# Multiobjective Hybrid Genetic Algorithms for Manufacturing Scheduling: Part II Case Studies of HDD and TFT-LCD

**Mitsuo Gen, Wenqiang Zhang and Lin Lin**

**Abstract** Manufacturing scheduling is one of the most important and complex combinatorial optimization problems, where it can have a major impact on the productivity of a production process. Moreover, most of manufacturing scheduling problems fall into the class of NP-hard combinatorial problems. In this paper, we introduce how to design hybrid genetic algorithms (HGA) and multiobjective hybrid genetic algorithms (Mo-HGA) for solving practical manufacturing scheduling problems for the hard disc device (HDD) and the thin-film transistor-liquid crystal display (TFT-LCD) manufacturing systems, respectively. In particularly, evolutionary representations and the fitness assignment mechanism as well as the hybrid genetic operations are introduced. Through a variety of computational experiments, the effectiveness of these HGA algorithm for HDD and Mo-HGA algorithm for TFT-LCD module assembly as the practical manufacturing scheduling problems are demonstrated. This paper introduces how to design Mo-HGAs for solving the practical multiobjective manufacturing scheduling problems expanded by a multiobjective flexible job-shop scheduling problem (Mo-FJSP; operation sequencing and resources assignment).

**Keywords** Hybrid genetic algorithms (HGA) · Multiobjective HGA (Mo-HGA) · Manufacturing scheduling · Flexible job-shop scheduling problem (FJSP) · Hard disc device (HDD) · Thin-film transistor-liquid crystal display (TFT-LCD)

M. Gen (✉)
Fuzzy Logic Systems Institute, Tokyo University of Science, Tokyo, Japan
e-mail: gen@flsi.or.jp

W. Zhang
College of Information Science & Engineering, Henan University of Technology, Henan, People's Republic of China

L. Lin
School of Software Technology,
Dalian University of Technology, Dalian, People's Republic of China

J. Xu et al. (eds.), *Proceedings of the Ninth International Conference on Management Science and Engineering Management*, Advances in Intelligent Systems and Computing 362, DOI 10.1007/978-3-662-47241-5_2

## 1 Introduction

The semiconductor industry has grown rapidly and subsequently production planning problems have raised many important research issues. Because of short product life cycles, it is crucial to rapidly respond to various customer needs and deliver products on time in high-tech semiconductor manufacturing industries such as the various semiconductor devices including IC chips, LSI chips and microprocessors, thin-film transistor-liquid crystal display (TFT-LCD) and hard disc device (HDD). Scheduling problems for semiconductor manufacturing have the features of scheduling with auxiliary resources, batching processes, multiple orders per job, internal and external scheduling of cluster tools, a large number of processing steps, random equipment failures, waiting time constraints [5, 18], job-shop scheduling problems with reentrance, sequence-dependent setup time (SDST), and sequence-dependent processing time (SDPT) [11]. Flexible manufacturing systems (FMS) allow the dynamic configuration of resources to process distinct products. It is possible to assign each of these products to more than one type of unrelated resource with various efficiencies. Multipurpose resources can perform a wide range of tasks, which allows schedulers to concentrate the workloads among these resources to improve the utilization and reduce resource requirements. A possible negative effect is the reduction of production effectiveness because of the time wasted when a machine performs changeover and configuration to accommodate for the next job. Thus, it is vital to arrange a production schedule that simultaneously considers the values of multiple resources to respond to production requirements rapidly and effectively [11, 19].

The HDD and TFT-LCD manufacturing is capital and technology intensive industry. Facing the fierce competitive pressures, it is important to enhance productivity and operational efficiency. Manufacturing scheduling of TFT-LCD module assembly system is a key issue to enhance manufacture efficiency that could satisfy customer demand on time [6].

By focusing on realistic settings, a module assembly process was formulated for use in the HDD and TFT-LCD industries as a generalization of the flexible job-shop scheduling problem (FJSP), respectively. On a flexible job-shop floor, workstations employ non-identical parallel machines scheduling (PMS) model and reentrant flowshop scheduling (RFS) model that exhibit distinct production velocities. An operation can be processed using an available machine from a given workstation. For example, the TFT-LCD module assembly scheduling problem can be divided into two subproblems: the routing (i.e., assigning each operation to machines) and scheduling problems (i.e., determining the start time of each operation to machines). The following factory-specific factors complicate the TFT-LCD module assembly scheduling problem. Bidotet [2] reported detail definitions to avoid ambiguity of terms commonly used by different communities: complete schedule, flexible schedule, conditional schedule, predictive schedule, executable schedule, adaptive scheduling system, robust predictive schedule and table predictive schedule. However, to find the optimal solutions of manufacturing scheduling gives rise to complex combinatorial optimization, unfortunately, most of them fall into the class of NP-hard combinatorial problems.

The rest of this paper is organized as follows: Sect. 2 introduces hybrid reentrant flow-shop scheduling (RFS) model in HDD manufacturing. The case study of hybrid RFS problem withtime window constraints for HDD module assembly system byhybrid genetic algorithm (HGA) with left-shift routine for improving and fuzzy logic controller (FLC) for tuning parameters introduces and the effectiveness of computational results demonstrates in Sect. 3. After introducing another case study of manufacturing model in TFT-LCD module assembly manufacturing system in Sect. 4, multiobjectivehybrid GA (Mo-HGA) algorithm with TOPSIS (technique for order preference by similarity to ideal solution) introduces how to design a chromosome, treat precedence relationship and fitness assignment mechanism, and clarifies effectiveness and efficiency in the best compromised solution as a quality of solution with reasonable interactive computational time. Finally, the conclusion of the paper and future research are drawn in Sect. 6.

## 2 Hybrid Reentrant Model in HDD Manufacturing

The semiconductor industry has grown rapidly, and subsequently production planning problems have raised many important research issues. The reentrant flow-shop scheduling (RFS) 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 makes pan of jobs is considered. This research addresses the HGA with auto-tuning parameters for the deterministic $F_m|fmls, rcrc, temp|C_{\max}$ problem.

A HDD (hard disk device) 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. HDD manufacturing consists of four Main processes as shown in Fig. 1.

| | |
|---|---|
| SF | (slider fabrication) |
| HGA | (head gimbals assembly) |
| HSA | (head stack assembly) |
| HDD | (hard disk drive assembly) |

HDD process is consisting of the following two serial one.

Assembly Lines and Backend Process, Backend Process consists of the following eight production stages [16]:

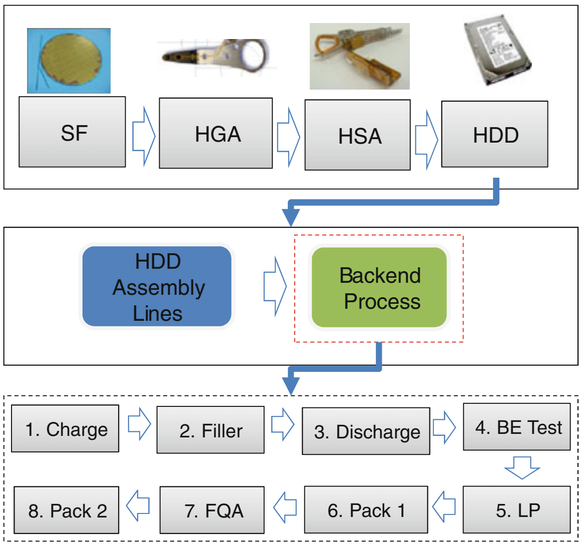


**Fig. 1** Main processes of HDD manufacturing

(1) Charge (Helium Charge);
(2) Filler Test;
(3) Discharge (Helium Discharge);
(4) BE test (Backend test);
(5) LP (Label Printing);
(6) Pack 1;
(7) FQA (Sampling constraint);
(8) Pack 2.

In this section, we consider assembly line process for HDD (hard disk drive) manufacturing system as shown in Fig. 2. This is the hybrid flow-shop in a real hard-disk manufacturing system, there are 9 processes with 17 workstations such as a complex parallel machine scheduling (PMS) model introduced in Sect. 2. 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 system [3, 4].

