Multiobjective Hybrid Genetic Algorithms for Manufacturing Scheduling: Part I Models and Algorithms

Mitsuo Gen, Lin Lin and Wenqiang Zhang

Abstract In real world manufacturing systems there are many combinatorial optimization problems (COP) imposing on more complex issues, such as complex structure, nonlinear constraints, and multiple objectives to be handled simultaneously. Manufacturing scheduling is one of the important and complex COP models, where it can have a major impact on the productivity of a production process. Moreover, the COP models make the problem intractable to the traditional optimization techniques because most of scheduling problems fall into the class of NP-hard combinatorial problems. In order to develop effective and efficient solution algorithms that are in a sense good, i.e., whose computational time is small as within 3 min, or at least reasonable for NP-hard combinatorial problems met in practice, we have to consider: quality of solution, computational time and effectiveness of the nondominated solutions for multiobjective optimization problem (MOP). When solving any NP-hard problem, a genetic algorithm (GA) based on principles from evolution theory is most powerful metaheuristics. In this paper, we concern with the design of hybrid genetic algorithms (HGA) and multiobjective HGA (Mo-HGA) to solve manufacturing scheduling problems. Firstly we introduce typical models in manufacturing scheduling systems such as parallel machines scheduling (PMS), flexible job-shop scheduling problem (FJSP) and assembly line balancing (ALB) problem. Secondly to solve NP-hard COP models, we introduce design scheme of HGA combined with fuzzy logic controller (FLC) and multiobjective HGA (Mo-HGA) with several fitness assignment mechanisms. For demonstrating computational experiments by HGA and Mo-HGA, the effectiveness of the HGA for the HDD (hard disc device) and Mo-HGA

M. Gen (✉)
Fuzzy Logic Systems Institute, Tokyo University of Science,
Tokyo 162-0825, Japan
e-mail: gen@flsi.or.jp

L. Lin
School of Software Technology, Dalian University of Technology,
Dalian 116024, People's Republic of China

W. Zhang
College of Information Science & Engineering, Henan University
of Technology, Zhengzhou 450001, 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_1

for TFT-LCD (thin-film transistor-liquid crystal display) module assembly problems as a practical manufacturing model, respectively is demonstrated in the concatenated paper Part II.

Keywords Hybrid genetic algorithms (HGA) · Multiobjective HGA (Mo-HGA) · Manufacturing scheduling · Parallel machines scheduling (PMS) · Hard disc device (HDD)

1 Introduction

In real world manufacturing systems there are many combinatorial optimization problems (COP) imposing on more complex issues, such as complex structure, non-linear constraints, and multiple objectives to be handled simultaneously. Manufacturing scheduling is one of the important and complex COP models, where it can have a major impact on the productivity of a production process. Moreover, the COP models make the problem intractable to the traditional optimization techniques because most of scheduling problems fall into the class of NP-hard combinatorial problems. In order to develop effective and efficient solution algorithms that are in a sense good, i.e., whose computational time is small as within 3 min, or at least reasonable for NP-hard combinatorial problems met in practice, we have to consider three important issues:

(1) Quality of solution, (2) Computational time and (3) Effectiveness of the non-dominated solutions for multiobjective optimization problem (MOP).

When solving any NP-hard problem, a genetic algorithm (GA) based on principles from evolution theory is one of very powerful metaheuristics [1].

The semiconductor industry has grown rapidly and subsequently production planning problems have raised many important research issues. Because of short product lifecycles, 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 & microprocessors, thin-film transistor-liquid crystal display (TFT-LCD) and hard disc device (HDD). 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 [2].

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) and assembly line balancing (ALB) model, respectively. On a flexible job-shop floor, workstations employ non-identical parallel machines scheduling (PMS) model and reentrant flow-shop 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. [3] reported detailed 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.

Furthermore, many researches are focusing on the manufacturing scheduling problems such as parallel machine system (PMS), job-shop scheduling problem (JSP), flexible job-shop scheduling problem (FJSP) and by genetic algorithms. The PMS model has been widely used in many manufacturing environments and [4] proposed a genetic algorithms for solving minmax earliness/tardiness scheduling in identical parallel machine system problem. Cheng and Gen [5] expanded the memetic algorithm to solve the minmax PMS problem. The memetic algorithm is a kind of hybrid version of the genetic algorithms and the local optimizer in order to enhance the performance of the genetic algorithms.

In the job-shop scheduling problem (JSP) consisting of a set of jobs and a set of machines, the JSP model is to establish a permutation of operations on each machine subject to precedence constraints to minimize production time completed. This problem is one of the best known models of the difficult combinatorial optimization problems. Cheng et al. [5, 6] reported a tutorial survey of JSP using genetic algorithms: representation and hybrid genetic search strategies, respectively. Kachitvichyanukul et al. [7] proposed a two-stage genetic algorithm (2S-GA) for multiobjective JSP (MoJSP). The 2S-GA is proposed with three criteria: minimizing makespan, minimizing total weighted earliness, and minimizing total weighted tardiness. Gao et al. [8] reported a hybrid genetic algorithm for solving flexible job-shop scheduling problem (FJSP) model with maintenances and Gao et al. [9] proposed a hybrid of genetic algorithm with bottleneck shifting method for solving multiobjective FJSP problems. Gao et al. [10] also proposed a hybrid genetic algorithm (HGA) combined with variable neighborhood descent (VND) method for solving multiobjective FJSP model (MoFJSP). Gen et al. [11] proposed a multistage-based GA (MsGA) with bottleneck shifting developed for treating the multiobjective FJSP model.

Since the 1960s, there has been being an increasing interest in imitating living beings to solve the hard optimization problems. An evolutionary algorithm (EA) such as a genetic algorithm (GA) is a generic population-based meta-heuristic optimization algorithm [12–14]. An EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. Handa et al. [15] gave a comprehensive overview of recent advances of evolutionary computation (EC) studies. GAs have attracted significantly attention with respect to complexity scheduling, which is referred to evolutionary scheduling, it is vital research domain at interface of two important sciences C artificial intelligence and operational research [16].

Even if EAs have attracted significantly attention with respect to above complexity scheduling problems, it has a disadvantage: we have to design a specialized EA for each practical scheduling problem with the problem's specificity. So that means each class of EAs doesn't have a wide range of applications on manufacturing scheduling. Recently Gen and Lin [1] surveyed multiobjective evolutionary algorithm for manufacturing scheduling problems. In order to design an effective GA with the problem's specificity, we have to consider (1) how to design a representation and a way of population initialization; (2) how to evaluate an individual by a fitness function; (3) how to improve population by evolutionary operators. In this paper, we focus on the effective multiobjective hybrid genetic algorithm for applying to manufacturing scheduling, in particular HDD and TFT-LCD module assembly problems.

The rest of this paper is organized as follows: Sect. 2 introduces typical manufacturing scheduling models such as parallel machine scheduling (PMS) model, flexible job-shop scheduling problem (FJSP) and assembly line balancing (ALB) problem. A design scheme for hybrid genetic algorithms (HGA) and fuzzy logic controller (FLC) for tuning GA parameters introduce in Sect. 3. The basic concept for a multiobjective optimization problem (MOP) summarizes and multiobjective HGA (Mo-HGA) with several fitness assignment mechanisms introduces in Sect. 4. Finally, the conclusion of this paper are drawn in Sect. 5.

