Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems

Shinn-Ying Ho, *Member, IEEE*, Li-Sun Shu, *Student Member, IEEE*, and Jian-Hung Chen, *Student Member, IEEE*

***Abstract*—This paper proposes two intelligent evolutionary algorithms IEA and IMOEA using a novel intelligent gene collector (IGC) to solve single and multiobjective large parameter optimization problems, respectively. IGC is the main phase in an intelligent recombination operator of IEA and IMOEA. Based on orthogonal experimental design, IGC uses a divide-and-conquer approach, which consists of adaptively dividing two individuals of parents into N pairs of gene segments, economically identifying the potentially better one of two gene segments of each pair, and systematically obtaining a potentially good approximation to the best one of all combinations using at most $2N$ fitness evaluations. IMOEA utilizes a novel generalized Pareto-based scale-independent fitness function for efficiently finding a set of Pareto-optimal solutions to a multiobjective optimization problem. The advantages of IEA and IMOEA are their simplicity, efficiency, and flexibility. It is shown empirically that IEA and IMOEA have high performance in solving benchmark functions comprising many parameters, as compared with some existing EAs.**

***Index Terms*—Evolutionary algorithm (EA), genetic algorithm (GA), intelligent gene collector (IGC), multiobjective optimization, orthogonal experimental design.**

I. Introduction

THE GREAT success for evolutionary computation techniques, including evolutionary programming (EP), evolutionary strategy (ES), and genetic algorithms (GAs), came in the 1980s when extremely complex optimization problems from various disciplines were solved, thus facilitating the undeniable breakthrough of evolutionary computation as a problem-solving methodology [1]. Inspired from the mechanisms of natural evolution, evolutionary algorithms (EAs) utilize a collective learning process of a population of individuals. Descendants of individuals are generated using randomized operations such as mutation and recombination. Mutation corresponds to an erroneous self-replication of individuals, while recombination exchanges information between two or more existing individuals. According to a fitness measure, the selection process favors better individuals to reproduce more often than those that are relatively worse [1].

Manuscript received January 6, 2002; revised June 2, 2003 and January 21, 2004. This work was supported in part by the National Science Council of R.O.C. under Contract NSC 91-2213-E-035-016.

S.-Y. Ho is with the Department of Biological Science and Technology, Institute of Bioinformatics, National Chiao Tung University, Hsin-Chu, Taiwan 300, R.O.C. (e-mail: syho@mail.nctu.edu.tw).

L.-S. Shu and J.-H. Chen are with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan 407, R.O.C.

Digital Object Identifier 10.1109/TEVC.2004.835176

EAs have been shown to be effective for solving NP-hard problems and exploring complex nonlinear search spaces as efficient optimizers. To solve many intractable engineering optimization problems comprising lots of parameters using EAs, these parameters are encoded into individuals where each individual represents a search point in the space of potential solutions. A large number of parameters would result in a large search space. Based on schema theorem, KrishnaKumar *et al.* [2] indicated that GAs need an enormously large population size and, hence, a large number of function evaluations for effectively solving a large parameter optimization problem (LPOP). Thierens *et al.* [3] derive time complexities for the boundary case which is obtained with an exponentially scaled problem (BinInt) having l building blocks (BBs) and show that this domino convergence time complexity is linear $O(l)$ for constant intensity selection (such as tournament selection and truncation selection) and exponential $O(2^l)$ for proportionate selection. Nowadays, it is difficult for conventional EAs to effectively solve LPOPs.

In this paper, we propose two intelligent evolutionary algorithms IEA and IMOEA using a novel intelligent gene collector (IGC) to solve single- and multiobjective LPOPs, respectively. IGC is the main phase in an intelligent recombination operator of IEA and IMOEA. Based on orthogonal experimental design [4]–[7], IGC uses a divide-and-conquer approach, which consists of adaptively dividing two individuals of parents into N pairs of gene segments, economically identifying the better one of two gene segments of each pair, and systematically obtaining a potentially good approximation to the best one of all combinations using at most $2N$ fitness evaluations. IMOEA utilizes a novel generalized Pareto-based scale-independent fitness function for efficiently finding a set of Pareto-optimal solutions to a multiobjective optimization problem. The advantages of IEA and IMOEA are their simplicity, efficiency, and flexibility. It will be shown empirically that IEA and IMOEA have high performance in solving benchmark functions comprising many parameters, as compared with some existing EAs.

The remainder of this paper is organized as follows. Section II presents the used orthogonal experimental design (OED) for IGC and the merits of the proposed OED-based EAs. Section III gives the proposed intelligent gene collector IGC. Section IV describes the IGC-based IEA and IMOEA. In Section V, we compare IEA with some existing EAs using benchmark functions comprising various numbers of parameters. In Section VI, we compare IMOEA with some existing multiobjective EAs by applying them to solving multiobjective 0/1 knapsack problems and benchmark functions. Section VII concludes this paper.

II. Orthogonal Experimental Design (OED)

A. Used OED

Experiments are carried out by engineers in all fields to compare the effects of several conditions or to discover something new. If an experiment is to be performed efficiently, a scientific approach to planning it must be considered. The statistical design of experiments is the process of planning experiments so that appropriate data will be collected, a minimum number of experiments will be performed to acquire necessary technical information, and suitable statistic methods will be used to analyze the collected data [4].

An efficient way to study the effect of several factors simultaneously is to use OED with both orthogonal array (OA) and factor analysis. The factors are the variables (parameters), which affect response variables, and a setting (or a discriminative value) of a factor is regarded as a level of the factor. A "complete factorial" experiment would make measurements at each of all possible level combinations. However, the number of level combinations is often so large that this is impractical, and a subset of level combinations must be judiciously selected to be used, resulting in a "fractional factorial" experiment [6], [7]. OED utilizes properties of fractional factorial experiments to efficiently determine the best combination of factor levels to use in design problems.

An illustrative example of OED using an objective function is given as follows:

$$\text{maximize } y(x_1, x_2, x_3) = 100\,x_1 - 10\,x_2 - x_3, \quad x_1 \in \{1,2\}, x_2 \in \{3,4\}, \text{ and } x_3 \in \{5,6\}. \quad (1)$$

This maximization problem can be regarded as an experimental design problem of three factors, with two levels each. Let factors 1, 2, and 3 be parameters x_1, x_2, and x_3, respectively. Let the smaller (larger) value of each parameter be the level 1 (level 2) of each factor. The objective function y is the response variable. A complete factorial experiment would evaluate $2^3 = 8$ level combinations, and then the best combination $(x_1, x_2, x_3) = (2, 3, 5)$ with $y = 165$ can be obtained. The factorial array and results of the complete factorial experiment are shown in Table I. A fractional factorial experiment uses a well-balanced subset of level combinations, such as the first, fourth, sixth, and seventh combinations. The best one of the four combinations is $(x_1, x_2, x_3) = (2, 3, 6)$ with $y = 164$. Using OED, we can reason the best combination (2, 3, 5) from analyzing the results of the four specific combinations, described below.

OA is a fractional factorial array, which assures a balanced comparison of levels of any factor. OA is an array of numbers arranged in rows and columns, where each row represents the levels of factors in each combination, and each column represents a specific factor that can be changed from each combination. The term "main effect" designates the effect on response variables that one can trace to a design parameter [5]. The array is called orthogonal because all columns can be evaluated independently of one another, and the main effect of one factor does not bother the estimation of the main effect of another factor [6], [7].

TABLE I
Example of a Complete Factorial Experiment

Combination no. t	Factor F_i			Parameter x_j			Response variable y_t	Rank of y_t
	F_1	F_2	F_3	x_1	x_2	x_3		
1	1	1	1	1	3	5	65	5
2	1	1	2	1	3	6	64	6
3	1	2	1	1	4	5	55	7
4	1	2	2	1	4	6	54	8
5	2	1	1	2	3	5	165	1
6	2	1	2	2	3	6	164	2
7	2	2	1	2	4	5	155	3
8	2	2	2	2	4	6	154	4

