IIE Transactions
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uiie20

Minimizing the total machine workload for the wafer probing scheduling problem

W.L. PEARN [a] , S.H. CHUNG [a] & M.H. YANG [a]
[a] Department of Industrial Engineering and Management , National Chiao Tung University , Hsinchu, Taiwan, ROC E-mail:
Published online: 17 Apr 2007.

To cite this article: W.L. PEARN , S.H. CHUNG & M.H. YANG (2002) Minimizing the total machine workload for the wafer probing scheduling problem, IIE Transactions, 34:2, 211-220, DOI: 10.1080/07408170208928862

To link to this article: http://dx.doi.org/10.1080/07408170208928862

IIE Transactions (2002) **34**, 211–220

Minimizing the total machine workload for the wafer probing scheduling problem

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

Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan, ROC
E-mail: roller@cc.nctu.edu.tw

Received November 1999 and accepted March 2001

The Wafer Probing Scheduling Problem (WPSP) is a practical generalization of the parallel-machine scheduling problem, which has many real-world applications, particularly, in the Integrated Circuit (IC) manufacturing industry. In the wafer probing factories, the jobs are clustered by their product types, which must be processed on groups of identical parallel machines and 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. 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, it is more difficult to solve than the classical parallel-machine scheduling problem. In this paper, we consider the WPSP and formulate the WPSP as an integer programming problem to minimize the total machine workload. We demonstrate the applicability of the integer programming model by solving a real-world example taken from a wafer probing shop floor in an IC manufacturing factory.

1. Introduction

Classical parallel-machine scheduling problems have been categorized (Cheng and Sin, 1990) into three types according to the job processing time characteristics, including the identical parallel-machine scheduling problem (Gabrel, 1995; Suer *et al.*, 1997), the uniform parallel-machine scheduling problem (Guinet, 1991; Randhawa and Kuo, 1997), and the unrelated parallel-machine scheduling problem (Piersma and Dijk, 1996; Herrmann *et al.*, 1997). For the identical parallel-machine scheduling problem, each job requires only a single operation, which may be processed on any of the parallel machines with an identical processing time. For the uniform parallel-machine scheduling problem, the job processing times are determined by the efficiencies of the machines and the job processing times are all the same on each single machine. For the unrelated parallel-machine scheduling problem, which is a generalization of the uniform parallel-machine scheduling problem, the efficiency of the machine depends on the type of jobs processed and the processing times of different jobs on the same machine may not be equal.

The Wafer Probing Scheduling Problem (WPSP) is a variation of the parallel-machine scheduling problem considered by Ovacik and Uzsoy (1995, 1996) and Centeno and Armacost (1997), which has many real-world applications, particularly, in the Integrated Circuit (IC) manufacturing industry. In a wafer probing factory, 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 times for two consecutive jobs of different product types (job clusters) on the same machine are sequentially dependent. 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, it is more difficult to solve than the classical parallel-machine scheduling problem.

Parker *et al.* (1977) considered a simple version of the WPSP with each job-cluster containing only one job, and provided a mathematical model. Parker *et al.* (1977) also presented a heuristic algorithm to solve the problem approximately. Unfortunately, their model does not consider the due date restriction that is essential and critical in practical situations. In fact, their formulation only includes the processing time without considering the setup time (changeover cost) in the machine capacity constraints, which may not reflect the real situations accurately.

Ovacik and Uzsoy (1996) presented another version of the WPSP with a different objective function. Ovacik and Uzsoy (1996) provided a class of heuristic procedures to

*Corresponding author

subject to

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

$$x_{0kk} = 1, \quad \text{for all } k, \tag{2}$$

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

$$(y_{iji'j'k} + y_{i'j'ijk}) - \mathrm{Q}(x_{ijk} + x_{i'j'k} - 2) \geq 1,$$
$$\text{for all } i,j,k, \tag{4}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) + \mathrm{Q}(x_{ijk} + x_{i'j'k} - 2) \leq 1,$$
$$\text{for all } i,j,k, \tag{5}$$

