Industrial Engineering
& Management Systems
Vol 12, No 4, December 2013, pp.306-316
ISSN 1598-7248 | EISSN 2234-6473 |

http://dx.doi.org/10.7232/iems.2013.12.4.306
© 2013 KIIE

Hybrid Genetic Algorithms for Solving Reentrant Flow-Shop Scheduling with Time Windows

Chettha Chamnanlor, Kanchana Sethanan*
Department of Industrial Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand

Chen-Fu Chien
Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu, Taiwan

Mitsuo Gen
Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu, Taiwan, Fuzzy Logic System Institute, Iizuka, Japan

(Received: January 21, 2013 / Revised: July 29, 2013 / Accepted: October 11, 2013)

ABSTRACT

The semiconductor industry has grown rapidly, and subsequently production planning problems have raised many important research issues. The reentrant flow-shop (RFS) scheduling problem with time windows constraint for hard-disk devices (HDD) manufacturing is one such problem of the expanded semiconductor industry. The RFS scheduling problem with the objective of minimizing the makespan of jobs is considered. Meeting this objective is directly related to maximizing the system throughput which is the most important of HDD industry requirements. Moreover, most manufacturing systems have to handle the quality of semiconductor material. The time windows constraint in the manufacturing system must then be considered. In this paper, we propose a hybrid genetic algorithm (HGA) for improving chromosomes/offspring by checking and repairing time window constraint and improving offspring by left-shift routines as a local search algorithm to solve effectively the RFS scheduling problem with time windows constraint. Numerical experiments on several problems show that the proposed HGA approach has higher search capability to improve quality of solutions.

Keywords: Reentrant Flow-Shop, Time Windows, Hybrid Genetic Algorithm, Local Search Method

* Corresponding Author, E-mail: ksethanan@gmail.com

1. INTRODUCTION

Scheduling in real production situations of a hard-disk manufacturing system contrasts to classical scheduling where each job visits each machine only once. Because the reentrant flow-shop (RFS) scheduling problem is often found in this system, the processing flow characteristic is that a job can visit certain machines more than once. Also, the flow-hop for the hard-disk device (HDD) manufacturing shop floor consists of several serial workstations. Each workstation is composed of only one machine for production of a total of *n* jobs. The jobs are divided into product family groups. Each job is provided with a different sequence of operations. Some workstations can produce some jobs depending on the processing steps of those jobs.

In this paper, the RFS scheduling problem with the objective of minimizing the makespan of jobs is considered. Minimizing makespan is directly related with maximizing the system throughput which is considered as the most important of HDD industry requirements. Moreover, most manufacturing systems have to cope with the quality of semiconductor material. The time windows constraint in the manufacturing system must then be considered. High quality processing with a critical control time is essential for hard-disk manufacturing. Time windows are controlled from the beginning step to the ending step. Hence, any job with its completion time exceeding the limited time window will lead to a loss. For example, production of certain materials that are sensitive to weather conditions, temperatures, or chemicals involved may lead to deterioration or wear during the process when the process requires or uses too much time (Chamnanlor and Sethanan, 2009; Monch *et al.*, 2011). This may result in rework or rejection and disposal (Chien and Chen, 2007).

To solve classical flow-shop problems, one metaheuristic which has been widely used and has performed well is a genetic algorithm (GA). Subsequently, hybrid genetic algorithms (HGA) have been used to enhance the performance of pure GA (Chen *et al.*, 2008a; Lin and Gen, 2009; Goncalves *et al.*, 2005). This paper therefore presents the HGA to solve the RFS problem with time windows constraint in the HDD industry. Since the production scheduling problem is intractably solved by an exact mathematical model, HGA will be developed in this paper to minimize makespan and reduce the loss.

In the next section, results of the review of related literature are presented. The characteristics of the reentrant model in an HDD manufacturing system are described in Section 3. This is followed by Section 4 presenting the HGA for solving the problem. Section 5 outlines the experimental research. Finally, a summary of the main findings is given in Section 6.

2. LITERATURE REVIEW

The most well known scheduling problem area is the flow-shop scheduling problem. Since the two-machine case of the regular flow-shop was pioneered in 1954 by Johnson (Chen *et al.*, 2008a; Ruiz *et al.*, 2005; Sethanan, 2001), most research has focused on developing heuristics for solving *m*-machine flow-shop problems, such as the well known NP-hard problem. Basically, flow-shop scheduling research is considered within the same set of constraints depending on industrial situations, for example, Ruiz *et al.* (2005) proposed advanced GAs to solve the permutation flow-hop scheduling problem with sequence dependent setup time and showed calibration of the GA's parameters. Even for semiconductor process scheduling, there were many constraints included such as Chien and Chen (2007) who developed a GA for batch sequencing combined with a novel timetabling algorithm for a furnace process for semiconductor fabrication. Their problem also was characterized by waiting time constraints, frequency-based setups, and capacity preoccupation.

For most of the previous research mentioned above, they are generally scheduling problem considering unidirectional flow. However, the complication of manufacturing systems has increased in current industries especially hard-disk manufacturing systems, so that they have reentrant flow processing. In research work on the RFS problem, Hwang and Sun (1997) studied the production sequencing problem with reentrant work flow and dependent setup times. They solved the problem with modified dynamic programming with the objective of the minimum makespan. Park *et al.* (2000) proposed an approximate method based on mean value analysis for estimating the average performance using the cycle time and the throughput of the RFS with single-job machines and batch machines. Pan and Chen (2003) proposed three extended mixed binary integer programming formulations and six extended effective heuristics for solving reentrant permutation flow-shop scheduling problems to minimize makespan. They also defined the corresponding scheduling problems with a re-entry property. Kang *et al.* (2007) proposed a heuristic algorithm that minimizes total weighted tardiness for RFS problems with sequence dependent setup time. The results showed that the proposed algorithms give very efficent schedues in terms of total weighted tardiness and computational effort. Chen *et al.* (2007) studied minimizing the makespan of the RFS problem, so the hybridization method was applied and used to enhance the performance of pure tabu search. In addition, they considered the RFS scheduling problem in which no passing is allowed, and also applied the hybrid tabu search to minimize the makespan of jobs in the next year (Chen *et al.*, 2008b). Jing *et al.* (2008) presented heuristic algorithms to solve a two-machine RFS scheduling problem with the objective of minimizing makespan. The experimental results show that the heuristics performance is significantly affected by the distribution of workloads on machines and some of them are excellent. Abe and Ida (2008) proposed genetic local search methods that are more valid than the other local search methods, such as swap, move, and swap-2 neighborhood for solving a RFS scheduling problem to get a better turn around time. Chen *et al.* (2008a) aimed to study minimizing makespan by using the genetic algorithm to move from local optimal solution to near optimal solution for the reentrant scheduling problem. Moreover, HGAs are proposed to enhance the performance of pure GA.