Let there be N factors, with two levels each, used in IGC. The total number of level combinations is 2^N for a complete factorial experiment. To use an OA of N factors, we obtain an integer $n = 2^{\lceil \log_2(N+1) \rceil}$, where the bracket represents a ceiling operation, build a two-level OA $L_n(2^{n-1})$ with n rows and $n-1$ columns, use the first N columns, and ignore the other $n-N-1$ columns. For example, if $N \in \{4,5,6,7\}$, then $n = 8$ and $L_8(2^7)$ is used. The numbers 1 and 2 in each column indicate the levels of the factors. Each column has an equal number of 1s and 2s. The four combinations of factor levels, (1,1), (1,2), (2,1), and (2,2), appear the same number of times in any two columns. The "L" stands for Latin, since most of these designs are based on classical Latin Square layouts, appropriately pruned for efficiency [5]. For example, an $L_4(2^3)$ is as follows:

$$\begin{array}{c} 1\,1\,1 \\ 1\,2\,2 \\ 2\,1\,2 \\ 2\,2\,1. \end{array}$$

Note that the OA $L_4(2^3)$ is the above-mentioned well-balanced subset of the complete factorial experiment in Table I. OA can reduce the number of combinations for factor analysis. The number of OA combinations required to analyze all individual factors is only n, where $N+1 \le n \le 2N$. An algorithm for generating a two-level OA for N factors is given as follows.

```
Algorithm Generate_OA(OA, N)
{
  n := 2^⌈log2(N+1)⌉;
  for i := 1 to n do
    for j := 1 to N do
      level := 0;
      k = j;
      mask := n/2;
      while k > 0 do
        if (k mod 2) and (bitwise_AND(i − 1,
        mask) ≠ 0) then
          level := (level + 1) mod 2;
        k := ⌊k/2⌋;
        mask := mask/2;
      OA[i][j] := level + 1;
} // bitwise_AND(α, mask) returns the
mth least significant bit of α, where
mask = 2^(m−1). //
```

For IGC, levels 1 and 2 of a factor represent selected genes from parents 1 and 2, respectively. After evaluation of the n combinations, the summarized data are analyzed using factor analysis. Factor analysis can evaluate the effects of individual factors on the objective (or fitness) function, rank the most effective factors, and determine the better level for each factor such that the function is optimized. Let y_t denote a function value of the combination t, where $t = 1, \ldots, n$. Define the main effect of factor j with level k as S_{jk}, where $j = 1, \ldots, N$ and $k = 1, 2$

$$S_{jk} = \sum_{t=1}^{n} y_t \cdot F_t \tag{2}$$

where $F_t = 1$ if the level of factor j of combination t is k; otherwise, $F_t = 0$. Considering the case that the optimization function is to be maximized, the level 1 of factor j makes a better contribution to the function than level 2 of factor j does when $S_{j1} > S_{j2}$. If $S_{j1} < S_{j2}$, level 2 is better. On the contrary, if the function is to be minimized, level 2 (level 1) is better when $S_{j1} > S_{j2}$ ($S_{j1} < S_{j2}$). If $S_{j1} = S_{j2}$, levels 1 and 2 have the same contribution. The main effect reveals the individual effect of a factor. The most effective factor j has the largest main effect difference $\mathrm{MED} = |S_{j1} - S_{j2}|$. Note that the main effect holds only when no or weak interaction exists, and that makes the OED-based IGC efficient.

After the better one of two levels of each factor is determined, a reasoned combination consisting of N factors with better levels can be easily derived. The reasoned combination is a potentially good approximation to the best one of the 2^N combinations. OED uses well-planned procedures in which certain factors are systematically set and modified, and then main effects of factors on the response variables can be observed. Therefore, OED using OA and factor analysis is regarded as a systematic reasoning method.

A concise example of OED for solving the optimization problem with (1) is described as follows (see Table II). First, use an $L_4(2^3)$, set levels for all factors as above mentioned, and evaluate the response variable y_t of the combination t, where $t = 1, \ldots, 4$. Second, compute the main effect S_{jk} where $j = 1$, 2, 3, and $k = 1$, 2. For example, $S_{21} = y_1 + y_3 = 229$. Third, determine the better level of each factor based on the main effect. For example, the better level of factor 1 is level 2 since $S_{11} < S_{12}$. Therefore, select $x_1 = 2$. Finally, the best combination (*Comb1*) can be obtained, which is $(x_1, x_2, x_3) = (2, 3, 5)$ with $y = 165$.[1] The most effective factor is x_1 with the largest $\mathrm{MED} = 200$. It can be verified from (1). The second best combination (*Comb2*) is $(x_1, x_2, x_3) = (2, 3, 6)$, which can be derived from the best one by replacing level 1 with level 2 of x_3 (the factor with the smallest MED). If only OA combinations without factor analysis are used, as did in the orthogonal GA with quantization (OGA/Q) [8], the best combination *Comb1* cannot be obtained.

[1]The reasoned combination is not guaranteed to be the best one in general cases.

TABLE II
CONCISE EXAMPLE OF OED USING AN $L_4(2^3)$

Combination no. t	Parameter x_j			y_t	Rank of y_t
	x_1	x_2	x_3		
1	1	3	5	65	5
2	1	4	6	54	8
3	2	3	6	164	2
4	2	4	5	155	3
S_{j1}	119	229	220		
S_{j2}	319	209	218		
Better level	2	1	1		
MED	200	20	2		
Comb1	2	3	5	165	1
Comb2	2	3	6	164	2

B. Merits of the Proposed OED-Based EAs

The OED-based methods have been recognized as a more efficient alternative to the traditional approach, such as one-factor-at-a-time method and acceptance sampling [4]–[7]. OED ensures an unbiased estimation of the effects of various factors on response variables and also plays an important role in designing an experiment of the Taguchi method [5], [9]. The merits of IEA and IMOEA are that they can simultaneously tackle the following four issues of the investigated LPOP by incorporating the advantages of OED into EAs. However, most existing studies on robust design using OED have addressed only part of the four issues, described below.

1) Full-Range Process of Continuous Parameters: If data type of parameters is continuous, it is hard for engineers to decide the number of factor levels and determine the best level settings, especially considering many parameters. Therefore, an iterative method is suggested to approximate the level settings with a small number of levels and then determine the most appropriate level settings by a nonlinear programming method [10]. Kunjur and Krishnamurty [11] proposed an iterative procedure to identify the feasible domains of parameters and then narrow the domains for tuning of a full-range process. Leung and Wang [8] proposed a single-objective GA using OAs for global numerical optimization that each continuous parameter is quantized into a finite number of values.

2) Multiple Objectives and Constraints: As to multiobjective optimization problems with constraints, the overall aggregating objective function using a penalty strategy is often used by the Taguchi method. However, the optimization of multiple objectives by using an aggregating function would possibly lead to erroneous results and fail to find a proper Pareto front, because of the difficulty involved in making the tradeoff decisions to arrive at this single expression [11]–[13]. OED-based approaches using the response surface methodology for optimization of multiple objectives vary from overlaying contours and surface plots of each objective function to the derivation of

complex functions [5]. These available optimization approaches using the response surface methodology do not enable the simultaneous optimization of multiple objectives [12], [13].

3) Large Number of Parameters: In traditional OED-based approaches, most of them dealt with few factors. It is not easy to design OED when the numbers of factors and levels increase, and the requirement of avoiding confounding effects due to interaction effects are imposed [5], [7], [10]. Some ad hoc methods such as multiple plots, engineering heuristics, and prior knowledge are used in OED-based approaches to optimizing multiple objectives [10]. Antony [13] uses Taguchi's quality loss function and principal component analysis for transforming a set of correlated responses into a small number of uncorrelated principal components to achieve an overall aggregating function. Elsayed and Chen [14] proposed an approach for product parameter setting that lists all possible values of the responses in a table and directly selects optimal levels of the factors. A major disadvantage of these ad hoc OED-based approaches is that the factors are often with unequal numbers of levels, so that some special design techniques must be performed by the experimenter, such as multilevel arrangements, dummy treatment, combination design, and idle column method, etc., [4], [7], [10].

4) Enumeration of Nondominated Solutions: A major drawback of the traditional OED-based approaches is that it is difficult to enumerate a set of nondomination solutions, because these approaches are designed for producing a single solution per single run. Song *et al.* [15] extended the Taguchi method and developed a discrete multiobjective optimization algorithm that incorporates the methods of dominated approximations and reference points to obtain nondominated solutions. The proposed discrete algorithm is shown to be more efficient for complex design problems involving many factors and multiple objectives. However, this approach is unable to effectively enumerate nondominated solutions to the large multiobjective optimization problems.

IEA and IMOEA take advantage of the reasoning ability of OED in selecting the potentially best combination of better levels of factors to efficiently identify good genes of parents and then generate good children. The divide-and-conquer mechanism of IGC based on OED can cope with the problem of many parameters. The evolution ability of EA will be used to tune the levels in the full-range process of continuous parameters. Furthermore, the evolution ability of IEA and IMOEA can compensate OED excluding the study of interactions existed in the generalized single- and multiobjective optimization problems with constraints. In addition, IMOEA can economically find satisfactory nondominated solutions without further interactive process or specified reference point [16] from decision makers to guide evolution to Pareto-optimal solutions.

III. Intelligent Gene Collector (IGC)

It is well recognized that divide-and-conquer is an efficient approach to solving large-scale problems. Divide-and-conquer mechanism breaks a large-scale problem into several subproblems that are similar to the original one but smaller in size, solves the subproblems concurrently, and then combines these solutions to create a solution to the original problem. IGC is the main phase in an intelligent recombination operator of IEA and IMOEA. It uses a divide-and-conquer approach, which consists of three phases: 1) division phase: divide large chromosomes into an adaptive number of gene segments; 2) conquest phase: identify potentially good gene segments such that each gene segment can possibly be one part of an optimal solution; and 3) combination phase: combine the potentially better gene segments of their parents to produce a potentially good approximation to the best one of all combinations of gene segments.