Without any generality of HDD manufacturing system, we can consider a small scale of there entrant flow-shop scheduling (RFS) problem for producing 4 different

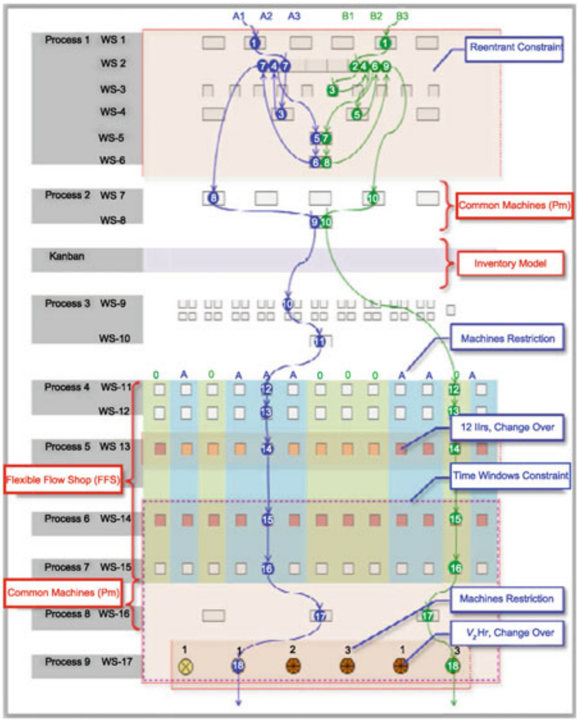


**Fig. 2** Real model of HDD manuf. system

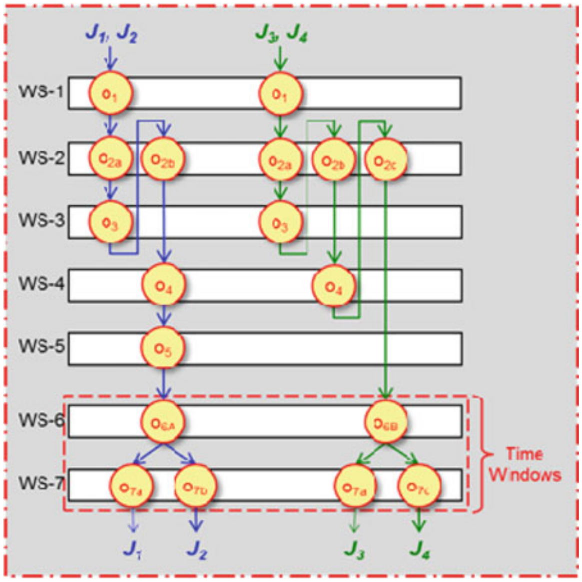
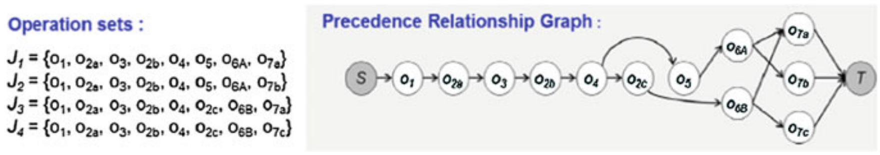


**Fig. 3** Processing flow of a simple RFS problem



**Operation sets :**

$J_1 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_5, o_{6A}, o_{7a}\}$
$J_2 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_5, o_{6A}, o_{7b}\}$
$J_3 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_{2c}, o_{6B}, o_{7a}\}$
$J_4 = \{o_1, o_{2a}, o_3, o_{2b}, o_4, o_{2c}, o_{6B}, o_{7c}\}$

**Fig. 4** Precedence relationship graph of simple RFS problem

HDD products under 7 workstations as shown in Fig. 3. Considering the processing flow of a simplified RFS scheduling problem as shown in Fig. 4, 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 ($t_w = 30$ 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 shown 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 as shown in Fig. 4.

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

## 3 Hybrid Genetic Algorithm with Left-Shift Routine and Computational Results

*Genetic Representation*: Gen et al. [9] 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 [10]. Furthermore, we can apply it to the RFS scheduling problem. After creating the chromosome by the encoding routine, the schedule can be generated. When generating it, an operation can be started whenever its predecessor has been finished and the machine to process it is available. The generated schedule of the example chromosome in Fig. 5 is shown as follows: Schedule $S = (o_{ij}, M_m, s_{ij} - c_{ij})$; $o_{ij}$ 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 $i$.

$S = \{(o_{1,1}, M_1, 0–6)$, $(o_{2,1}, M_1, 6–12)$, $(o_{3,1}, M_1, 12–18)$, $(o_{4,1}, M_1, 18–24)$, $(o_{1,2a}, M_2, 6–13)$, $(o_{1,3}, M_3, 13–18)$, $(o_{2,2a}, M_2, 13–20)$, $(o_{2,3}, M_3, 20–25)$,$(o_{3,2a}$, $M_2, 20–27)$, $(o_{3,3}, M3, 27–32)$, $(o_{4,2a}, M_2, 27–34)$, $(o_{4,3}, M_3, 34–39)$, $(o_{1,2b}, M_2$,

**Table 1** The operations of a simple example RFS 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 | $o_{7a}$ | – | $o_{7b}$ | – | $o_{7a}$ | – | $o_{7c}$ | – |

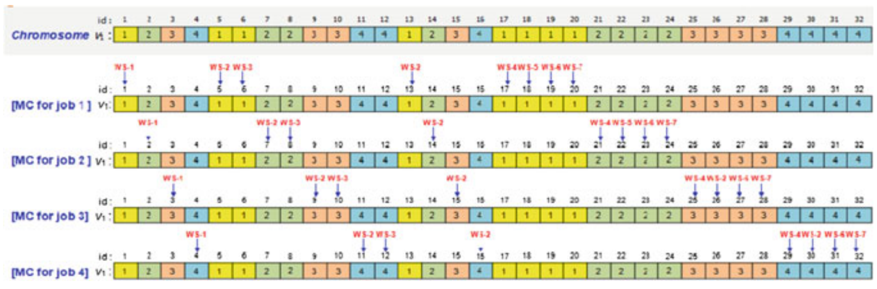


**Fig. 5** The illustration of a chromosome with operation-based representations

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 Fig. 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 Fig. 6. So, this sequence solution has made a 284-min makespan and no loss because there are no jobs exceeding the time windows (30 min between WS-6 and WS-7).

*Genetic Operations*: For creating offspring by genetic operations we used two-cut point crossover, swap-mutation and insert-mutation operations. For the detailed calculated results by genetic operations, we can get them in [3].

*Fitness Function*: In the RFS scheduling problem, the objective is to minimize the makespan (zi), so it is directly related with maximizing the system throughput. RFS scheduling in the hard-disk manufacturing system as in this paper, also has to consider the lost lot which exceeded the critical processing time. Equation (1) shown the fitness function of GA where vi is a chromosome vector i; the population is popSize.

$$\text{eval}(v_i) = 1/z_i, i = 1, 2, \ldots, \text{popSize}. \quad (1)$$

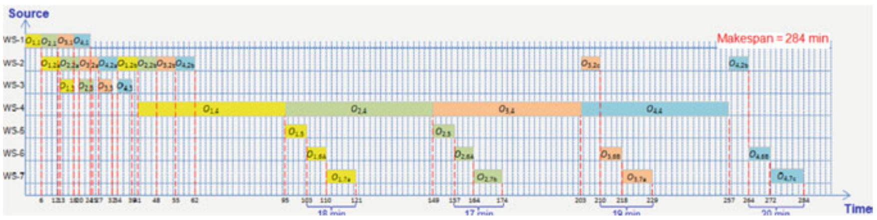
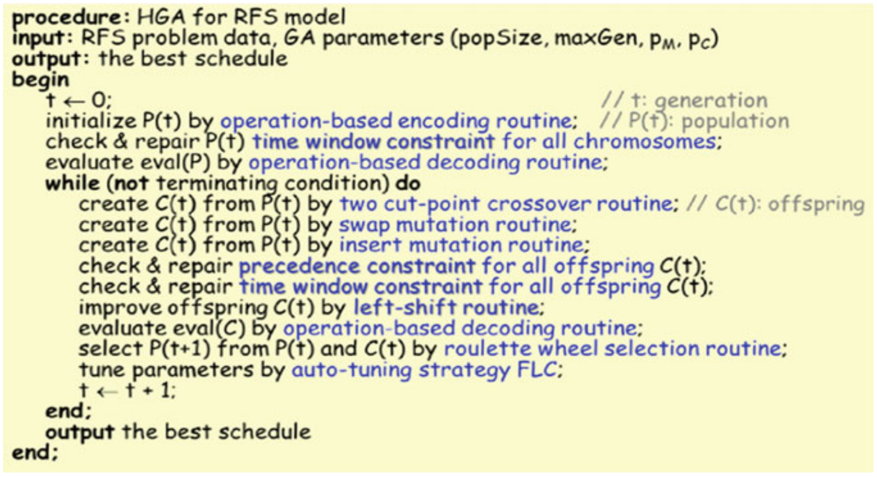


**Fig. 6** A Gantt chart of the generated chromosome



```
procedure: HGA for RFS model
input: RFS problem data, GA parameters (popSize, maxGen, $p_M$, $p_C$)
output: the best schedule
begin
    t ← 0;                                                          // t: generation
    initialize P(t) by operation-based encoding routine;   // P(t): population
    check & repair P(t) time window constraint for all chromosomes;
    evaluate eval(P) 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 & repair precedence constraint for all offspring C(t);
        check & repair time window constraint for all offspring C(t);
        improve offspring C(t) by left-shift routine;
        evaluate eval(C) by operation-based decoding routine;
        select P(t+1) from P(t) and C(t) by roulette wheel selection routine;
        tune parameters by auto-tuning strategy FLC;
        t ← t + 1;
    end;
    output the best schedule
end;
```

**Fig. 7** Overall procedure in the pseudo-code of hybrid GA for solving RFS problem

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.