2 Manufacturing Scheduling Models

2.1 Parallel Machine Scheduling Model

Scheduling is to simply assign a set of jobs to a set of machines with respect to operational constraints such that optimal usage of available resources is obtained. Because of the growing cost of raw material, labor, energy and transportation, scheduling plays an essential role in production planning of manufacturing systems, and it is one of the most important issues for survival in the modern competitive market place. Parallel machine scheduling (PMS) has been widely used in many manufacturing environments [4]. It is an extension of the fundamental single-machine scheduling, and can also be regarded as a production scheduling stage in a flexible flow shop or flexible job-shop. In the PMS problem an objective is to minimize the maximum weighted absolute lateness. It is known as one of non-regular performance measures. In recent years, scheduling research involving non-regular performance measures has received much attention from practitioners as well as researchers to respond to the increasing competitive pressure in domestic and international market. There are two non-regular performance measures commonly used in machine scheduling: minsum and minmax. A minsum problem attempts to minimize the sum of weighted absolute deviation of job completion time about the due date to reduce customers' aggregate disappointment; while a minmax problem attempts to minimize the maximum weighted absolute deviation of job completion time about the due date to reduce a customer's maximum disappointment. Li and Cheng [17] have shown that the minmax scheduling problem is NP-complete even for a single machine system.

In general, we know that there are two essential issues to be dealt with in PMS problems: (1) Job partition among machines (combination nature) and (2) Job sequence within each machine (permutation nature).

That is, the problem involves both permutation and combination components. Most PMS problems have been shown to be NP-complete [13]. That is, the PMS problem involves a combination of the permutation component and the combination component and is much difficult to solve than the single machine scheduling problem. Parallel machine scheduling problem has been one of the topics of interest for many researchers in the past few decades. Most of these studies considered the machines as only resource which is restricted. However, in real life manufacturing systems other resources such as machine operators and tools are constrained and it is illogical to consider that there are always enough resources for processing a job.

Cheng and Gen [12] applied the memetic algorithm to solve the minmax parallel machine scheduling problem. The memetic algorithm is a kind of hybrid version of the genetic algorithms and the local optimizer in order to enhance the performance of the genetic algorithms. The genetic algorithm is used to evolve the job partition among machines and the job sequence within each machine whereas a local optimizer is used to adjust the job order to form a V-shaped scheduler for each machine. Experiment results shown that the hybrid approach outperforms the genetic algorithms and conventional heuristic.

Balin [18] proposed genetic algorithm a new crossover operator and a new optimality criterion for solving the non-identical parallel machine scheduling (Nonident-MPS) problem consisting of the n independent jobs and the m non-identical parallel machines in order to minimize makespan with a definite processing time with a different processing time on each machine. The GA proposed here is suitable for the Nonident-MPS problem of minimizing the maximum completion time (makespan). GA can be applied to non-identical parallel machine scheduling problem involving setup times, ready times and/or due dates in order to minimize the maximum flow time, number of tardy jobs or total tardiness [18]. Balin [19] proposed genetic algorithm embedded with a simulation model for solving non-identical parallel machine scheduling problem with fuzzy processing times (Nonident-FuzzyPMS) by minimizing maximum completion time (makespan).

Li et al. [20] proposed metaheuristics and exact methods to solve a multiobjective parallel machines scheduling (MoPMS) problem, $Pm|s_{ij}, r_j|C_{\max}, T_j$ in which consists in scheduling n independent jobs on m identical parallel proposed machines and the job data such as processing times, release dates, due dates and sequence dependent setup times are considered where s_{ij} and r_j are the sequence dependent setup times if job j is the immediate successor of the job i on the same machine and the release of job j, respectively and $C_{\max}$, T_j are the makespan and the tardiness of job j. They formulated a mixed integer linear program (MILP) model for the MoPMS problem for solving it by CPLEX solver and proposed the nondominated sorting genetic algorithm (FLC-NSGA-II) coupled with a fuzzy logic controller (FLC) to solve the problem. The role of the fuzzy logic is to better set the crossover and the mutation probabilities in order to update the search ability as demonstrated in hybrid genetic algorithm (HGA) by Yun and Gen [16] as introducing in Subsection 3.2 Fuzzy Logic Controller for Tuning Parameters and also auto-tuning strategy for balancing between exploration and exploitation by Lin and Gen [21]. The experimental results for the MoPMS problem shown the advantages and the efficiency of the FLC-NSGA-II [20].

The flexible flow-shop scheduling (FSS) problem may be seen as a generalization of two particular types of scheduling problems: the PMS problem and the flow-shop scheduling problem (FSP). The key decision of the PMS problem is the allocation of jobs to machines whereas the key decision of FSP is the sequence of jobs through the manufacturing shop floor. The FSS problem has been widely studied in the literature. The machines considered in each stage of the flexible flow-shop problem are different and can be identical machines [22]. The main consequence of the reentrant flow nature is that wafers at different stages in their semiconductor manufacturing cycle have to compete with each other for the same machines. This is a reentrant flow-shop scheduling (RFS) model, in the RFS problem all jobs have the same routing over the machines of the shop and the same sequence is traversed several times to complete the jobs. Chen et al. [23] proposed a hybrid genetic algorithm for solving the reentrant flow-shop scheduling problem.

Recently hybrid genetic algorithms (HGAs) are proposed to solve the complex re-entrant scheduling problem with time windows constraint in manufacturing HDD devices with lot size. This problem can be formulated as a deterministic $Fm|fmls, rcrc, temp|C_{MAX}$ problem for finding the scheduling operations of machines in a flow-shop environment processing *fmls* job families with the objective of minimising the makespan, C_{MAX} [24, 25]. Sangsawang et al. [26] proposed metaheuristics optimization approaches for solving the two-stage reentrant FFS (RFFS) problem with blocking constraint (FFS|2-stage,rcrc,block|Cmax) in which they applied a hybrid GA and a hybrid particle swarm optimization (HPSO) with Cauchy distribution.

2.2 Flexible Job-Shop Scheduling Model

Machine scheduling problems arise in diverse areas such as flexible manufacturing system, production planning, computer design, logistics, communication, etc. A common feature of many of these problems is that no efficient solution algorithm is known yet for solving it to optimality in polynomial time. The classical job-shop scheduling problem (JSP) is one of the most well-known machine scheduling problems. JSP is among the hardest combinatorial optimization problems [27]. Because of its inherent intractability, heuristic procedures are an attractive alternative. Most conventional heuristic procedures use a priority rule, i.e., a rule for choosing an operation from a specified subset of as yet unscheduled operations. In recent years, an interest in using probabilistic search methods to solve JSP has been growing rapidly. These approaches comprise the emergence of promise for conquering the combinatorial explosion in a variety of decision-making arenas. Van Laarhoven et al. [28] proposed a simulated annealing for solving job-shop scheduling problem and Dell'Amico and Trubian [29] proposed a tabu search for solving the job-shop scheduling problem. Gen et al. [30] proposed a genetic algorithm for solving the job-shop scheduling problem. Cheng et al. [6, 31] reported a tutorial survey of JSP using genetic algorithms: representation and hybrid genetic search strategies, respectively.

Flexible job-shop scheduling problem (FJSP) is an extension of the traditional job-shop scheduling problem, which provides a closer approximation to real scheduling problems. In the job-shop scheduling problem (JSP), there are n jobs that must be processed on a group of m machines. Each job i consists of a sequence of m operations $(o_{i1}, o_{i2}, \ldots, o_{im})$, where o_{ik} (the kth operation of job i) must be processed without interruption on a predefined machine m_{ik} for p_{ik} time units. The operations $o_{i1}, o_{i2}, \ldots, o_{im}$ must be processed one after another in the given order and each machine can process at most one operation at a time. In a flexible job-shop, each job i consists of a sequence of n_i operations $(o_{i1}, o_{i2}, \ldots, o_{ini})$. The FJSP extends JSP by allowing an operation o_{ik} to be executed by one machine out of a set A_{ik} of given machines. The processing time of operation o_{ik} on machine j is $p_{ikj} > 0$. The FJSP problem is to choose for each operation o_{ik} a machine $M(o_{ik}) \in A_{ik}$ and a starting time s_{ik} at which the operation must be performed.

