Available online at www.sciencedirect.com
SCIENCE DIRECT®
Int. J. Production Economics 88 (2004) 257–267
international journal of production economics
www.elsevier.com/locate/dsw

A case study on the multistage IC final testing scheduling problem with reentry

W.L. Pearn[a,*], S.H. Chung[a], A.Y. Chen[a], M.H. Yang[b]

[a] Department of Industrial Engineering and Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, ROC
[b] Department of Business Management, National Lien-Ho Institute of Technology, Kung-Ching Li, Miao-Li, Taiwan, ROC

Received 8 October 2001; accepted 30 May 2003

Abstract

The integrated-circuit final testing scheduling problem (ICFTSP) with reentry is a variation of the complex flow-shop scheduling problem, which is also a generalization of the classical reentrant flow batch process problem, and the identical parallel machine problem. In this paper, we present a case study on the ICFTSP with reentry, which is taken from a final testing shop floor in an integrated circuit manufacturing factory. For the case investigated, the jobs are clustered by their product types, which must be processed on groups of parallel machines at various process stages following the manufacturing sequence, which 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 is to schedule jobs without violating all constraints, while the total machine workload is minimized. Since the ICFTSP has reentry characteristic, and involves job processing precedence, serial-processing stage, batch-processing stage, job clusters, job-cluster dependent processing time, due dates, machine capacity, and sequence dependent setup time, it is more difficult to solve than the classical flow-shop scheduling problem. We present three fast network algorithms to efficiently solve the ICFTSP with reentry and provide a performance comparison between the three algorithms on eight test problems.

Keywords: Reentrant flow lines; Complex flow shop; Parallel-machine scheduling problem; Sequence dependent setup time; Due date

1. Introduction

The integrated-circuit final testing scheduling problem (ICFTSP) with reentry is a scheduling problem related to the reentrant processes considered by Kumar (1993) and Connors et al. (1996), which is a variation of the parallel-machine scheduling problem considered by Centeno and Armacost (1997), Schutten and Leussink (1996), Ovacik and Uzsoy (1995, 1996), and Yalaoui and Chu (2002). The ICFTSP with reentry is also a practical generalization of the complex flow-shop with batch process problem, which has many real-world applications in the integrated circuit (IC) manufacturing industry. In the IC final testing factory, the jobs are clustered by their product types, which must be processed on any of the parallel machines at each process stage and be

*Corresponding author. Tel.: +886-35-714261; fax: +886-35-722392.
E-mail address: roller@cc.nctu.edu.tw (W.L. Pearn).

0925-5273/03/$ - see front matter
doi:10.1016/S0925-5273(03)00186-5

completed before the due dates. The jobs reenter the same critical work centers (the testers) multiple times, and jobs of different product types share with each other for critical resources (which are the most expensive equipment in the shop floor and must be utilized efficiently). Further, the job processing time may vary, depending on the product type (job cluster) of the jobs processed on. Setup times between two consecutive jobs of different product types (job clusters) on the same machine are sequentially dependent. Machines are arranged in parallel at each stage of the process, and jobs to be processed are typically in hundreds of product types.

Kumar (1993) proposed a queuing network model for the reentrant lines problem, using simulation technique to analyze the performance of a wide variety of scheduling policies based on buffer priorities with due dates. Connors et al. (1996) presented an open queuing network model for a quick performance analysis on semiconductor manufacturing facilities. They assumed the dispatching rule first-come first-serve (FCFS) at all nodes in the queuing network, and used a decomposition approach to analyze the model. Some dispatching rules other than FCFS are also considered. Hwang and Sun (1997) presented a modified dynamic programming method to minimize makespan for the jobs processed in reentrant lines with sequence dependent setup time. Vargas-Villamil and Rivera (1997) developed a two-layer hierarchical structure control method to handle the optimization of the scheduling problem of discrete event reentrant semiconductor manufacturing lines. Sawik (2000) also presented a mixed integer programming formulation for solving a flexible-flow line problem with limited buffers to minimize the total completion time. Unfortunately, most of those works only focused on the front end of IC manufacturing such as the wafer fabrication.

Ovacik and Uzsoy (1996) provided some heuristic procedures to approximately solve the scheduling problem arising in final test phase of the semiconductor manufacturing. Chen et al. (1995) provided an integer-programming model for this problem to minimize the total weighted tardiness, which solves the problem using Lagrangian relaxation approach. Yoo and LA (1997) developed heuristic scheduling algorithms to minimize the number of tardy jobs using decomposition methods. Chikamura et al. (1997) applied the event-driven simulation analysis to evaluate the effect of express lots on the production dispatching rule scheduling and the associated cost. In those research works, the characteristic of reentry in the IC final testing factories has never been considered, therefore, do not reflect the real situations accurately.