The overall procedure in the pseudo-code of Hybrid GA for solving the reentrant flow-shop scheduling (RFS) problem for HDD manufacturing system, can be designed in Fig. 7. Regular genetic operations might construct illegal offspring in the next generation. In particular crossover and mutation operations, which are exploration and exploitation for the search space, might not produce a feasible solution. Then, checking and repairing offspring with system constraints should be done carefully.

*Checking Precedence Constraint*: 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:

**Step 1**. Transform all offspring $C(t)$ by the decoding routine.
**Step 2**. Compare each offspring $C(t)$ with operation sequences for each job.
**Step 3**. If there is an illegal offspring $C(t)$, repair it by the operation sequences based on the job.

*Checking Time Window Constraint*: In the same situation, the 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:

**Step 1**. Transform all offspring $C(t)$ by decoding routine.
**Step 2**. Calculate a time difference between first and last operations in time window zone for each job in $C(t)$.

**Step 3**. Compare a time difference in each job with the time window constraint to find an illegal job.
**Step 4**. If there is an illegal job, the first operation shifts to right before the last operation on the same machine.

*Local Search (Left-shift Algorithm)*: 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 [1] is suitable to apply the RFS scheduling problem in this paper. The left-shift procedure is shown by the steps as follows:

To improve the decoded schedule after drawing the Gantt chart, a heuristic of local search, namely the Left-shift algorithm [7] could be used for solving the RFS problem too. It will help to better minimize the makespan than the old Gantt chart [4].

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 (Table 2). 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 2. Both of the tables also include the time window details.

**Step 1**. Transform all offspring $C(t)$ by decoding routine.
**Step 2**. Calculate all idle times on each machine.
**Step 3**. Check all idle times to move left side for an operation in each partial sequence by comparing the precedence relationship on the same machine.
**Step 4**. Repeat steps 2-3 until there are no more left-shift operations.

When considering lot sizes, data is shown in Table 3 for the standardized problem and 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, this table shows 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 3, 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. This table also includes the time window details. When considering lot sizes, data is shown in Table 4 for the standardized problem and 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, this table shows 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 more detailed computational results by the proposed hybrid GA with local search and fuzzy logic controller demonstrated the effectiveness and efficiency by solving a real case of an HDD manufacturing system. The parameters $p_C$ and $p_M$ were variously changed to automatically regulate a suitable balance between exploitation and exploration during the evolutionary process of the hybrid genetic algorithm with fuzzy logic controller (HGA.FLC). Recently Sangsawang et al. [16] proposed metaheuristics optimization approaches for solving the two-stage reentrant FFS (RFFS) problem with blocking constraint (FFS|2-stage,rcrc,block|$C_{\max}$)

**Table 2** Data set of processing time and time window of the standardized and simple problems

| Machines | Operations | ProcTime (m) | |
|---|---|---|---|
| | | Real data ($x_i$) | Standardized data ($10(y_i + 1)$) |
| WS-1 | $o_1$ | 6.5 | 6 |
| | $o_{2a}$ | 10 | 7 |
| | $o_{2b}$ | 10 | 7 |
| WS-2 | $o_{2c}$ | 10 | 7 |
| | $o_{2d}$ | 10 | 7 |
| WS-3 | $o_3$ | 16 | 9 |
| WS-4 | $o_4$ | 2 | 5 |
| WS-5 | $o_5$ | 0.3 | 5 |
| WS-6 | $o_6$ | 0.3 | 5 |
| WS-7 | $o_7$ | 210 | 54 |
| WS-8 | $o_8$ | 10 | 7 |
| WS-9 | $o_9$ | 15 | 8 |
| WS-10 | $o_{10}$ | 8 | 7 |
| WS-11 | $o_{11A}$ | 20 | 10 |
| | $o_{11B}$ | 20 | 10 |
| WS-12 | $o_{12A}$ | 8 | 7 |
| | $o_{12B}$ | 8 | 7 |
| WS-13 | $o_{13A}$ | 10.2 | 7 |
| | $o_{13B}$ | 11.5 | 8 |
| WS-14 | $o_{14A}$ | 4.8 | 6 |
| | $o_{14B}$ | 5.6 | 6 |
| WS-15 | $o_{15A}$ | 1.5 | 5 |
| | $o_{15B}$ | 1.5 | 5 |
| WS-16 | $o_{16}$ | 35 | 13 |
| | $o_{17a}$ | 26.5 | 11 |
| WS-17 | $o_{17b}$ | 23 | 10 |
| | $o_{17c}$ | 32 | 12 |
| WS-1 | $o_1$ | | 6 |
| | $o_{2a}$ | | 7 |
| WS-2 | $o_{2b}$ | | 7 |
| | $o_{2c}$ | | 7 |
| WS-3 | $o_3$ | | 5 |
| WS-4 | $o_4$ | | 54 |
| WS-5 | $o_5$ | | 8 |
| WS-6 | $o_{6A}$ | | 7 |
| | $o_{6B}$ | | 8 |

WS-14 to WS-17: time windows = 300 min/lot, WS-6 to WS-7: time windows = 30 min/lot

**Table 3** Data set of product types and number of lots of the standardized and simple problems

| | | No. of lots | |
|---|---|---|---|
| Product types | Products | Real data($x_i$) | Standardized data ($10(y_i + 2)$) |
| A1 | P-1 | 100 | 9 |
| A1 | P-2 | 175 | 11 |
| A2 | P-3 | 1,243 | 32 |
| A1 | P-4 | 866 | 25 |
| A2 | P-5 | 1,137 | 30 |
| A3 | P-6 | 228 | 12 |
| B1 | P-7 | 228 | 12 |
| B2 | P-8 | 510 | 17 |
| B3 | P-9 | 841 | 24 |
| B2 | P-10 | 1533 | 38 |
| B3 | P-11 | 139 | 10 |
| A1 | P-1 | | 9 |
| A2 | P-2 | | 30 |
| B1 | P-3 | | 12 |
| B3 | P-4 | | 10 |

in which they applied a hybrid GA and a hybrid particle swarm optimization (HPSO) with Cauchy distribution.

## 4 Manufacturing Model in TFT-LCD Module Assembly

Because of short product lifecycles, it is crucial to rapidly respond to various customer needs and deliver products on time in high-tech industries such as the thin-film transistor-liquid crystal display (TFT-LCD) and semiconductor manufacturing industries. The TFT-LCD manufacturing is capital and technology intensive industry. Facing the fierce competitive pressures, it is important to enhance productivity and operational efficiency. Manufacturing scheduling of TFT-LCD Module Assembly system is a key issue to enhance manufacture efficiency that could satisfy customer demand on time [6]. By focusing on realistic settings, a module assembly process was formulated for use in the TFT-LCD industry as a generalization of the flexible job-shop scheduling problem (FJSP). On a flexible job-shop floor, workstations employ non-identical parallel machines scheduling (PMS) that exhibit distinct production velocities. An operation can be processed using an available machine from a given workstation. The TFT-LCD module assembly scheduling problem can be divided into two subproblems: the routing (i.e., assigning each operation to machines) and scheduling problems (i.e., determining the start time of each operation to machines).

**Table 4** Computational result by GA and HGA for the standardized and the simple problems

| Problem types | Parameters | | | GA without time window | | | HGA with time window | |
|---|---|---|---|---|---|---|---|---|
| | popSize | maxGen | Makspan (m) | Loss | CPU (m) | Makespan (m) | Loss | CPU (m) |
| Simple without lot size | 10 | 100 | 267.2 | 1.2 | 0.03 | 266.6 | 0 | 0.09 |
| | 20 | 1000 | 266 | 1.1 | 0.62 | 266 | 0 | 1.21 |
| Standard without lot size | 10 | 1000 | 784.9 | 0.8 | 2.35 | 720.5 | 0 | 11.83 |
| | 20 | 2000 | 770.8 | 0.7 | 9.11 | 718.2 | 0 | 29.4 |
| Simple with lot size | 10 | 100 | 3,908.30 | 0.9 | 0.04 | 3,854.60 | 0 | 0.08 |
| | 20 | 1000 | 3,829.60 | 0.4 | 0.56 | 3,820.30 | 0 | 1.09 |
| Standard with lot size | 10 | 1000 | 16,270.80 | 1.9 | 2.36 | 14,332.80 | 0 | 11.72 |
| | 20 | 2000 | 15,789.40 | 0.7 | 8.97 | 14,128.00 | 0 | 29.51 |

The following factory-specific factors complicate the TFT-LCD module assembly scheduling problem.

