Control ISSN 0953–7287 print/ISSN 1366–5871 online
http://www.tandf.co.uk/journals
DOI: 10.1080/0953728011006159 3

uling problem. The WPSP is formulated as an integer programming problem and the problem solved using the powerful CPLEX with effective implementation strategies. An efficient solution procedure to solve the WPSP near-optimally is proposed.

1. Introduction

The wafer probing scheduling problem (WPSP) is a variation of the parallel-machine scheduling problem considered by Centeno and Armacost (1997) and Ovacik and Uzsoy (1995, 1996), which has many real-world applications, particularly, in the integrated circuit (IC) manufacturing industry. The major four stages of process for IC product are wafer fabrication, wafer sort, assembly, and final testing. Both wafer sort and final testing are testing related processes, where the testers are critical and expensive resources. Therefore, developing efficient scheduling methods to minimize the total workload and enhance the utilization of testers is essential and important. For the WPSP case investigated, the jobs are clustered by their product types, which must be processed on identical parallel machines and be completed before the due dates. Further, the job processing time may vary, depending on the product type (job cluster) of the job processed on. Setup time for two consecutive jobs of different product types (job clusters) on the same machine are sequentially dependent.

In this paper, a general version of WPSP with each job cluster containing multiple jobs is considered, and a case study on the WPSP presented which covers two workload levels: low and high. The case is taken from a wafer probing shop floor in an IC manufacturing factory located in the Science-based Industrial Park at Hsinchu, Taiwan, where the total machine workload must be minimized. The WPSP is formulated as an integer programming problem to minimize the total machine workload. The programming model considers the due date restrictions, which includes the processing time and the setup time in the capacity constraints, thus reflects the real situations more accurately than those considered by Parker *et al.* (1977) and Chen *et al.* (1995). To illustrate the applicability of the linear integer programming model, the integer programming model is implemented using the IP software CPLEX to solve the WPSP. In addition, we propose an efficient solution procedure is proposed called the Generalized-Saving algorithm to solve the WPSP near-optimally. The model and algorithms developed for WPSP in this paper can be applied to general parallel machine problems covering a wide class of scheduling problems such as the die mounting and wire bonding scheduling problems in IC packaging factories, the knitting machine scheduling problem in textile factories, and so on.

Parker *et al.* (1977) considered a simple version of the WPSP with each job cluster containing only one job, and presented a heuristic algorithm to solve the problem approximately. Unfortunately, their model does not consider the due date restrictions, which only includes the processing time without considering the setup time (changeover cost) in the machine capacity constraints. Ovacik and Uzsoy (1996) presented another version of WPSP minimizing the maximum lateness for the scheduling problem arising in the final test phase of semiconductor manufacturing. However, they have simplified the complexity of the problem by designing the decomposition procedures to divide the testers into a number of workcentres with each containing a single machine. Chen *et al.* (1995) discussed an analogous version of WPSP, which covers both stages of wafer sorting (wafer probing) and IC testing. But, Chen *et al.* (1995) simplified the problem by adding the setup time to the processing time in their model. Therefore, the models considered by those authors do not reflect the real situations accurately. Since the wafer probing scheduling problem involves constraints on job clusters, job-cluster dependent processing time, due dates, machine capacity, and sequentially dependent setup time, obviously it is considerably more difficult to solve than those classical parallel-machine scheduling problems investigated by Ho and Chang (1995), Gabrel (1995), Schutten and Leussink (1996), Cheng and Gen (1997), Lee and Pinedo (1997), Park and Kim (1997), Ruiz-Torres *et al.* (1997).

2. The wafer probing process

The operations in the wafer probing factory can be separated into four stages: lot queuing, wafer sorting, ink marking, and quality control inspection, as illustrated in figure 1. The wafer lots, shipped from the wafer fabrication factory, dynamically arrive at the wafer probing factory. To prevent the dynamic arrival of lots from causing frequent setup of testers, certain level of lot inventory is allowed. Equipment including tester, prober, and some hardware (such as load board and probe card) are needed for wafer sorting. Testers are expensive and critical resources, which determine whether the individual dice on wafer is good or bad. This is done by running the test codes on testers. Prober, load board, and probe card are necessary resources for wafer sorting. The prober is on the position where the wafer is fixed and tested. The load

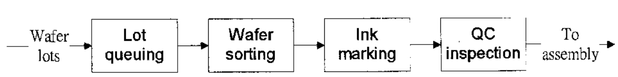


Figure 1. Material flow in wafer probing factories.

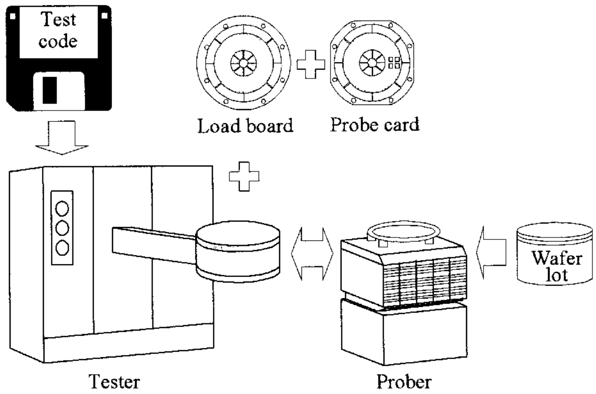


Figure 2. The hardware connections in the wafer probing process.

board is the interface board connecting the tester and probe card via pogo pins. The probe card is the interface between the tester and the prober transferring the testing signals. The connections between the tester and the prober, and other related resources are displayed in figure 2. During the testing, the individual dice on wafer is contacted with the probes on the probe card and the defective dice is marked, as shown in figure 3.