Mastrolilli and Gambardella [32] proposed two neighborhood functions for the FJSP problem. They proposed a tabu search procedure and provided an extensive computational study on 178 FJSP problems and 43 JSP problems. The flexible job-shop scheduling problem is to assign each operation to an available machine and sequence the operations assigned on each machine in order to minimize the makespan, that is, the time required to complete all jobs. The multiobjective FJSP model (MoFJSP) will be formulated as a multiobjective 0–1 mixed integer programming (M0-1MIP) model as follows [11]:

$$\min t_M = \max_{i,k} \{c_{ik}\}, \tag{1}$$

$$\min W_M = \max_{j} \left\{W_j\right\}, \tag{2}$$

$$\min W_T = \sum_{j=1}^{m} W_j, \tag{3}$$

$$s.t. c_{ik} - t_{ikj}x_{ikj} - c_{i(k-1)} \geq 0, \quad k = 2, \ldots, K_i; \forall i, j \tag{4}$$

$$\left[\left(c_{hg} - c_{ik} - t_{hgj}\right) x_{hgj}x_{ikj} \geq 0\right] \vee \left[\left(c_{ik} - c_{hg} - t_{ikj}\right) x_{hgj}x_{ikj} \geq 0\right], \forall (i,k), (h,g), j \tag{5}$$

$$\sum_{j=1}^{m} x_{ikj} = 1, \forall k, i \tag{6}$$

$$x_{ikj} \in \{0, 1\}, \forall j, k, i \tag{7}$$

$$c_{ik} \geq 0, \forall k, i \tag{8}$$

The objective functions accounts Eq. (1) is to minimize the makespan, Eq. (2) is to minimize the maximal machine workload (i.e., the maximum working time spent at any machine), Eq. (3) is to minimize the total workload (i.e., the total working time over all machines). Equation (4) states that the successive operation has to be started after the completion of its precedent operation of the same job, which represents the operation precedence constraints. Equation (5) is a disjunctive constraint, where one or the other constraint must be observed. Equation (6) states that one machine must be selected for each operation. Equations (7) and (8) are variable restrictions. Gao et al. [8] developed a new hybrid GA to solve the flexible job-shop scheduling problem with non-fixed availability constraints. Gao et al. [10] also proposed a hybrid genetic algorithm (HGA) combined with variable neighborhood descent (VND) method for solving multiobjective FJSP model (MoFJSP). Gen et al. [11] proposed a multistage-based GA (MsGA) with bottleneck shifting developed for treating the multiobjective FJSP model. The realistic FJSP model combined sequence-dependent setup time (SDST) constraints will be considered manufacturing scheduling systems for the TFT-LCD (thin-film transistor-liquid crystal display).

2.3 Assembly Line Balancing Model

An assembly line (AL) is a manufacturing process consisting of various tasks in which interchangeable parts are added to a product in a sequential manner at a station to produce a finished product. Assembly lines are the most commonly used method in a mass production environment, because they allow the assembly of complex products by workers with limited training, by dedicated machines and/or by robots. The assembly line balancing (ALB) model is not only include the operation sequence between stations, but also need to consider that how to group the operations among stations so that the precedence relations are not violated and a given objective function is optimized.

Since, ALB model falls into the NP-hard class of combinatorial optimization problems [33], in recent years, to provide an alternative to traditional optimization techniques, numerous research efforts have been directed towards the development of heuristics and meta-heuristics. While heuristic methods generating one or more feasible solutions were mostly developed until mid 90s; meta-heuristics such as tabu search [34], simulated annealing, genetic algorithms [29] and ant colony optimization [35] have been the focus of researchers in the last decade.

Among the meta-heuristics, the application of Genetic Algorithms (GAs) received a considerable attention from the researchers, since it provides an alternative to traditional optimization techniques by using directed random searches to locate optimum solutions in complex landscapes and it is also proven to be effective in various combinatorial optimization problems. GAs are powerful and broadly applicable stochastic search and optimization techniques based on principles from evolutionary theory [13]. Falkenauer and Delchambre [36] were the first to solve ALB problem with GAs. Following Falkenauer and Delchambre, application of GAs for solving ALB model was studied by many researchers, [37–39]. However, most of the researchers focused on the simplest version of the problem, with single objective and ignored the recent trends, i.e., mixed-model production, u-shaped lines, robotic lines and etc., in the complex assembly environments, where ALB models are multiobjective in nature [40, 41].

The basic version of the ALB model is the simple assembly line balancing (sALB) model. The simple assembly line is a single-model assembly line that is capable of producing only one type of product. A simple ALB model capable of producing only one type of product consists of stations ($i = 1, \ldots, m$) arranged along a conveyor belt or a similar mechanical material handling equipment. The workpieces (jobs) are consecutively launched down the line and are moved from station to station. At each station, certain tasks are repeatedly performed regarding the cycle time (maximum or average time available for each workcycle). The decision problem of optimally partitioning, i.e., balancing, the assembly work among the stations with respect to a given objective function is known as sALB problem. Manufacturing a product on an assembly line requires partitioning the total amount of work into a set of elementary operations named tasks $j = 1, \ldots, n$. Performing a task j takes certain task time t_j and requires certain equipment of machines and/or skills of workers. Due to technological and organizational conditions precedence constraints between the tasks have to be observed. These elements can be summarized and visualized by a precedence network. It contains a node for each task, node weights for the task processing times and arcs for the precedence constraints.

Any type of sALB model consists in finding a feasible line balance, i.e., an assignment of each task to a station such that the precedence constraints are fulfilled. The set of tasks S_i assigned to a station i ($i = 1, \ldots, m$) constitutes its station load, the cumulated task time:

$$t(S_i) = \sum_{j \in S_i} t_j, \tag{9}$$

is called station time. When a fixed common cycle time c_T is given, a line balance is feasible only if the station time of neither station exceeds c_T. In case of $t_{Si} < c_T$, the station i as an idle time of $(c_T - t_{Si})$ time units in each cycle.

After defining the indices, parameters and decision variable [11, 21, 42], we formulate the following multiobjective 0–1 integer programming model:

$$\max E = \frac{1}{mc_T} \sum_{j=1}^{n} t_j x_{ij} \tag{10}$$

$$\min \; m = \sum_{i=1}^{M} \max_{1 \le j \le n} \{x_{ij}\} \tag{11}$$

$$\min V = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(u_i - \overline{u})^2} \tag{12}$$

$$s.t. \sum_{i=1}^{M} x_{ij} = 1, \forall j \tag{13}$$

$$\sum_{i=1}^{M} i x_{ik} \le \sum_{i=1}^{M} i x_{ij}, \forall j; \forall k \in \Pr e(j) \tag{14}$$

$$t(S_i) = \sum_{j \in S_i} t_j = \sum_{j=1}^{n} t_j x_{ij} \le c_T, \forall i \tag{15}$$

$$x_{ij} = 0 \; or \; 1, \forall i, j. \tag{16}$$

In this mathematical model, the first objective Eq. (10) of the model is to maximize the line efficiency. The second objective Eq. (11) is to minimize the number of stations actually employed. The third objective Eq. (12) of the model is to minimize the variation of workload. The constraints given in Eqs. (13)–(16) are used to formulate the general feasibility of the problem. The constraint given in Eq. (13) states that each task must be assign to one and only one station. Equation (14) represents the precedence constraints and it states that the direct predecessor of task j must be assign to a station, which is in front of or the same as the station that task j is assigned in. This constraint stresses that if a task is assigned to a station, then the predecessor of this task must be already assigned to a station. Equation (15) denotes that the available time at each station should be less than or equal to the given cycle time. Constraint given in Eq. (16) represents the usual integrity restriction.