The TFT-LCD module assembly production is one of FJSP models that is critical to satisfy the customer demands on time. On the module assembly shop floor, each workstation has identical and non-identical parallel machines that access the jobs at various processing velocities depending on the product families. To satisfy the various jobs, the machines need to be set up as the numerous tools to conduct consecutive products. This study aims to propose a novel approach to address the TFT-LCD module assembly scheduling problem by simultaneously considering the following multiple and often conflicting objectives such as the makespan, the weighted number of tardy jobs, and the total machine setup time, subject to the constraints of product families, non-identical parallel machines, and sequence-dependent setup times [6].

The TFT-LCD manufacturing process is divided into three main stages: Array/CF (color filter) process, Cell process, and Module process (Fig. 8). The Array/CF process is similar to semiconductor wafer fabrication except that transistors are built up on the glass substrate instead of silicon wafer, and the processes are also re-entrant flow. The Cell process attaches the Array substrate and CF substrate together, and fills the liquid crystal between two substrates. The Module process, the final stage, is to assemble all customized components as the finished goods. The Module process stage involves six workstations that assemble customized components (e.g., integrated circuit, printed circuit board, driver board, backlight, and chassis) onto the panels to complete the final TFT-LCD production:

(1) The IC (integrated circuit) bonding.
(2) The PCB (printed circuit board) bonding.
(3) The components assembly.
(4) The burn-in test.
(5) The inspection.
(6) The packing and shipping.

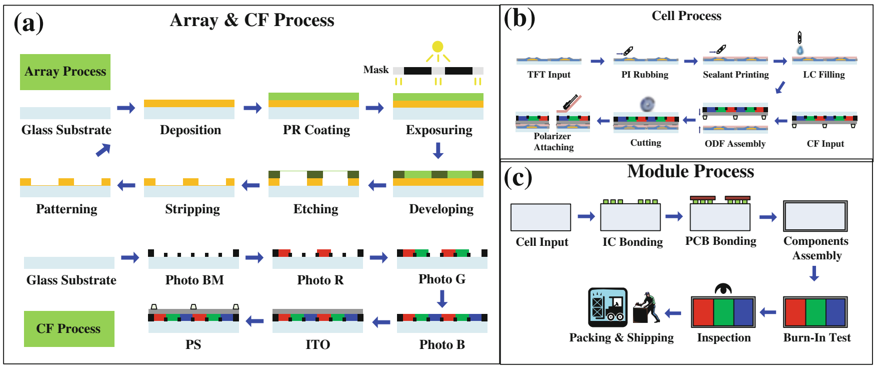


**Fig. 8** Three main stages of the TFT-LCD manufacturing process: **a** Array process, **b** Cell process, and **c** Module process

The module assembly process is the final stage of TFT-LCD manufacturing process, and the finished goods are directly shipping to customers. Therefore, it is very important to deliver product on time. In order to keep efficiency and commit customer's due date, the Module assembly scheduling problem should be considered the following multiple and conflicting objectives simultaneously: minimizing makespan, minimizing total workload, and maximizing confirmed line item performance (CLIP) rate. The CLIP is a measure of customer satisfaction reliability, meaning the percentage of order requests that was delivered as promised (i.e., commit customer's due date). The CLIP has been used as the major key performance indicator in high-tech industry [14]. However, if the scheduling in order to commit the customer's due date, its lead to a larger makespan and workload [17]. It's dilemma to minimize makespan, total workload and maximize CLIP rate at the same time in the shop-floor manufacturing environment. The minimizing makespan and minimizing total work load are the effective objectives, and the maximizing CLIP rate is the objective that directly related to customer service by keeping manufacture efficiency and committing customer's due date. Therefore, this paper considers the conflicting objectives simultaneously to find a suitable compromised schedule.

On the module assembly shop floor, the manufacturing process consisting of five workstations with 10 machines (WS-1: JI with 3 m/c, WS-2: 3D VAS with 1 m/c, WS-3: Packer with 1 m/c, WS-4:MA with 2 m/c and WS-5:3D Cal. with 1 m/c) depend on the product family. Figure 9 shows the jobs with different product families and access to different routes. For instance, for Job 4 in Fig. 9, one of the products is a 3D-type panel, which requires laminating the 3D glass substrate onto the panel after the cell process. It must then pass through the IC and PCB bonding, components assembly, and the test workstations. Before shipping, 3D products must pass through the 3D calibration workstation to calibrate the 3D product picture settings. In addition, two types of shipments exist depending on customer demand: semi-finished and finished goods. Semi-finished goods do not require assembling the customized electric components onto the panel.

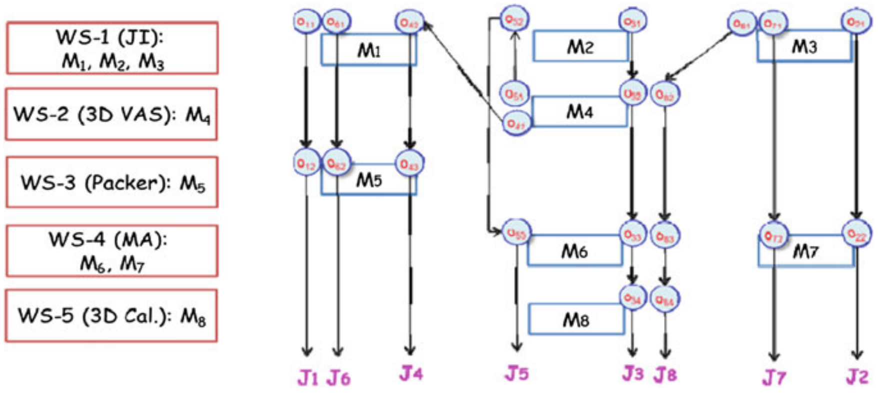


**Fig. 9** Processing flow in module assembly for the TFT-LCD manufacturing

Jobs can be assigned to non-identical machines and a high speed processing velocity is chosen to improve production efficiency. Choosing the fast processing time, the job could be finished quickly with earlier completion time. However, if all jobs are assigned to the fast processing time machine, the job must queue and also delay the completion time. Furthermore, jobs in varying product families require machine setup tools, increasing the total setup time and prolonging the job completion time.

Various product families can be produced using the varying process on a non-identical machine shop floor. The module assembly scheduling system requires selecting which job operation passes through which machine and determining the start time of each operation in each machine. Therefore, this study was conducted to solve the complex scheduling problem that considers incompatible product families, non-identical parallel machines, and sequence-dependent setup time (SDST) constraints in a real production system. There are several product types based on its specification (panel size, display type, shipping type, etc.) and Table 5 is product family case in Module Assembly for the TFT-LCD manufacturing.

The module process stage involves five workstations in which customized components (i.e., integrated circuit (IC), printed circuit board (PCB), driver board, backlight, and chassis) are assembled onto the panel.

(1) IC and PCB bonding (Workstation 1): Bonding the IC and PCB components onto the panel.
(2) The 3D substrate lamination (Workstation 2): Laminating the 3D substrate on the cell panel if the product is a 3D type.
(3) Semi-finished goods packing (Workstation 3): Semi-finished goods are packed to ship to the customer.
(4) Component assembly and testing (Workstation 4): Assembling customized electric components onto the panel.
(5) The 3D calibration (Workstation 5): Calibrating 3D product picture setting.

**Table 5** Product family cases in module assembly for the TFT-LCD manufacturing

| Job | Lot size | Due date | Display type | Operation sequence | Product type |
|---|---|---|---|---|---|
| $i$ | $q_i$ | $d_i$ (k s) | | $o_{ik}$ | |
| 1 | 250 | 45 | 2D | $o_{11}, o_{12}$ | Small Size_SKD |
| 2 | 500 | 65 | 2D | $o_{21}, o_{22}$ | Large Size_Module |
| 3 | 400 | 60 | $3D\_OGS$ | $o_{31}, o_{32}, o_{33}, o_{34}$ | Large Size_Module |
| 4 | 200 | 60 | $3D\_GPR$ | $o_{41}, o_{42}, o_{43}$ | Small Size_SKD |
| 5 | 500 | 95 | $3D\_GPR$ | $o_{51}, o_{52}, o_{53}$ | Large Size_Module |
| 6 | 600 | 32 | 2D | $o_{61}, o_{62}$ | Small Size_SK |
| 7 | 400 | 75 | 2D | $o_{71}, o_{72}$ | Large Size_Module |
| 8 | 600 | 95 | $3D\_OGS$ | $o_{81}, o_{82}, o_{83}, o_{84}$ | Large Size_Module |

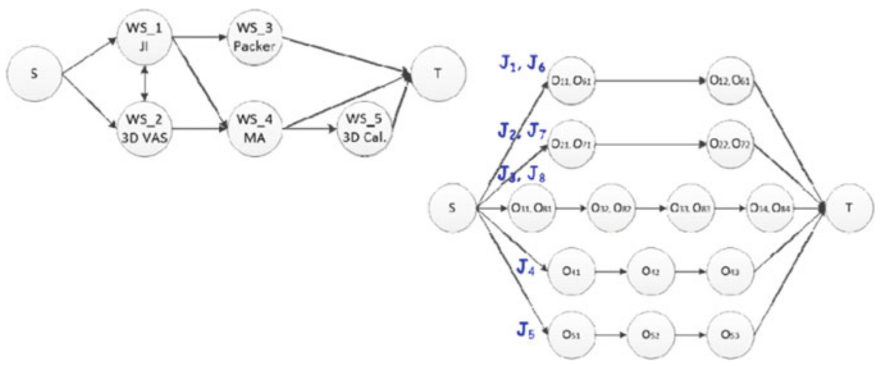


**Fig. 10** Precedence relationship graph in module assembly for the TFT-LCD manufacturing

Based on the precedence relationship of the processing flow in TFT-LCD module assembly, the following five different types of routes created as shown in Fig. 10.