Since different product type wafer must be probed with some specific type of load board and probe card, setup operations may be required. The steps of setup operations include:

(1) Obtaining a suitable load board and probe card and bringing them to the tester assigned.
(2) Loading the load board and the probe card into the prober and adjusting it to fit the wafer.
(3) Attaching the prober to the tester head.
(4) Downloading the required test code, which is determined by the product type of the wafer to be tested.
(5) Heating the prober to the required level of temperature, which is determined by the product type of the wafer to be tested.

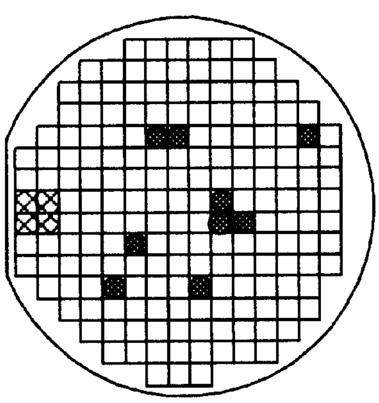


Figure 3. The probed wafer with several defective dies.

The time required for changing the load board and probe card can be regarded as a fixed constant. In the situation where the previous job is tested under high temperature, the next job would have to be put on hold until the temperature falls. On the other hand, if the next job will be tested under high temperature while the current one is tested under room temperature, then the next job would have to be put on hold until the temperature goes up. Thus, the setup time required for switching one product type to another depends on the load board and probe card, and the level of temperature.

3. Case study

Consider the following case with two different workload levels taken from a wafer probing shop floor in an IC manufacturing factory located on the Science-based Industrial Park at Hsinchu, Taiwan. For the case investigated, there are 12 test codes (with each code representing a specific product type) to be processed on 9 identical testers (testing machine) arranged in parallel, as shown in figure 4. The jobs are to be completed on the parallel-tester before due dates in the following three days. Therefore, the machine capacity is set to 4320 (min). 'Minute' is used here as the time unit of the processing time and due date of jobs, and also as the time unit of the machine capacity. The setup time required for switching one product type to another are shown in table 1. The

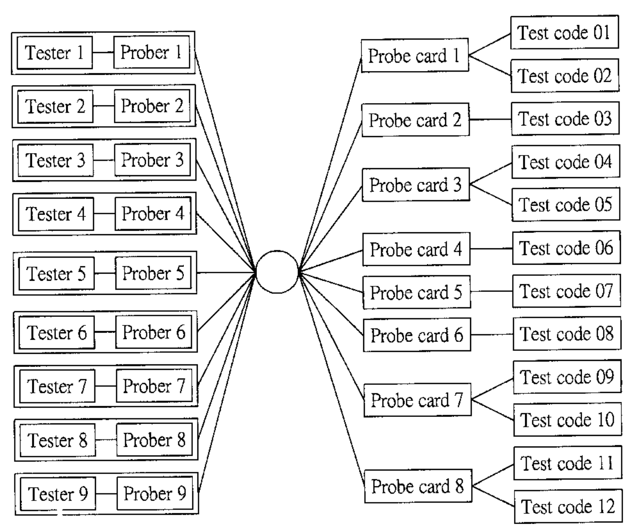


Figure 4. The relationships between the twelve test codes, eight probe cards, nine probers and testers.

Table 1. Setup time required for switching one product type to another for the twelve product types.

From \ To	00	01	02	03	04	05	06	07	08	09	10	11	12
00	0	70	70	70	40	40	40	40	70	70	70	70	70
01	80	0	0	110	80	80	80	80	110	110	110	110	110
02	80	0	0	110	80	80	80	80	110	110	110	110	110
03	80	110	110	0	80	80	80	80	110	110	110	110	110
04	40	70	70	70	0	0	40	40	70	70	70	70	70
05	40	70	70	70	0	0	40	40	70	70	70	70	70
06	40	70	70	70	40	40	0	40	70	70	70	70	70
07	40	70	70	70	40	40	40	0	70	70	70	70	70
08	80	110	110	110	80	80	80	80	0	110	110	110	110
09	80	110	110	110	80	80	80	80	110	0	0	110	110
10	80	110	110	110	80	80	80	80	110	0	0	110	110
11	80	110	110	110	80	80	80	80	110	110	110	0	0
12	80	110	110	110	80	80	80	80	110	110	110	0	0

time required for changing the probe card is 40 minutes. The time required to increase the temperature is 30 minutes and the time required to decrease the temperature is 40 minutes.

On the shop floor, the workloads might vary. Two workload levels are considered: the lower level with 20 jobs (wafer lot) and the higher level with 35 jobs (wafer lot), which is shown in table 2 and table 3. Each job is associated with a due date and a processing time, which must be tested at a certain level of temperature with a specific test code and probe card. The jobs listed in table 2 include 11 product types and the jobs listed in table 3 include 12 product types.

Table 2. The product types, probe card, testing temperatures, processing time, and due dates for the 20 jobs in the case.

Job I.D.	Product type	Probe card	Testing temperature	Processing time	Due date
1	01	1	High	1200	4320
2	01	1	High	1200	4320
3	02	1	High	1108	1440
4	02	1	High	1108	1440
5	03	2	High	511	4320
6	03	2	High	511	4320
7	04	3	Room	262	1440
8	05	3	Room	277	1440
9	05	3	Room	277	1440
10	06	4	Room	410	2880
11	06	4	Room	410	2880
12	07	5	Room	2215	4320
13	08	6	High	300	4320
14	09	7	High	343	2880
15	09	7	High	343	4320
16	09	7	High	343	4320
17	10	7	High	351	2880
18	10	7	High	351	2880
19	11	8	High	585	4320
20	11	8	High	585	4320

Table 3. The product types, probe card, testing temperatures, processing time, and due dates for the 35 jobs in the case.