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

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

$$(y_{iji'j'k} + y_{i'j'ijk}) - \mathrm{Q}(x_{ijk} + x_{i'j'k} + 1) \leq 0,$$
$$\text{for all } i,j,k, \tag{8}$$

$$y_{iji'j'k} \geq z_{iji'j'k} \quad \text{for all } i,j,k, \tag{9}$$

$$\sum_{i=0}^{I}\sum_{j=1}^{J_i} x_{ijk} - \sum_{r_{ij}\neq r_{i'j'}} z_{iji'j'k} = 1, \quad \text{for all } k, \tag{10}$$

$$t_{ijk} + n_{ij}p_i + s_{ii'} - t_{i'j'k} + \mathrm{Q}(y_{iji'j'k} - 1) \leq 0,$$
$$\text{for all } i,j,k, \tag{11}$$

$$t_{ijk} + n_{ij}p_i + s_{ii'} - t_{i'j'k} - \mathrm{Q}(y_{iji'j'k} + z_{iji'j'k} - 2) \geq 0,$$
$$\text{for all } i,j,k, \tag{12}$$

$$y_{iji''j''k} + z_{iji''j''k} - \mathrm{Q}(y_{iji''j''k} + z_{iji''j''k} - 2)$$
$$- \mathrm{Q}(y_{iji'j'k} - z_{iji'j'k} - 1) \geq 2, \quad \text{for all } i,j,k, \tag{13}$$

$$t_{ijk} \geq b_{ij}x_{ijk}, \quad \text{for all } i,j,k, \tag{14}$$

$$t_{ijk} \leq e_{ij}x_{ijk}, \quad \text{for all } i,j,k, \tag{15}$$

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

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

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

The objective function seeks to minimize the sum of the total processing time $\sum_{i=0}^{I}\sum_{j=1}^{J_i} x_{ijk}n_{ij}p_i$ and the total setup time $\sum_{i=0}^{I}\sum_{j=1}^{J_i}(\sum_{i'=0}^{I}\sum_{j'=1}^{J_{i'}} z_{iji'j'k}s_{ii'})$ over the K machines. The constraints in (1) guarantee that job r_{ij} is processed by one machine exactly once. The constraints in (2) guarantee that only one pseudo-job r_{0j} is scheduled on a machine. The constraints in (3) state that each machine workload does not exceed the machine capacity. The constraints in (4) and (5) 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 (4) and (5) are satisfied for $x_{ijk} + x_{i'j'k} - 2 < 0$. The constraints in (6) 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 (7) and (8) 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. The constraints in (7) 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 (8) 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 (9) 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 (10) state that there should exist $n - 1$ direct-precedence variables, which are set to one, on the schedule with n jobs. The constraints in (11) ensure the satisfaction of the inequality $t_{ijk} + n_{ij}p_i + s_{ii'} \leq t_{i'j'k}$, if the jobs r_{ij} preceding 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 (11) are satisfied when $y_{iji'j'k}$ is equal to zero or one. For example, we can choose $\mathrm{Q} = \sum_{i=1}^{I}\sum_{j=1}^{J_i}(n_{ij}p_i + \max_{i'}\{s_{ii'}\})$. The constraints in (12) ensure the satisfaction of the inequality $t_{ijk} + n_{ij}p_i + s_{ii'} \geq t_{i'j'k}$ if the jobs r_{ij} preceding job $r_{i'j'}$ directly $(y_{iji'j'k} + z_{iji'j'k} - 2 = 0)$. Therefore, the constraints in (11) and (12) 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 (13) 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 $y_{iji''j''k} + z_{iji''j''k} \geq 2$. The constraints in (14) and (15) 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}.

For a parallel-machine problem 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 $N_I K$ variables of x_{ijk}, $N_I K$ variables of t_{ijk}, $N_I K$ $(N_I - 1)$ variables of $y_{iji'j'k}$, and $N_I K$ $(N_I - 1)$ variables of $z_{iji'j'k}$ (including $z_{iji''j''k}$). Further, the constraint set in (1) contains N_I equations, the constraint set in (2) contains K equations, the constraint sets in (3) and (10) each contains K equations, constraint sets in (4) $\sim$ (8) each contains

$N_l K(N_l - 1)/2$ equations, the constraint sets in (9), (11), and (12) each contains $N_l K$ $(N_l - 1)$ equations, the constraints in (13) contains $N_l K$ $(N_l - 1)(N_l - 2)$ equations, and the constraint sets in (14) and (15) each contains $N_l K$ equations. Thus, the total number of variables is $2N_l^2 K$, and the total number of equations is $N_l^3 K + (5/2)N_l^2 K - (3/2)N_l K + N_l + 3K$.