A. Division Phase

Before the IGC operation, we ignore the parameters having identical values in two parents such that the chromosomes can be temporally shortened resulting in using small OAs. Let L be the size of chromosomes represented by using bit string, bit matrix, integer, or real number, etc. Using the same division scheme with/without prior knowledge, divide two chromosomes of nonidentical parents into $N > 1$ pairs of gene segments GS_i with sizes l_i,$i = 1, \ldots, N$, such that

$$\sum_{i=1}^{N} l_i = L \quad \text{and} \quad GS_i \cap GS_j = \phi,\ i \neq j \tag{3}$$

and the alleles of gene segments GS_i in the same pair are not identical. If the maximal value of N equals one, the IGC operation is not necessarily applied. One IGC operation can explore the search space of 2^N combinations by reasoning to obtain a good child using at most $2N$ fitness evaluations. If there exists no or weak interactions among gene segments, a larger value of N can make IGC more efficient. Generally, a larger value of l_i can make the estimated main effects of gene segments more accurate. Considering the tradeoff, the best values of N and l_i depend on the interaction degree of problem's encoding parameters in a genotype. An efficient bi-objective division criterion is to minimize the degree of epistasis, which is taken as a measure of the amount of interactions in a chromosome [17], while maximizing the value of N. If the epistasis is low, the commonly used value of N is $N = 2^{\lfloor \log_2(M+1) \rfloor} - 1$, where the bracket represents a floor operation and, thus, the used OA is $L_{N+1}(2^N)$, where M is the number of parameters participated in the division phase. This value of N is a maximal one which can make efficient use of all columns of OA. The $N - 1$ cut points are randomly specified from the $M - 1$ candidate cut points which separate individual parameters. Since gene segment is the unit of exchange between parents in the IGC operation, the feasibility maintenance is often considered in the division phase such that all combinations of gene segments are feasible. If feasibility maintenance is considered or the epistasis is high, the best value of N is problem-dependent.

A simple example of using an OED-based EA for solving a polygonal approximation problem (PAP) is given to illustrate the efficient division of chromosomes [18]. The statement of PAP is as follows: for a given number r of approximating vertices which are a subset of the original with p points, the objective is to minimize the error between the digital contour of a 2-D shape and the approximating polygon. The optimization

problem is NP-hard having a search space of $C(p, r)$ instances, i.e., the number of ways of choosing r vertices out of p points. The feasible solution is encoded using a binary string that the numbers of 1's and 0's are r and $p - r$, respectively. The bit has value 1 when the corresponding point is selected. Since the neighboring points have larger interactions than distant ones, two neighboring points are encoded as two neighboring bits in genotypes.

The criterion of chromosome division is to determine a maximal number N of pairs of gene segments such that the numbers of 1's in every pair of gene segments of parents are equal and these alleles of two gene segments are not identical. If no such cut points exist, the recombination operation will not be applied before performing mutation operations. For instance, let $p = 10$ and $r = 5$. Two feasible parents P_1 and P_2 can be divided using $N = 3$ as follows: $P_1 = \underline{10}\ \underline{10}\ \underline{111000}$ and $P_2 = \underline{01}\ \underline{01}\ \underline{010101}$. Consequently, an $L_4(2^3)$ is used, where one gene segment is treated as a factor. This division criterion guarantees that all possible combinations of gene segments always remain feasible.

It is shown that the necessity of maintaining feasibility can make the OED-based recombination more efficient than the penalty approach [18]. If it is difficult to divide the chromosomes such that all possible combinations of gene segments always remain feasible, an additional repair operation on individual gene segments before the recombination can be advantageously used [19]. For a 0/1 optimization problem with no constraints and low epistasis, 1 bit can be treated as a gene segment [20].

B. Conquest Phase

IGC seeks the best combination consisting of a set of good gene segments. The aim of the conquest phase is to identify good gene segments according to the main effect of factors (gene segments). It is desirable to evolve these good gene segments based on the evolution ability of EA such that a set of optimal gene segments can exist in a population. Consequently, all these optimal gene segments can be collected to form an optimal solution through the combination phase. The following aspects may be helpful in obtaining optimal gene segments.

1) Initial Population: In general, an initial population is randomly generated. OED has been proven optimal for additive and quadratic models, and the selected combinations are good representatives for all of the possible combinations [21]. Since OA specifies a small number of representative combinations that are uniformly distributed over the whole space of all possible combinations, IEA and IMOEA can generate these representative combinations and select a number of combinations having the best fitness performance as the initial population, as similarly did in OGA/Q [8]. Generally, this OA-based initial population is helpful for optimizing functions with high epistasis. For LPOP, the OA-based initial population often takes a large number of fitness evaluations because this method is effective when the resolution of quantization is relatively high. For a set of real-world intractable engineering problems, only a small number of samples are available or a fitness evaluation takes many costs. In such cases, this approach to generating an initial population is not practical.

2) Increasing Diversity: Since gene segment is the unit of inheritance from parents for each IGC operation, it is important to increase the diversity of gene segments in the evolutionary process. Besides the above-mentioned initial population, it is a useful strategy to use variable division configurations and dynamic values of N and l_i. Generally, cut points/lines can be randomly specified provided that the meaningful gene segments behaving as tightly linked BBs are not easily destroyed if possible. The values of N and l_i may vary in each recombination step for various purposes, such as maintaining feasibility of recombination [18], using a coarse-to-fine approach where the sizes of gene segments are gradually decreased [19], and speeding up the evolution process by ignoring the genes having the same alleles in two parents to temporally shorten chromosomes [20].

C. Combination Phase

The combination phase aims to efficiently combine good gene segments from two parents P_1 and P_2 to breed two good children C_1 and C_2 using IGC at a time. How to perform an IGC operation with the combination phase is described as follows.

- Step 1) Ignore the parameters having identical values in two parents such that the chromosomes can be temporally shortened resulting in using small OAs.
- Step 2) Adaptively divide parent chromosomes into N pairs of gene segments, where one gene segment is treated as a factor.
- Step 3) Use the first N columns of an $L_n(2^{n-1})$, where $n = 2^{\lceil \log_2(N+1) \rceil}$.
- Step 4) Let levels 1 and 2 of factor j represent the jth gene segments of chromosomes coming from parents P_1 and P_2, respectively.
- Step 5) Compute the fitness value y_t of the combination t, where $t = 1, \ldots, n$.
- Step 6) Compute the main effect S_{jk}, where $j = 1, \ldots, N$ and $k = 1, 2$.
- Step 7) Determine the better one of two levels of each factor based on main effect.
- Step 8) The chromosome of C_1 is formed using the combination of the better gene segments from the derived corresponding parents.
- Step 9) The chromosome of C_2 is formed similarly as C_1, except that the factor with the smallest MED adopts the other level.

In Step 1), the number of parameters participating in the IGC operation would be gradually decreased, while the evolution proceeds with a decreasing number of nondeterminate parameters. This behavior can reduce the number of fitness evaluations for one IGC operation and, thus, can helpfully solve LPOPs. Note that C_1 and C_2 are only different in one factor. In some applications of LPOPs, one factor may represent one parameter. In other words, if C_1 and C_2 are almost the same, it may be not necessary to generate C_2 as a child using Step 9). Various variants of generating C_2 can be adaptively used depending on evolutionary computation techniques and parameter specifications of EA.

An additional step [Step 10)] can be appended to the IGC procedure for the elitist strategy of IEA as follows.

Step 10) Select the best two individuals from the n generated combinations, C_1, C_2, and P_2 as the final children.

Note that P_1 is the combination 1 in Step 5). Step 10) can provide valuable information for determining an effective value of N. If C_1 and C_2 are often superior to the n conducted combinations, it means the used value of N is appropriate. Otherwise, it means the main effect is inaccurate resulting from high epistasis such that the value of N can be further decreased.

IV. Proposed Algorithms IEA and IMOEA

The main power of IEA and IMOEA arises mainly from IGC. The merit of IGC is that the reasoning ability of OED is incorporated in IGC to economically identify potentially better gene segments of parents and intelligently combine these better gene segments to generate descendants of individuals. The superiority of IGC over conventional recombinations for solving LPOPs arises from that IGC replaces the generate-and-test search for descendants with a reasoning search method and applies a divide-and-conquer strategy to cope with large-scale problems.

A. Chromosome Encoding

A suitable way of encoding and dividing the chromosome into N gene segments plays an important role in efficiently using IGC. The modifying genetic operator strategy invents problem-specific representation and specialized genetic operators to maintain the feasibility of chromosomes. Michalewicz *et al.* have pointed out that often such systems are much more reliable than any other GAs based on penalty approaches [22]. An experienced engineer of using IGC tries to use appropriate parameter transformation to reduce interactions among parameters and confine genetic search within feasible regions.