For designing a chromosome representation with encoding method, there are several methods to suit the characteristics of ALB model [41]: Task-based Encoding, Embryonic Encoding, Workstation-based Encoding, Grouping-based Encoding, and Heuristic-based (Indirect) Encoding. In the past decades, robots have been extensively used in assembly lines as called robotic assembly lines (rALs). An assembly robot can work 24 h a day without worries or fatigue. Goals for implementation of robotic assembly lines include high productivity, quality of product, manufacturing flexibility, safety, decreasing demand of skilled labor, and so on. Different robot types may exist at the assembly facility. Unlike manual assembly lines, where actual processing times for performing task vary considerably and optimal balance is rather of theoretical importance, the performance of rALs depends strictly on the quality of its balance. As extended from sALB, robotic assembly line balancing (rALB) is also NP-hard [43]. Lin et al. [44] proposed a hybrid genetic algorithm for robot-based ALB problem, Gao et al. [45] reported an efficient approach Based on GA for solving type II robotic assembly line balancing problems, Zhang et al. [46] and Zhang and Gen [47] proposed multiobjective GA for ALB Problems with worker allocation and considering demand ratio-based cycle time, respectively.

3 Hybrid Genetic Algorithms

3.1 Genetic Representations and Operations

Major Advantages of GA: GA has received considerable attention regarding their potential as a novel optimization technique. Three major advantages exist when applying GA to various combinatorial optimization problems [11]:

Adaptability: GA does not have much mathematical requirements about the optimization problems. Because of the evolutionary nature, GA will search for solutions without regard to the specific inner workings of the problem. GA can handle any kind of objective functions and any kind of constraints, i.e., linear or nonlinear, defined on discrete, continuous, or mixed search spaces.

Robustness: The use of evolution operators makes GA very effective in performing global search (in probability), whereas most of conventional heuristics usually perform local search. It has been proved by many studies that GA is more efficient and more robust in locating an optimal solution and reducing a computational effort than other conventional heuristics.

Flexibility: GA provides us with a great flexibility to hybridize with domain-dependent heuristics to make an efficient implementation for a specific problem.

Applicability: The practical software packages based on GA such as ERP and Simulation are standard usages in the real world.

The original form of genetic algorithms (GAs) was described by Goldberg [48] and Michalewicz [14] expanded GA to evolution programs by combined with data structure. Gen and Lozano [49] proposed fuzzy logic controllers for adapting parameters in genetic algorithm. GA is a stochastic search technique based on the mechanism of natural selection and natural genetics. The central theme of research on GA is to keep a balance between exploitation and exploration in its search to the optimal solution for survival in many different environments. Features for self-repair, self-guidance, and reproduction are the rules in biologic systems, whereas they barely exist in the most sophisticated artificial systems. GA has been theoretically and empirically proved to provide a robust search in complex search spaces. Many research papers and dissertations have established the validity of GA approach in function optimization problems and application problems [11–13, 50].

Genetic Representation: In general, two ways exist to generate the initial population, i.e., a set of solution candidates, heuristic initialization and random initialization, by using an encoding procedure satisfying system constraints and/or a boundary condition. How to present a solution of the scheduling problem into a chromosome is a key issue for GAs. For evaluating the effectiveness of the different chromosome representation, there are several critical issues are summarized by Gen and Lin [21].

Space: Chromosomes should not require extravagant amounts of memory.

Time: The time complexities of evaluating, recombining, and mutating chromosomes should be small.

Feasibility: All chromosomes, particularly those generated by simple crossover (i.e., one-cut point crossover) and mutation, should represent feasible solutions.

Uniqueness: The mapping from chromosomes to solutions (decoding) may belong to one of the following three cases: 1-to-1 mapping, n-to-1 mapping and 1-to-n mapping. The 1-to-1 mapping is the best one among three cases and 1-to-n mapping is the most undesired one.

Heritability: Offspring of simple crossover (i.e., one-cut point crossover) should represent solutions that combine substructures of their parental solutions.

Locality: A mutated chromosome should usually represent a solution similar to that of its parent.

We need to consider these critical issues carefully when designing an appropriate representation so as to build an effective GA. As known, scheduling problem is the implement of production plan, with considering production processes, lot-size, amount and customer requirements etc. And scheduling problem is how to decide the resources assignment to the production, with considering constrains of resources capabilities and capacities. There are two decision making parts for scheduling optimization: (1) operation sequencing and (2) resources assignment.

Representation for Operation Sequencing and Resource Assignment: In the past few decades, the following 6 representations for the job-shop scheduling problem (JSP, an operation sequencing problem with considering the precedence constraints of operations) have been proposed [13]:

Operation-based representation, Job-based representation, Preference list-based representation, Priority rule-based representation, Completion time-based representation, Random key-based representation.

The flexible job-shop scheduling problem (FJSP) is expanded from the traditional JSP, which possesses wider availability of machines for all the operations (a combinatorial optimization problem considering both of the operation sequence and the resource assignment). The following 4 representations for FJSP have been proposed [13].

Parallel machine-based representation, Parallel jobs representation, Operations machines-based representation, Multistage operation-based representation.

Cheng and Gen [51] proposed a priority-based representation firstly in evolution program for solving resource-constrained project scheduling problem (RcPSP). This representation encodes a schedule as a sequence of operations and each gene stands for one operation [12]. As known, a gene in a chromosome is characterized by two factors: locus, i.e., the position of the gene located within the structure of chromosome, and allele, i.e., the value the gene takes. In this encoding method, the position of a gene is used to represent operation ID and its value is used to represent the priority of the operation for constructing a schedule among candidates. A schedule can be uniquely determined from this priority-based encoding. However, the nature of the priority-based representation is a kind of permutation representations. Generally, this representation will yield illegal offspring when using one-cut point crossover or other simple crossover operators. That means some node's priority may be duplicated in the offspring.

Genetic Operations: When GAs are used, both the search direction to optimal solution and the search speed should be considered as an important factor, in order to keep a balance between exploration and exploitation in search space. In general, the exploitation of the accumulated information resulting from a GA search is done by the selection mechanism, whereas the exploration to new regions of the search space is accounted for by genetic operators. The genetic operators mimic the process of heredity of genes to create new offspring at each generation. The operators are used to alter the genetic composition of individuals during representation. In essence, the operators perform a random search and cannot guarantee to yield an improved offspring. Three common genetic operators exist: crossover, mutation, and selection.

Crossover is the main genetic operator. It operates on two chromosomes at a time and generates offspring by combining both chromosomes' features such as one-cut point, two-cut point, multi-cut point, or uniform crossover. There are several crossover operators proposed for permutation representation, such as partial-mapped crossover (PMX), order crossover (OX), position-based crossover (PX), heuristic crossover, etc.