In this paper, we present a case study on the ICFTSP covering all manufacturing process stages in the IC final testing factory, which include the stages processing jobs serially and the stages processing jobs in a batch manner. The case is taken from an IC final testing factory located on the Industrial Park in Taiwan. For the case investigated, jobs of various product types are to be processed on the flow line with reentry at critical resource stage (final testing) many times. Each single stage of ICFTSP with serial processing characteristic can be viewed as a parallel-machine scheduling problem, which can be solved by network algorithms designed for the vehicle routing problem with time windows (VRPTW) (Pearn et al., 2002). Since the case investigated here involves reentry characteristic, with constraints on processing precedence, serial stage, batch stage, job clusters, job-cluster dependent processing time, due dates, machine capacity, and sequence dependent setup time, it is more difficult to solve than the classical flow-shop scheduling problem. We present three efficient network algorithms to solve the problem, based on the stage-by-stage solution strategy and full-load policy applied to batch-processing stages, so that the capacity of critical resources testers would be efficiently utilized. Finally, we provide a performance comparisons between the three algorithms on eight test problems, considering the variation of processing time, the variation of setup time, and the tightness of due dates.

2. Multistage ICFTSP with reentry

The logic IC, such as LAN, AUDIO, and VEDIO, and the memory IC, such as EPROM,

MASKROM, SRAM, DRAM, and FLASH are the two major types of products in the IC final testing process. Due to the large increasing demand in memory IC products, most IC final testing factories allocate the machine capacity to the memory IC product. Therefore, we concentrate on the scheduling problem of memory products in the IC final testing factory. For the memory IC, the manufacturing process includes the following nine stages, (1) FT-1, (2) Cycling, (3) FT-2, (4) Burn-In, (5) FT-3 (6) Laser Mark, (7) VM/Scan, (8) Bake/Package, and (9) Shipping, as displayed in Fig. 1. We note that jobs in the following three stages, FT-1, FT-2, and FT-3, share the same group of resources (testers).

For the FT-1 (final test 1) stage, the process equipment consists of testers (with appropriate testing programs), handlers, assorted handling accessories such as load boards and socket bases, as shown in Figs. 2 and 3. At FT-1 stage, the tester combined with handler is used to test the basic functions and conductivity of chips, 100% inspection, at room temperature, $25\pm3^{\circ}C$, as shown in Fig. 4. In most cases, the chips are tested individually through a specific load board with socket bases, and ordered through a handler afterwards. Then, the chips are collected and moved to the ovens for cycling operation at higher temperature. For the FT-2 (final test 2) stage,

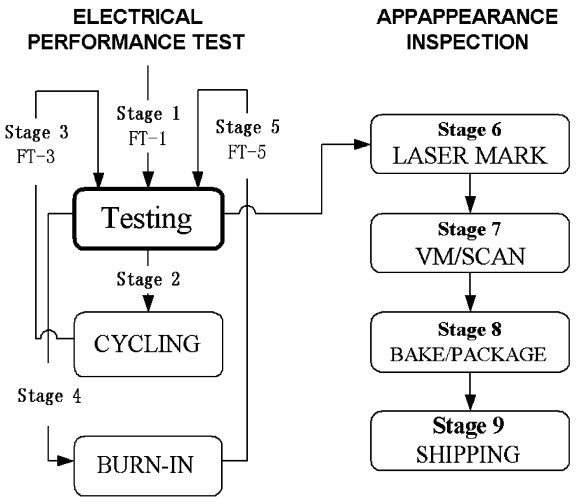

Fig. 1. The nine stages in the IC final testing process.

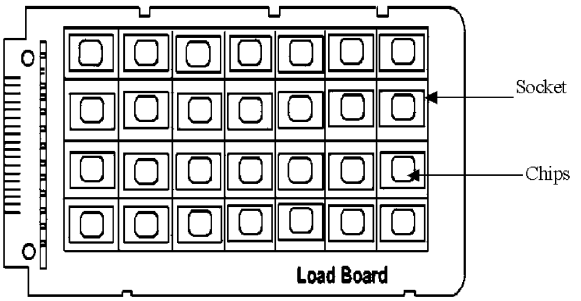

Fig. 2. Load board used in the chip cycling and burn-in processes.