Job I.D.	Product type	Probe card	Testing temperature	Processing time	Due date
1	01	1	High	1200	2880
2	01	1	High	1200	2880
3	01	1	High	1200	4320
4	01	1	High	1200	4320
5	02	1	High	1108	1440
6	02	1	High	1108	1440
7	02	1	High	1108	1440
8	02	1	High	1108	2880
9	02	1	High	1108	2880
10	03	2	High	511	4320
11	03	2	High	511	4320
12	04	3	Room	262	1440
13	04	3	Room	262	1440
14	04	3	Room	262	1440
15	05	3	Room	277	1440
16	05	3	Room	277	1440
17	06	4	Room	410	2880
18	06	4	Room	410	2880
19	06	4	Room	410	2880
20	07	5	Room	2215	4320
21	07	5	Room	2215	4320
22	07	5	Room	2215	4320
23	08	6	High	300	4320
24	09	7	High	343	2880
25	09	7	High	343	4320
26	09	7	High	343	4320
27	10	7	High	351	2880
28	10	7	High	351	2880
29	11	8	High	585	1440
30	11	8	High	585	4320
31	11	8	High	585	4320
32	11	8	High	585	4320
33	12	8	High	497	2880
34	12	8	High	497	2880
35	12	8	High	497	2880

The objective in the case is to find a schedule for the jobs, which satisfies the due date restrictions without violating the machine capacity constraints, while the total machine workload must be minimized. Reducing the total setup time is essential to the minimization of the total machine workload.

4. An integer programming formulation

Define $R = \{R_0, R_1, R_2, \ldots, R_I\}$ as the $I + 1$ clusters of jobs to be processed with each job cluster $R_i = \{r_{ij} | j = 1, 2, \ldots, \mathcal{J}_i\}$ containing $\mathcal{J}_i$ jobs. Thus, job cluster $R_0 = \{r_{01}, r_{02}, \ldots, r_{0\mathcal{J}_0}\}$ contains $\mathcal{J}_0$ jobs, job cluster $R_1 = \{r_{11}, r_{12}, \ldots r_{1\mathcal{J}_1}\}$ contains $\mathcal{J}_1$ jobs, job cluster $R_i = r_{i1}, r_{12}, \ldots r_{i\mathcal{J}_1}\}$ contains $\mathcal{J}_i$ jobs, and job cluster $R_I = \{r_{I1}, \ldots r_{I2}, \ldots r_{I\mathcal{J}_I}\}$ contains $\mathcal{J}_I$ jobs. Define $M = \{m_1, m_2, \ldots, m_K\}$ as the group of machines containing a set of K identical machines. Note that job cluster R_0 including $\mathcal{J}_0 = K$ jobs is a pseudo cluster, which is used to indicate the K machines are in idle status. Let W be the predetermined machine capacity (set to a constant in this case) expressed in terms of processing time units. Further, let n_{ij} be the lot size (number of wafers) of job r_{ij}, and p_i be the job processing time of job r_{ij} in cluster $R_i (r_{ij} \in R_i)$. Therefore, the job processing time for job r_{ij} is $n_{ij}p_i$. Let $s_{ii'}$ be the sequentially dependent setup time between any two consecutive jobs $r_{ij} (\in R_i)$ and $r_{i'j'} (\in R_i)$ from different job clusters $(i \neq i')$.

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 to be processed on machine m_k, and $x_{ijk} = 0$ otherwise. Let b_{ij} be the ready time of job r_{ij} and d_{ij} be the due date of job r_{ij}. Define the variable t_{ijk} as the starting time for job r_{ij} to be processed on machine m_k. The starting processing time t_{ijk} should not be less than the earliest starting processing time b_{ij}, which depends on the ready time of job r_{ij}, and not be greater than the latest starting processing time e_{ij}, which relates to the due date d_{ij} and can be computed as $e_{ij} = D_{ij} - n_{ij}p_i$. If job r_{ij} is ready to be processed initially, then b_{ij} may be set to 0. It is noted that the processing time and due dates for the jobs in R_0 should be set to 0 so that these pseudo jobs can be scheduled as the first jobs on each machine, which indicates that each machine is initially in idle status. Let $y_{iji'j'k}$ be the precedence variable, where $y_{iji'j'k}$ should be set to 1 if the two jobs r_{ij} and $r_{i'j'}$ are scheduled on machine m_k and job r_{ij} precedes job $r_{i'j'}$ (not necessarily directly), and where $y_{iji'j'k} = 0$ otherwise. Further, let $z_{iji'j'k}$ be the direct-precedence variable, where $z_{iji'j'k}$ should be set to 1 if the two jobs r_{ij} and $r_{i'j'}$ scheduled on machine m_k and job r_{ij} precedes job $r_{i'j'}$ directly, and where $z_{iji'j'k} = 0$ otherwise.

To find a schedule for those jobs which minimizes the total machine workload without violating the machine capacity and the service time window constraints, the following integer programming model is considered.

Objective function:

The objective function seeks to minimize the sum of the total processing time $\sum_{i=0}^{I} \sum_{j=1}^{\mathcal{J}_i} x_{ijk} n_{ij} p_i$ and the total setup time $\sum_{i=0}^{I} \sum_{j=1}^{\mathcal{J}_i} (\sum_{i'=0}^{I} \sum_{j'=1}^{\mathcal{J}_{i'}} z_{iji'j'k} s_{ii'})$ over the K machines

$$\sum_{k=1}^{K} \left\{ \sum_{i=0}^{I} \sum_{j=1}^{\mathcal{J}_i} x_{ijk} n_{ip} p_i + \sum_{i=0}^{I} \sum_{j=1}^{\mathcal{J}_i} \left(\sum_{i'=0}^{I} \sum_{j'=1}^{\mathcal{J}} z_{iji'j'k} s_{ii'} \right) \right\}$$

Job constraints:

The constraints in equation (1) guarantees that job r_{ij} is processed by one machine exactly once.