Mutation is a background operator that produces spontaneous random changes in various chromosomes such as inversion mutation, insertion mutation, displacement mutation, and swap mutation. In GAs, mutation serves the crucial role of either

(1) Replacing the genes lost from the population during the selection process so that they can be tried in a new context or

(2) Providing the genes that were not present in the initial population. Selection (or reproduction) operator is intended to improve the average quality of the population by giving the high-quality chromosomes a better chance to get copied into the next generation.

Selection provides the driving force in a GA. One of the common proportional selections is the so-called roulette wheel selection, and other selection types like tournament selection, elitist selection, (μ, λ) selection, and $(\mu + \lambda)$ selection deterministic procedures that select the best chromosomes from parents and offspring [13].

Multistage-based Hybrid Genetic Algorithms: Considering the GA approach proposed by Kacem et al. [52], it is complex even when you take all the objectives in count, because all the crossover and mutation are based on the chromosome which is described as a constructor of table. Therefore, it will spend more CPU-time for finding solutions. To avoid such kind of loosing and improve the convergence speed at the same time, multistage-based genetic algorithm (MsGA) for solving the flexible job-shop scheduling problems is very useful [53]. There are several stages separating the route from the starting node to the terminal one, and in each stage several states are offered for choosing as shown in Fig. 1. After we make all the decisions for choosing states, we can get a solution, and the fitness of the result is in terms of the different decisions made in the route [50].

It is obvious to find the stage number is just the total operation numbers, and also in each stage, the machines available are treating as the state correspondingly.

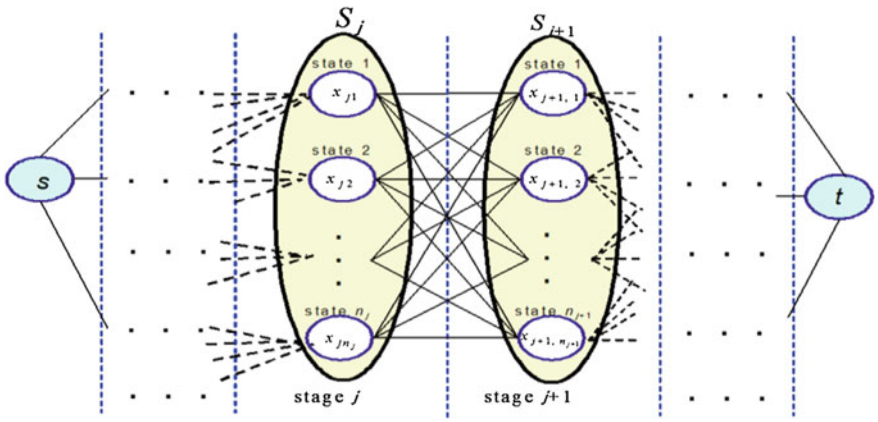


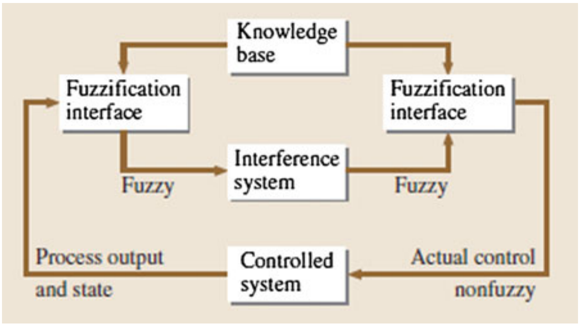
Fig. 1 Multi-stage decision making model of multioperation based genetic algorithm

3.2 Fuzzy Logic Controller for Tuning Parameters

Fuzzy logic is much closer in spirit to human thinking and natural language than the traditional logical systems. In essence, the fuzzy logic controller (FLC) provides an algorithm which can convert a linguistic control strategy based on expert knowledge into an automatic control strategy. In particular, this methodology appears very useful when the processes are too complex for analysis by conventional techniques or when the available sources of information are interpreted qualitatively, inexactly, or with uncertainty [49]. The pioneering work to extend the fuzzy logic technique to adjust the strategy parameters of genetic algorithms dynamically was carried out by Xu and Vukovich [50]. The main idea is to use a FLC to compute new strategy parameter values that will be used by the genetic algorithms [16]. A fuzzy logic controller is comprised of four principal components as shown in the generic structure of a FLC in Fig. 2:

Knowledge base, Fuzzification interface, Inference system, Defuzzification interface.

Fig. 2 Generic structure of a fuzzy logic controller (FLC)



The experts' knowledge is stored in the knowledge base in the form of linguistic control rules. The inference system is the kernel of the controller, which provides an approximate reasoning based on the knowledge base.

One of the main problems related to GA is to find the optimal control parameter values required by the technique. Furthermore, different values may be necessary during the course of a run. There has been more work dedicated to finding the optimal parameters of genetic algorithms, which require different techniques for different problems. A hybrid genetic algorithms (HGA) has been built so that selected control parameters may be dynamically adjusted during the course of evolving a problem solution. In our paper, we use the basic concept of Yun and Gen [16] to adaptively regulate GA operators using fuzzy logic control (FLC). The main scheme of this concept is to use two FLCs: the crossover FLC and mutation FLC as shown in Fig. 3 and it is based on the coordinated strategy between the FLC and GA combined the change of the average fitness at each generation. These two FLCs are implemented independently to adaptively regulate the rates of crossover and mutation operators during genetic search process. For example, in minimization problem, we can set the change of the average fitness at generation t, $\nabla f_{avg}(t)$ as follows:

$$\Delta f_{avg} = \frac{1}{\text{popsize}} \sum_{k=1}^{\text{popsize}} f_k(t) - \frac{1}{\text{offsize}} \sum_{k=1}^{\text{offsize}} f_k(t), \tag{17}$$

where parSize and offSize are the parent size and offspring size, respectively. For the defuzzification table simplified for determining the action of the crossover FLC and mutation FLC with the input and output results of discrimination, it can refer in Yun and Gen [16].

Yun and Gen [16] proposed auto-tuning routine for the parameter crossover and mutation rates based on FLC and the change of the average fitness for enhancing the evolutionary process of the GA as shown in Fig. 4. Lin and Gen [21] also proposed another enhanced auto-tuning strategy for GA for keeping a balance between exploration and exploitation during the evolutionary process. The practical design scheme for HGA combined with FLC routine is introduced in Sect. 2 of the concatenated paper Part II [2].

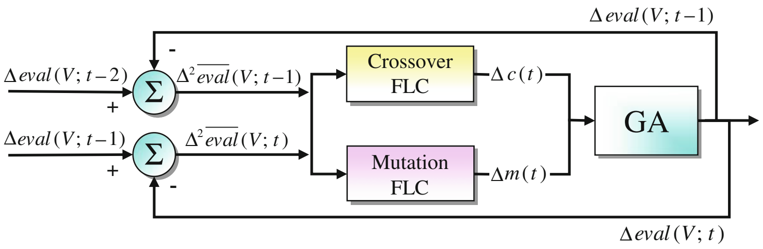
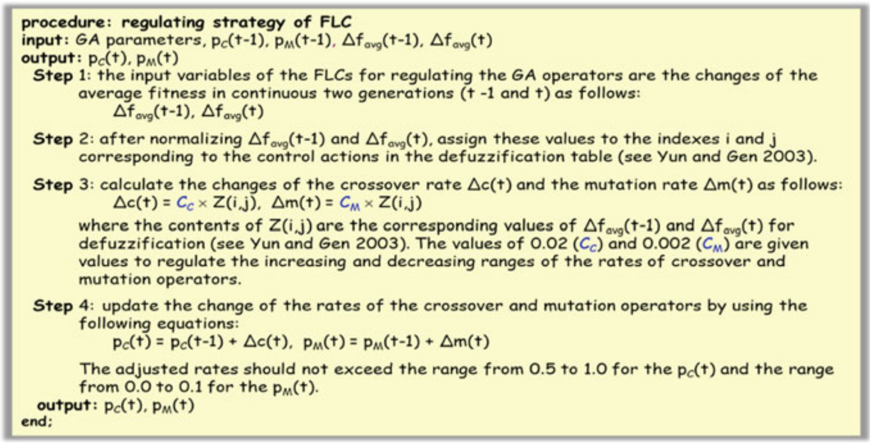