4. Solutions for the WPSP

To solve the integer programming problem for the WPSP example described in Section 2, we write a C++ programming code to generate the constraints and variables of the model. For the WPSP example with two machines, three job clusters, and seven jobs, the model contains 324 variables and 1851 equations. We run the integer programming model using the IP software CPLEX 4.0 on a Pentium II 266 MHz PC. Table 3 displays the output solution of the integer programming model.

The variables X011 = 1, X111 = 1, X311 = 1, and X321 = 1 indicate that the jobs r_{01}, r_{11}, r_{31}, and r_{32} are scheduled on machine m_1. The variables Z01321 = 1, Z32311 = 1, and Z31111 = 1 imply that job r_{01} precedes job r_{32} directly, job r_{32} precedes job r_{31} directly, and job r_{31} precedes job r_{11} directly. Thus, there are two product type changes, one from R_0 (r_{01}) to R_3 (r_{32}) and the other one from R_3 (r_{31}) to R_1 (r_{11}). The starting processing times (t_{ijk}) for the jobs on machine m_1 are shown in Table 4.

The variables X022 = 1, X122 = 1, X212 = 1, X222 = 1, and X232 = 1 indicate that jobs r_{02}, r_{12}, r_{21}, r_{22}, and r_{23} are scheduled on machine m_2. The variables Z02122 = 1, Z12232 = 1, Z23222 = 1, and Z22212 = 1 imply that job r_{02} precedes job r_{12} directly, job r_{12} precedes job r_{23} directly, job r_{23} precedes job r_{22} directly, and job r_{22} precedes job r_{21} directly. Thus, there are two product type changes, one from R_0 (r_{02}) to R_1 (r_{12}) and the other one from R_1 (r_{12}) to R_2 (r_{23}). The starting processing times (t_{ijk}) for the jobs on machine m_2 are shown in Table 5. We note that the integer programming solution of the problem is indeed identical to that depicted in Section 2.

5. A real-world application

To demonstrate the applicability of the integer programming model in real situations, we consider the following example taken from a wafer probing shop-floor in an IC manufacturing factory located in the Science-based

Table 3. The integer programming solution (optimal) for the WPSP example in Section 2 solved by CPLEX 4.0

The objective value and the solution time

Integer optimal solution: Objective = 1.430 000 0000e + 002
Solution time = 152.36 seconds Iterations = 292 758 Nodes = 29 417

The statistics of the model

Constraints:	1851	[Less: 452, Greater: 1386, Equal: 13]
Variables:	324	[Nneg: 18, Binary: 306]
Constraint nonzeros:	7108	
Objective nonzeros:	106	
RHS nonzeros:	1599	

The values for all variables

Name	*Value*	*Name*	*Value*	*Name*	*Value*
X011	1.000 000	Z12232	1.000 000	Y12222	1.000 000
X111	1.000 000	Z22212	1.000 000	Y12232	1.000 000
X311	1.000 000	Z23222	1.000 000	Y22212	1.000 000
X321	1.000 000	Y01111	1.000 000	Y23212	1.000 000
Z01321	1.000 000	Y01311	1.000 000	Y23222	1.000 000
Z31111	1.000 000	Y01321	1.000 000	Y32311	1.000 000
Z32311	1.000 000	Y02122	1.000 000	T111	33.000 000
X022	1.000 000	Y02212	1.000 000	T311	20.000 000
X122	1.000 000	Y02222	1.000 000	T321	10.000 000
X212	1.000 000	Y02232	1.000 000	T122	10.000 000
X222	1.000 000	Y31111	1.000 000	T212	69.000 000
X232	1.000 000	Y32111	1.000 000	T222	54.000 000
Z02122	1.000 000	Y12212	1.000 000	T232	39.000 000