The module process is the final stage of TFT-LCD manufacturing system, and the finished products are directly shipping to customers. The module assembly scheduling problem consider the multiple conflicting objectives simultaneously:

(1) Minimizing makespan.
(2) Minimizing total workload.
(3) Maximizing CLIP (confirmed line item performance).

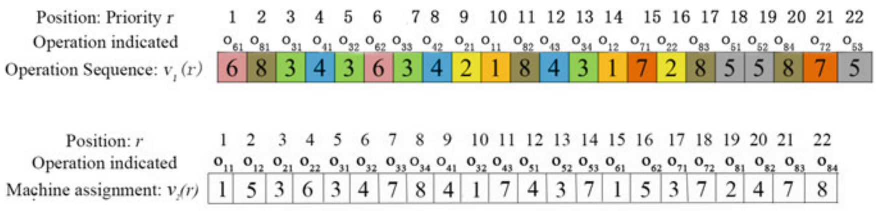
Constraints:

(1) Incompatible product families: The jobs requiring the same recipe can be regarded as a product family and have the same processing time in one machine. Incompatible product families cannot be processed together in one machine.
(2) Parallel machines.
(3) Sequence dependent setup time (SDST): With variety customers and product families, a machine setup time is required if two consecutive jobs of different product families in the same machine.

The TFT-LCD module assembly scheduling problem is formulated as a multi-objective mixed-integer linear programming (Mo-MILP) model as shown in Chou et al. [6].

## 5 Multiobjective Hybrid GA with TOPSIS and Computational Result

*Genetic Representation*: A common representation of the FJSP problem is designed using a two-vector chromosome that names all the operations of a job by using the same symbol, and interprets them according to the order of occurrence in the sequence



| Position: Priority $r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Operation indicated | $o_{61}$ | $o_{81}$ | $o_{31}$ | $o_{41}$ | $o_{32}$ | $o_{62}$ | $o_{33}$ | $o_{42}$ | $o_{21}$ | $o_{11}$ | $o_{82}$ | $o_{43}$ | $o_{34}$ | $o_{12}$ | $o_{71}$ | $o_{22}$ | $o_{83}$ | $o_{51}$ | $o_{52}$ | $o_{84}$ | $o_{72}$ | $o_{53}$ |
| Operation Sequence: $v_1(r)$ | 6 | 8 | 3 | 4 | 3 | 6 | 3 | 4 | 2 | 1 | 8 | 4 | 3 | 1 | 7 | 2 | 8 | 5 | 5 | 8 | 7 | 5 |

| Position: $r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Operation indicated | $o_{11}$ | $o_{12}$ | $o_{21}$ | $o_{22}$ | $o_{31}$ | $o_{32}$ | $o_{33}$ | $o_{34}$ | $o_{41}$ | $o_{32}$ | $o_{43}$ | $o_{51}$ | $o_{52}$ | $o_{53}$ | $o_{61}$ | $o_{62}$ | $o_{71}$ | $o_{72}$ | $o_{81}$ | $o_{82}$ | $o_{83}$ | $o_{84}$ |
| Machine assignment: $v_2(r)$ | 1 | 5 | 3 | 6 | 3 | 4 | 7 | 8 | 4 | 1 | 7 | 4 | 3 | 7 | 1 | 5 | 3 | 7 | 2 | 4 | 7 | 8 |

**Fig. 11** Scheme of operation sequence vector v1 and machine assignment vector $v_2$

of a given chromosome [10]. The TFT-LCD module assembly scheduling problem is a combination of operation scheduling and machine assignment decisions. Therefore, in this study, the chromosome was designed as two parts: an operation sequence vector ($v_1$) and a machine assignment vector ($v_2$). The evaluated TFT-LCD module assembly scheduling problem comprised eight jobs and eight machines, where each job required several operations. The operation sequence vector $v_1$ shows that each job $i$ appears nitimes, indicating niordered operations. An example of the operation sequence vector is shown in Fig. 11. The machine assignment vector $v_2(r)$ shows that the machine selected for the operation is indicated at position $r$, and shown in Fig. 11. For example, the position 1 in $v_1(1)$ indicates $o_{61}$ (i.e., the first operation of Job 6), and Position 1 in $v_2(1)$ denotes that machine 1 is assigned to $o_{11}$. The main advantage of a two-vector representation is that each possible chromosome always depicts a feasible candidate.

*Population Initialization (Encoding Routine)*: To guarantee the quality and diversity of an initial population, a mixed strategy is used to generate chromosomes that include an operation sequence vector and a machine assignment vector. First, a random rule strategy is applied to randomly initialize the operations in a sequence vector. Second, minimal processing time and random rule strategies are applied to generate the machine assignment vector, as follows:

(1) The minimal processing time strategy [13, 15] is used to locate the machine that exhibits a minimal processing time for the permuted operation, and then adds its processing time to every subsequent entry.

(2) The random rule strategy randomly assigns a machine to each operation.

In this study, the random rule strategy was used to initialize the operation sequences, in which 50 % of the machine assignment vectors were generated using the minimal processing time strategy, and the remaining 50 % were generated using the random rule strategy.

*Left-shift Based Decoding*: This study used left-shift based decoding, where each operation was shifted left until it was as compact as possible to reduce the machine idle time. This strategy is used to search for the earliest available time interval $[t_i^E, t_i^L]$ to allocate the permuted operation to a machine based on the operation sequence vector. If the time span is sufficient from the beginning to ending, is allocated in the time interval; otherwise, is allocated at the end of machine. The following relationship formulated in Eq. 2):

$$\begin{cases} \max\left\{t_j^E + s_{hgikj},\ t_{i,k-1}^C\right\} + p_{ikj} \leqslant t_j^L, & \text{if } k \geqslant 2 \\ t_j^E + p_{ikj} \leqslant t_j^L, & \text{if } k = 1. \end{cases} \tag{2}$$

Left-shift based decoding sequentially allocates each operation to an assigned machine in the order represented in the operation sequence vector. The detailed example of the process by the left-shift based decoding is shown in Chow et al. [6].

*Genetic Operations*: In the two-vector representation, each gene of the operation sequence vector does not indicate a specific job's operation but refer to its context-dependent. This reason causes the crossover procedure cannot inherit their parental characteristics at the form of two-vector representation. We used the order crossover in this study for the operation sequence vector and the procedure is as follows.

**Step 1**. Randomly select a subsection of the operation from one parent.
**Step 2**. Conduct an offspring by copying the subsection of the parent, including the operation sequence and machine assignment in the corresponding position.
**Step 3**. Delete the operations that are already in the offspring from the second parent.
**Step 4**. Allocate the operations and assigned machines to the unfixed positions of the offspring from left to right, according to the sequence in the second parent.

The conventional mutation operator is used to randomly generate offspring [13]. In this study, the objective functions of makespan and total machine setup time are used to minimize the production time. An artificial mutation was developed, combining the minimal processing time concept and mutation operator, reallocating the machine that exhibits the minimal processingtime to the operation [15]. In the operation sequence vector, a gene of certain probability is selected and the operations are randomly exchanged with the machine that was assigned to the operation. In the machine assignment vector, the job that exhibits the longest total processingtime is selected and the machine that exhibits the maximal processing time is reallocated to the minimal processingtime to the corresponding operation. The offspring generated by artificial mutation may exhibit a superior makespan compared with the makespan before the convergence was accelerated. In addition, immigration strategy was used to randomly generate new chromosomes and prohibit rapid convergence.

*Local Search (Variable Neighborhood Descent)*: Local search can be used to improve the convergence speed, yielding superior solutions. The variable neighborhood descent (VND) approach is considered a local search algorithm that produces a new solution from the current population by making a slight change before it is inserted into the population [8]. The VND approach is employed to sequentially identify and exchange critical operations and find a new schedule that exhibits a small makespan in the multiobjective module assembly scheduling problem. The makespan of a scheduling solution is defined by the length of its critical path; that is, the makespan cannot be reduced while adjusting the current critical paths. Any operationon the critical path is called a critical operation.