Arguably, one way to achieve an improved EA is to find out a set of BBs in genotypes that are responsible for the properties of interest using statistical analysis. Afterwards, the EA attempts to produce the next population of the organizm using genotypes that contain more useful BBs utilizing various types of intervention in the crossover operator [23]. The improper encoding of chromosomes may result in loose BBs with high order. Estimation of distribution algorithms (EDAs) may consider interactions among high-order BBs to facilitate EAs in efficiently mixing the BBs [24]. Statistic methods used by EDAs for extracting information need lots of samples and, consequently, cost much computation time, especially in solving LPOPs. Since it is difficult to detect the loose linkage groups and inherit the high-order BBs, it is desirable to incorporate the linkage group existent in the underlying structure of the problem in encoding chromosomes.

The behavior of gene segments is to be expected as that of tightly linked BBs with low order. If epistasis is too high, the performance of EAs will be very poor [17]. To accurately estimate the main effect of OED, chromosome should be encoded so as to reduce the degree of epistasis and maintain feasibility of all conducted combinations. A proper encoding scheme of chromosomes that the parameters with strong interactions (if prior knowledge is available) are encoded together would enhance the effectiveness of chromosome division. An illustrative example using an effective parameter transformation in encoding chromosomes to reduce the degree of epistasis and confine genetic search within feasible regions for designing genetic-fuzzy systems can be referred in [25]. IEA and IMOEA can work well without using linkage identification in solving LPOPs.

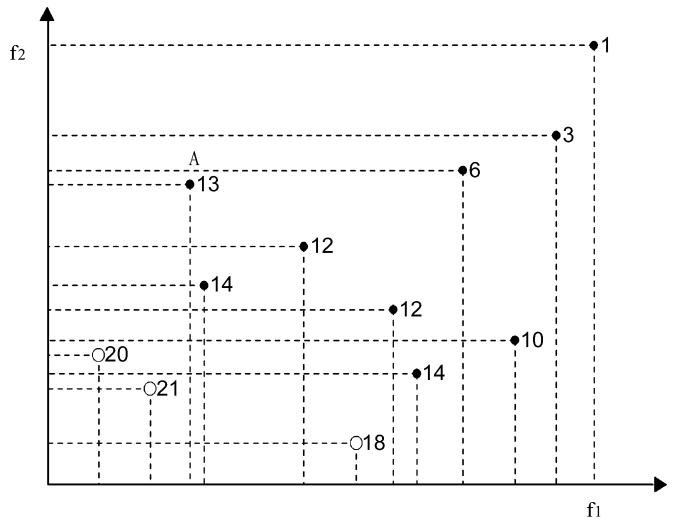

Fig. 1. Fitness values of 12 participant individuals in the objective space of a bi-objective minimization problem. The fitness value of the dominated individual A using GPSIFF is $3 - 2 + 12 = 13$.

B. Fitness Function GPSIFF

Many multiobjective EAs differ mainly in the fitness assignment strategy which is known as an important issue in solving multiobjective optimization problems (MOOPs) [26]–[29]. IMOEA uses a generalized Pareto-based scale-independent fitness function (GPSIFF) considering the quantitative fitness performances in the objective space for both dominated and nondominated individuals. GPSIFF makes the best use of Pareto dominance relationship to evaluate individuals using a single measure of performance. Let the fitness value of an individual X be a tournament-like score obtained from all participant individuals by the following function:

$$\mathrm{GPSIFF}(X) = p - q + c \tag{4}$$

where p is the number of individuals which can be dominated by X, and q is the number of individuals which can dominate X in the objective space. Generally, a constant c can be optionally added in the fitness function to make fitness values positive. In this paper, c is the number of all participant individuals.

GPSIFF uses a pure Pareto-ranking fitness assignment strategy, which differs from the traditional Pareto-ranking methods, such as nondominated sorting [23] and Zitzler and Thiele's method [27]. GPSIFF can assign discriminative fitness values to not only nondominated individuals but also dominated ones. IGC can take advantage of this assignment strategy to accurately estimate the main effect of factors and, consequently, can achieve an efficient recombination using IGC. It is less efficient for IGC to use Zitzler and Thiele's method where the fitness values of dominated individuals in a cluster are always identical. Fig. 1 shows an example for illustrating the fitness

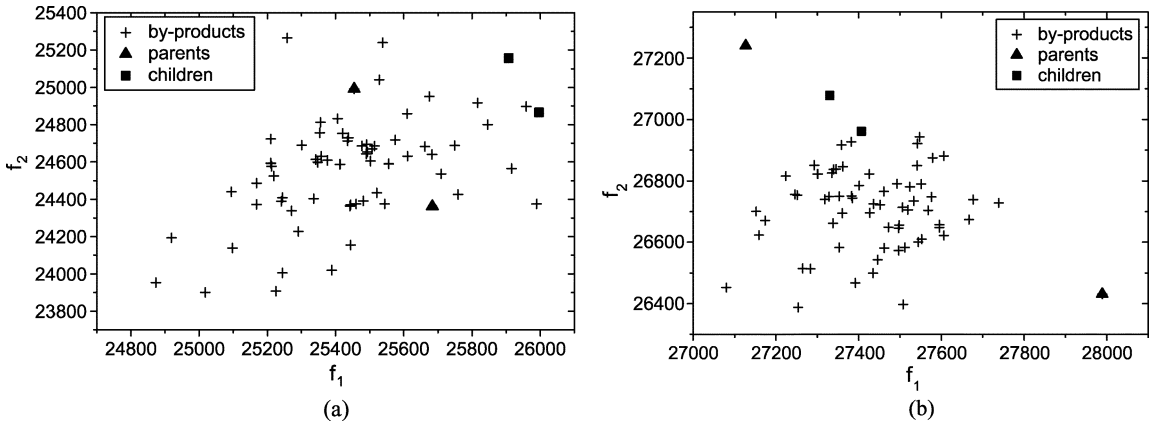

Fig. 2. Examples of an IGC operation in solving a bi-objective maximization problem. (a) Good children can be efficiently generated by the multiobjective IGC. (b) IGC can obtain a well-distributed Pareto front by generating nondominated individuals in the gap between two parents.

value using GPSIFF for a bi-objective minimization problem. For example, three individuals are dominated by $A(p = 3)$ and two individuals dominate $A(q = 2)$. Therefore, the fitness value of A is $3 - 2 + 12 = 13$. It can be found that one individual has a larger fitness value if it dominates more individuals. On the contrary, one individual has a smaller fitness value if more individuals dominate it.

C. IEA

IEA with IGC is designed toward being an efficient general-purpose optimization algorithm for solving LPOPs. Of course, domain knowledge, heuristics, and auxiliary genetic operations, which can enhance the conventional EAs can also improve the performance of IEA. The simple IEA can be written as follows.

- Step 1) Initialization: Randomly generate an initial population of $N_{\rm pop}$ individuals.
- Step 2) Evaluation: Compute fitness values of all individuals.
- Step 3) Selection: Perform a conventional selection operation. For example, truncation selection is used: the best $(1-p_s)\cdot N_{\rm pop}$ individuals are selected to form a new population, where ps is a selection probability. Let $I_{\rm best}$ be the best individual in the population.
- Step 4) Recombination: Randomly select $p_c \cdot N_{\rm pop}$ parents including $I_{\rm best}$ for IGC, where p_c is a recombination probability. Perform the IGC operations for all selected pairs of parents.
- Step 5) Mutation: Apply a conventional mutation operation (e.g., bit-inverse mutation) with a mutation probability p_m to the population. To prevent the best fitness value from deteriorating, mutation is not applied to the best individual.
- Step 6) Termination test: If a stopping condition is satisfied, stop the algorithm. Otherwise, go to Step 2).

D. IMOEA

Since it has been recognized that the incorporation of elitism may be useful in maintaining diversity and improving the performance of multiobjective EAs [26]–[29], IMOEA uses an elite set E with capacity $N_{\rm Emax}$ to maintain the best nondominated individuals generated so far. The simple IMOEA can be written as follows.

- Step 1) Initialization: Randomly generate an initial population of $N_{\rm pop}$ individuals and create an empty elite set E and an empty temporary elite set E'.
- Step 2) Evaluation: Compute all objective function values of each individual in the population. Assign each individual a fitness value by using GPSIFF.
- Step 3) Update elite sets: Add the nondominated individuals in both the population and E' to E, and empty E'. Considering all individuals in E, remove the dominated ones. If the number N_E of nondominated individuals in E is larger than $N_{\rm Emax}$, randomly discard excess individuals.
- Step 4) Selection: Select $N_{\rm pop} - N_{ps}$ individuals from the population using binary tournament selection and randomly select N_{ps} individuals from E to form a new population, where $N_{ps} = N_{\rm pop} \cdot p_s$. If $N_{ps} > N_E$, let $N_{ps} = N_E$.
- Step 5) Recombination: Perform the IGC operations for $N_{\rm pop} \cdot p_c$ selected parents. For each IGC operation, add nondominated individuals derived from by-products OA combinations (by-products) and two children to E'.
- Step 6) Mutation: Apply a conventional mutation operation with p_m to the population.
- Step 7) Termination test: If a stopping condition is satisfied, stop the algorithm. Otherwise, go to Step 2).