All other variables in the range 1–324 are zero

Table 4. The starting times for the jobs on machine m_1

	Starting time
t_{011}	0
t_{321}	10 ($t_{321} = t_{011} + s_{03}$)
t_{311}	20 ($t_{311} = t_{321} + n_{32}p_3$)
t_{111}	33 ($t_{111} = t_{311} + n_{31}p_3 + s_{31}$)

Table 5. The starting times for the jobs on machine m_2

	Starting time
t_{022}	0
t_{122}	10 ($t_{122} = t_{022} + s_{01}$)
t_{232}	39 ($t_{232} = t_{122} + n_{12}p_1 + s_{12}$)
t_{222}	54 ($t_{222} = t_{232} + n_{23}p_2$)
t_{212}	69 ($t_{212} = t_{222} + n_{22}p_2$)

Industrial Park at Hsinchu, Taiwan. For the case we investigated, there are six test codes (product type) being processed on three identical testers arranged in parallel with each tester connected to the same type of prober, at which the wafer lot is positioned and tested, as shown in Fig. 4. We note that the test codes are the program executed in the tester when the wafer is probed with a specific probe card fixed on the prober.

This real example contains 10 wafer lots with due dates and processing times, which should be tested under certain levels of temperature with six test codes and five probe cards, as shown in Table 6. The product type of a wafer lot is determined by the code used during the testing operations. Thus, there are six product types and 10 jobs in this example. These jobs are to be completed on the three parallel testers within 3 days. Therefore, the machine capacity is set to 4320 minutes. We have set the "minute" as the unit of the processing time, setup time, due date, machine workload, and machine capacity in our investigation.

Before the testing of a job, setup operations including probe card change and temperature setting are needed. The time required to change a probe card can be regarded as a fixed constant. In the case where the previous job is tested under a high temperature, we would have to wait for the temperature to fall. On the other hand, if the next job is to be tested at high temperature whilst the current one is a room temperature test then we would have to wait for the temperature to increase. Thus, the setup time required for switching one product type to another depends on the probe card and the testing temperature as is shown in Table 7. The time to change a probe card is 40 minutes in this case. The time to increase the temperature to the required level is 30 minutes and the time to decrease the temperature is 40 minutes.

Obtaining the optimal solution, as shown in Fig. 5, with a total load of 9486 for the real example requires roughly 5.42 hours, (see Table A1 in the Appendix). However, in solving the integer programming problem, we implement a depth-first search strategy by choosing the most recently created node, incorporating a strong branching rule causing variable selection based on partially solving a number of sub-problems with tentative branches to find the most promising branch. The depth-first search strategy looks for a quick good solution by following and searching good lower bounds right to the bottom of the tree. 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. Table 8 shows various memory node limits, the corresponding feasible solutions, run times, and solution quality in terms of the deviation from the optimality. Tables A2–A4 (see Appendix) show the output solutions for the real example with different memory node limits. The feasible solutions obtained are remarkably good.

6. Conclusion

In this paper, we considered the Wafer Probing Scheduling Problem (WPSP), a variation of the parallel-machine scheduling problem, which has many real-world applica-

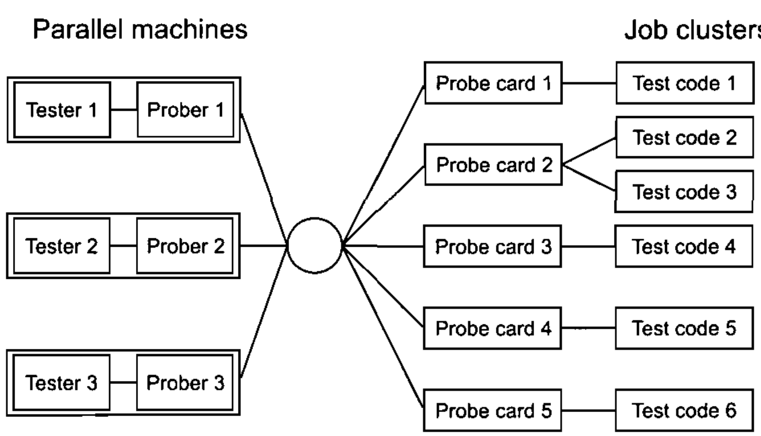

Fig. 4. The relationships between the six test codes, five probe cards, three probers and testers.