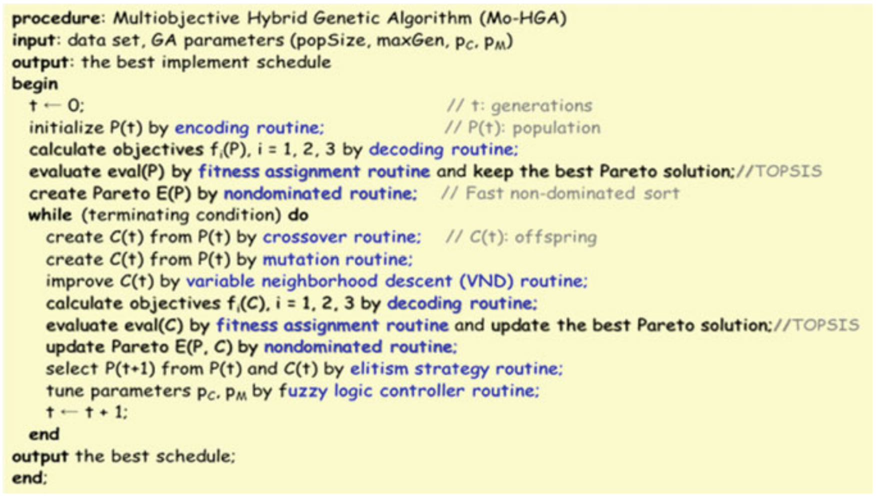
This study employed the VND approach to determine a schedule that yielded a small makespan. To reduce computational loading, only one critical operation is moved at a time and inserted into an available idle time interval. Therefore, the single moving operation of the VND procedure is as follows:

(1) Deleting a critical operation;
(2) Finding an assignable idle time interval;
(3) Allocating the deleted into the found time interval.

*TOPSIS*: The technique for order preference by similarity to the ideal solution (TOPSIS) is to derive the best compromised solution among Pareto optimal solutions. TOPSIS evaluation mechanism quickly conduct a best compromised schedule into the manufacturing system and TOPSIS is considering the best alternative should have the shortest distance from the ideal solution $(A^+)$ [12]. To prioritize the Pareto non-dominated solutions based on objective functions that decision maker concerned. TOPSIS evaluation mechanism ranks the best compromised scheduling from all alterative solutions and the detailed procedure is shown in Chou et al. [6]. Decision makers design the preference of objective functions:

Normalized weight vector; $w_1 + w_2 + w_3 = 1$;
Minimize makespan: weight $w_1$;
Minimize total workload: weight $w_2$;
Maximize CLIP weight: $w_3$

*Overall Procedure of Mo-HGA*: This study combined the auto-tuning strategy [20] to dynamically regulate the parameters for the multiobjective hybrid genetic algorithm (Mo-HGA) by employing a fuzzy logic controller (FLC). Two FLCs, the crossover and mutation FLCs, were implemented to adaptively regulate the rates of crossover and mutation operators during the genetic search process as introduced in [20] FLC for tuning parameters. This enabled the automatic tuning of the parameters of the Mo-HGA depending on the convergence situation of the current generation.



```
procedure: Multiobjective Hybrid Genetic Algorithm (Mo-HGA)
input: data set, GA parameters (popSize, maxGen, $p_C$, $p_M$)
output: the best implement schedule
begin
  $t \leftarrow 0$;                                        // t: generations
  initialize P(t) by encoding routine;                     // P(t): population
  calculate objectives $f_i(P)$, i = 1, 2, 3 by decoding routine;
  evaluate eval(P) by fitness assignment routine and keep the best Pareto solution;//TOPSIS
  create Pareto E(P) by nondominated routine;    // Fast non-dominated sort
  while (terminating condition) do
    create C(t) from P(t) by crossover routine;    // C(t): offspring
    create C(t) from P(t) by mutation routine;
    improve C(t) by variable neighborhood descent (VND) routine;
    calculate objectives $f_i(C)$, i = 1, 2, 3 by decoding routine;
    evaluate eval(C) by fitness assignment routine and update the best Pareto solution;//TOPSIS
    update Pareto E(P, C) by nondominated routine;
    select P(t+1) from P(t) and C(t) by elitism strategy routine;
    tune parameters $p_C$, $p_M$ by fuzzy logic controller routine;
    $t \leftarrow t + 1$;
  end
output the best schedule;
end;
```

**Fig. 12** Overall procedure of Mo-HGA for solving TFT-LCD module assembly scheduling problem

The overall procedure in the pseudo-code of Mo-HGA for solving designed for solving TFT-LCD module assembly scheduling problem as shown in Fig. 12.

A small-scale case was used to demonstrate the applicability of the proposed Mo-HGA. In this case, eight jobs involving 22 operations were arranged to eight machines on the shop floor, and the job quantity, product family, and operation sequences are listed in Table 5. The operation processing time of this small-scale case is shown in Table 6. The processing time of these 22 operations were designed on the basis of empirical setting that depends on the job quantity and production velocity of the machine. The sequence-dependent setup time between the product families is shown in Table 7.

Schedule $S = \{(o_{ij}, M_m, s_{ij} - c_{ij})\}$; $o_{ij}$ denotes operation $j$ in job $i$; $M_m$ is machine $m$; $s_{ij}$ means starting operation $j$ in job $i$ and $c_{ij}$ is completing operation $j$ in job $i$. Here is one of Pareto optimal solutions with three objective function values and Gantt chart (Fig. 13) by Mo-HGA as follows:

**Table 6** Processing time of the small-scale case

| Processing time (s) | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| $o_{11}$ – | 5,750 | – | – | – | – | – | – | – |
| $o_{12}$ – | – | – | – | – | 6,250 | – | – | – |
| $o_{21}$ – | – | 25,000 | 17,500 | – | – | – | – | – |
| $o_{22}$ – | – | – | – | – | – | 25,000 | 15,000 | – |
| $o_{31}$ | – | 20,000 | 14,000 | – | – | – | – | – |
| $o_{32}$ | – | – | – | 6,000 | – | – | – | – |
| $o_{33}$ | – | – | – | – | – | 20,000 | 12,000 | – |
| $o_{34}$ | – | – | – | – | – | – | – | 24,000 |
| $o_{41}$ | – | – | – | 3,000 | – | – | – | – |
| $o_{42}$ | 4,600 | – | – | – | – | – | – | – |
| $o_{43}$ | – | – | – | – | – | – | 5,000 | – |
| $o_{51}$ | – | – | – | 7,500 | – | – | – | – |
| $o_{52}$ | – | 25,000 | 17,500 | – | – | – | – | – |
| $o_{53}$ | – | – | – | – | – | 25,000 | 15,000 | – |
| $o_{61}$ | 13,800 | – | – | – | – | – | – | – |
| $o_{62}$ | – | – | – | – | 15,000 | – | – | – |
| $o_{71}$ | – | 20,000 | 14,000 | – | – | – | – | – |
| $o_{72}$ | – | – | – | – | – | 20,000 | 12,000 | – |
| $o_{81}$ | – | 30,000 | 21,000 | – | – | – | – | – |
| $o_{82}$ – | – | – | – | 9,000 | – | – | – | – |
| $o_{83}$ – | – | – | – | – | – | 30,000 | 18,000 | – |
| $o_{84}$ | – | – | – | – | – | – | – | 36,000 |