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

Schedule initialization constraints:

The constraints in equation (2) guarantee that only one pseudo job r_{0j} is scheduled on a machine.

$$\sum_{j=1}^{\mathcal{J}_0} x_{0jk} = 1 \qquad \text{for all } k \tag{2}$$

Machine capacity constraints:

The constraints in equation (3) state that each machine workload does not exceed the machine capacity W.

$$\sum_{i=0}^{I} \sum_{j=1}^{\mathcal{J}_i} x_{ijk} n_{ij} p_i + \sum_{i=0}^{I} \sum_{j=1}^{\mathcal{J}_i} \left(\sum_{i'=0}^{I} \sum_{j'=1}^{\mathcal{J}_{i'}} \mathcal{J}_{i'}, z_{iji'j'k} s_{ii'} \right) \leq \text{W} \qquad \text{for all } k \tag{3}$$

Starting processing time constraints:

The constraints in equations (4) and (5) ensure that $t_{ijk} + n_{ij}p_i + s_{ii'} = t_{i'j'k}$ if job r_{ij} precedes job $r_{i'j'}$ directly ($y_{iji'j'k} = 1$ and $z_{iji'j'k} = 1$). The constraints in equation (4) ensure the satisfaction of the inequality $t_{ijk} + n_{ij}p_i + s_{ii'} \leq t_{i'j'k}$, if the jobs r_{ij} proceeding job $r_{i'j'} (y_{iji'j'k} - 1 = 0)$. The number Q is a constant, which is chosen to be sufficiently large so that the constraints in equation (4) are satisfied for $y_{iji'j'k} = 0$ or 1. For example, one can choose $\text{Q} = \sum_{i=1}^{I} \sum_{j=1}^{\mathcal{J}_i} (n_{ij}p_i + \max_{i'}\{s_{ii'}\})$. The constraints in equation (5) ensure the satisfaction of the inequality $t_{ijk} + n_{ij}p_i + s_{ii'} \geq t_{i'j'k}$ the jobs r_{ij} proceeding job $r_{i'j'}$ directly $(y_{iji'j'k} + z_{iji'j'k} - 2 = 0)$.

$$t_{ijk} + n_{ij}p_i + s_{ii'} - t_{i'j'k} + Q(y_{iji'j'k} - 1) \leq 0 \quad \text{for all } i, j, k \tag{4}$$

$$t_{ijk} + n_{ij}p_i + s_{ii'} - t_{i'j'k} - Q(y_{iji'j} + z_{iji'j'k} - 2) \geq 0 \quad \text{for all } i, j, k \tag{5}$$

Due date constraints:

The constraints in equations (6) and (7) state that the starting processing time t_{ij} for each job r_{ij} scheduled on machine $m_k(x_{ijk} = 1)$ should not be less than the earliest starting processing time b_{ij} and not be greater than the latest starting processing time e_{ij}.

$$t_{ijk} \geq b_{ij}x_{ijk} \quad \text{for all } i, j, k \tag{6}$$

$$t_{ijk} \leq e_{ij}x_{ijk} \quad \text{for all } i, j, k \tag{7}$$

Precedence constraints:

The constraints in equations (8) and (9) ensure that one job should precede another $(y_{iji'j'k} + y_{i'j'ijk} = 1)$ if two jobs are scheduled on the same machine $(x_{ijk} + x_{i'j'k} - 2 = 0)$. The number Q is a constant, which is chosen to be sufficiently large so that the constraints in equations (8) and (9) are satisfied for $x_{ijk} + x_{i'j'k} - 2 < 0$. The constraints in equation (10) ensure that the precedence variables $y_{iji'j'k}$ and $y_{i'j'ijk}$ should be set to zero $(y_{iji'j'k} + y_{i'j'ijk} \leq 0)$ 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 constraints in equations (11) and (12) ensure that the precedence variables $y_{iji'j'k}$ and $y_{ipj'ijk}$ should be set to zero $(y_{iji'j'k} + y_{i'j'ijk} \leq 0)$ if any two jobs r_{ij} and $r_{i'j'}$ are not scheduled on the machine m_k. The constraints in equation (11) indicate 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 equation (12) indicate the case that job $r_{i'j'}$ is scheduled on machine m_k and the job r_{ij} is scheduled on another machine $(x_{ijk} - x_{i'j'k} + 1 = 0)$.

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{ijk} + x_{i'j'k} - 2) \geq 1 \quad \text{for all } i, j, k \tag{8}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) + Q(x_{ijk} + x_{i'j'k} - 2) \leq 1 \quad \text{for all } i, j, k \tag{9}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{ijk} + x_{i'j'k}) \leq 0 \quad \text{for all } i, j, k \tag{10}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{i'j'k} - x_{ijk} + 1) \leq 0 \quad \text{for all } i, j, k \tag{11}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{ijk} - x_{i'j'k} + 1) \leq 0 \quad \text{for all } i, j, k \tag{12}$$

Directly precedence constraints:

The constraints in equation (13) 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 job $r_{i'j'}(y_{iji'j'k} = 0)$. The constraints in equation (14) state that there should exist $n - 1$ directly precedence variables, which are set to 1, on the schedule with n jobs. The constraints in equation (15) state: when the job r_{ij} proceeds job $r_{i'j'}$ but not consecutively $(y_{iji'j'k} = 1$ and $z_{iji'j'k} = 0)$, then there must exist another job $r_{i^*j^*}$ scheduled after job r_{ij} directly $(y_{iji^*j^k} = 1$ and $z_{iji^*j^*k} = 1)$ and ensuring the satisfaction of the inequality $y_{iji^*j^*k} + z_{iji^*j^*k} \geq 2$.

$$y_{iji'j'k} \geq z_{iji'j'k} \quad \text{for all } i, j, k \tag{13}$$

$$\sum_{i=0}^{I} \sum_{j=1}^{J_i} x_{ijk} - \sum_{r_{ij} \neq r_{i'j'}} z_{iji'j'k} = 1, \quad \text{for all } k \tag{14}$$

$$y_{iji^*j^*k} + z_{iji^*j^*k} - Q(y_{iji^*j^*k} + z_{iji^*j^*k} - 2) - Q(y_{iji'j'k} - z_{iji'j'k} - 1) \geq 2 \quad \text{for all } i, j, k \tag{15}$$