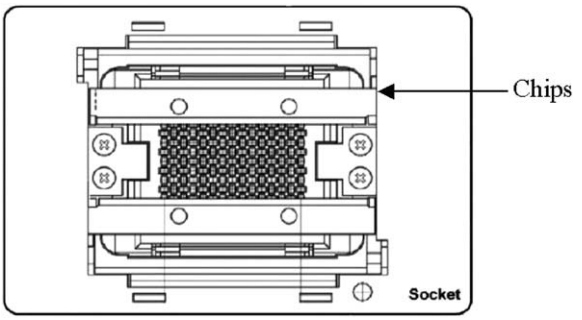

Fig. 3. Socket base used in the final testing process.

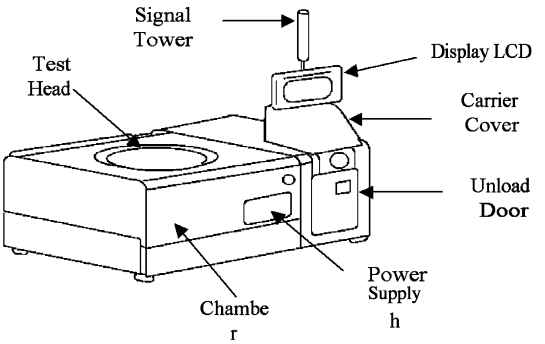

Fig. 4. Tester and handler used in the final testing process.

chips must be tested within a certain amount of time, 96 hours, where the testing temperature is set to high ($100\pm3^{\circ}C$). FT-2 determines the failure rate (such as open/short) and detects potential reliability problems of IC before shipping out to customers.

For the burn-in stage, the process equipment consists of the burn-in ovens (as shown in Fig. 5),

a robot handler, testing lots, an input waiting line, and an output waiting line. At burn-in stage, ten thousands of chips are arrayed in the oven and kept under high temperature ($125\pm3^{\circ}C$) and high voltage to speed up the deterioration rate to detect infant defects, where the burn-in operation is performed at the same time; therefore, it is called the batch-processing. The burn-in operation consumes the longest time compared to the operations at other stages in the final testing process, which is generally considered as a secondary capacity constraint. For the FT-3 (final test 3) stage, basic testing such as the conductivity testing is optionally executed as the temperature goes down to normal ($0\pm3^{\circ}C$). Generally, the testers in the IC final-testing factories are the most expensive equipment and should be utilized efficiently. Therefore, FT-1, FT-2, and FT-3 are the critical stages in the IC final testing process, where jobs of different product types compete with each other for the limited capacity of tester, as shown in Fig. 6.

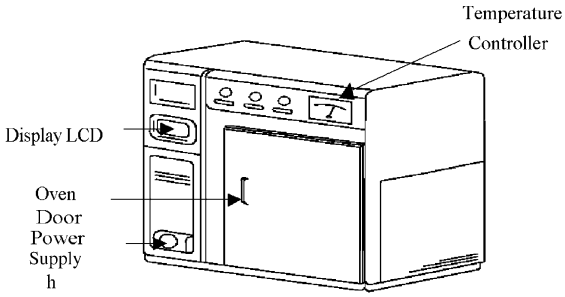

Fig. 5. Oven used in the burn-in process.

After completing all testing operations, chips are sorted by their grades in performance, and some legible information is marked on the top surface of the chips, such as the device code, manufacturing date, the manufacturer, and the country of origin. For the UM/Scan stage, the appearance of each chip is examined manually or by computer scanner. The leads of IC chips also need to be examined if they are tightly contacted with the same plane. The approved chips are baked and packaged with plastic tubes according to customers' requirements. Typically, there are three different types of package, namely, the normal package, the vacuumed package, and the tape/reel package. After the packaging, the encased chips are ready to be shipped to the warehouse or directly to the customers.

3. A case study

In the following, we consider a case taken from an IC final testing factory located on the Science-Based Industrial Park in Taiwan. For the case investigated, jobs of six product types must be