Fig. 3 Coordinated strategy between the FLC and GA



```
procedure: regulating strategy of FLC
input: GA parameters, $p_C(t-1)$, $p_M(t-1)$, $\Delta f_{avg}(t-1)$, $\Delta f_{avg}(t)$
output: $p_C(t)$, $p_M(t)$
  Step 1: the input variables of the FLCs for regulating the GA operators are the changes of the
          average fitness in continuous two generations (t -1 and t) as follows:
              $\Delta f_{avg}(t-1)$, $\Delta f_{avg}(t)$
  Step 2: after normalizing $\Delta f_{avg}(t-1)$ and $\Delta f_{avg}(t)$, assign these values to the indexes i and j
          corresponding to the control actions in the defuzzification table (see Yun and Gen 2003).
  Step 3: calculate the changes of the crossover rate $\Delta c(t)$ and the mutation rate $\Delta m(t)$ as follows:
              $\Delta c(t) = C_C \times Z(i,j)$,  $\Delta m(t) = C_M \times Z(i,j)$
          where the contents of Z(i,j) are the corresponding values of $\Delta f_{avg}(t-1)$ and $\Delta f_{avg}(t)$ for
          defuzzification (see Yun and Gen 2003). The values of 0.02 ($C_C$) and 0.002 ($C_M$) are given
          values to regulate the increasing and decreasing ranges of the rates of crossover and
          mutation operators.
  Step 4: update the change of the rates of the crossover and mutation operators by using the
          following equations:
              $p_C(t) = p_C(t-1) + \Delta c(t)$,  $p_M(t) = p_M(t-1) + \Delta m(t)$
          The adjusted rates should not exceed the range from 0.5 to 1.0 for the $p_C(t)$ and the range
          from 0.0 to 0.1 for the $p_M(t)$.
  output: $p_C(t)$, $p_M(t)$
end;
```

Fig. 4 Auto-tuning routine for the parameter crossover and mutation rates based on FLC

4 Multiobjective Hybrid Genetic Algorithms

he multiobjective optimization problems (MOP) have been receiving growing interest from researchers with various backgrounds since early 1960 [40]. There are a number of scholars who have made significant contributions to the problem. Among them, Pareto is perhaps one of the most recognized pioneers in the field. Recently, GAs have been received considerable attention as a novel approach to MOPs, resulting in a fresh body of research and applications known as evolutionary multiobjective optimization (EMO). The basic feature of GAs is the multiple directional and global searches by maintaining a population of potential solutions of potential solutions from generation to generation. The population-to-population approach is hopeful to explore all Pareto solutions. GAs are essentially a kind of metastrategy methods. When applying the GAs to solve a given problem, it is necessary to refine upon each of the major components of GAs, such as encoding methods, recombination operators, fitness assignment, selection operators, constraints handling, and so on, in order to obtain a best solution to the given problem. Because the MOPs are the natural extensions of constrained and combinatorial optimization problems, so many useful methods based on EAs developed during the past two decades. One of special issues in the MOPs is fitness assignment mechanism. Since the 1980s, several fitness assignment mechanisms have been proposed and applied in MOPs [44]. Although most fitness assignment mechanisms are just different approach and suitable to different cases of MOPs, in order to understanding the development of multiobjective GAs (MoGAs), we classify algorithms according to proposed years of different approaches.

MOPs arise in the design, modeling, and planning of many real complex systems in the areas of industrial production, urban transportation, capital budgeting, forest management, reservoir management, layout and landscaping of new cities, energy distribution, etc. [1]. Since the 1990s, EAs have been received considerable attention as a novel approach to MOPs, resulting in a fresh body of research and applications known as EMO. Without loss of generality, a MOP with q objective functions conflicting each other and m constraints can be formally represented as follows:

$$\begin{aligned} &\max \left\{z_1 = f_1(\mathbf{x}), z_2 = f_2(\mathbf{x}), \ldots, z_q = f_q(\mathbf{x})\right\} \\ &\text{s.t.}\ \ g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m \\ &\qquad \mathbf{x} \in R^n. \end{aligned} \tag{18}$$

We sometimes graph the MOP problem in both decision space and criterion space. S is used to denote the feasible region in the decision space and Z is used to denote the feasible region in the criterion space respectively as follows:

$$S = \left\{\mathbf{x} \in R^n \left| g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m \right.\right\}, \tag{19}$$

$$Z = \left\{z \in R^q \left| z_1 = f_1(\mathbf{x}), z_2 = f_2(\mathbf{x}), \ldots, z_q = f_q(\mathbf{x}), \mathbf{x} \in S \right.\right\}, \tag{20}$$

where $x \in R^n$ is a vector of values of q objective functions. In the other words, Z is the set of images of all points in S. Although S is confined to the nonnegative region of R^n and Z is not necessarily confined to the nonnegative region of R^q. There usually exists a set of solutions for the multiple objective cases which cannot be simply compared with each other. Such kind of solutions are called nondominated solutions or Pareto optimal solutions, for which no improvement in any objective function is possible without sacrificing on at least one of other objectives.

Definition 1 For a given point $z_o \in Z$, it is nondominated if and only if there does not exist another point $z \in Z$ such that for the maximization case,

$$\begin{aligned} &z_k > z_k^0, \quad \text{for some } k \in \{1, 2, \ldots, q\}, \\ &z_l \geq z_l^0, \quad \text{for some } 1 \neq k, \end{aligned}$$

where z_0 is a dominated point in the criterion space Z with q objective functions.

When applying the GAs to solve a given MOP problem, it is necessary to refine upon each of the major components of GAs, such as encoding methods, recombination operators, fitness assignment, selection operators, and constraints handling, and so on, in order to obtain a best solution to the given problem. One of special issues in the multiobjective optimization problems is fitness assignment mechanism. Although most fitness assignment mechanisms are just different approach and suitable to different cases of multiobjective optimization problems, in order to understanding the development of multiobjective EAs (MOEAs), we classify algorithms according to proposed years of different approaches:

Nondominated Sorting Genetic Algorithm II [54]: Srinivas and Deb developed a Pareto ranking-based fitness assignment and it called NSGA. In each method, the nondominated solutions constituting a nondominated front are assigned the same dummy fitness value. These solutions are shared with their dummy fitness values (phenotypic sharing on the decision vectors) and ignored in the further classification process. Finally, the dummy fitness is set to a value less than the smallest shared fitness value in the current nondominated front. Then the next front is extracted. The procedure of NSGA II is repeated until all individuals in the population are classified [54].

Random-weight Genetic Algorithm [55]: Ishibuchi et al. proposed a weighted-sum based fitness assignment method. Weighted-sum approach can be viewed as an extension of methods used in the multiobjective optimizations to GAs. It assigns weights to each objective function and combines the weighted objectives into a single objective function. To search for multiple solutions in parallel, the weights are not fixed and able to uniformly the sample area towards to the whole frontier.