From the research mentioned, it is clear that there are many studies on the RFS scheduling problem. There are almost no appoaches for solving RFSs with time windows constraint as the focus of our research. In those RFS scheduling problems, many approaches were conducted as solutions. As if the GA might be recognized by high performance since in the past few decades the number of related GA papers indexed by Science Citation Index has shown importance (Lin *et al.*, 2012). In addition, the GA is still available combined with heuristics. Also, Lin and Gen (2009) proposed a HGA combined with a fuzzy logic controller for an auto-tuning strategy that adaptively regulates for taking the balance among the stochastic search and local search probabilities based on the change of the average fitness of parents and offspring which occurs at each generation. They also used an auto-tuning strategy with fuzzy logic control to auto-tune the balance between stochastic search and heuristic search probabilities based on the change of the average fitness of the current and last generations (Lin *et al.*, 2009).

It can be see that a hybrid approach which combines a GA and local search technique has been reported by many researchers. Gao *et al.* (2006) proposed a new HGA to solve the flexible job shop scheduling problem with non-fixed availability constraints. Also, Gen *et al.* (2009) proposed a multistage-based GA with bottleneck shifting developed for the flexible job shop scheduling problem.

3. HYBRID REENTRANT MODEL IN HDD MANUFACTURING SYSTEM

Naturally, a HDD manufacturing system is one of the most complicated systems depending on several constraints, such as various product families with different processing time and processing flow, high flexibility machines, and one or more time operations on a workstation in the reentry flow of a job. Moreover, controlling processing time constraints is an important issue for an industry which requires high quality production especially in a hard-disk manufacturing system.

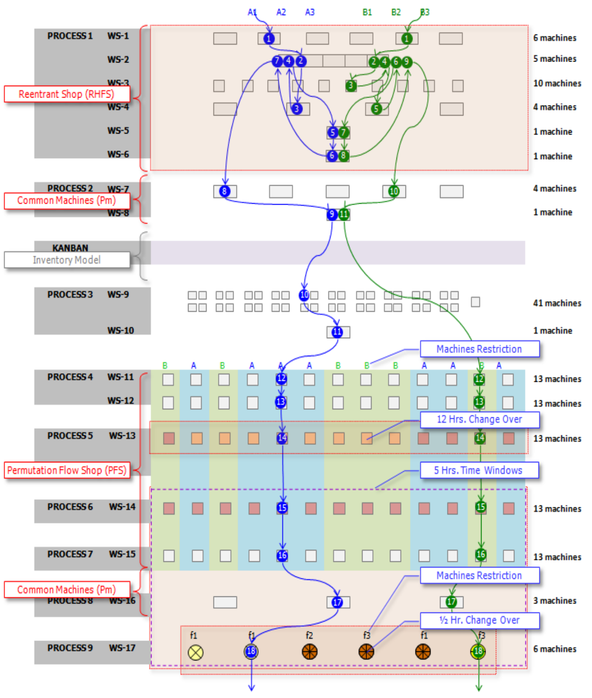


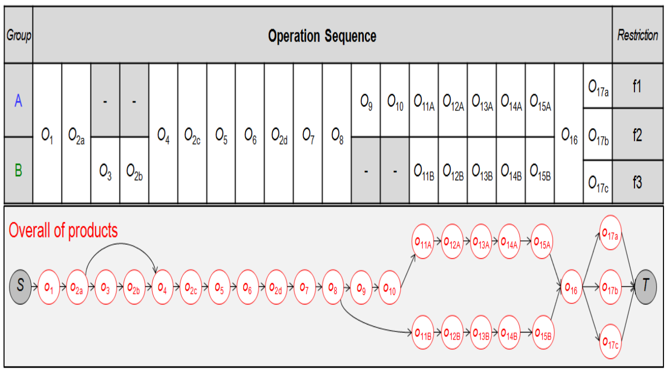
Figure 1. Real hard-disk device manufacturing system model.

As shown in Figure 1, of the hybrid flow-shop in a real hard-disk manufacturing system, there are 9 processes with 17 workstations. Each of them has a different number of machines which also have different efficiencies. Some machines might be limited by production constraints, such as machine eligibility restriction and sequence dependent setup time. Moreover, the system still consists of several sub-systems for example, reentrant shop, common machine shop, and permutation shop. Unfortunately, these were located in the single system; it was very difficult to solve all by the optimization techniques.

Nevertheless, planning and scheduling in the above system might be reduced by a simplification. Decomposition of the problem and decrement of the problem size were usually included by many researchers. This should be done to understand and clarify a complicated manufacturing system.

In complex hard-disk manufacturing, from the real defined problem to the simplified one, sequencing and scheduling jobs in this manufacturing shop floor can be considered on the production model as a RFS consisting of *m* workstations (stages). Each workstation consists of exactly one machine in order to produce *n* jobs. Also, each job includes different amounts of workload (lots) for example, 6 products of 2 different family groups as shown in Figure 1. Each job is processed in various workstations depending on its flow. Some workstations can produce more than one operation according to the reentrant constraint. However, the job can be processed within the limited time windows.

However, a precedence relationship analysis (as shown in Figure 2) should be conducted for clarity of the complex sequences even if there are a lot of jobs. In the case of a hard-disk manufacturing system, types of products should be produced in the same time of types of machines and materials. Since there are many job types and many flow types of the sequence, the precedence relationship is very important in this research, because it is able to control generating a legal chromosome.



Group	Operation Sequence																			Restriction
A	O_1	O_{2a}	-	-	O_4	O_{2c}	O_5	O_6	O_{2d}	O_7	O_8	O_9	O_{10}	O_{11A}	O_{12A}	O_{13A}	O_{14A}	O_{15A}	O_{16}	O_{17a}	f1
																				O_{17b}	f2
B			O_3	O_{2b}								-	-	O_{11B}	O_{12B}	O_{13B}	O_{14B}	O_{15B}		O_{17c}	f3

Figure 2. Precedence relationship graph of real reentrant flow-shop scheduling problem.