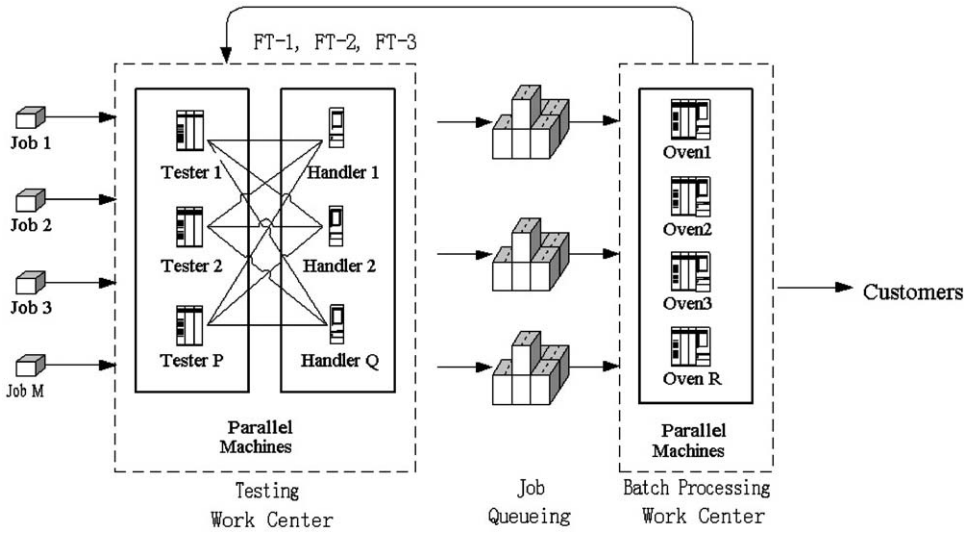

Fig. 6. Reentrant flow and critical resources at FT-1, FT-2, FT-3, and batch stages in the IC final testing process.

processed at all nine process stages and be completed before the due dates, at where a set of identical machines are arranged in parallel at each stage with two types of equipment in the shop floor. The first type of equipment (at work center performing the testing) is dedicated to the serial processing, and the other type of equipment is dedicated to the batch processing. Among the nine stages, stages 1, 3, 5, 6, 7, and 9 are dedicated for serial processing, which processes one single job each time, and stages 2, 4, and 8 are dedicated for batch processing, which processes a batch of jobs up to the maximum batch size (the oven capacity) at the same time. In our case, the maximum batch size is 10 lots (jobs) for each batch machine.

Due to the reentry characteristic, all jobs are required to visit the testing work center multiple times. Table 1 shows the process stage, processing type, and the number of machines at each work center. We note that the operations for all jobs at stages 1, 3, 5 are completed in the same work center, the work center performing the testing. Each job is a sub-lot divided from a specific order, which includes a variety of product type containing the required quantity of IC chips, various number of lots, the received date, and the due date, which is normally set between 2 and 6 days. Table 2 describes the two orders considered in our case, which includes 65 lots in the first order and 55 lots in the second order. The machine capacity (the maximum work load) is set to 1440 minutes and the planning horizon is set to 7 days (10080 minutes). "Minute" is used here as the time unit for the machine capacity, the job processing time, and the due date of jobs.

Since the job processing time depends on the product type and the machine setup time for two consecutive jobs of different product types (job clusters) on the same machine are sequential dependent, we design eight test problems, D1, D2, E1, E2, F1, F2, G1, and G2, to reflect such problem features. The design of those problems considers three factors: (a) the variation of the job processing time for different product types, (b) the variation of the setup time for switch product types, and (c) the tightness of due date for each job, as displayed in Table 3. Table 4(a) and (b) display the two levels of the job processing time required for the six product types at each stage of the entire process. Table 5(a) and (b) display the two levels of the machine setup time for the six product types, which is required at each process stage for switching one product type to another. After receiving orders, all sub-lots (jobs) must be completed before the due dates, which is normally set between 2 and 6 days. The objective in the case is to find the schedule solution for the jobs on the parallel machines at each stage, which satisfies the

Table 1
The process stage, processing type, and the number of machines at each work center

Work center	Testing			Cycling	Burn-in	Laser mark	VM/scan	Bake	Package
Process stage	1	3	5	2	4	6	7	8	9
Processing type	Serial			Batch	Batch	Serial	Serial	Batch	Serial
Number of machines	5			12	13	8	9	6	10

Table 2
The product type, the required quantity of chips, the number of lots, and the due date for the two orders in the case

Order	1						2	
Product type	1	2	3	4	5	6	1	3
Required quantity of chips	10,000	16,000	15,000	12,000	4000	8000	30,000	25,000
Number of lots	10	16	15	12	4	8	30	25
Lead time before due date (in days)	2	3	2.5	5	4	6	3	4.5

Table 3
Related information about the eight problem sets

Problem set	Factor					
	Variation of processing time for different product types		Variation of setup time for switching product types		Tightness of due date	
	Small	Large	Small	Large	Tight	Loose
D1	⋁		⋁		⋁	
D2	⋁		⋁			⋁
E1	⋁			⋁	⋁	
E2	⋁			⋁		⋁
F1		⋁	⋁		⋁	
F2		⋁	⋁			⋁
G1		⋁		⋁	⋁	
G2		⋁		⋁		⋁