Adaptive Weight Genetic Algorithm [13]: Gen and Cheng utilized some useful information from the current population to readjust weights to obtain a search pressure toward a positive ideal point. For the examined solutions at each generation, we define two extreme points for the kth objective (maximum: z^+, minimum: z^-) as follows:

$$z_k^{\max} = \max\{f_k(\mathbf{x}) |\ \mathbf{x} \in P\}, \qquad k = 1,\ 2,\ \ldots,\ q, \tag{21}$$

$$z_k^{\min} = \min\{f_k(\mathbf{x}) |\ \mathbf{x} \in P\}, \qquad k = 1,\ 2,\ \ldots,\ q. \tag{22}$$

The weighted-sum objective function for a given chromosome x is given by the following equation:

$$\text{eval}(\mathbf{x}) = \sum_{k=1}^{q} w_k(z_k - z_k^{\min}) = \sum_{k=1}^{q} \frac{z_k - z_k^{\min}}{z_k^{\max} - z_k^{\min}} = \sum_{k=1}^{q} \frac{f_k(\mathbf{x}) - z_k^{\min}}{z_k^{\max} - z_k^{\min}}, \tag{23}$$

where w_k is adaptive weight for the kth objective function as shown in the following equation:

$$w_k = \frac{1}{z_k^{\max} - z_k^{\min}}, k = 1, 2, \ldots, q. \tag{24}$$

The Eq. (7) driven above is a hyperplane defined by the following extreme points in current solutions:

Strength Pareto Evolutionary Algorithm [56]: Zitzler and Thiele proposed strength Pareto Evolutionary Algorithm [57] and an extended version SPEA2 [56] that combines several features of previous MOGA in a unique manner. The fitness assignment procedure is a two-stage process. The individuals in the external nondominated set P' are ranked.

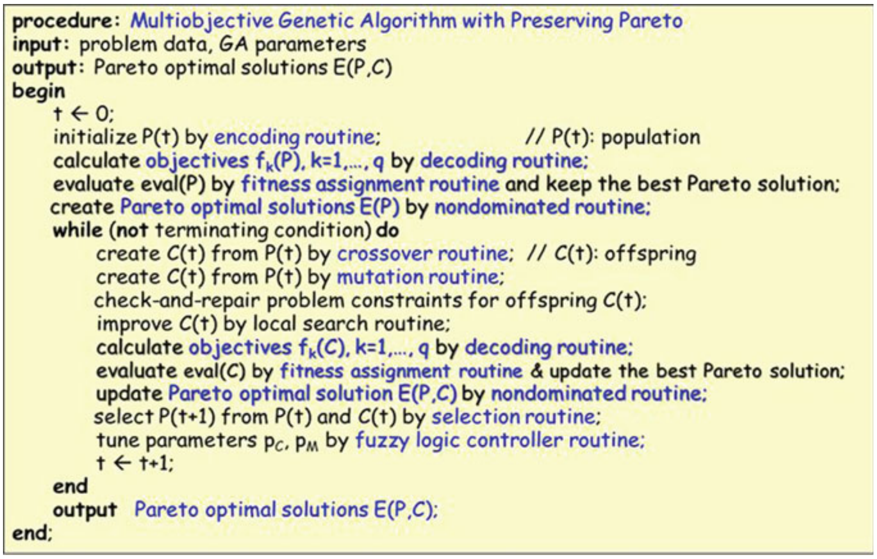
Interactive Adaptive-weight Genetic Algorithm [21]: Lin and Gen proposed an interactive AWGA, which is an improved adaptive-weight fitness assignment approach with the consideration of the disadvantages of weighted-sum approach and Pareto ranking-based approach. They combined a penalty term to the fitness value for all of dominated solutions. Firstly, we calculate the adaptive weight $w_i = 1/(Z_i^{\max} - Z_i^{\min})$ for each objective by using AWGA. Afterwards, we calculate the penalty term $p(v_k) = 0$, if v_k is nondominated solution in the nondominated set P. Otherwise $p(v'_k) = 1$ for dominated solution v'_k. Lastly, we calculate the fitness value of each chromosome by combining the i-AWGA method:

$$\text{eval}(v_k) = \sum_{i=1}^{q} w_i(z_i^k - z_i^{\min}) + p(v_k), \quad \forall k \in \text{popSize}. \tag{25}$$

Hybrid Sampling Strategy-based EA: Zhang et al. [58] proposed a hybrid sampling strategy-based evolutionary algorithm (HSS-EA). A Pareto dominating and dominated relationship-based fitness function (PDDR-FF) is proposed to evaluate the individuals. The PDDR-FF of an individual Si is calculated by the following function:

$$\text{eval}(S_i) = q(S_i) + \frac{1}{p(S_i + 1)}, \quad i = 1, 2, \ldots, \text{popSize}, \tag{26}$$

where $p()$ is the number of individuals which can be dominated by the individual S. $q()$ is the number of individuals which can dominate the individual S. The PDDR-FF can set the obvious difference values between the nondominated and dominated individuals. The general structure in the pseudo code of multiobjective hybrid genetic algorithm (Mo-HGA) is described as shown in Fig. 5.



```
procedure: Multiobjective Genetic Algorithm with Preserving Pareto
input: problem data, GA parameters
output: Pareto optimal solutions E(P,C)
begin
    t ← 0;
    initialize P(t) by encoding routine;                // P(t): population
    calculate objectives $f_k(P)$, k=1,…, q by decoding routine;
    evaluate eval(P) by fitness assignment routine and keep the best Pareto solution;
    create Pareto optimal solutions E(P) by nondominated routine;
    while (not terminating condition) do
        create C(t) from P(t) by crossover routine;  // C(t): offspring
        create C(t) from P(t) by mutation routine;
        check-and-repair problem constraints for offspring C(t);
        improve C(t) by local search routine;
        calculate objectives $f_k(C)$, k=1,…, q by decoding routine;
        evaluate eval(C) by fitness assignment routine & update the best Pareto solution;
        update Pareto optimal solution E(P,C) by nondominated routine;
        select P(t+1) from P(t) and C(t) by selection routine;
        tune parameters $p_C$, $p_M$ by fuzzy logic controller routine;
        t ← t+1;
    end
    output  Pareto optimal solutions E(P,C);
end;
```

Fig. 5 Procedure in the pseudo code of multiobjective hybrid genetic algorithm

5 Conclusion

In real world manufacturing systems there are many combinatorial optimization problems (COP) imposing on more complex issues, such as complex structure, non-linear constraints, and multiple objectives to be handled simultaneously. Manufacturing scheduling is one of the important and complex COP models, where it can have a major impact on the productivity of a production process. Moreover, the COP models make the problem intractable to the traditional optimization techniques because 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 and also for developing effective and efficient solution algorithms that are in a sense good, i.e., whose computational time is small as within 3 minutes, or at least reasonable for NP-hard combinatorial problems met in practice, 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, as the typical models in manufacturing scheduling systems we introduced parallel machines scheduling (PMS), flexible job-shop scheduling problem (FJSP) and assembly line balancing (ALB) problem. Secondly to solve NP-hard COP models, we introduced the design scheme of HGA combined with fuzzy logic controller (FLC) to tune GA parameters and multiobjective HGA (Mo-HGA) with local search such as the variable neighborhood descent (VND) routines and several fitness assignment mechanisms for multiobjective optimization problems (MOP). For demonstrating computational experiments by HGA and Mo-HGA, the effectiveness and efficiency of the HGA for the HDD (hard disc device) and Mo-HGA for TFT-LCD (thin-film transistor-liquid crystal display) module assembly problems as a practical manufacturing model, respectively is demonstrated in the concatenated paper Part II [2].