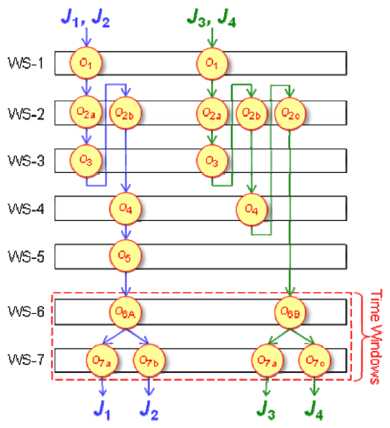


Figure 3. Processing flow of a simple reentrant flow-shop scheduling problem.

In this paper, a simple RFS scheduling problem, which is similar to the real problem, is conducted to test the algorithm. It was generated by decreasing the problem size of the real problem.

Considering the processing flow of a simplified RFS scheduling problem as shown in Figure 3, there are 4 products (J_1, J_2, J_3, and J_4) with no consideration of lot sizes of jobs. This means all jobs have the same lot sizes. Also, this manufacturing system has 7 workstations with the reentrant workstations (Table 1). Moreover, in this example, there is the time windows constraint (tw = 30 min/lot) for controlling production starting from WS-6 to WS-7.

From the data set, the precedence relationship can be defined by the successors and operation sequence (as show in Table 1). Within these workstations, the jobs will be produced depending on their operation sequences. The graph is then drawn for preparing a chromosome for initial generation, and repairing the chromosome after genetic operators (Figure 4).

Indeed, the system complexity comes from three important restrictions. First, all products can be produced depending on reentrant flow; they have to produce

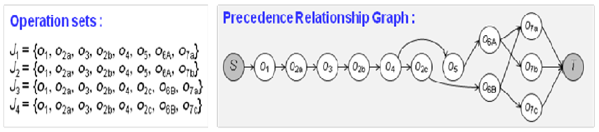

Operation sets :
$J_1 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_5, o_{6A}, o_{7a}\}$
$J_2 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_5, o_{6A}, o_{7b}\}$
$J_3 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_{2c}, o_{6B}, o_{7a}\}$
$J_4 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_{2c}, o_{6B}, o_{7c}\}$
Precedence Relationship Graph :
S
o1
o2a
o3
o2b
o4
o2c
o5
o6A
o6B
o7a
o7b
o7c
T

Figure 4. Precedence relationship graph of simple reentrant flow-shop scheduling problem.

two family product groups at some workstations. Lastly, they have to produce all products with the completion time of each under the time windows. The objective is to minimize makespan and reduce loss.

4. HYBRID GENETIC ALGORITHM

4.1 Genetic Algorithm Representation for RFS

GA is a class of general purpose search methods combining elements of directed and stochastic search which can produce a remarkable balance between exploration and exploitation of the search space. Even though there are newer methods currently available, GA is still accepted especially when applying with the hybrid method (Gen and Cheng, 2000).

In the case of RFS with time windows constraint, there are complications with checking the condition of the constraint. It can generate a chromosome with illegal rules.

4.1.1 Operation-based representations

In 1994, Gen, Tsujimura, and Kubota proposed an implementation of GA for solving the job-shop scheduling problem. The operation-based representation encoded a schedule as a sequence of operations and each gene standing for one operation was proposed by them (Gen *et al*., 2008). Furthermore, this paper can be applied to the RFS scheduling problem.

After generating the chromosome, the schedule can be generated. When generating it, an operation can be started whenever its predecessor has been finished and

Table 1. Define the operations of a simple example reentrant flow-shop scheduling problem

Step	Machine	In the same group of family A				In the same group of family B			
		J_1		J_2		J_3		J_4	
		Operation	*Successor*	*Operation*	*Successor*	*Operation*	*Successor*	*Operation*	*Successor*
1	WS-1	o_1	o_{2a}	o_1	o_{2a}	o_1	o_{2a}	o_1	o_{2a}
2	WS-2	o_{2a}	o_3	o_{2a}	o_3	o_{2a}	o_3	o_{2a}	o_3
3	WS-3	o_3	o_{2b}	o_3	o_{2b}	o_3	o_{2b}	o_3	o_{2b}
4	WS-2	o_{2b}	o_4	o_{2b}	o_4	o_{2b}	o_4	o_{2b}	o_4
5	WS-4	o_4	o_5	o_4	o_5	o_4	o_{2c}	o_4	o_{2c}
6	WS-2	-	-	-	-	o_{2c}	o_{6B}	o_{2c}	o_{6B}
7	WS-5	o_5	o_{6A}	o_5	o_{6A}	-	-	-	-
8	WS-6	o_{6A}	o_{7a}	o_{6A}	o_{7b}	o_{6B}	o_{7a}	o_{6B}	o_{7c}
9	WS-7	o_{7a}	-	o_{7b}	-	o_{7a}	-	o_{7c}	-

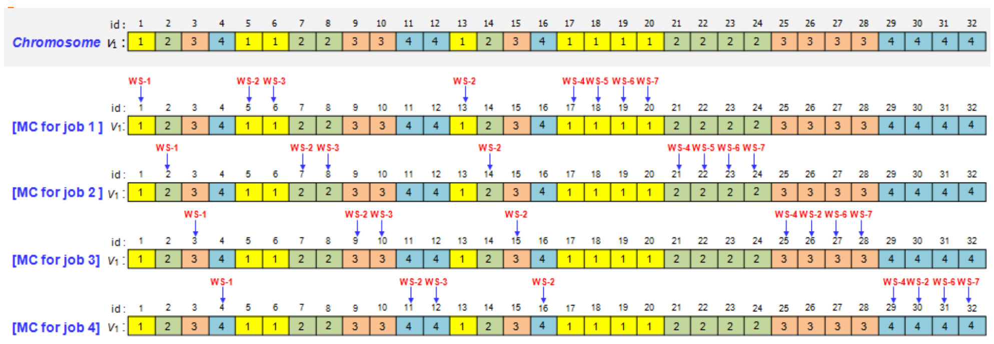


Figure 5. The illustration of a chromosome with operation-based representations.