Table 4

Process stage	Product type					
	1	2	3	4	5	6
(a) Processing time with large variation for the six product types at each process stage						
1	15	35	20	60	50	75
2	360	360	360	360	720	720
3	15	35	20	60	50	75
4	1440	1440	1440	1440	1440	1440
5	15	35	20	60	50	75
6	10	30	10	50	10	20
7	15	35	20	60	50	75
8	0	600	0	360	0	600
9	15	35	20	60	50	75
(b) Processing time with small variation for the six product types at each process stage						
1	30	30	30	30	35	30
2	360	360	360	360	720	720
3	30	30	30	30	35	30
4	1440	1440	1440	1440	1440	1440
5	30	30	30	30	35	30
6	30	30	30	30	35	30
7	30	30	30	30	35	30
8	0	600	0	360	0	600
9	30	30	30	30	35	30

Table 5

Product type	0	1	2	3	4	5	6
(a) Constant setup time required for switching product types at serial process stages							
0	0	20	20	20	20	20	20
1	0	0	20	20	20	20	20
2	0	20	0	20	20	20	20
3	0	20	20	0	20	20	20
4	0	20	20	20	0	20	20
5	0	20	20	20	20	0	20
6	0	20	20	20	20	20	0
(b) Setup time required for switching product types at serial process stages							
0	0	150	120	100	180	150	160
1	0	0	10	20	15	5	20
2	0	10	0	25	10	20	5
3	0	20	25	0	5	10	5
4	0	5	10	5	0	5	5
5	0	30	20	10	5	0	15
6	0	20	5	5	5	15	0

due date restrictions without violating the machine capacity constraints, while the total machine workload is minimized at the critical resource stages. Reducing the total setup time at the critical stages (the testers) is essential to the minimization of the total machine workload.

4. Algorithms for the multistage ICFTSP with reentry

In this section, we introduce three efficient solution procedures to solve the ICFTSP with reentry approximately, which can effectively handle large-scale problems. Noting that the IC final testing process consists of multiple manufacturing stages (serial and batch), one solution strategy we may take is to solve each single stage of the

ICFTSP problem sequentially to obtain the stage solutions, and then combine them into a complete ICFTSP solution. The completion time of the jobs determined at the previous stage will be used as the ready time of the jobs at next stage. We develop network algorithms to obtain the job schedules at the serial processing stages, which share the most critical resources (the testers), with full-load policy applied to jobs at the batch-processing stages.

Each single stage of ICFTSP that processes the jobs in serial can be viewed as a parallel-machine scheduling problem, which can be solved using the network algorithms designed for the VRPTW. For the serial processing stages of ICFTSP, we present three algorithms, sequential savings algorithm (SSA), matching-based savings algorithm (MBSA), and parallel insertion algorithm (PIA) with some modifications to obtain the machine schedule, which can effectively utilize the testers in the work center. As regarding the scheduling of jobs at batch-processing stages, the dispatching rule of first-come first-serve (FCFS) combined with a full-loaded policy, is used to collect jobs of the same product type until meeting the maximum batch size, and then assign to an available machine to perform the operation for the entire batch at the same time.

4.1. Sequential saving algorithm

The development of the SSA is essentially based on the well-known savings procedure of Clark and Wright (1964) for the vehicle routing problem, with some modifications. The procedure proceeds sequentially by adding a new pair of jobs at either one of the two ends of the currently constructed schedule, guided by measures on the savings list. SSA initially calculates the setup-time savings of all pairs of jobs, and creates a job list by sorting the savings in descending order. Starting with the top of the savings list, SSA 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. Note that a selected pair of jobs is feasible and is added to the machine schedule only if the capacity constraint and the due date restrictions are not violated. If the current constructed schedule cannot be expanded, another new schedule is created by selecting the first feasible pair of jobs from the top of the remaining list. Repeat the above procedures until all jobs are scheduled. The procedure steps of the SSA are described as the following:

Step 1 (*Initialization*): Calculate the setup time savings, $SA_{ijp,i'j'p}$, defined as the following formula, for all pairs of jobs r_{ijp}, $r_{i'j'p}$:

$$SA_{ijp,i'j'p} = s_{i'Up} + s_{i'Up} - s_{ii'p},$$

where the $s_{ii'p}$ represents setup time between any two consecutive jobs $r_{ijp}(\in R_i)$ and $r_{i'j'p}(\in R_{i'})$ from different product types ($i \neq i'$) at the same process stage, and U denotes the machine is in idle status.

Step 2: Sort the savings on a list in descending order of magnitude.

Step 3 (*Schedule construction*): Choose the first feasible link in the list to start a new schedule (initialization of the first schedule). Starting from the top of the savings list, proceed the following two sub-steps.