**Table 7** Sequence-dependent setup time between the product families

| Setup Time (s) | $o_{11}$ | $o_{12}$ | $o_{21}$ | $o_{22}$ | $o_{31}$ | $o_{32}$ | $o_{33}$ | $o_{34}$ | $o_{41}$ | $o_{42}$ | $o_{43}$ | $o_{51}$ | $o_{52}$ | $o_{53}$ | $o_{61}$ | $o_{62}$ | $o_{71}$ | $o_{72}$ | $o_{81}$ | $o_{82}$ | $o_{83}$ | $o_{84}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $o_{11}$ | – | – | – | – | – | – | – | – | – | 7,200 | – | – | – | – | – | – | – | – | – | – | – | – |
| $o_{12}$ | – | – | – | – | – | – | – | – | – | – | 1,800 | – | – | – | – | – | – | – | – | – | – | – |
| $o_{21}$ | – | – | – | – | 7,200 | – | – | – | – | – | – | – | 7,200 | – | – | – | – | – | 7,200 | – | – | – |
| $o_{22}$ | – | – | – | – | – | – | 3,600 | – | – | – | – | – | – | 3,600 | – | – | – | – | – | – | 3,600 | – |
| $o_{31}$ | – | – | 7,200 | – | – | – | – | – | – | – | – | – | 7,200 | – | – | – | 7,200 | – | – | – | – | – |
| $o_{32}$ | – | – | – | – | – | – | – | – | 7,200 | – | – | 7,200 | – | – | – | – | – | – | – | – | – | – |
| $o_{33}$ | – | – | – | 3,600 | – | – | – | – | – | – | – | – | – | 3,600 | – | – | – | 3,600 | – | – | – | – |
| $o_{34}$ | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| $o_{41}$ | – | – | – | – | – | 7,200 | – | – | – | – | – | 7,200 | – | – | – | – | – | – | – | 7,200 | – | – |
| $o_{42}$ | 7,200 | – | – | – | – | – | – | – | – | – | – | – | – | – | 7,200 | – | – | – | – | – | – | – |
| $o_{43}$ | – | 1,800 | – | – | – | – | – | – | – | – | – | – | – | – | – | 1,800 | – | – | – | – | – | – |
| $o_{51}$ | – | – | – | – | – | 7,200 | – | – | 7,200 | – | – | – | – | – | – | – | – | – | – | 7,200 | – | – |
| $o_{52}$ | – | – | 7,200 | – | 7,200 | – | – | – | – | 7,200 | – | – | – | – | – | – | 7,200 | – | 7,200 | – | – | – |
| $o_{53}$ | – | – | – | 3,600 | – | – | 3,600 | – | – | – | 3,600 | – | – | – | – | – | – | 3,600 | – | – | 3,600 | – |
| $o_{61}$ | 7,200 | – | – | – | – | – | – | – | – | 7,200 | – | – | – | – | – | – | – | – | – | – | – | – |
| $o_{62}$ | – | 1,800 | – | – | – | – | – | – | – | – | 1,800 | – | – | – | – | – | – | – | – | – | – | – |
| $o_{71}$ | – | – | – | – | 7,200 | – | – | – | – | – | – | – | 7,200 | – | – | – | – | – | 7,200 | – | – | – |
| $o_{72}$ | – | – | – | – | – | – | 3,600 | – | – | – | – | – | – | 3,600 | – | – | – | – | – | – | 3,600 | – |
| $o_{81}$ | – | – | 7,200 | – | – | – | – | – | – | – | – | – | 7,200 | – | – | – | 7,200 | – | – | – | – | – |
| $o_{82}$ | – | – | – | – | – | – | – | – | 7,200 | – | – | 7,200 | – | – | – | – | – | – | – | – | – | – |
| $o_{83}$ | – | – | – | 3,600 | – | – | – | – | – | – | – | – | – | 3,600 | – | – | – | 3,600 | – | – | – | – |
| $o_{84}$ | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

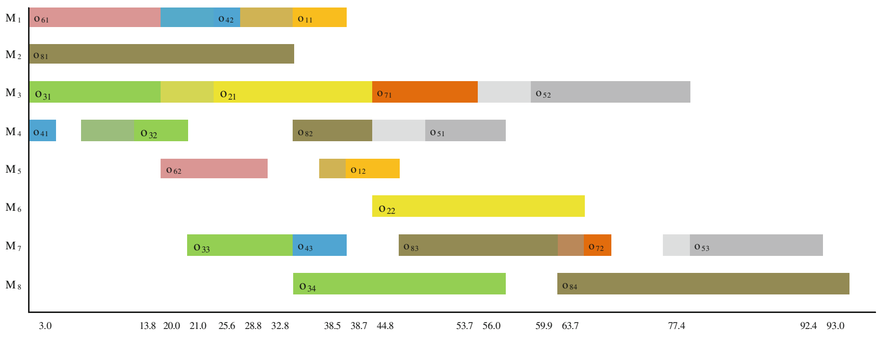


**Fig. 13** Gantt chart of Pareto optimal solution

Schedule: $S = o_{61}, o_{81}, o_{31}, o_{41}, o_{32}, o_{62}, o_{33}, o_{42}, o_{21}, o_{11}, o_{82}, o_{43}, o_{34}, o_{12}, o_{71}, o_{22}, o_{83}, o_{51}, o_{52}, o_{84}, o_{72}, o_{53} = (o_{61}, M_1, 0\text{–}13.8), (o_{81}, M_2, 0\text{–}30.0), (o_{31}, M_3, 0\text{–}14.0), (o_{41}, M_4, 0\text{–}3.0), (o_{32}, M_4, 14.0\text{–}20.0), (o_{62}, M_5, 13.8\text{–}28.8), (o_{33}, M_7, 20.0\text{–}32.0), (o_{42}, M_7, 21.0\text{–}25.6), (o_{21}, M_3, 21.2\text{–}38.7), (o_{11}, M_1, 32.8\text{–}38.55), (o_{82}, M_4, 30.0\text{–}39.0), (o_{43}, M_7, 32.0\text{–}37.0), (o_{34}, M_8, 23.0\text{–}56.0), (o_{12}, M_5, 38.55\text{–}44.8), (o_{71}, M_3, 38.7\text{–}52.7), (o_{22}, M_6, 38.7\text{–}63.7), (o_{83}, M_7, 39.0\text{–}57.0), (o_{51}, M_4, 46.2\text{–}53.7), (o_{52}, M_3, 59.9\text{–}77.4), (o_{72}, M_7, 60.6\text{–}72.6), (o_{53}, M_7, 77.4\text{–}92.4), (o_{84}, M_8, 57.0\text{–}93.0)$

Three objectives: Makespan: 93 (K s), Total workload: 310.9 (K s), CLIP: 100 (%).

Tables 8 and 9 is shown objective function values by Mo-GA, Mo-HGA.VND and Mo-HGA.VND.FLC for 8-Job/8-Machine and 40-Job/22-Machine respectively. Comparing the Mo-GA, Mo-HGA with VND, Mo-HGA with VND & FLC experiment results using TOPSIS, the Mo-HGA with VND & FLC could get the best compromised schedule.

The each objective function value by Mo-HGA.VND.FLC for 40-Job/22-Machine in Table 10 is shown three cases of the different computational time such as 60, 180 (s) and more than 22 generations.

To solve the multiobjective scheduling problem for the TFT-LCD module assembly system by using LINGO for comparing with the Mo-HGA proposed, Chou et al. [6] also formulated a fuzzy multiobjective mixed-integer linear programming (FMo-MILP) model to obtain the compromised solution as a benchmark by using fuzzy multiobjective programming, a fuzzy goal, and fuzzy constraints [21] A quality criterion, the optimality gap, was defined to show the percentage of deviation of the values of the multiobjective programming approaches (i.e., the FMo-MILP and the Mo-HGA) from the values of LINGO, according to the following equation [17]:

$$\text{Optimality Gap} = (\text{Approach - LINGO})/\text{LINGO} \times 100\,(\%). \qquad (3)$$

The optimality gaps of the small-scale test problems, compared to the compromised solution of multiobjective programming approaches with an aspiration level, are shown in Tables 11 and 12. The results show that the FMO-MILP and the MO-

**Table 8** Objective function values by Mo-GA, Mo-HGA.VND and Mo-HGA.VND.FLC for 8-Job/8Mc

| 8 Jobs with 8 Machines | MO-GA | | | MO-HGA with VND | | | MO-HGA with VND & FLC | | |
|---|---|---|---|---|---|---|---|---|---|
| Terminating Time (s) | CM (K s) | WT (K s) | CLIP (%) | CM (K s) | WT (K s) | CLIP (%) | CM (K s) | WT (K s) | CLIP (%) |
| 60 | 94.2 | 308.9 | 85.5 | 93 | 310.9 | 100 | 93 | 310.9 | 100 |
| 180 | 94.2 | 308.9 | 85.5 | 93 | 310.9 | 100 | 93 | 310.9 | 100 |

Terminating with same objective values over 50 generations

| 8 Jobs with 8 Machines | MO-HGA | | | | MO-HGA with VND with VND & FLC | | | | MO-HGA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| popSize: 200 | CM (K s) | WT (K s) | CLIP (%) | CPU (s) | CM (K s) | WT (K s) | CLIP (%) | CPU (s) | CM (K s) | WT (K s) | CLIP (%) | CPU (s) |
| | 94.2 | 308.9 | 85.5 | 22 | 93 | 310.9 | 100 | 112 | 93 | 310.9 | 100 | 118 |

**Table 9** Objective function values by Mo-GA, Mo-HGA.VND and Mo-HGA.VND.FLC for 40-Job/22Mc

| 40 Jobs with 22 Machines | MO-GA | | | MO-HGA with VND | | | MO-HGA with VND & FLC | | |
|---|---|---|---|---|---|---|---|---|---|
| Terminating Time (s) | CM (K s) | WT (K s) | CLIP (%) | CM (K s) | WT (K s) | CLIP (%) | CM (K s) | WT (K s) | CLIP (%) |
| 60 | 5057.5 | 43386.6 | 64.55 | 2121 | 7606.1 | 69.13 | 278.8 | 1429.5 | 91.82 |
| 180 | 7007.5 | 51296.9 | 67.43 | 256.5 | 1385.5 | 97.71 | 259 | 1377 | 94.44 |