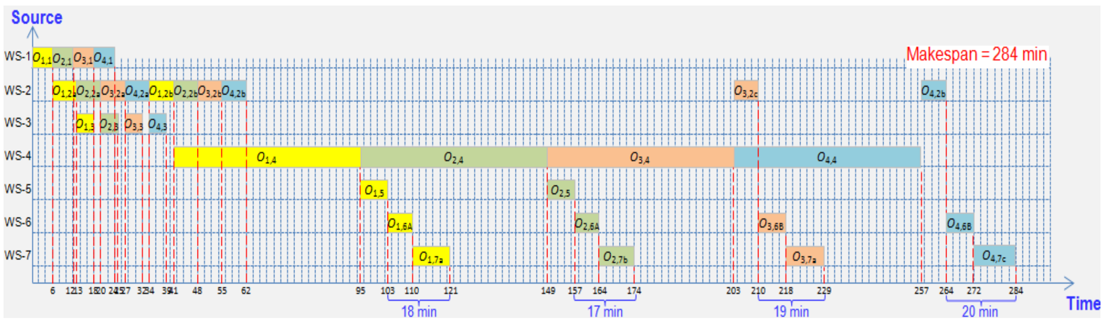


Figure 6. A Gantt chart of the generated chromosome.

the machine to process it is available. The generated schedule of the example chromosome in Figure 5 is shown as follows: Schedule S = {(o_{ij}, M_m, s_{ij}, c_{ij})}; $o_{i,j}$ denotes operation *j* in job *i*; M_m is machine *m*; s_{ij} means starting operation *j* in job *i* and is completing operation *j* in job *i*.

S = {($o_{1,1}$, M_1, 0-6), ($o_{2,1}$, M_1, 6-12), ($o_{3,1}$, M_1, 12-18), ($o_{4,1}$, M_1, 18-24), ($o_{1,2a}$, M_2, 6-13), ($o_{1,3}$, M_3, 13-18), ($o_{2,2a}$, M_2, 13-20), ($o_{2,3}$, M_3, 20-25), ($o_{3,2a}$, M_2, 20-27), ($o_{3,3}$, M_3, 27-32), ($o_{4,2a}$, M_2, 27-34), ($o_{4,3}$, M_3, 34-39), ($o_{1,2b}$, M_2, 32-41), ($o_{2,2b}$, M_2, 41-48), ($o_{3,2b}$, M_2, 48-55), ($o_{4,2b}$, M_2, 55-62), ($o_{1,4}$, M_4, 41-95), ($o_{1,5}$, M_5, 95-103), ($o_{1,6A}$, M_6, 103-110), ($o_{1,7a}$, M_7, 110-121), ($o_{2,4}$, M_4, 95-149), ($o_{2,5}$, M_5, 149-157), ($o_{2,6A}$, M_6, 157-164), ($o_{2,7b}$, M_7, 164-174), ($o_{3,4}$, M_4, 149-203), ($o_{3,2c}$, M_2, 203-210), ($o_{3,6B}$, M_6, 210-218), ($o_{3,7a}$, M_7, 218-229), ($o_{4,4}$, M_4, 203-257), ($o_{4,2c}$, M_2, 257-264), ($o_{4,6B}$, M_6, 264-272), ($o_{4,7c}$, M_7, 272-284)}.

The corresponding Gantt chart of this schedule can be drawn as shown in Figure 6.

From the Gantt chart, it is clear that the operation-based method can be used for generating the suitable candidate chromosome as shown in Figure 6. So, this sequence solution has made a 284-minute makespan and no loss because there are no jobs exceeding the time windows (30 minutes between WS-6 and WS-7).

4.1.2 Two-cut point crossover representations

As a general rule, crossover is the main genetic operator. It operates on two chromosomes at a time and generates offspring by combining both chromosomes' features (Gen *et al.*, 2008). Two cut-point is one of the conventional crossover operators which has good performance, and is usually use for evolutional search. Based on the characteristics of the chromosome, the two cut-point crossover operator will randomly select two cutting points within the second segments of parents and exchange the substrings in the middle of the two cutting points between parents. Subsequently, a repairing technique is adopted to convert an illegal chromosome to a legal one, whenever it is infeasible (Figure 7).

4.1.3 Swap-mutation representations

The swap-mutation operator will select two allele values in a string at random, and swap their positions. Moreover, the swap process will be repeated depending on the string length in proportion (Figure 8).

4.1.4 Insert-mutation representations

The insert-mutation operator will select two allele values in a string at random. Next, the second allele will be moved to precede the first, and then shift the rest along to accommodate. In addition, the insert process will be repeated depending on the string length in proportion (Figure 9).

4.1.5 Fitness function

In the RFS scheduling problem, the objective is to minimize the makespan (z_i), so it is directly related with maximizing the system throughput. RFS scheduling in the hard-disk manufacturing system as in this paper, also

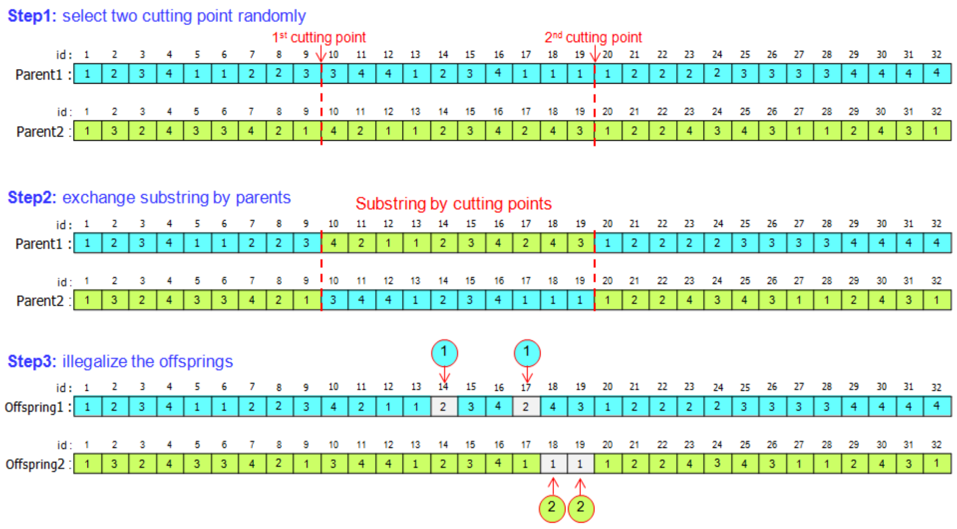

Step1: select two cutting point randomly
1st cutting point
2nd cutting point
Parent1 :
Parent2 :
Step2: exchange substring by parents
Substring by cutting points
Step3: illegalize the offsprings
Offspring1 :
Offspring2 :

Figure 7. Illustration of two cut-point crossover steps for solving a simple reentrant flow-shop scheduling problem.