Step 3.1: Find the first feasible link in the list, which can be used to extend one of the two ends of the currently constructed schedule without violating the machine capacity constraints and the job due date restrictions.

Step 3.2: If the current schedule cannot be expanded, choose the first feasible link in the list to start a new schedule.

Step 4: Repeat step 3 until all jobs are scheduled.

4.2. Matching-based saving algorithm

The development of the MBSA is based on the procedures proposed by Altinkemer and Gavish (1991), originally proposed for the vehicle routing problem. Initially, each job is assigned to a machine schedule. That is, each parallel machine has a schedule containing a single job. All feasible combinations of merging two machine schedules into a new schedule are evaluated based on the measure of the setup-time savings. A combination of two machine schedules is feasible only when the jobs on the combined schedule does not violate the machine capacity and the due date restrictions. A new schedule is created by choosing the combination with the largest saving in setup time among all feasible combinations. Repeat the procedures until no more feasible schedule can be merged and the

set of machine schedules is the desired solution. The notations used in the MBSA are described as the following.

Z_u	the set of jobs in schedule u, this is the set of jobs assign to schedule Z,
SA_{uv}	the savings obtained by merging schedules u and v,
V_n	the set of schedules after nth computation,
$I(\alpha)$	the function represents the product type of job α,
$y_{\alpha\beta}$	the variable that represents whether any two jobs α and β are directly connected in schedule u: For instance, $y_{\alpha\beta}=1$ represents job β is scheduled directly after job α, and $y_{\alpha\beta}=0$ represents β is scheduled directly after job α.
u_1	the first job in schedule u,
u_f	the last job in schedule u.

The steps of the MBSA are described as the following.

Step 0 (*Initialization*): For the N_I jobs ($u=1,2,\ldots,N_I$), create N_I schedules and form a schedule set V with each job set $Z_u=\{u\}$ containing one job, $n=1$, $V_0=V$.

Step 1: For every u, v, and u, $v\in V_{n-1}$, merge schedules u and v (insert schedule u before schedule v) and do:

(a) If $\sum_{\alpha\in Z_u\cup Z_v} p_{I(\alpha)}+\sum_{\alpha,\beta\in Z_u\cup Z_v} y_{\alpha\beta}s_{I(\alpha)I(\beta)}>W$ or the violation of the due date restriction of any job in Z_v caused by the schedule merging, go to the next combination of u and v, where W denotes the maximum workload allowed on a machine.

(b) Otherwise, compute savings SA_{uv} according to the formula

$$SA_{uv}=\begin{cases}0 & \text{if } SA_{\mathrm{U}V}<0,\\ s_{I(u_l)\mathrm{U}}+s_{\mathrm{U}I(v_f)}-s_{I(u_l)I(v_f)} & \text{otherwise.}\end{cases}$$

Step 2: Consider the maximum weighted matching problem with V_{n-1} as its node set and SA_{uv} as the cost entry for arc (u,v). Merge the schedule combinations with the largest savings (u^*,v^*) in the matching problem, where $(u^*,v^*)=\{(u,v)|\max[SA_{uv}]\}$.

Step 3: If all mergers are non-admissible, then stop. Otherwise, increase n by one, update V_{n-1} to include all sub-schedules and go to Step 1.

4.3. Parallel insertion algorithm

The PIA is essentially based on the parallel insertion procedure presented in Potvin and Rousseau's (1993) for the VRPTW, with some modifications. At the initialization step, the PIA constructs a set of machine schedules simultaneously. The PIA uses a generalized regret measure over all schedules and select the best unscheduled job, which looks ahead what can be lost later if a given job is not immediately inserted into its best alternative machine. The regret measure summaries the differences of insertion cost between the best alternative machine and all other alternative machines. Hence, unscheduled jobs with larger regret value will be inserted with higher priority. The notations used in the PIA are described as the following.

PS_{pk}	The partial schedule of machine m_{k_p} at process stage p.
$\lambda_{pk}(u_{pk(n-1)},r_{ijp},u_{pkn})$	The additional setup time occurred if job r_{ijp} is inserted between the $(n-1)$th and nth positions of the partial schedule PS_{pk}.
$\sigma(r_{ijp})$	The regret value of job r_{ijp}.
$I(u_{pkn})$	The function that returns the product type of the job being scheduled on the nth position of partial schedule PS_{pk}.

4.4. Parallel insertion procedure

Step 1: In each process stage, derive the partial schedule for each machine using the uncompleted schedule of previous planning.

Step 2: Follow three sub-steps to execute the scheduling procedures. Repeat the procedures until each lot of the machine group is scheduled.