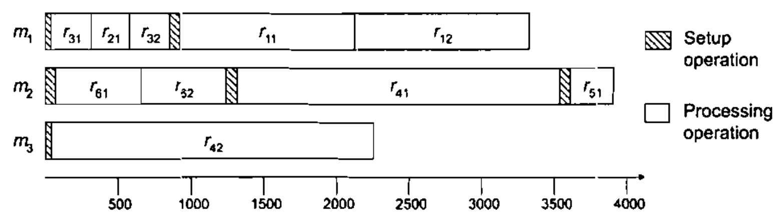

Fig. 5. The optimal schedule for the application example.

Table 6. The product types, probe card, testing temperatures, processing times, and due dates for the 10 jobs in the real example

Job I.D.	*Product type*	*Probe card*	*Testing temperature*	*Processing time*	*Due date*
1	1	1	High	1200	4320
2	1	1	High	1200	4320
3	2	2	Room	262	1440
4	3	2	Room	277	1440
5	3	2	Room	277	1440
6	4	3	Room	2215	4320
7	4	3	Room	2215	4320
8	5	4	High	300	4320
9	6	5	High	585	1440
10	6	5	High	585	4320

Table 7. Setup times required for switching one product type to another in the real example

To / *From*	*0*	*1*	*2*	*3*	*4*	*5*	*6*
0	0	70	40	40	40	70	70
1	80	0	80	80	80	110	110
2	40	70	0	0	40	70	70
3	40	70	0	0	40	70	70
4	40	70	40	40	0	70	70
5	80	110	80	80	80	0	110
6	80	110	80	80	80	110	0

Table 8. The run times (in minutes), solutions, and solution quality for the real example with various node limits

Node limits	*Run time*	*Solution*	*Deviation from optimality (%)*
1E02	0.55	96.36	1.58
1E03	3.35	95.26	0.42
1E04	30.89	94.86	0.00
1E05	291.51	94.86	0.00
1E06	324.99	94.86	–

tions. The WPSP involves constraints on job clusters, job-cluster dependent processing time, due dates, machine capacity, and a sequentially dependent setup time, which is more difficult to solve then the classical parallel-machine scheduling problem. We formulated the WPSP as an integer programming model to minimize the total machine workload. We demonstrated the applicability of the integer programming model by solving a real-world example taken from a wafer probing shop-floor in an IC manufacturing factory using the powerful software CPLEX 4.0. We also implemented the depth-first search strategy with strong branching rules to effectively solve the WPSP example investigated, to obtain the desired solution within a reasonable amount of computation time.

Acknowledgements

This paper was supported in part by the National Science Council, Taiwan, ROC, under 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, T.C.E. and Sin, C.C.S. (1990) A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, **47**, 271–292.

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

Guinet, A. (1991) Textile production systems: a succession for non-identical parallel processors shops. *Journal of the Operational Research Society*, **42**(8), 655–671.

Herrmann, J., Porth, J.M. and Sauer, N. (1997) Heuristics for unrelated machine scheduling with precedence constraints. *European Journal of Operational Research*, **102**, 528–537.

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.

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.

Piersma, N. and Dijk, W.V. (1996) A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search. *Mathematical Computer Modeling*, **24**(9), 11–19.

Randhawa, S.U. and Kuo, C.-H. (1997) Evaluating scheduling heuristics for non-identical parallel processors. *International Journal of Production Research*, **35**(4), 969–981.

Suer, G.A., Pico, F. and Santiago, A. (1997) Identical machine scheduling to minimize the number of tardy jobs when lot-splitting is allowed. *Computers and Industrial Engineering*, **33**(1, 2), 277–280.