If many nondominated solutions are needed, especially in solving large MOOPs with many objectives, one may use an additional external set to store all the nondominated individuals found so far.

Fig. 2 illustrates two typical examples gleaned from the IGC operation using an $L_n(2^{n-1})$ with $n = 64$ in solving a bi-objective 0/1 knapsack problem, described in Section VI. Fig. 2 reveals the following.

1) For one IGC operation, the two children are more promising to be new nondominated individuals, as shown in Fig. 2(a). The individuals corresponding to OA combinations are called by-products of IGC. Because the by-products are well planned and systematically

TABLE III
PERFORMANCE OF VARIOUS EAS FOR SOLVING AN LPOP WITH A GLOBAL OPTIMUM 121.598

	SGA	Sensitivity GA	GA-Local	Stochastic GA	SAGA	AGA	OEGA	BOA	IEA	IEA	OGA/Q	OGA/Q
Chromosome length	1000 (Bit)	300 (Bit)	300 (Bit)	500 (Bit)	500 (Bit)	500 (Bit)	500 (Bit)	500 (Bit)	500 (Bit)	1400 (Bit)	100 (Real)	100 (Real)
N_{pop}	51	51	51	31	30	30	30	500	30	30	200	200
p_s	NA	NA	NA	NA	0.1	0.1	0.1	0.5	0.2	0.2	0.1	0.1
p_c	NA	NA	NA	NA	1.0	--	1.0	--	0.8	0.8	0.1	0.1
p_m	NA	NA	NA	NA	0.05	--	0.05	--	0.05	0.05	0.02	0.02
Iteration	750	NA	400	200	199	232	196	48	21	86	0	1,000
N_{eval}	38,250	11,259	37,493	12,000	12,000	12,000	12,000	12,000	12,000	58,156	102,010	388,010
Fitness value	69	114	105	115	88.114	45.816	79.800	100.667	120.890	121.598	110.715	121.594

TABLE IV
BENCHMARK FUNCTIONS

Test functions	x_i domain	Optimum
$f_1 = -\sum_{i=1}^{D}\left[\sin(x_i)+\sin\left(\frac{2x_i}{3}\right)\right]$	[3, 13]	1.21598D(max)
$f_2 = -\sum_{i=1}^{D-1}\left[\sin(x_i+x_{i+1})+\sin\left(\frac{2x_ix_{i+1}}{3}\right)\right]$	[3, 13]	≈2D (max)
$f_3 = \sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	[-100, 100]	0 (min)
$f_4 = \sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	[-5.12, 5.12]	0 (min)
$f_5 = \sum_{i=1}^{D}x_i^2$	[-5.12, 5.12]	0 (min)
$f_6 = \sum_{i=1}^{D}(x_i\sin(10\pi x_i))$	[-1.0, 2.0]	1.85D (max)
$f_7 = \sum_{i=1}^{D}\left\lvert\frac{\sin(10x_i\pi)}{10x_i\pi}\right\rvert$	[-0.5, 0.5]	0 (min)
$f_8 = 20+e-20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D}x_i^2}{D}}}-e^{\frac{\sum_{i=1}^{D}\cos(2\pi x_i)}{D}}$	[-30, 30]	0 (min)
$f_9 = 418.9829D-\sum_{i=1}^{D}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500, 500]	0 (min)
$f_{10} = \sum_{i=1}^{D-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	[-5.12, 5.12]	0 (min)
$f_{11} = 6D+\sum_{i=1}^{D}\lfloor x_i\rfloor$	[-5.12, 5.12]	0 (min)
$f_{12} = \frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]	0 (min)

sampled within the hypercube formed by parents, some of them are promising to be nondominated individuals. For example, there are three and four by-products being nondominated individuals in Fig. 2(a) and (b), respectively.

2) If the final Pareto front obtained has a gap without sufficient solutions, two nondominated individuals comprising the gap can be chosen as parents of IGC to fill the gap, as shown in Fig. 2(b). For one IGC operation, the number of nondominated individuals is increased from two to eight, including two parents, two children, and four by-products.

V. PERFORMANCE COMPARISONS OF IEA

We use three experiments to evaluate the performance of IEA by comparing with some existing EAs without heuristics using benchmark functions consisting of various numbers of parameters. The used parameters of IEA are $p_s = 0.2$, $p_c = 0.8$,

TABLE V
MEAN FITNESS VALUES AND RANKS FOR FUNCTIONS WITH $D = 10$. THE MINIMAL AND MAXIMAL FITNESS VALUES OF SOLUTIONS IN 30 RUNS ARE SHOWN IN BRACKETS

Test functions	IEA	OEGA	UEGA	TEGA	BLXGA	BOA	OGA
f_1	12.116 (**4**)	12.119(**3**)	12.110(**6**)	12.113(**5**)	12.150(**2**)	12.151(**1**)	12.109 (**7**)
	[12.077, 12.150]	[12.089, 12.149]	[12.07, 12.1499]	[12.086, 12.140]	[12.117, 12.157]	[12.129, 12.159]	[12.068, 12.140]
f_2	15.32 (**5**)	16.19(**3**)	16.39(**2**)	15.89(**4**)	16.52(**1**)	12.29(**7**)	15.24 (**6**)
	[14.34, 16.76]	[14.38, 17.68]	[14.37, 16.98]	[14.34, 17.79]	[15.18, 17.30]	[10.7, 14.182]	[12.52, 17.45]
f_3	5.13 (**4**)	5.53(**6**)	5.10(**3**)	4.90(**2**)	5.83(**7**)	0.77(**1**)	5.30 (**5**)
	[2, 11]	[2, 9]	[1, 10]	[2, 14]	[2, 22]	[0, 2]	[1, 9]
f_4	15.42 (**6**)	8.94(**3**)	9.57(**4**)	11.63(**5**)	6.48(**2**)	5.32 (**1**)	17.62 (**7**)
	[12.28, 19.68]	[2.41, 16.78]	[3.76, 25.07]	[4.78, 20.22]	[3.93, 10.30]	[2.11, 8.26]	[11.35, 49.20]
f_5	0.0003(**1**)	0.0004(**4**)	0.0003(**1**)	0.0006(**5**)	0.0092(**7**)	0.0077(**6**)	0.0003 (1)
	[2×10^{-4}, 8×10^{-4}]	[2×10^{-4}, 1.3×10^{-3}]	[2×10^{-4}, 3.5×10^{-3}]	[2×10^{-4}, 7.5×10^{-3}]	[0.003, 0.035]	[0.0014, 0.0440]	[2.5×10^{-4}, 3.2×10^{-4}]
f_6	14.60 (**6**)	15.82(**3**)	15.54(**4**)	15.24(**5**)	15.85(**2**)	18.01(**1**)	14.01 (**7**)
	[12.46, 16.36]	[13.90, 17.50]	[13.32, 17.40]	[13.26, 17.21]	[12.31, 17.91]	[17.42, 18.42]	[11.64, 17.02]
f_7	0.054(**6**)	0.035(**3**)	0.040(**5**)	0.039(**4**)	0.017(**1**)	0.019(**2**)	0.072(**7**)
	[0.047, 0.067]	[0.017, 0.124]	[0.016, 0.194]	[0.011, 0.229]	[0.007, 0.048]	[0.011, 0.030]	[0.016, 0.131]
f_8	1.00 (**5**)	0.23(**1**)	0.36(**2**)	0.75(**3**)	1.93(**7**)	0.93(**4**)	1.70(**6**)
	[0.09, 1.73]	[0.11, 1.36]	[0.25, 2.01]	[0.18, 1.92]	[1.28, 3.02]	[0.32, 1.65]	[0.182, 2.68]
f_9	667.4(**3**)	848.1(**5**)	1859.5(**7**)	1132.0(**6**)	714.7(**4**)	8.4 (**1**)	584.2 (**2**)
	[512.0, 1169.1]	[249.9, 1669.0]	[1027.0, 2384.0]	[360.9, 2072.5]	[243.6, 1139.6]	[1.7, 43.3]	[523.0, 1282.7]
f_{10}	116.44 (**7**)	41.39(**5**)	39.56(**4**)	23.00(**3**)	22.63(**2**)	8.93(**1**)	94.57 (**6**)
	[3.70, 210.79]	[3.21, 139.62]	[7.08, 148.88]	[5.34, 152.50]	[2.14, 70.02]	[1.50, 53.72]	[7.83, 529.45]
f_{11}	0.338 (**4**)	0.03(**1**)	0.067(**2**)	0.170(**3**)	1.030(**6**)	10.067(**7**)	0.900(**5**)
	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 3]	[10, 11]	[0, 4]
f_{12}	0.999 (**1**)	1.001(**2**)	1.030(**6**)	1.002(**3**)	1.030(**6**)	1.008(**5**)	1.002 (**3**)
	[0.9994, 0.9996]	[0.999, 1.002]	[1.002, 1.184]	[1.000, 1.025]	[1.012, 1.173]	[1.001, 1.021]	[0.999, 1.087]
Averaged rank	4.33	3.25	3.83	4.00	3.92	3.08	5.17
Final rank	6	2	3	5	4	1	7

and $p_m = 0.05$. Let M be the number of parameters participated in the division phase and $N = 2^{\lfloor \log_2(M+1) \rfloor} - 1$. How to effectively formulate and solve real-world applications of LPOP using IEA with various variants of IGC can be referred in [18]–[20], and[25].