Step 2.1: For each unscheduled lot, first compute its best feasible insertion place, by

$\lambda^*_{pk}(u_{pk,n^*-1}, r_{ijp}, u_{pk,n^*})$ in each machine's partial schedule:

$$\lambda_{pk}(u_{pk(n-1)}, r_{ijp}, u_{pkn}) = s_{I(u_{pk(n-1)})ip} + s_{iI(u_{pkn})p} - s_{I(u_{pk(n-1)})I(u_{pkn})p}.$$

Step 2.2: Compute each lot's regret value, $\sigma(r_{ijp})$. Choose the next inserted lot i, i^*, with the largest $\sigma(r_{ijp})$ among all unscheduled lots:

$$\sigma(r_{ijp}) = \sum_{pk \neq pk'} [\lambda^*_{pk}(u_{pk(n^*-1)}, r_{ijp}, u_{pkn^*}) - \lambda^*_{pk'}(u_{pk'(n^*-1)}, r_{ijp}, u_{pk'n^*})],$$

$$\sigma(r^*_{ijp}) = \max_{r_{ijp}}[\sigma(r_{ijp})].$$

Step 2.3: The best lot i^* is insert into the lowest insertion cost position of the machine determined by the lowest insertion cost $\lambda^*_{pk}(r_{ijp})$.

$$\lambda^*_{pk}(u_{pk(n^*-1)}, r_{ijp}, u_{pkn^*}) = \min_{n=1,\ldots,l}[\lambda_{pk}(u_{pk(n-1)}, r_{ijp}, u_{pkn})].$$

Step 3: Repeat step 1–step 2 until each machine is scheduled.

5. Computational results

To solve the multistage ICFTSP case, three heuristic algorithms were coded in Java programming language, and run on Pentium-III 600 personal computer. All computations are exact arithmetic without any truncations. Further, for the three proposed algorithms, computational experiments are carried out to test their performance with respect to the total machine workload. The total machine workloads of the three proposed algorithms on the eight test problems are tabulated in Table 6. The workload at each stage of the three proposed algorithms on the eight test problems are shown in Table 7(a)–(d).

As the due date of each job gets tighter, the total machine workload increases for all eight testing problems. Larger variations in the setup time for switching different product types also causes an increase of the total machine workload. Based on the performance measure with respect to the total machine workload, the MBSA performs better than both the SSA and the PIA for all the eight test problems. Although the PIA performs poor in six out of the eight test problems, it obtains better solutions than the SSA algorithm in test problems F1 and F2.

6. Conclusions

In this paper we presented a case study on the multistage ICFTSP with reentry, which covers both the serial processing and the batch processing stages. The case investigated includes six product types of jobs to be processed at nine process stages. The scheduling problem we investigated is a generalization of the classical reentrant flow line problems, batch process problems, and the identical parallel machine problems, which has many real-world applications particularly in the IC manufacturing industry. The ICFTSP involves constraints on reentrant flow, job processing precedence, serial-processing stage, batch-processing stage, job clusters, job-cluster dependent processing time, due dates, machine capacity, and sequence dependent setup time, it is more difficult to solve than the classical flow-shop scheduling problem.

In this paper, we introduced three efficient solution procedures to solve the job scheduling

Table 6
Total workload of the three algorithms on the testing problems

Algorithm	Problem							
	D1	D2	E1	E2	F1	F2	G1	G2
SSA	52,265	51,995	57,160	55,445	51,425	51,245	56,740	54,620
MBSA	51,925	51,845	56,645	55,160	51,085	50,765	56,350	54,380
PIA	52,445	52,085	60,420	56,615	51,305	51,005	59,675	56,240

Table 7
Stage workload of the three algorithms on testing problem sets: (a) D1 and D2, (b) E1 and E2, (c) F1 and F2, (d) G1 and G2

Process stage	(a) Algorithm					
	D1			D2		
	SSA	MBSA	PIA	SSA	MBSA	PIA
Stage 1	3820	3780	3800	3800	3780	3820
Stage 2	6040	6040	6040	6040	6040	6040
Stage 3	3820	3780	3800	3800	3780	3820
Stage 4	20,580	20,580	20,580	20,580	20,580	20,580
Stage 5	3860	3800	3960	3800	3780	3820
Stage 6	3860	3800	3920	3810	3780	3820
Stage 7	3860	3800	3920	3810	3780	3820
Stage 8	2545	2545	2545	2545	2545	2545
Stage 9	3880	3800	3880	3810	3780	3820