Appendix

Table A1. The integer programming solution (optimal) for the application example solved by CPLEX 4.0

The objective value and the solution time

Integer optimal solution:	Objective = 9.486 000e+001	
Solution time = 19 499.10 seconds	Iterations = 1363 236	Nodes = 111 727

The statistics of the model

Constraints:	7825	[Less: 1446, Greater: 6357, Equal: 22]
Variables:	1014	[Nneg: 39, Binary: 975]
Constraint nonzeros:	30 594	
Objective nonzeros:	444	
RHS nonzeros:	7045	

The values for all variables

Name	*Value*	*Name*	*Value*	*Name*	*Value*
X011	1.000 000	X033	1.000 000	Y31211	1.000 000
X111	1.000 000	X423	1.000 000	Y21321	1.000 000
X121	1.000 000	Z03423	1.000 000	Y31321	1.000 000
X211	1.000 000	Y01111	1.000 000	Y41512	1.000 000
X311	1.000 000	Y01121	1.000 000	Y61412	1.000 000
X321	1.000 000	Y01211	1.000 000	Y62412	1.000 000
Z01311	1.000 000	Y01311	1.000 000	Y61512	1.000 000
11121	1.000 000	Y01321	1.000 000	Y62512	1.000 000
Z21321	1.000 000	Y02412	1.000 000	Y61622	1.000 000
Z31211	1.000 000	Y02512	1.000 000	T111	9.260 000
Z32111	1.000 000	Y02612	1.000 000	T121	21.260 000
X022	1.000 000	Y02622	1.000 000	T211	3.170 000
X412	1.000 000	Y03423	1.000 000	T311	0.400 000
X512	1.000 000	Y11121	1.000 000	T321	5.790 000
X612	1.000 000	Y21111	1.000 000	T412	13.200 000
X622	1.000 000	Y31111	1.000 000	T512	36.050 000
Z02612	1.000 000	Y32111	1.000 000	T612	0.700 000
Z41512	1.000 000	Y21121	1.000 000	T622	6.550 000
Z61622	1.000 000	Y31121	1.000 000	T423	0.400 000
Z62412	1.000 000	Y32121	1.000 000		

All other variables in the range 1–1014 are zero

Table A2. A feasible solution for the application example solved by CPLEX 4.0 with the node limit set to 1E02

The objective value and the solution time

Node limit, integer feasible: Objective = 9.636 000 0000e+001
Solution time = 32.95 seconds Iterations = 1818 Nodes = 100

The values for all variables

Name	*Value*	*Name*	*Value*	*Name*	*Value*
X011	1.000 000	Z31423	1.000 000	Y21321	1.000 000
X111	1.000 000	Z42623	1.000 000	Y51211	1.000 000
X121	1.000 000	Z61313	1.000 000	Y31423	1.000 000
X211	1.000 000	Y01111	1.000 000	Y61313	1.000 000
X321	1.000 000	Y01121	1.000 000	Y31623	1.000 000
X511	1.000 000	Y01211	1.000 000	Y51321	1.000 000
Z01511	1.000 000	Y01321	1.000 000	Y61423	1.000 000
Z11121	1.000 000	Y01511	1.000 000	Y42623	1.000 000
Z21321	1.000 000	Y02412	1.000 000	Y61623	1.000 000
Z32111	1.000 000	Y03313	1.000 000	T111	10.590 000
Z51211	1.000 000	Y03423	1.000 000	T121	22.590 000
X022	1.000 000	Y03613	1.000 000	T211	4.500 000
X412	1.000 000	Y03623	1.000 000	T321	7.120 000
Z02412	1.000 000	Y11121	1.000 000	T511	0.700 000
X033	1.000 000	Y21111	1.000 000	T412	0.400 000
X313	1.000 000	Y32111	1.000 000	T313	7.350 000
X423	1.000 000	Y51111	1.000 000	T423	10.520 000
X613	1.000 000	Y21121	1.000 000	T613	0.700 000
X623	1.000 000	Y32121	1.000 000	T623	33.370 000
Z03613	1.000 000	Y51121	1.000 000		

All other variables in the range 1–1014 are zero

Table A3. A feasible solution for the application example solved by CPLEX 4.0 with the node limit set to 1E03

The objective value and the solution time

Node limit, integer feasible: Objective = 9.526 000 0000e+001
Solution time = 201.14 seconds Iterations = 14 672 Nodes = 1000