Terminating with same objective values over 20 generations

| 40 Jobs with 22 Machines | MO-GA | | | | MO-HGA with VND | | | | MO-HGA with VND & FLC | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CM (K s) | WT (K s) | CLIP (%) | CPU (s) | CM (K s) | WT (K s) | CLIP (%) | CPU (s) | CM (K s) | WT (K s) | CLIP (%) | CPU (s) |
| | 5242.5 | 44870.4 | 25.7 | 35.6 | 259.5 | 1368.5 | 97.71 | 416 | 258 | 1372.5 | 97.71 | 311 |

**Table 10** Objective function values by Mo-HGA.VND.FLC for 40-Job/22Mc

| 40 Jobs with 22 Machines | MO-HGA with VND & FLC | | |
|---|---|---|---|
| Terminating Condition | $C_m$ (k s) | $W_t$ (k s) | *CLIP* (%) |
| Computational time>60 s | 278.8 | 1429.5 | 91.8 |
| Computational time>180 s | 259.0 | 1377.0 | 94.4 |
| Same result over 20 generations | 258.0 | 1372.5 | 97.7 |

**Table 11** Experimental result of FMo-MILP and Mo-HGA

| Test Problem | MO-MILP (LINGO) | | | FMO-MILP (LINGO) | MO-HGA | Aspiration Level |
|---|---|---|---|---|---|---|
| | $C_{\max}$ | $W_{NT}$ | $S_T$ | $(C_{\max}, W_{NT}, S_T)$ | $(C_{\max}, W_{NT}, S_T)$ | |
| P1 | 50.0 | 0.37 | 0 | (69.7, 0.37, 7.2) | (69.7, 0.37, 7.2) | (50.0, 0.37, 0) |
| P2 | 60.5 | 0.34 | 10.8 | (72.5, 0.34, 10.8) | (72.5, 0.34, 10.8) | (60.5, 0.34, 10.8) |
| P3 | 76.2 | 0.16 | 10.8 | (78.6, 0.18, 14.4) | (78.6, 0.18, 14.4) | (76.2, 0.16, 10.8) |

**Table 12** Optimality gaps of small-scale test problems

| Test problem | FMO-MILP | | | MO-HGA | | |
|---|---|---|---|---|---|---|
| | $C_{\max}$ (%) | $W_{NT}$ (%) | $S_T$ (%) | $C_{\max}$ (%) | $W_{NT}$ (%) | $S_T$ (%) |
| P1 | 39.4 | 0 | | 39.4 | 0 | |
| P2 | 19.8 | 0 | 0 | 19.8 | 0 | 0 |
| P3 | 3.1 | 12.5 | 25.0 | 3.1 | 11.1 | 25.0 |

HGA yield the same optimality gaps for each test problem.These two approaches have the values close to the aspiration level. Detailed computational results refer Chou et al. [6].

As introduced computational results for the various scale of multiobjective scheduling problems for the TFT-LCD module assembly system by using LINGO for comparing with the Mo-HGA, it clearly demonstrated that multiobjective hybrid genetic algorithm with VND and FLC routines (Mo-HGA.VND.FLC) proposed is effectiveness and efficiency in the best compromised solution as a quality of solution with reasonable interactive computational time for NP-hard multiobjective optimization problem in practice.

## 6 Conclusions

Manufacturing scheduling is one of the important and complex combinatorial optimization problems, where it can have a major impact on the productivity of a production process. Moreover, most of scheduling problems fall into the class of NP-hard combinatorial problems. Recently, many manufacturing companies are faced with global market demands for a variety of low cost products with a high quality. For responding rapidly to demand fluctuations and reducing costs related to manufacturing scheduling and logistics networks, hybrid genetic algorithm (HGA) and multiobjective HGA (Mo-HGA)have received considerable attention regarding their potential for solving various complex manufacturing and logistics problems.

In this paper, we introduced how to design Hybrid GA and Mo-HGA with parameter tuning by the fuzzy logic controller (FLC) and local search such as the left-ship routine and variable neighborhood descent (VND) routines to solve manufacturing scheduling problems for hard disc device (HDD) and thin-film transistor-liquid crystal display (TFT-LCD), respectively. In particularly, the FLC for tuning crossover and mutation rates, the fitness assignment mechanism for multiobjective optimization problems (MOP) and genetic representations as well as the hybrid evolutionary operations are combined. Through a variety of numerical experiments, the effectiveness and efficiency of the HGA for HDD and Mo-HGA.VND.FLC for TFT-LCD module assembly as the practical applications of manufacturing scheduling problems are demonstrated. This paper also introduced how to design Mo-HGAs for applying a multiobjective flexible job-shop scheduling problem (Mo-FJSP; operation sequencing and resources assignment) to the practical manufacturing scheduling problems.

As future researches in multiobjective scheduling problems, it is to apply hybrid sampling strategy-based evolutionary algorithms [9] to real case study. Another topics is to enhance the evolutionary process by combining hybrid genetic algorithm with another metaheuristics such as PSO, DE or EDA.

**Acknowledgments** First author is appreciated for cooperating the project in Taiwan and Thailand with Prof. C-F Chien & Dr. C-W Chou and Prof. K. Sethanan & Dr. C. Chamnanlor in 2012-2014, respectively. This work is partly supported by JSPS: Grant-in-Aid for Scientific Research (C; No.24510219), Taiwan NSF (NSC 101-2811-E-007-004, NSC 102-2811-E-007-005), National NSF of China (No. U1304609), the Fundamental Research Funds for the Henan Provincial Colleges and Universities (No. 2014YWQQ12), the Plan of Nature Science Fundamental Research in Henan Univ. of Tech. (No. 2012JCYJ04), and the Fundamental Research Funds (Software+X) of Dalian Univ. of Tech. (No. DUT12JR05, No. DUT12JR12).

## References

1. Abe K, Ida K (2008) Genetic local search method for re-entrant flow shop problem. In: Dagli CH et al (eds) Artificial neural networks in engineering, vol 17, pp 381–387
2. Bidot J, Vidal T et al (2009) A theoretic and practical framework for scheduling in a stochastic environment. J Sched 12:315–344
3. Chamnanlor C, Sethanan K et al (2013) Hybrid genetic algorithms for solving reentrant flow-shop scheduling with time windows. Ind Eng Manag Syst 12:306–316
4. Chamnanlor C, Sethanan K et al (2014) Re-entrant flow shop scheduling problem with time windows using hybrid genetic algorithm based on autotuning strategy. Int J Prod Res 52:2612–2629
5. Chien CF (2007) Made in Taiwan: shifting paradigms in high-tech industries. Ind Eng 39:47–49
6. Chou CW, Chien CF, Gen M (2014) A multiobjective hybrid genetic algorithm for TFT-LCD module assembly scheduling. IEEE Trans Autom Sci Eng 10:692–705
7. Danping L, Lee CKM, Wu Z (2012) Integrating analytical hierarchy process to genetic algorithm for re-entrant flow shop scheduling problem. Int J Prod Res 50:1813–1824
8. Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 35:2892–2907
9. Gen M, Tsujimura Y, Kubota E (1994) Solving job-shop scheduling problems by genetic algorithm. Int Conf IEEE Syst Man Cybern 2:1577–1582
10. Gen M, Cheng R, Lin L (2008) Network models and optimization: multiobjective genetic algorithm approach. Springer Science and Business Media
11. Hao XC, Wu JZ et al (2014) The cooperative estimation of distribution algorithm: a novel approach for semiconductor final test scheduling problems. J Intell Manuf 25:867–879
12. Hwang CL, Yoon K (1981) Multiple attribute decision making: methods and applications. Springer, Berlin
13. Kacem I, Hammadi S, Borne P (2002) Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Trans Syst Man Cybern Part C 32:408–419
14. Lin JT, Wang FK, Kuo PC (2005) A parameterized-dispatching rule for a Logic IC sort in a wafer fabrication. Prod Plan Control 16:426–436
15. Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 35:3202–3212
16. Sangsawang C, Sethanan K et al (2015) Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking. Expert Syst Appl 42:2395–2410
17. Tay JC, Ho NB (2008) Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput Ind Eng 54:453–473
18. Wu JZ, Chien CF, Gen M (2012) Coordinating strategic outsourcing decisions for semiconductor assembly using a bi-objective genetic algorithm. Int J Prod Res 50:235–260
19. Wu JZ, Hao XC, Chien CF (2012) A novel bi-vector encoding genetic algorithm for the simultaneous multiple resources scheduling problem. J Intell Manuf 23:2255–2270
20. Yun YS, Gen M (2003) Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics. Fuzzy Optim Decis Mak 2:161–175
21. Zimmerman HJ (1979) Fuzzy programming and linear programming with several objective functions. Fuzzy Sets Syst 1:45–55