Process stage	(b) Algorithm					
	E1			E2		
	SSA	MBSA	PIA	SSA	MBSA	PIA
Stage 1	4485	4300	4755	4310	4255	4565
Stage 2	6040	6040	6040	6040	6040	6040
Stage 3	4485	4300	4755	4310	4255	4565
Stage 4	20,580	20,580	20,580	20,580	20,580	20,580
Stage 5	4665	4500	5310	4310	4255	4565
Stage 6	4680	4675	5470	4450	4410	4585
Stage 7	4680	4675	5470	4450	4410	4585
Stage 8	2545	2545	2545	2545	2545	2545
Stage 9	5000	5030	5495	4450	4410	4585

Process stage	(c) Algorithm					
	F1			F2		
	SSA	MBSA	PIA	SSA	MBSA	PIA
Stage 1	3680	3620	3620	3680	3600	3640
Stage 2	6040	6040	6040	6040	6040	6040
Stage 3	3680	3620	3620	3680	3600	3640
Stage 4	20,580	20,580	20,580	20,580	20,580	20,580
Stage 5	3700	3640	3640	3680	3600	3640
Stage 6	3680	3660	3720	3680	3600	3640
Stage 7	3720	3680	3800	3680	3600	3640
Stage 8	2545	2545	2545	2545	2545	2545
Stage 9	3800	3700	3740	3680	3600	3640

Process stage	(d) Algorithm					
	G1			G2		
	SSA	MBSA	PIA	SSA	MBSA	PIA
Stage 1	4225	4310	4630	4235	4195	4420
Stage 2	6040	6040	6040	6040	6040	6040
Stage 3	4225	4310	4630	4235	4195	4420
Stage 4	20,580	20,580	20,580	20,580	20,580	20,580
Stage 5	4560	4450	4975	4235	4195	4420
Stage 6	4855	4705	5425	4250	4210	4605
Stage 7	4855	4705	5425	4250	4210	4605
Stage 8	2545	2545	2545	2545	2545	2545
Stage 9	4855	4705	5425	4250	4210	4605

problem at the serial processing stages, accompanied with the stage-by-stage solution strategy and full-load policy applied to the batch-processing stages to solve this multistage ICFTSP with reentry approximately. Total workload is measured on the eight testing problems to evaluate the performance of the three algorithms. A design on the test problems considering three factors is conducted for the computational testing and comparisons. The computational testing results reveal that the MBSA performs better than the SSA and the PIA for all test problems (with different problem characteristics). All three proposed algorithms can effectively solve the large-scale ICPSP. Details of the machine schedules, and the workload on each individual machine, for the eight test problems, are also provided.

Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments, which significantly improved the paper.

References

Altinkemer, K., Gavish, B., 1991. Parallel savings based heuristics for the delivery problem. Operations Research 39 (3), 456–469.

Centeno, G., 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., Kao, J., 1995. Scheduling for IC sort and test with preemptiveness via

Lagrangian relaxation. Transactions on Systems, Man, and Cybernetics 25 (8), 1249–1256.

Chikamura, A., Nakamae, K., Fujioka, H., 1997. Effect of express lots on production dispatching rule scheduling and cost in final test process of LSI manufacturing system. IEEE International Symposium on Semiconductor Manufacturing Conference Proceedings, D23–D26.

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

Connors, D.P., Feigin, G.E., Yao, D.D., 1996. A queuing network model for semiconductor manufacturing. IEEE Transactions on Semiconductor manufacturing 9 (3), 412–427.

Hwang, H., Sun, J.U., 1997. Production sequencing problem with reentrant work flows and sequence dependent setup times. Computers and Industrial Engineering 33 (3–4), 773–776.

Kumar, P.R., 1993. Re-entrant lines. Queueing Systems: Theory and Applications 13, 87–110.

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

Pearn, W.L., Chung, S.H., Yang, M.H., 2002. The wafer probing scheduling problem (WPSP). Journal of the Operational Research Society 53, 864–874.

Potvin, Y., Rousseau, J.M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66, 331–340.

Sawik, T., 2000. Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. Mathematical and Computer Modelling 31 (13), 39–52.

Schutten, J.M.J., 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.

Vargas-Villamil, F.D., Rivera, D.E., 1997. Scheduling of reentrant manufacturing lines using a model predictive control (MPC) approach. Proceedings of the 1997 American Control Conference, American Automatic Control Council, Vol. 1(3), pp. 1919–1923.

Yalaoui, F., Chu, C., 2002. Parallel machine scheduling to minimize total tardiness. International Journal of Production Economics 76 (3), 265–279.

Yoo, W.S., LA, M.V., 1997. A decomposition methodology for scheduling semiconductor test operations for number of tardy job measures. Journal of Electronics Manufacturing 17 (1), 51–61.