The values for all variables

Name	*Value*	*Name*	*Value*	*Name*	*Value*
X011	1.000 000	Z03313	1.000 000	Y21321	1.000 000
X111	1.000 000	Z31613	1.000 000	Y31423	1.000 000
X121	1.000 000	Z61623	1.000 000	Y31613	1.000 000
X211	1.000 000	Z62423	1.000 000	Y31623	1.000 000
X321	1.000 000	Y01111	1.000 000	Y41512	1.000 000
Z01211	1.000 000	Y01121	1.000 000	Y61423	1.000 000
Z11121	1.000 000	Y01211	1.000 000	Y62423	1.000 000
Z21321	1.000 000	Y01321	1.000 000	Y61623	1.000 000
Z32111	1.000 000	Y02412	1.000 000	T111	6.490 000
X022	1.000 000	Y02512	1.000 000	T121	18.490 000
X412	1.000 000	Y03313	1.000 000	T211	0.400 000
X512	1.000 000	Y03423	1.000 000	T321	3.020 000
Z02412	1.000 000	Y03613	1.000 000	T412	0.400 000
Z41512	1.000 000	Y03623	1.000 000	T512	23.250 000
X033	1.000 000	Y11121	1.000 000	T313	0.400 000
X313	1.000 000	Y21111	1.000 000	T423	16.370 000
X423	1.000 000	Y32111	1.000 000	T613	3.870 000
X613	1.000 000	Y21121	1.000 000	T623	9.720 000
X623	1.000 000	Y32121	1.000 000		

All other variables in the range 1–1014 are zero

Table A4. A feasible solution for the application example solved by CPLEX 4.0 with the node limit set to 1E04

The objective value and the solution time

Node limit, integer feasible: Objective = 9.486 000 0000e+001
Solution time = 1853.24 seconds Iterations = 118 814 Nodes = 10 000

The values for all variables

Name	*Value*	*Name*	*Value*	*Name*	*Value*
X011	1.000 000	X033	1.000 000	Y31211	1.000 000
X111	1.000 000	X423	1.000 000	Y21321	1.000 000
X121	1.000 000	Z03423	1.000 000	Y31321	1.000 000
X211	1.000 000	Y01111	1.000 000	Y41512	1.000 000
X311	1.000 000	Y01121	1.000 000	Y61412	1.000 000
X321	1.000 000	Y01211	1.000 000	Y62412	1.000 000
Z01311	1.000 000	Y01311	1.000 000	Y61512	1.000 000
Z11121	1.000 000	Y01321	1.000 000	Y62512	1.000 000
Z21321	1.000 000	Y02412	1.000 000	Y61622	1.000 000
Z31211	1.000 000	Y02512	1.000 000	T111	9.260 000
Z32111	1.000 000	Y02612	1.000 000	T121	21.260 000
X022	1.000 000	Y02622	1.000 000	T211	3.170 000
X412	1.000 000	Y03423	1.000 000	T311	0.400 000
X512	1.000 000	Y11121	1.000 000	T321	5.790 000
X612	1.000 000	Y21111	1.000 000	T412	13.200 000
X622	1.000 000	Y31111	1.000 000	T512	36.050 000
Z02612	1.000 000	Y32111	1.000 000	T612	0.700 000
Z41512	1.000 000	Y21121	1.000 000	T622	6.550 000
Z61622	1.000 000	Y31121	1.000 000	T423	0.400 000
Z62412	1.000 000	Y32121	1.000 000		

All other variables in the range 1–1014 are zero

Biographies

Wen Lea Pearn is a Professor in the Industrial Engineering and Management Department, National Chiao-Tung University, Hsinchu, Taiwan, ROC. He received his Ph.D. degree in Operations Research from Maryland University, College Park. He worked for the Switch Network Control and Process Quality Centers at AT&T Bell Laboratories. He has published and presented research papers in the areas of networks, scheduling, and process capability.

Shu-Hsing Chung is a Professor in the Industrial Engineering and Management Department, National Chiao-Tung University, Hsinchu, Taiwan, ROC. She received a Ph.D. degree in Industrial Engineering from Texas A&M University, College Station, in 1986. 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 graduate of the Industrial Engineering and Management Department, National Chiao-Tung University, Hsinchu, Taiwan, ROC. He received an M.S. degree in Industrial Engineering and Management from the National Chiao-Tung University. He is a Lecturer in the Industrial Engineering and Management Department, National Lien Ho Institute of Technology, Miao Li, Taiwan, ROC. His research interests include production planning and scheduling.