A. Performance Comparisons on LPOPs

KrishnaKumar *et al.* [2] proposed three approaches, stochastic GA, sensitivity GA, and GA-local search, and provided reasonable success on LPOPs using a multimodal function f_1 with no interaction among $D = 100$ parameters

$$\max . f_1 = -\sum_{i=1}^{D} \left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right) \right] \tag{5}$$

where parameters $x_i \in [3, 13]$. The performances of the above-mentioned three methods and a simple genetic algorithm (SGA) are obtained from [2], as shown in Table III. IEA additionally compares with the following EAs: BOA [24], a simple GA with elitist strategy using one-point crossover (OEGA), simulated annealing GA (SAGA) [30], GA with adaptive probabilities of crossover and mutation (AGA) [31], and OGA/Q [8]. Let N_{eval} be the number of fitness evaluations. To directly compare with the best one of the three proposed GAs in [2], i.e., the stochastic GA, we specify $N_{\text{pop}} = 30$, chromosome length 500, and the stopping condition using $N_{\text{eval}} = 12\,000$ for the compared EAs, except BOA and OGA/Q. For obtaining high performance by using the suggested moderate population size, BOA uses $N_{\text{pop}} = 500$ [24] and OGA/Q uses $N_{\text{pop}} = 200$ [8]. The average performances of all compared EAs using ten independent runs are shown in Table III.

TABLE VI
MEAN FITNESS VALUES AND RANKS FOR FUNCTIONS WITH $D = 100$. THE MINIMAL AND MAXIMAL FITNESS VALUES OF SOLUTIONS IN 30 RUNS ARE SHOWN IN BRACKETS

Test function	IEA	OEGA	UEGA	TEGA	BLXGA	OGA
f_1	120.44 (**1**)	90.52(**3**)	89.36(**4**)	87.86(**6**)	88.9(**5**)	116.71 (**2**)
	[120.04, 120.77]	[85.94, 95.11]	[84.66, 95.52]	[76.46, 89.92]	[82.15, 92.24]	[114.57, 118.11]
f_2	153.15 (**1**)	94.59(**3**)	89.49(**5**)	94.27(**4**)	74.47(**6**)	139.71 (**2**)
	[149.23, 164.65]	[79.88, 106.96]	[76.63, 95.52]	[84.96, 106.55]	[60.72, 78.87]	[134.44,147.61]
f_3	621 (**1**)	55027(**4**)	56647(**5**)	60252(**6**)	26320(**3**)	6107 (**2**)
	[535, 824]	[36021, 71917]	[41103, 67435]	[40471, 72663]	[20553, 29867]	[4926, 7929]
f_4	213.46 (1)	795.75(**5**)	819.65(**6**)	791.56(**4**)	763.42(**3**)	367.51 (**2**)
	[181.82, 257.95]	[724.08, 978.49]	[752.34, 891.67]	[698.53, 860.99]	[693.14, 846.44]	[313.33, 404.96]
f_5	1.60 (**1**)	145.52(**4**)	151.43(**5**)	154.97(**6**)	67.89(**3**)	14.96 (**2**)
	[1.06, 3.13]	[123.03, 169.34]	[114.02, 190.93]	[105.51, 189.85]	[62.58, 93.28]	[10.27, 19.68]
f_6	131.31 (**1**)	76.72(**3**)	74.50(**5**)	76.06(**4**)	51.53(**6**)	115.48 (**2**)
	[122.33, 141.66]	[63.88, 80.79]	[63.56, 81.19]	[57.32, 79.72]	[45.14, 56.37]	[106.22,121.56]
f_7	0.65(**1**)	3.86(**4**)	3.94(**5**)	3.82(**3**)	5.45(**6**)	1.63 (**2**)
	[0.55, 0.82]	[2.49, 7.61]	[3.33, 4.66]	[1.02, 8.36]	[5.13, 6.29]	[1.22, 1.72]
f_8	3.69 (**1**)	16.93(**5**)	16.83(**4**)	16.97(**6**)	14.35(**3**)	9.47 (**2**)
	[2.71, 4.81]	[16.44, 17.86]	[16.08, 17.40]	[16.32, 17.51]	[13.86, 15.15]	[7.31, 11.48]
f_9	8011 (**1**)	24645(**4**)	24160(**3**)	24723(**5**)	24794(**6**)	12294 (**2**)
	[6743, 9479]	[24319, 24834]	[24114, 24207]	[24517, 24832]	[24737, 24893]	[9118, 14359]
f_{10}	2081(**1**)	96556(**5**)	89514(**4**)	97990(**6**)	16086(**3**)	5282 (**2**)
	[879, 2803]	[50898, 138273]	[54798, 118653]	[58985, 170483]	[12879, 26941]	[3780, 8226]
f_{11}	43.94 (**1**)	159.90(**4**)	154.37(**3**)	172.03(**5**)	233.63(**6**)	65.19(**2**)
	[36, 51]	[130, 176]	[130, 178]	[141, 203]	[218, 248]	[51, 73]
f_{12}	32.86 (**1**)	511.93(**4**)	1002.00(**6**)	537.09(**5**)	237.84(**3**)	48.25 (**2**)
	[14.10, 74.85]	[389.73, 677.39]	[906, 1225]	[394.73, 645.26]	[171.95, 282.96]	[12.30, 63.22]
Averaged rank	1.00	4.00	4.58	5.00	4.42	2.00
Final rank	1	3	5	6	4	2

For BOA, a commonly used constraint restricts the networks to have at most $\kappa(\kappa = 4$ in this test) incoming edges into each node. The truncation selection with $\tau = 50\%$ is used. The solution of BOA is only superior to those of SGA, OEGA, AGA, and SAGA. BOA averagely takes about 1,115.95 s for a single run, while the other EAs only take less than 1 s using the CPU AMD 800 MHz. It reveals that BOA is hard to solve LPOPs within a limited amount of computation time.

OGA/Q applies the orthogonal design to generate an initial population of points that are scattered uniformly over the feasible solution space, so that the algorithm can evenly scan the search space once to locate good points for further exploration in subsequent iterations [8]. OGA/Q divides the feasible solution space along one of all dimensions into S subspaces, and then employs an $L_Q 2(Q^{Q+1})$ of Q levels with Q^2 rows and $Q + 1$ columns to generate Q^2 points in each subspace. In this test, $S = 10$ and $Q = 101$. OGA/Q generates SQ^2 individuals and selects $N_{\text{pop}} = 200$ combinations having the best fitness performance as the initial population. Therefore, OGA/Q takes $N_{\text{eval}} = SQ^2 = 102\,010$ to generate an initial population. The best solution among the initial population is 110.715. OGA/Q uses an orthogonal crossover with quantization which acts on two parents. The crossover operator quantizes the solution space defined by these parents into a finite number of points, and then applies orthogonal design to select a small, but representative sample of points as the potential offspring. For obtaining the optimum 121.598, OGA/Q using 1000 iterations ($N_{\text{eval}} = 388\,010$) has a mean fitness value 121.594. OGA/Q can obtain an accurate solution but a large number of fitness evaluations are needed.