Binary variables:

$$x_{ijk} \in \{0, 1\} \quad \text{for all } i, j, k \tag{16}$$

$$y_{iji'j'k} \in \{0, 1\} \quad \text{for all } i, j, k \tag{17}$$

$$z_{iji'j'k} \in \{0, 1\} \quad \text{for all } i, j, k \tag{18}$$

For a WPSP with I job clusters and K machines, containing a total of $N_I = J_0 + J_1 + J_2 + \cdots + J_I$ jobs, the integer programming model contains N_IK variables of x_{ijk}, N_IK variables of t_{ijk}, $N_IK(N_I - 1)$ variables of $y_{iji'j'k}$, and $N_IK(N_I - 1)$ variables of $z_{iji'j'k}$ (including $z_{iji^*j^*k}$). Further, the constraint set in equation (1) contains N_I equations, the constraint set in equation (2) contains K equations, the constraint sets in equations (3) and (14) each contains K equations, constraint sets in equations (8)–(12) each contains $N_IK\ (N_I - 1)/2$ equations, the constraint sets in equations (4), (5), and (13) each contains $N_IK(N_I - 1)$ equations, the constraints in equation (15) contains $N_IK(N_I - 1)(N_I - 2)$ equations, and the constraint sets in equations (6) and (7) each contains N_IK equations. Thus, the total number of variables is $2N_I^2K$, and the total number of equations is $K_I^3K + (5/2)N_I^2K - (3/2)N_IK + N_I + 3K$.

5. Solutions for the WPSP

The integer programming model is implemented using the software CPLEX to solve the WPSP case with 20 jobs described in section 3. The constraints and variables in the model are generated using a C++ programming code. For the WPSP case with 4 machines, 11 job clusters, and 20 jobs, the model contains 4576 variables and 249784 equations. In solving the integer programming problem, the depth-search strategy is implemented by choosing the most recently created node, incorporating with the strong branching rule causing variable selection based on partially solving a number of subproblems with tentative branches to find the most promising branch.

Table 4. The run times and workloads for the WPSP with various node limits.

Node limit	Run time	Workload
2.03E02	25.42	13940
1.08E05	5066.40	13830

The implementation thus allows various limits on the number of memory nodes to be set so that feasible solutions may be obtained efficiently within reasonable amount of computer time. Table 4 shows the two feasible solutions, the corresponding memory node limits and run times on a Pentium-II 350 PC. The first feasible solution of the integer programming model requires a total workload of 13940 minutes, where run time is 25.42 minutes with node limits setting to 2.03E02. The second feasible solution of the integer programming model requires a total workload of 13830 minutes, where run time is 5066.40 minutes with node limits setting to 1.08E05.

Table 5 displays the output of all variables in the first feasible solution of the integer programming model,

Table 5. The feasible solution for the 20-job WPSP solved by CPLEX

The objective value and the solution time

Node limit, integer feasible: Objective = 1.3940000000e + 002
Solution time = 1525.17 sec. Iterations = 8107 Nodes = 210

The statistics of the model

Constraints:	63160	[Less: 8932, Greater: 54192, Equal: 36]
Variables:	4576	[Nneg: 96, Binary: 4480]
Constraint nonzeros:	249784	
Objective nonzeros:	2048	
RHS nonzeros:	59656	

The values for all variables

Name	Value	Name	Value	Name	Value	Name	Value
X0011	1.00	X0924	1.00	Y0220113	1.00	Y0810912	1.00
X0311	1.00	X0934	1.00	Y0610113	1.00	Y0930924	1.00
X0411	1.00	X1024	1.00	Y1010113	1.00	Y1020924	1.00
X0511	1.00	X1114	1.00	Y0120324	1.00	Y0921114	1.00
X0521	1.00	X1124	1.00	Y0120924	1.00	Y0921124	1.00
X0711	1.00	Z0040124	1.00	Y0120934	1.00	Y0931024	1.00
Z0010411	1.00	Z0120324	1.00	Y0121024	1.00	Y0931114	1.00
Z0310711	1.00	Z0320934	1.00	Y0121114	1.00	Y0931124	1.00
Z0410511	1.00	Z0921114	1.00	Y0121124	1.00	Y1021114	1.00
Z0510521	1.00	Z0931024	1.00	Y0210622	1.00	Y1021124	1.00
Z0520311	1.00	Z1020924	1.00	Y0210812	1.00	Y1111124	1.00
X0022	1.00	Z1111124	1.00	Y0210912	1.00	T0311	9.26
X0212	1.00	Y0010311	1.00	Y0220613	1.00	T0411	0.40
X0622	1.00	Y0010411	1.00	Y0221013	1.00	T0511	3.02
X0812	1.00	Y0010511	1.00	Y0410311	1.00	T0521	5.79
X0912	1.00	Y0010521	1.00	Y0510311	1.00	T0711	15.17
Z0020212	1.00	Y0010711	1.00	Y0520311	1.00	T0212	0.70
Z0210622	1.00	Y0020212	1.00	Y0310711	1.00	T0622	12.58
Z0620812	1.00	Y0020622	1.00	Y0320924	1.00	T0812	17.38
Z0810912	1.00	Y0020812	1.00	Y0320934	1.00	T0912	21.48
X0033	1.00	Y0020912	1.00	Y0321024	1.00	T0113	21.99
X0113	1.00	Y0030113	1.00	Y0321114	1.00	T0223	0.70
X0223	1.00	Y0030223	1.00	Y0321124	1.00	T0613	12.58
X0613	1.00	Y0030613	1.00	Y0410511	1.00	T1013	17.38
X1013	1.00	Y0031013	1.00	Y0410521	1.00	T0124	0.70
Z0030223	1.00	Y0040124	1.00	Y0410711	1.00	T0324	13.80
Z0220613	1.00	Y0040324	1.00	Y0510521	1.00	T0924	26.95
Z0611013	1.00	Y0040924	1.00	Y0510711	1.00	T0934	20.01
Z1010113	1.00	Y0040934	1.00	Y0520711	1.00	T1024	23.44
X0044	1.00	Y0041024	1.00	Y0611013	1.00	T1114	31.48
X0124	1.00	Y0041114	1.00	Y0620812	1.00	T1124	37.33
X0324	1.00	Y0041124	1.00	Y0620912	1.00		

All other variables in the range 1–4576 are zero.