Acknowledgments 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), the Fundamental Res. Funds (Software+X) of Dalian Univ. of Tech. (No. DUT12JR05, No. DUT12JR12) and 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 Res. in Henan Univ. of Tech. (No. 2012JCYJ04).

References

1. Gen M, Lin L (2014) Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. J Intell Manuf 25:849–866
2. Gen M, Lin L, Zhang WQ (2015) Multiobjective hybrid genetic algorithms for manufacturing scheduling: part II case studies of HDD and TFT-LCD. In: The 9th international conference

on management science and engineering management
3. Bidot J, Vidal T et al (2009) A theoretic and practical framework for scheduling in a stochastic environment. J Sched 12:315–344
4. Cheng R, Gen M, Tozawa T (1995) Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. Comput Ind Eng 29:513–517
5. Cheng R, Gen M (1997) Parallel machine scheduling problems using memetic algorithms. Comput Ind Eng 33:761–764
6. Cheng R, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms: I. Representation. Comput Ind Eng 30:983–997
7. Kachitvichyanukul V, Siriwan S (2011) A two-stage genetic algorithm for multi-objective job shop scheduling problems. J Intell Manuf 22:355–365
8. Gao J, Gen M, Sun L (2006) Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. J Intell Manuf 17:493–507
9. Gao J, Gen M et al (2007) A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Comput Ind Eng 53:149–162
10. 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
11. Gen M, Gao J, Lin L (2009a) Multistage-based genetic algorithm for flexible job-shop scheduling problem. Intell Evol Syst 187:183–196
12. Gen M, Cheng R (1997) Genetic algorithms and engineering design. Wiley, New York
13. Gen M, Cheng R (2000) Genetic algorithms and engineering optimization. Wiley, New York
14. Michalewicz Z (1994) Genetic algorithm + data structure = evolution programs. Springer, New York
15. Handa H, Kawakami H, Katai O (2008) Recent advances in evolutionary computation. IEEJ Trans Electron Inf Syst 128:334–339
16. Yun YS, Gen M (2003) Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics. Fuzzy Optim Decis Mak 2:161–175
17. Li C, Cheng T (1993) The parallel machine minmax weighted absolute lateness scheduling problem. Nav Res Logist 41:33–46
18. Balin S (2011) Non-identical parallel machine scheduling using genetic algorithm. Expert Syst Appl 38:6814–6821
19. Balin S (2012) Non-identical parallel machine scheduling with fuzzy processing times using genetic algorithm and simulation. Int J Adv Manuf Technol 61:1115–1127
20. Li X, Yalaoui F et al (2012) Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. J Intell Manuf 23:1179–1194
21. Lin L, Gen M (2009) Auto-tuning strategy for evolutionary algorithms: Balancing between exploration and exploitation. Soft Comput 13:157–168
22. Jungwattanakita J, Reodechaa M et al (2009) A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. Comput Oper Res 36:358–378
23. Chen JS, Pan CH, Lin CM (2008) A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. Expert Syst Appl 34:570–577
24. Chamnanlor C, Sethanan K et al (2013) Hybrid genetic algorithms for solving reentrant flow-shop scheduling with time windows. Ind Eng Manag Syst 12:206–316
25. 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
26. 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
27. Garey M, Johnson D, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1:117–129
28. Laarhoven PV, Aarts E, Lenstra J (1992) Job shop scheduling by simulated annealing. Oper Res 40:113–125

29. Dell'Amico M, Trubian M (1993) Applying tabu search to the job shop scheduling problem. Ann Oper Res 40:231–252
30. Gen M, Tsujimura Y, Kubota E (1994) Solving job-shop scheduling problems by genetic algorithm. Proc Int Conf IEEE Syst Man Cybern 2:1577–1582
31. Cheng R, Gen M, Tsujimura Y (1999) A tutorial survey of job-shop scheduling problems using genetic algorithms: II. Hybrid genetic search strategies. Comput Ind Eng 36:343–364
32. Mastrolilli M, Gambardella LM (2000) Effective neighborhood functions for the flexible job shop problem. J Sched 3:3–20
33. Karp RM (1972) Reducibility among combinatorial problems. Plenum Press, New York
34. Scholl A (1999) Balancing and sequencing of assembly lines. Physica-Verlag, Heidelberg
35. Bautista J, Pereira J (2002) Ant algorithms for assembly line balancing. Lect Notes Comput Sci 2463:65–75
36. Falkenauer E, Delchambre A (1992) A genetic algorithm for bin packing and line balancing. In: Proceeding of IEEE international conference on robotics and automation, pp 1189–1192
37. Gen M, Tsujimura Y, Li Y (1996) Fuzzy assembly line balancing using genetic algorithms. Comput Ind Eng 31:631–634
38. Kim YK, Kim YJ, Kim YH (1996) Genetic algorithms for assembly line balancing with various objectives. Comput Ind Eng 30:397–409
39. Leu YY, Matheson LA, Rees LP (1994) Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria. Decis Sci 15:581–606
40. Hwang CL, Yoon K (1981) Multiple attribute decision making: methods and applications. Springer, Berlin
41. Tasan SO, Tunali S (2008) A review of the current applications of genetic algorithms in assembly line balancing. J Intell Manuf 19:49–69
42. Zhang WQ, Lin L, Gen M (2008) A multiobjective genetic algorithm based approach to assembly line balancing problem with worker allocation. J Soc Plant Eng Jpn 19:61–72
43. Levitin G, Rubinovitz J, Shnits B (2006) A genetic algorithm for robotic assembly line balancing. Eur J Oper Res 168:811–825
44. Lin L, Gen M, Gao J (2008) Optimization and improvement in robot-based assembly line system by hybrid genetic algorithm. IEEE Trans Electron Inf Syst 128:424–431
45. Gao J, Wang L, Gen M (2009) An efficient approach for type II robotic assembly line balancing problems. Comput Ind Eng 56:1065–1080
46. Zhang WQ, Lin L, Gen M (2008) Using multi-objective genetic algorithm with fuzzy logic controller for assembly line balancing problem with worker allocation. Int J Inf Syst Logist Manag 3:79–88
47. Gen M, Zhang WQ, Lin L (2009) Survey of evolutionary algorithms in advanced planning and scheduling. J Korean Inst Ind Eng 35:15–39
48. Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading
49. Herrera F, Lozano M (1996) Adaptation of genetic algorithm parameters based on fuzzy logic controllers. Physica-Verlag, Berlin, pp 95–125
50. Gen M (2006) Genetic algorithms and their applications. Springer, New York
51. Cheng R, Gen M (1994) Evolution program for resource constrained project scheduling problem. In: Proceeding of first IEEE conference on evolutionary computation, pp 736–741
52. 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
53. Zhang H, Gen M (2005) Multistage-based genetic algorithm for flexible job-shop scheduling problem. Complex Int 11:223–232
54. Deb K (2001) Multiobjective optimization using evolutionary algorithms. Wiley, Chichester
55. Ishibuchi H, Murata T (1998) A multiobjective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans Syst Man Cybern 28:392–403
56. Zitzler E, Thiele L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. Technical report 103. Computer Engineering and Communication Networks Lab (TIK)

57. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3:257–271
58. Zhang WQ, Gen M, Jo JB (2014) Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem. J Intell Manuf 25:881–897