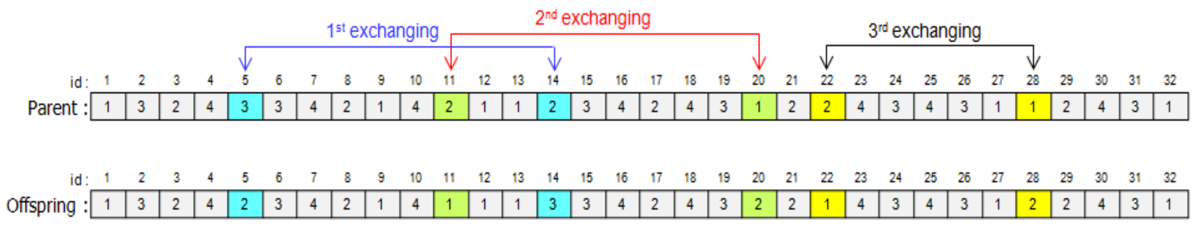


Figure 8. The illustration of swap-mutation operation for solving a simple reentrant flow-shop scheduling problem.

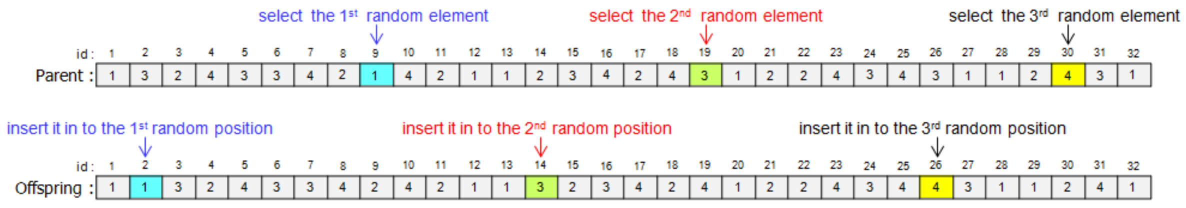

select the 1st random element
select the 2nd random element
select the 3rd random element
Parent :
insert it in to the 1st random position
insert it in to the 2nd random position
insert it in to the 3rd random position
Offspring :

Figure 9. Illustration of insert-mutation for solving simple reentrant flow-shop scheduling problem.

has to consider the lost lot which exceeded the critical processing time. Eq. (1) shown the fitness function of GA where v_i is a chromosome vector i; the population is *popSize*.

$$eval(v_i) = \frac{1}{z_i}, \quad i = 1, 2, \cdots popSize \tag{1}$$

In addition, RFS scheduling in a hard-disk manufacturing system as in this paper, also has to consider the number of losses as jobs which exceeded the critical processing time. It also is an indication of the effective system even if the loss cannot be reworked.

4.2 Combining Genetic Algorithm by TW and LS

The overall pseudo-code procedure of HGA is shown in Figure 10.

```
procedure: HGA for RFS model
input: RFS problem data, GA parameters (popSize, maxGen, $p_M$, $p_C$)
output: the best schedule
begin
    $t \leftarrow 0$;                                   // t: generation number
    initialize P(t) by operation-based encoding routine; // P(t): population
    check and repair P(t) time window constraint for all chromosomes;
    evaluate P(t) by operation-based decoding routine;
    while (not terminating condition) do
        create C(t) from P(t) by two cut-point crossover routine;
                                                          // C(t): offspring
        create C(t) from P(t) by swap mutation routine;
        create C(t) from P(t) by insert mutation routine;
        check and repair precedence constraint for all offspring C(t);
        check and repair time window constraint for all offspring C(t);
        improve C(t) by left-shifts routine;
        evaluate eval(P, C) by operation-based decoding routine;
        select P(t+1) from P(t) and C(t) by roulette wheel selection
        routine;
        $t \leftarrow t + 1$;
    end
    output the best schedule
end;
```

Figure 10. The implementation structure of hybrid genetic algorithm for the reentrant flow-shop scheduling problem.

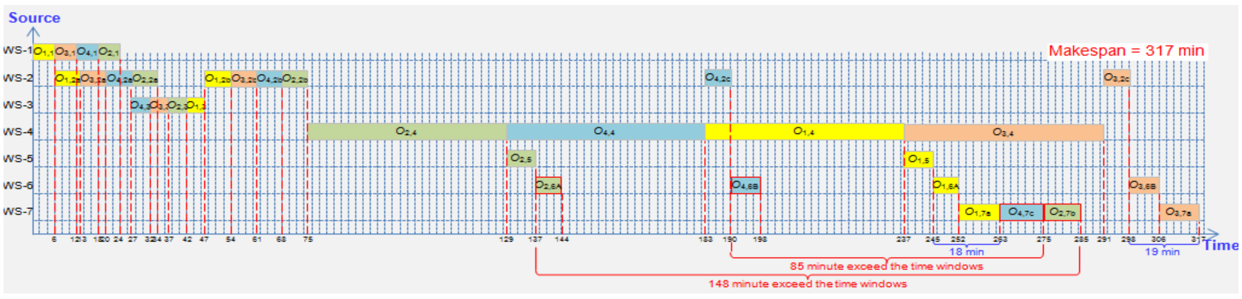


Figure 11. A Gantt chart with some lost products.

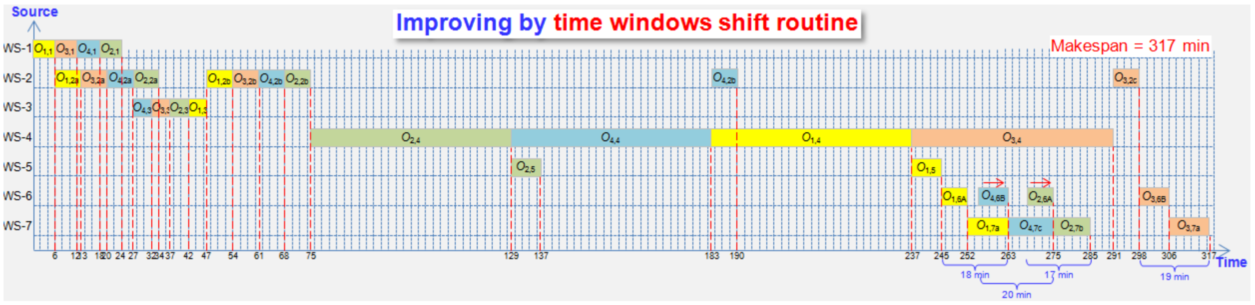


Figure 12. Illustration of improving the Gantt chart by the time window satisfied algorithm.