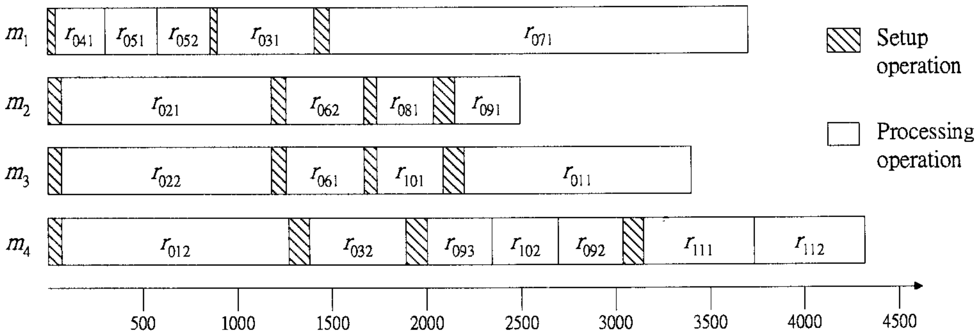


Figure 5. A feasible schedule for the 20-job WPSP obtained by solving the integer programming model.

Table 6. The starting times set for the 20 jobs in the WPSP solution.

The starting times	Values	The starting times	Values
t_{0011}	0	t_{0223}	70 ($t_{0223} = t_{0033} + s_{0002}$)
t_{0411}	40 ($t_{0411} = t_{0011} + s_{0004}$)	t_{0613}	1258 ($t_{0613} = t_{0223} + n_{022}p_{02} + s_{0206}$)
t_{0511}	302 ($t_{0511} = t_{0411} + n_{041}p_{04}$)	t_{1013}	1738 ($t_{1013} = t_{0613} + n_{061}p_{06} + s_{0610}$)
t_{0521}	579 ($t_{0521} = t_{0511} + n_{051}p_{05}$)	t_{0113}	2199 ($t_{0113} = t_{1013} + n_{101}p_{10} + s_{1001}$)
t_{0311}	926 ($t_{0311} = t_{0521} + n_{052}p_{05} + s_{0503}$)	t_{0044}	0
t_{0711}	1517 ($t_{0711} = t_{0311} + n_{031}p_{03} + s_{0307}$)	t_{0124}	70 ($t_{0124} = t_{0044} + s_{0001}$)
t_{0022}	0	t_{0324}	1380 ($t_{0324} = t_{0124} + n_{012}p_{01} + s_{0103}$)
t_{0212}	70 ($t_{0212} = t_{0022} + s_{0002}$)	t_{0934}	2001 ($t_{0934} = t_{0324} + n_{032}p_{03} + s_{0309}$)
t_{0622}	1258 ($t_{0622} = t_{0212} + n_{021}p_{02} + s_{0206}$)	t_{1024}	2344 ($t_{1024} = t_{0934} + n_{093}p_{09} + s_{0910}$)
t_{0812}	1738 ($t_{0812} = t_{0622} + n_{062}p_{06} + s_{0608}$)	t_{0924}	2695 ($t_{0924} = t_{1024} + n_{102}p_{10} + s_{1009}$)
t_{0912}	2148 ($t_{0912} = t_{0812} + n_{081}p_{08} + s_{0809}$)	t_{1114}	3148 ($t_{1114} = t_{0924} + n_{092}p_{09} + s_{0911}$)
t_{0033}	0	t_{1124}	3733 ($t_{1124} = t_{1114} + n_{111}p_{11}$)

which is reformed to the corresponding machine schedules, as shown in figure 5.

The variables X0011 = 1, X0411 = 1, X0511 = 1, X0521 = 1, X0311 = 1, and X0711 = 1 indicate that the jobs r_{001}, r_{041}, r_{051}, r_{052}, r_{031}, and r_{071} are scheduled on machine m_1. The variables Z00104111 = 1, Z0410511 = 1, Z0510521 = 1, Z0520311 = 1, and Z0310711 = 1 imply that job r_{001} proceeds job r_{041} directly, job r_{041} proceeds job r_{051} directly, job r_{051} proceeds job r_{052} directly, job r_{052} proceeds job r_{031} directly, and job r_{031} proceeds job r_{071} directly. Thus, there are three product type changes, first from R_{00} (r_{001}) to R_{04} (r_{041}), secondly from R_{05} (r_{052}) to R_{03} (r_{031}), and finally from R_{03} (r_{031}) to R_{07} (r_{071}). The starting processing time (t_{ijk}) for the 20 jobs are tabulated in table 6.

6. An algorithm for the WPSP

For small and moderate size of WPSP, the integer programming model provides optimal or near-optimal solutions within reasonable amount of computer time. For large size of WPSP, solving the integer programming model is computationally inefficient. Therefore, in the following, a Weighted-Saving algorithm is presented to generate feasible solutions for the WPSP efficiently. The algorithm takes the merits of the well-known savings function defined by Clark and Wright (1964) with some modifications, which considers the setup time saving when combing two single-job schedules. However, only considering a combination with larger setup time saving may cause too much advancing in job starting time. Therefore, the Weighted-Saving algorithm considers both the setup time savings and starting time slackness savings when combing two single-job schedules, to achieve the benefits of setup time and job slackness reduction.