IEA obtains the best mean solution 120.890, compared with all EAs. For evaluating the ability to obtain an optimal or near-optimal solution, the chromosome length is increased to 1400. Using the stopping condition that the global optimal solution

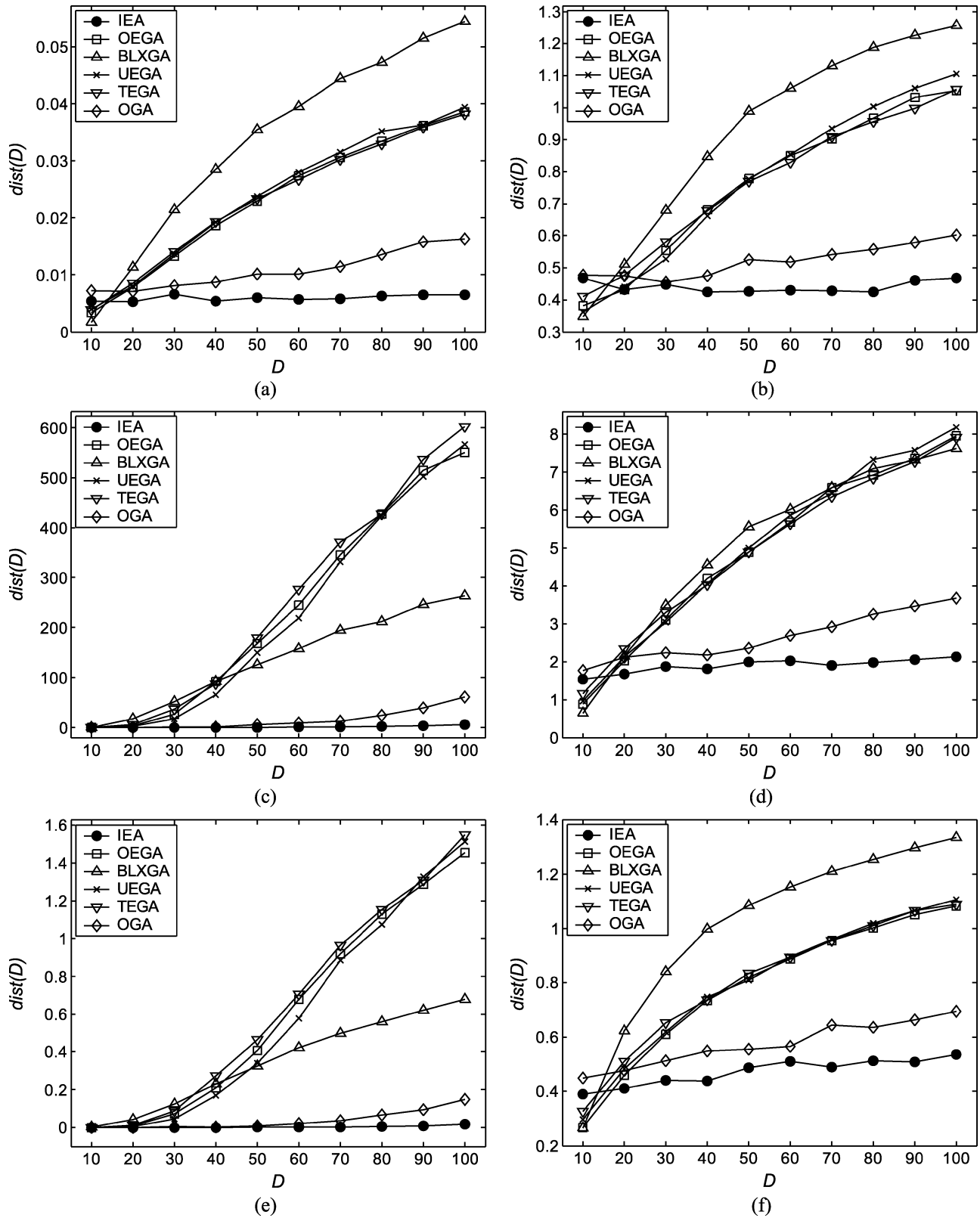

Fig. 3. Comparisons of various GAs using $\mathrm{dist}(D)$ curves for functions $f_1, f_2, \ldots$, and f_{12} in (a), (b), ..., and (l), respectively.

121.598 is obtained, IEA averagely spends 58 156 fitness evaluations. The experiment shows that IEA can efficiently search for a very good solution to the LPOP with low epistasis.

B. Test Functions With Various Dimensions

To evaluate the efficiency of IGC for solving optimization problems with various dimensions, 12 benchmark functions

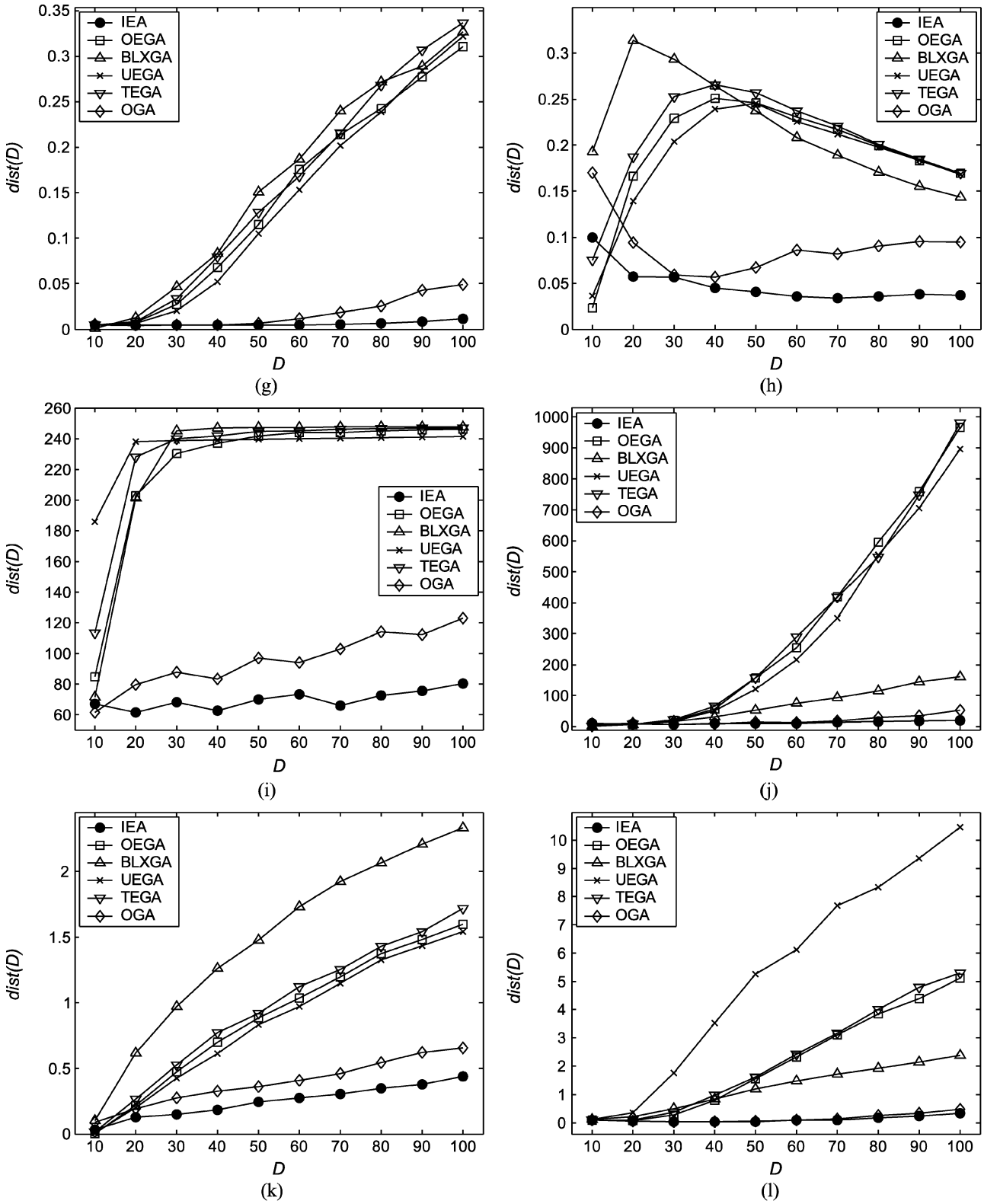

Fig. 3. (*Continued.*) Comparisons of various GAs using $\mathrm{dist}(D)$ curves for functions $f_1, f_2, \ldots$, and f_{12} in (a), (b), . . ., and (l), respectively.

gleaned from literature, including unimodal, and multimodal functions, as well as continuous and discontinuous functions, are tested in the experimental study. The test function, parameter domain, and global optimum for each function are listed in Table IV. Each parameter is encoded using 10 bits for all test functions except that functions f_9 and f_{12} use 24 bits. The dimensions of the twelve test functions vary on $D = 10, 20, \ldots, 100$ in the experiments. For all compared EAs, $N_{\mathrm{pop}} = 10$, $p_s = 0.2$, $p_c = 0.8$, and $p_m = 0.05$. The stopping condition uses $N_{\mathrm{eval}} = 10\,000$.