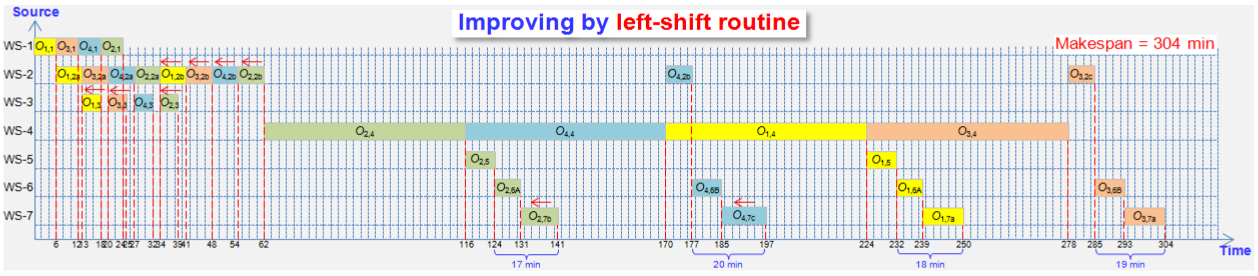


Figure 13. Illustration of improving the Gantt chart by left-shift algorithm.

The operation sequence vector (v_1) of a chromosome can be generated at random. With the locus of the operation o_{ij} in machines, the encoding procedure is shown in Figure 5. However, the operation may be shifted to the left as compactly as possible. So, a shift is called local left-shift if some operations can be started earlier than at present.

Nevertheless, the first algorithm is time window satisfied routine (TW). It would be moved into a suitable position when there is any operation exceeding the time window. The second algorithm, which is left-shift routine (LS), is an area restriction type shift search. It compares processing time and end time as saved, does not set the lot in the area to smaller than the processing time, and does not insert it in the double range of processing time.

An example of a Gantt chart including lot-losses is shown in Figure 11. It can be recovered by the concept of improving the Gantt chart that is to use time windows shift. The last job of the loss will be considered first by shifting some operation of the job to the right in order to fit the window of the controlling time. Then, the next to last job will be considered in the same way, and so on. After satisfying time windows of the Gantt chart in Figure 12, some parts can be improved by the left shift algorithm appropriately. So, the new solution has made a 304-minute makespan. There also is no loss, as shown in Figure 13.

When expanding the vital points, an illegal chromosome might occur by regular genetic operations. In case of two cut-point crossover routines and mutation routines including in the HGA, they simply use exploration and exploitation of the search space. They cannot arrive at a feasible solution as a legal chromosome. That is the reason for conducting check and repair of pre-offspring with precedence constraint limitation.

When expanding the vital points, an illegal chromosome might occur by regular genetic operations. In case of two cut point crossover routines and mutation routines including in the HGA, they simply use exploration and exploitation of the search space. They cannot arrive at a feasible solution as a legal chromosome. That is the reason for conducting check and repair of pre-offspring with precedence constraint limitation.

This paper introduces checking and repairing precedence constraint for all offspring $C(t)$. The steps of the check and repair routine for the precedence constraint are given as follows:

Step1: Transform all offspring $C(t)$ by the decoding routine.
Step2: Compare each offspring $C(t)$ with operation sequences for each job.
Step3: If there is an illegal offspring $C(t)$, repair it by the operation sequences based on the job.

In the same situation, $C(t)$ time window constraints for all chromosomes/offspring are checked and repaired. The steps for the checking and repairing routine for the time window constraint are as follows:

Step1: Transform all offspring $C(t)$ by decoding routine.
Step2: Calculate a time difference between first and last operations in time window zone for each job in $C(t)$.
Step3: Compare a time difference in each job with the time window constraint to find an illegal job.
Step4: If there is an illegal job, the first operation shifts to right before the last operation on the same machine.

After drawing the Gantt chart of a chromosome or an offspring, a local search can be conducted to improve $C(t)$ in order to reduce the idle time. Left-shift algorithm by Abe and Ida (2008) is suitable to apply the RFS scheduling problem in this paper. The left-shift procedure is shown by the steps as follows:

Step1: Transform all offspring $C(t)$ by decoding routine.
Step2: Calculate all idle times on each machine.
Step3: Check all idle times to move left side for an operation in each partial sequence by comparing the precedence relationship on the same machine.
Step4: Repeat steps 2–3 until there are no more left-shift operations.

5. EXPERIMENTAL RESEARCH

Two types of data problems were generated for all data. They were derived by standardization of industrial case, such as the standardized problem for 17 workstations, and the simple problem for 7 workstations by covered selection (Tables 2–5).

In Table 2, all of the processing times on each machine by operations is detailed for the standardized problem data set. Another data set is the simple problem as listed in Table 3. Both of the Tables also include the time window details.

Table 2. Data set of processing time and time window of the standardized problem

Machine	Operation	Processing time (min)
		Standardized data
WS-1	o_1	6
WS-2	o_{2a}	7
	o_{2b}	
	o_{2c}	
	o_{2d}	
WS-3	o_3	9
WS-4	o_4	5
WS-5	o_5	5
WS-6	o_6	5
WS-7	o_7	54
WS-8	o_8	7
WS-9	o_9	8
WS-10	o_{10}	7
WS-11	o_{11A}	10
	o_{11B}	
WS-12	o_{12A}	7
	o_{12B}	
WS-13	o_{13A}	7
	o_{13B}	8
WS-14	o_{14A}	6
	o_{14B}	6
WS-15	o_{15A}	5
	o_{15B}	
WS-16	o_{16}	13
WS-17	o_{17a}	11
	o_{17b}	10
	o_{17c}	12

Time window 300 min/lot

Table 3. Data set of processing time and time window of the simple problem

Machine	Operation	Processing time (min)
		Simple data
WS-1	o_1	6
WS-2	o_{2a}	7
	o_{2b}	
	o_{2c}	
WS-3	o_3	5
WS-4	o_4	54
WS-5	o_5	8
WS-6	o_{6A}	7
	o_{6B}	8
WS-7	o_{7a}	11
	o_{7b}	10
	o_{7c}	12

Time window 30

When considering lot sizes, data is shown in Table 4 for the standardized problem and Table 5 for the simple problem. So, 11 jobs with 220 lots per period is the problem size for the standardized problem; 4 jobs with

61 lots per period is the simple problem. Additionally, both tables show the different product types with "A" being the first type and "B" being the last type. Also, all of them can be divided into three sub-types.