The Weighted-Saving algorithm, initially, calculates the weighted savings of all pairs of jobs and creates a list by sorting the weighted savings in descending order of their magnitudes. The algorithm then selects the first feasible pair of jobs from the top of the list to start a new schedule (initialization of the first schedule). Note that a selected pair of jobs is feasible and will be added to the machine schedule if it does not violate the machine capacity constraints and the job due date restrictions. Starting from the top of the savings list, the Weighted-

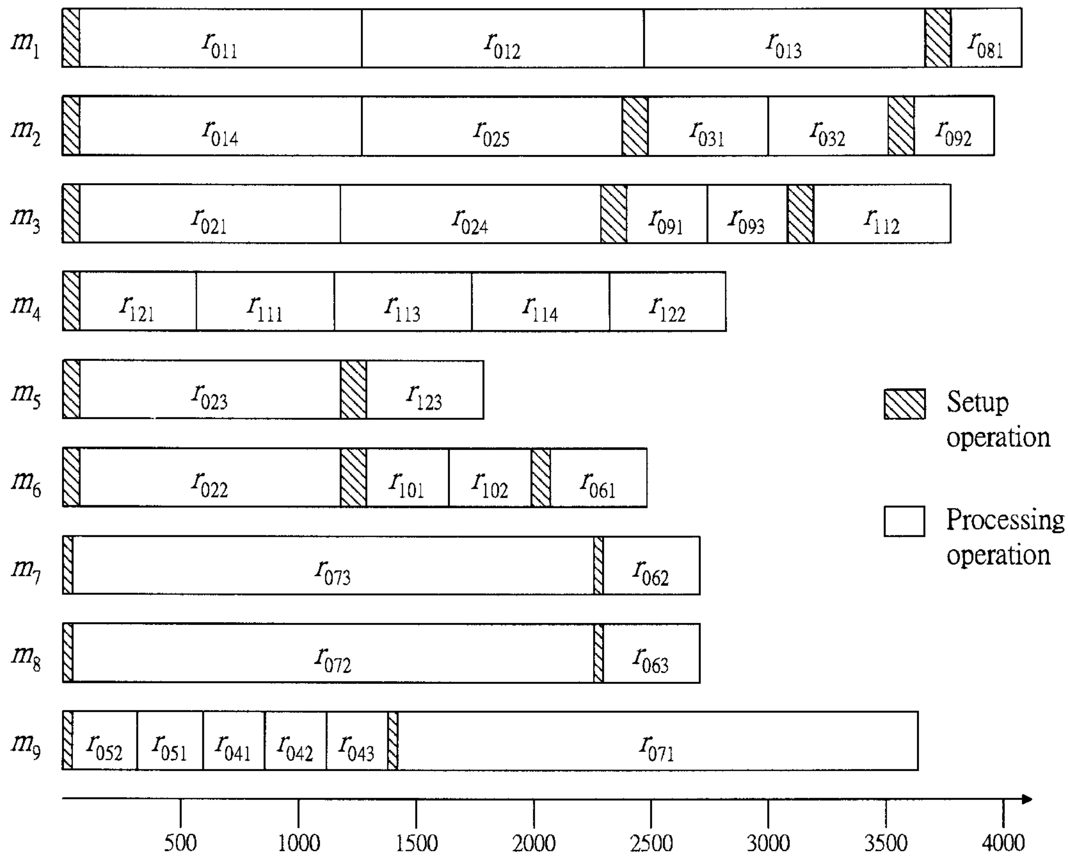


Figure 6. A feasible schedule for the 35-jobWPSP solved by the Weighted-Savings algorithm.

Saving algorithm expands the schedule by finding the first feasible pair of jobs on the list then adding it to either one of the two ends of the schedule. If the current schedule cannot be expanded, choose the first feasible pair of jobs from the top of the list to start another new schedule. Repeat such steps until all jobs are scheduled.

The weighted saving $WSA_{iji'j'}$ considered in the algorithm is defined as $WSA_{iji'j'} = \alpha(s_{i'\mathrm{U}} + s_{i'\mathrm{U}} - s_{ii'}) + (1-\alpha)(e_{i'j'} - s_{\mathrm{U}i} - n_{ij}p_i - s_{ii'})$, for all pairs of jobs r_{ij} and $r_{i'j'}$, where α represents the weight of setup time savings, $s_{ii'}$ represents the setup time between any two consecutive jobs r_{ij} and $r_{i'j'}$, $e_{i'j'}$, represents the latest starting time of job $r_{i'j'}$, and $n_{ij}p_i$ represents the processing time of job r_{ij} on machine m_i.

The Weighted-Savings algorithm is implemented in Visual Basic programming language. The Weighted-Saving algorithm runs quite efficiently. In fact, for the WPSP case investigated with 35 jobs, the algorithm takes less than 2 CPU seconds to obtain a feasible solution, where the total machine workload is 27949. The machine schedules of the feasible solution are depicted in figure 6. The starting processing time (t_{ijk}) for the 35 jobs are tabulated in table 7.

7. Conclusion

In this paper, a case study on the WPSP has been presented, which is taken from a wafer probing shop floor in an IC manufacturing factory. For the WPSP case investigated, the jobs are clustered by their product types, which are processed on groups of identical parallel machines and must be completed before the due dates. The job processing time depends on the product type, and the machine setup time is sequentially dependent on the orders of jobs processed. The WPSP is formulated as an integer programming model to minimize the total machine workload. To demonstrate the applicability of the integer programming model, a real-world WPSP was applied using the powerful CPLEX with effective implementation strategies, so that solutions may be obtained within reasonable amount of time. In addition, an efficient solution procedure was proposed called the Weighted-Saving algorithm to solve the WPSP near-optimally. The model and algorithms developed for WPSP in this paper can be applied to a wide class of scheduling problems such as the die mounting and wire bonding scheduling problems in IC packaging factories,

Table 7. The starting times set for the 35 jobs in the WPSP solution.