In Table 2, all of the processing times on each machine by operations is detailed for the standardized problem data set. Another data set is the simple problem as listed in Table 3. Both of the Tables also include the time window details.

When considering lot sizes, data is shown in Table 4 for the standardized problem and Table 5 for the simple problem. So, 11 jobs with 220 lots per period is the problem size for the standardized problem; 4 jobs with 61 lots per period is the simple problem. Additionally, both tables show the different product types with "A" being the first type and "B" being the last type. Also, all of them can be divided into three sub-types.

The parameters for each data type were defined from pilot simulation results as shown in Tables 6 and 7. First, finding the appropriate GA parameters included crossover rate (p_C) and mutation rate (p_M). The simple problem was computationally tested by four numerical experiments giving different parameters with two levels of p_C (0.6 and 0.8), and two levels of p_M (0.1 and 0.2). The number of populations (*popSize*) is fixed at 10, and the maximum number of generations (*maxGen*) is fixed at 100. Lastly, testing each combination takes place 10 times.

Table 4. Data set of product types and number of lots of the standardized problem

Product type	Job	No. of lots
		Standardized data
A1	J_1	9
A1	J_2	11
A2	J_3	32
A1	J_4	25
A2	J_5	30
A3	J_6	12
B1	J_7	12
B2	J_8	17
B3	J_9	24
B2	J_{10}	38
B3	J_{11}	10

Table 5. Data set of product types and number of lots of the simple problem

Product type	Product	No. of lots
		Simple data
A1	J_1	9
A2	J_2	30
B1	J_3	12
B3	J_4	10

Table 6. Different genetic algorithm parameters with testing a simple RFS scheduling problem without lot sizes

No.	Simple problem without lot sizes							
	$p_C = 0.6$				$p_C = 0.8$			
	$p_M = 0.1$		$p_M = 0.2$		$p_M = 0.1$		$p_M = 0.2$	
	C_{max} (min)	CPU (s)	C_{max} (min)	CPU (s)	C_{max} (min)	CPU (s)	C_{max} (min)	CPU (s)
1	267	**1**	**266**	**2**	**266**	2	**266**	2
2	**266**	2	**266**	**2**	**266**	2	267	**1**
3	268	2	**266**	**2**	**266**	2	267	2
4	267	2	268	**2**	268	2	267	2
5	**266**	2	267	**2**	267	**1**	267	2
6	**266**	2	267	**2**	**266**	2	**266**	2
7	270	2	267	**2**	267	2	**266**	2
8	**266**	2	268	**2**	**266**	2	**266**	2
9	267	2	268	**2**	268	2	268	2
10	268	2	267	**2**	268	2	**266**	2
Best	266	1	266	1	266	1	266	1
Mean	267.1	1.9	267	2	266.8	1.9	266.6	1.9
SD	1.29	0.32	0.82	0.00	0.92	0.32	0.70	0.32

RFS: reentrant flow-shop, p_C: crossover rate, p_M: mutation rate, C_{max}: makespan, SD: standard deviation.

Table 7. Different genetic algorithm parameters with testing a simple RFS scheduling problem with lot sizes

No.	Simple problem with lot sizes							
	$p_C = 0.6$				$p_C = 0.8$			
	$p_M = 0.1$		$p_M = 0.2$		$p_M = 0.1$		$p_M = 0.2$	
	C_{max} (min)	CPU (s)	C_{max} (min)	CPU (s)	C_{max} (min)	CPU (s)	C_{max} (min)	CPU (s)
1	4,051	**1**	3,900	**2**	**3,864**	2	3,937	**2**
2	3,864	2	3,909	**2**	3,869	2	3,965	**2**
3	3,964	2	3,856	**2**	3,985	2	3,898	**2**
4	3,909	2	3,886	**2**	**3,864**	2	3,877	**2**
5	4,090	2	3,966	**2**	3,882	2	3,924	**2**
6	3,891	2	4,023	**2**	3,940	**1**	3,934	**2**
7	**3,849**	**1**	3,897	**2**	3,942	2	**3,831**	**2**
8	3,987	2	4,017	**2**	4,012	2	3,922	**2**
9	3,972	2	4,056	**2**	3,972	2	3,890	**2**
10	4,080	2	**3,828**	**2**	3,955	2	3,879	**2**
Best	3,849	1	3,828	2	3,864	1	3,831	2
Mean	3,963.9	1.8	3,933.8	2	3,928.5	1.9	3,906.5	2
SD	87.81	0.42	77.09	0.00	54.87	0.32	38.19	0.00

RFS: reentrant flow-shop, p_C: crossover rate, p_M: mutation rate, C_{max}: makespan, SD: standard deviation.

After testing GA parameters from Tables 6 and 7, the GA parameters p_C = 0.8 and p_M = 0.2 have been selected to give the best values. Next, three experimental pro-

Table 8. The values of parameters with different problem type and summarized solutions with the average

Problem type	Parameter			SA			GA			HGA		
	Searched solutions	*popSize*	*maxGen*	Makspan (min)	Loss	CPU (min)	Makspan (min)	Loss	CPU (min)	Makspan (min)	Loss	CPU (min)
Without lot size												
Simple	1000	10	100	268.2	1.0	0.01	267.2	1.2	0.03	266.6	0	0.09
	20000	20	1000	266	0.8	0.32	266.0	1.1	0.62	266.0	0	1.21
Standard	10000	10	1000	796	1.4	1.99	784.9	0.8	2.35	720.5	0	11.83
	40000	20	2000	780.6	0.6	8.2	770.8	0.7	9.11	718.2	0	29.40
	10000	10	1000	2,585.7	5.0	2.06	2,579.5	4.4	2.42	2,489.5	0	13.23
Real	40000	20	2000	2,553.9	3.2	8.9	2,553.8	4.2	9.10	2,484.1	0	32.25
With lot size												
Simple	1000	10	100	3,924.8	0.8	0.01	3,908.3	0.9	0.04	3,854.6	0	0.08
	20000	20	1000	3,823.6	0.2	0.45	3,829.6	0.4	0.56	3,820.3	0	1.09
Standard	10000	10	1000	16,348.4	1.8	1.77	16,270.8	1.9	2.36	14,332.8	0	11.72
	40000	20	2000	15,734.4	0.8	7.93	15,789.4	0.7	8.97	14,128.0	0	29.51
Real	10000	10	1000	41,481.7	6.4	1.86	41,455.0	4.6	2.27	38,692.4	0	17.02
	40000	20	2000	40,964.6	5.0	8.47	40,866.6	4.6	9.20	38,276.9	0	40.65

SA: simulated annealing, GA: genetic algorithm, HGA: hybrid genetic algorithm, popSize: the number of populations, maxGen: the maximum number of generations, Loss: the number of lot-losses.

blems will be compared for different sizes of *popSize* and *maxGen*. Also, the replications are considered at several levels.

The proposed algorithms were run with MATLAB on a 2.27 GHz PC, with 2 G-Byte of RAM, for testing and evaluation. The solutions to be tested and evaluated are: the makespan (C_{max}) and the number of lot-losses (Loss). The results of the average values, the standard deviation values, and the best solution obtained from the different computations were obtained from 10 calculations for each combination. Lastly, all of the problems are summarized in Table 8 by different problem type. Additionally, the proposed algorithms were also compared with simulated annealing (SA) in order to evaluate their performance with the same number of searched solutions (*popSize*×*maxGen*). The computational results reveal that GA yields better solutions than SA for most test problems (Table 8).

6. CONCLUSION

In this research, HGA were developed for solving the RFS scheduling problems with consideration of critical production time control (i.e., time windows constraint) according to the situations of the hard-disk manufacturing system. The simple test problems and the standardization problems were brought to experiment on the efficiency of the approaches. Eventually, the real processing time and reducing lot sizes could be conducted as real problem values. The experimental results prove that the HGA can be applied for solving all problem types with the best solution by average. Moreover, it can improve the jobs exceeding time windows by minimizing lot losses. However, the related computational time showed that there are different CPU times under different test problems. Nevertheless, a reentrant hybrid flow-shop system will be expanded from the simplification of the RFS system presented in this paper to find the best solutions for the industrial case in future research.

ACKNOWLEDGMENT

This project was financially supported by the Industry/University Cooperative Research Center (I/UCRC) in HDD Components, the Faculty of Engineering, Khon Kaen University and National Electronics and Computer Technology Center, National Science and Technology Development Agency, National Tsing Hua University, Taiwan (NSC 101-2811-E-007-004; NSC 102-2811-E-007-005; NSC 100-2628-E-007-017-MY3; NTHU 101-N2073E1) and Fuzzy Logic Systems Institute, Japan (JSPS-the Grant-in-Aid for Scientific Research C; No. 24510219).

REFERENCES

Abe, K. and Ida, K. (2008), Genetic local search method for re-entrant flow shop problem, *Proceedings of the Artificial Neural Networks in Engineering Conference*, St. Louis, MO, 381-387.

Chamnanlor, C. and Sethanan, K. (2009), Mixed integer programming models for scheduling hybrid flow-shop with unrelated machines and time windows constraints, *Proceedings of the 10th Asia Pacific*

Industrial Engineering and Management System Conference, Kitakyushu, Japan, 2331-2335.

Chen, J. S., Pan, J. C. H., and Lin, C. M. (2008a), A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem, *Expert Systems with Applications*, **34**(1), 570-577.

Chen, J. S., Pan, J. C. H., and Wu, C. K. (2007), Minimizing makespan in reentrant flow-shops using hybrid tabu search, *International Journal of Advanced Manufacturing Technology*, **34**(3/4), 353-361.

Chen, J. S., Pan, J. C. H., and Wu, C. K. (2008b), Hybrid tabu search for re-entrant permutation flow-shop scheduling problem, *Expert Systems with Applications*, **34**(3), 1924-1930.

Chien, C. F. and Chen, C. H. (2007), A novel timetabling algorithm for a furnace process for semiconductor fabrication with constrained waiting and frequency-based setups, *OR Spectrum*, **29**(3), 391-419.

Gao, J., Gen, M., and Sun, L. (2006), Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, *Journal of Intelligent Manufacturing*, **17**(4), 493-507.

Gen, M. and Cheng, R. (2000), *Genetic Algorithms and Engineering Optimization*, John Wiley and Sons, New York, NY.

Gen, M., Cheng, R., and Lin, L. (2008), *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*, Springer, London.

Gen, M., Gao, J., and Lin, L. (2009), Multistage-based genetic algorithm for flexible job-shop scheduling problem, In: *Intelligent and Evolutionary Systems*, Springer, Heidelberg, Germany, 183-196.

Gen, M., Tsujimura, Y., and Kubota, E. (1994), Solving job-shop scheduling problems by genetic algorithm, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, TX, 1577-1582.

Goncalves, J. F., de Magalhaes Mendes, J. J., and Resende, M. G. (2005), A hybrid genetic algorithm for the job shop scheduling problem, *European Journal of Operational Research*, **167**(1), 77-95.

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

Jing, C., Tang, G., and Qian, X. (2008), Heuristic algorithms for two machine re-entrant flow shop, *Theoretical Computer Science*, **400**(1), 137-143.

Kang, Y. H., Kim, S. S., and Shin, H. J. (2007), A scheduling algorithm for the reentrant shop: an application in semiconductor manufacture, *International Journal of Advanced Manufacturing Technology*, **35**(5/6), 566-574.

Lin, L. and Gen, M. (2009), Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation, *Soft Computing*, **13**(2), 157-168.

Lin, L., Gen, M., and Wang, X. (2009), Integrated multistage logistics network design by using hybrid evolutionary algorithm, *Computers and Industrial Engineering*, **56**(3), 854-873.

Lin, L., Hao, X. C., Gen, M., and Jo, J. B. (2012), Network modeling and evolutionary optimization for scheduling in manufacturing, *Journal of Intelligent Manufacturing*, **23**(6), 2237-2253.

Monch, L., Fowler, J. W., Dauzere-Peres, S., Mason, S. J., and Rose, O. (2011), A survey of problems, solution techniques, and future challenges in scheduleing semiconductor manufacturing operations, *Journal of Scheduling*, **14**(6), 583-599.

Pan, J. H. and Chen, J. S. (2003), Minimizing makespan in re-entrant permutation flow-shops, *Journal of the Operational Research Society*, **54**(6), 642-653.

Park, Y., Kim, S., and Jun, C. H. (2000), Performance analysis of re-entrant flow shop with single-job and batch machines using mean value analysis, *Production Planning and Control*, **11**(6), 537-546.

Ruiz, R., Maroto, C., and Alcaraz, J. (2005), Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics, *European Journal of Operational Research*, **165**(1), 34-54.

Sethanan, K. (2001), *Scheduling flexible flowshops with sequence dependent setup times*, Ph.D. dissertation, West Virginia University, Morgantown, WV.