The starting times	Values	The starting times	Values
t_{0111}	70 ($t_{0111} = s_{0001}$)	t_{1224}	2322 ($t_{1224} = t_{1144} + n_{114}p_{11}$)
t_{0121}	1270 ($t_{0121} = t_{0111} + n_{011}p_{01}$)	t_{0235}	70 ($t_{0235} = s_{0002}$)
t_{0131}	2470 ($t_{0131} = t_{0121} + n_{012}p_{01}$)	t_{1235}	1288 ($t_{1235} = t_{0235} + n_{023}p_{02} + s_{0212}$)
t_{0811}	3780 ($t_{0811} = t_{0131} + n_{013}p_{01} + s_{0108}$)	t_{0226}	70 ($t_{0226} = s_{0002}$)
t_{0142}	70 ($t_{0142} = s_{0001}$)	t_{1016}	1288 ($t_{1016} = t_{0226} + n_{022}p_{02} + s_{0210}$)
t_{0252}	1270 ($t_{0252} = t_{0142} + n_{014}p_{01}$)	t_{1026}	1639 ($t_{1026} = t_{1016} + n_{101}p_{10}$)
t_{0312}	2488 ($t_{0312} = t_{0252} + n_{025}p_{02} + s_{0203}$)	t_{0616}	2070 ($t_{0616} = t_{1026} + n_{102}p_{10} + s_{1006}$)
t_{0322}	2999 ($t_{0322} = t_{0312} + n_{031}p_{03}$)	t_{0737}	40 ($t_{0737} = s_{0007}$)
t_{0922}	3620 ($t_{0922} = t_{0322} + n_{032}p_{03} + s_{0309}$)	t_{0627}	2295 ($t_{0627} = t_{0737} + n_{072}p_{07} + s_{0706}$)
t_{0213}	70 ($t_{0213} = s_{0002}$)	t_{0728}	40 ($t_{0728} = s_{0007}$)
t_{0243}	1178 ($t_{0243} = t_{0213} + n_{021}p_{02}$)	t_{0638}	2295 ($t_{0638} = t_{0728} + n_{072}p_{07} + s_{0706}$)
t_{0913}	2396 ($t_{0913} = t_{0243} + n_{024}p_{02} + s_{0209}$)	t_{0529}	40 ($t_{0529} = s_{0005}$)
t_{0933}	2739 ($t_{0933} = t_{0913} + n_{091}p_{09}$)	t_{0519}	317 ($t_{0519} = t_{0529} + n_{052}p_{05}$)
t_{1123}	3192 ($t_{1123} = t_{0933} + n_{093}p_{09} + s_{0911}$)	t_{0419}	594 ($t_{0419} = t_{0519} + n_{051}p_{05}$)
t_{1214}	70 ($t_{1214} = s_{0012}$)	t_{0429}	856 ($t_{0429} = t_{0419} + n_{041}p_{04}$)
t_{1114}	567 ($t_{1114} = t_{1214} + n_{121}p_{12}$)	t_{0439}	1118 ($t_{0439} = t_{0429} + n_{042}p_{04}$)
t_{1134}	1152 ($t_{1134} = t_{1114} + n_{111}p_{11}$)	t_{0719}	1420 ($t_{0719} = t_{0439} + n_{043}p_{04} + s_{0407}$)
t_{1144}	1737 ($t_{1144} = t_{1134} + n_{113}p_{11}$)		

the knitting machine scheduling problem in textile factories, which include critical process containing parallel machines.

Acknowledgements

This paper was supported in part by the National Science Council, Taiwan, R. O. C., under the contract NSC 89-2213-E-009-034 .

References

Centeno, G., and Armacost, R. L., 1997, Parallel machine scheduling with release time and machine eligibility restrictions. *Computers and Industrial Engineering*, **33**(1–2), 273–276.

Chen, T. R., Chang, T. S., Chen, C. W., and Kao, J., 1995, Scheduling for IC sort and test with preemptiveness via Lagrangian relaxation. *Transactions on Systems, Man, and Cybernetics*, **25**(8), 1249–1256.

Cheng, R., and Gen, M., 1997, Parallel machine scheduling problem using memetic algorithms. *Computers and Industrial Engineering*, **33**(3–4), 761–764.

Clark, G., and Wright, J., 1964, Scheduling vehicles from a central depot to a number of delivery points. *Operations Research*, **12**, 568.

Gabrel, V., 1995, Scheduling jobs within time windows on identical parallel machines: new model and algorithms. *European Journal of Operational Research*, **83**, 320–329.

Ho, J. C., and Chang, Y. L., 1995, Minimizing the number of tardy jobs for m parallel machines. *European Journal of Operational Research*, **84**, 343–355.

Lee, Y. H., and Pinedo, M., 1997, Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, **100**, 464–474.

Ovacik, I. M., and Uzsoy, R., 1995, Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup time. *International Journal of Production Research*, **33**(11), 3173–3192.

Ovacik, I. M., and Uzsoy, R., 1996, Decomposition methods for scheduling semiconductor testing facilities. *The International Journal of Flexible Manufacturing Systems*, **8**, 357–388.

Park, M. W., and Kim, Y. D., 1997, Search heuristics for a parallel machine scheduling problem with ready time and due dates. *Computers and Industrial Engineering*, **33**(3–4), 793–796.

Parker, R. G., Deane, R. H., and Holmes, R. A., 1977, On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent changeover costs. *AIIE Transactions*, **9**(2), 155–160.

Ruiz-Torres, A. J., Enscore, E. E., and Barton, R. R., 1997, Simulated annealing heuristics for the average flow-time and the number of tardy jobs bi-criteria identical parallel machine problem. *Computers and Industrial Engineering*, **33**(1–2), 257–260.

Schutten, J. M. J., and Leussink, R. A. M., 1996, Parallel machine scheduling with release dates, due dates and family setup times. *International Journal of Production Economics*, **46–47**, 119–125.