To compare IGC with commonly used crossover operators without heuristics, one-point, two-point, uniform, BLX-α

TABLE VII
PERFORMANCE COMPARISONS OF OGA/Q AND IEA

Test functions	Mean number of function evaluations		Mean function value (Standard deviation)		Globally minimal function value
	OGA/Q	IEA	OGA/Q	IEA	
g_1	302,166	54,706	-12569.4537 (6.447×10^{-4})	-12569.49 (6.079×10^{-3})	-12569.5
g_2	224,710	8,420	0 (0)	0 (0)	0
g_3	112,421	8,420	4.440×10^{-16} (3.989×10^{-17})	0 (0)	0
g_4	134,000	16,840	0 (0)	0 (0)	0
g_5	134,556	128,260	6.019×10^{-6} (1.159×10^{-6})	1.442299×10^{-6} (4.5325×10^{-6})	0
g_6	134,143	163,638	1.869×10^{-4} (2.615×10^{-5})	1.1351×10^{-5} (1.5401×10^{-20})	0
g_7	302,773	218,068	-92.83 (2.615×10^{-2})	-97.64365 (1.1269×10^{-1})	-99.2784
g_8	190,031	402,064	4.672×10^{-7} (1.293×10^{-7})	1.6999×10^{-7} (1.6586×10^{-7})	0
g_9	245,930	184,711	-78.300 (6.288×10^{-3})	-78.33232 (6.3530×10^{-6})	-78.33236
g_{10}	167,863	102,020	7.520×10^{-1} (1.140×10^{-1})	0 (0)	0
g_{11}	112,559	16,840	0 (0)	0 (0)	0
g_{12}	112,652	8,420	6.301×10^{-3} (4.069×10^{-4})	0 (0)	0
g_{13}	112,612	8420	0 (0)	0 (0)	0
g_{14}	112,576	16,840	0 (0)	0 (0)	0
g_{15}	112,893	16,840	0 (0)	0 (0)	0

[32], and orthogonal [33] crossover operators are used. Due to the long computation time, BOA is tested on the twelve test functions with $D = 10$ only. The compared EAs with elitist strategy and the associated crossover are denoted as (IEA, IGC), (OEGA, one-point), (TEGA, two-point), (UEGA, uniform), (BLXGA, BLX-α), and (OGA, orthogonal).

OGA divides each parent string into k parts, sample these parts from n parents based on the m combinations in an $L_m(n^k)$ to produce m binary strings, and then select j of them to be the offspring. In this experiment, we use the OA $L_9(3^4)$ and $j = 2$.

The comparisons of average performance using 30 independent runs for $D = 10$ and $D = 100$ are listed in Tables V and VI, respectively. To illustrate the performance comparisons on various numbers of parameters, we use a distance function $\mathrm{dist}(D)$ for describing the mean distance between the optimal solution $f_{\mathrm{opt}}(D)$ and the obtained best solution $f_{\mathrm{best}}(D)$ for one parameter as follows:

$$\mathrm{dist}(D) = \frac{|f_{\mathrm{opt}}(D) - f_{\mathrm{best}}(D)|}{D}. \tag{6}$$

The results of average performance for all test functions with $D = 10, \ldots, 100$ are shown in Fig. 3.

Table V reveals that BOA and OEGA have the best performances with final ranks 1 and 2, respectively. However, BOA is only slightly superior to OEGA according to the averaged rank, but the computation time of BOA is much longer than that of OEGA. On the other hand, IEA and OGA have the worst performances with final ranks 6 and 7, respectively. The two OA-based algorithms take more fitness evaluations for one recombination operation than conventional EAs but cannot effectively take advantage of OA experiments in solving the small-scale optimiza-

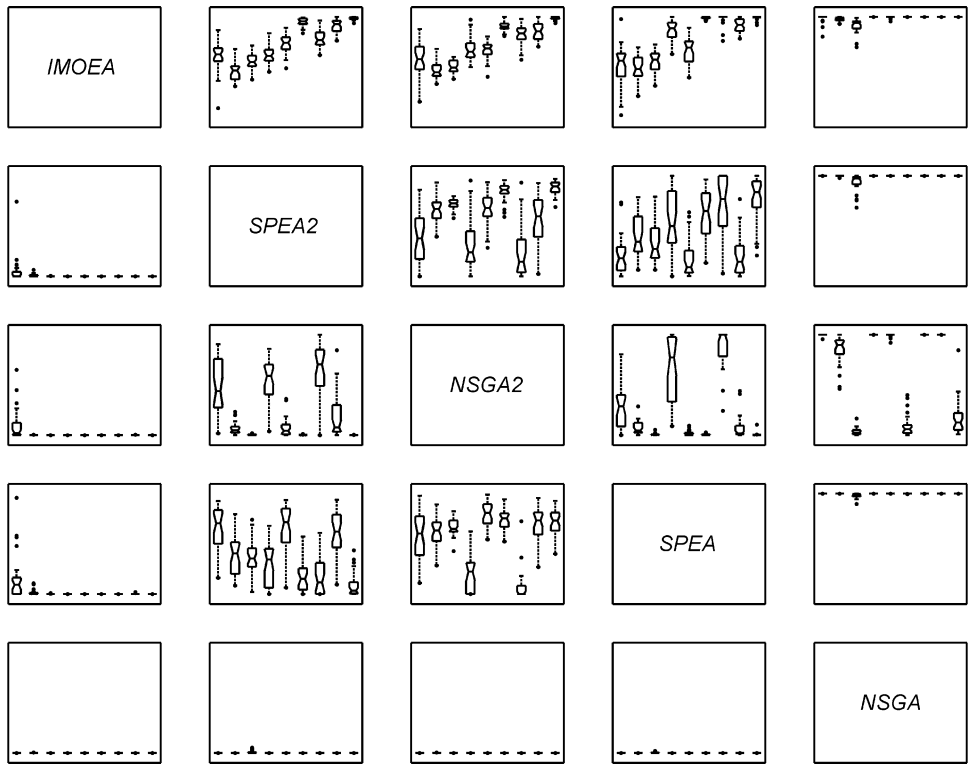

Fig. 4. Box plots based on the cover metric for multiobjective 0/1 knapsacks problems. The box plot relates to {(2, 250), (3, 250), (4, 250), (2, 500), (3, 500), (4, 500), (2, 750), (3, 750), (4, 750)} (knapsack, item) problems from left to right. Each rectangle refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives nine box plots representing the distribution of the cover metric $C(A, B)$. The scale is 0 at the bottom and 1 at the top per rectangle.

tion problems with $D = 10$. Note that IEA is superior to OGA for most test functions. From the simulation results, it can be found that SGA with elitist strategy using one-point crossover (OEGA) performs well in solving small parameter optimization problems.

Table VI reveals that IEA and OGA having the averaged ranks 1.00 and 2.00, respectively, outperform all other EAs. In fact, it can be found from Fig. 3 that IEA and OGA have averaged ranks 1.00 and 2.00 not only for $D = 100$, but also for $D = 20, 30, \ldots, 90$. These simulation results reveal the following.

1) The OA-based recombination operations of IEA and OGA using sampling representative candidate solutions perform well in solving optimization problems with a large number of parameters.
2) IGC performs better than the orthogonal crossover of OGA. It reveals that the reasoned combination using factor analysis is effective in IGC.
3) The mean distance value $\mathrm{dist}(D)$ of IEA slightly increases, while D increases from 10 to 100, compared with other EAs. IEA can effectively solve LPOPs in a reasonable among of computation time.
4) Fig. 3 reveals that the curves of OEGA, TEGA, and UEGA form a group and the value $\mathrm{dist}(D)$ obviously increases, while D increases. The performance of BLXGA is similar to those of the three SGAs. It means that the conventional SGA is impractical for solving LPOPs.

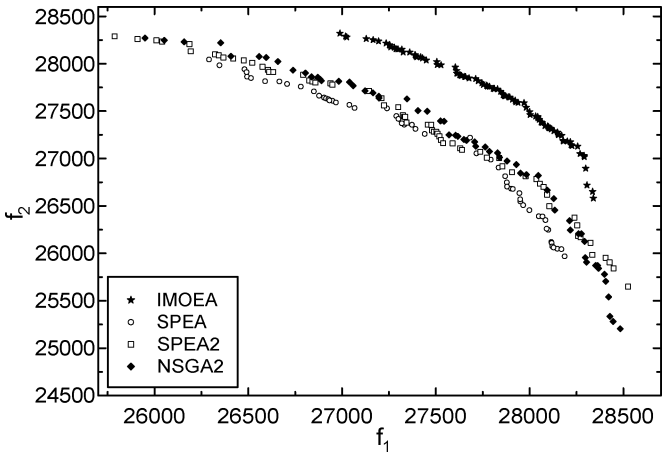

Fig. 5. Pareto fronts of the problem with two knapsacks and 750 items.

C. Comparisons With OGA/Q

In Section V-B, it has been shown that IEA is superior to OGA with a randomly generated initial population. In this section, IEA is compared with OGA/Q by examining the effectiveness of IGC with an OA-based initial population. For comparison with OGA/Q, the test functions are the same as those in [8].

The OA-based initial population of IEA is similar to that of OGA/Q, described as follows. After generating Q^2 points in each subspace like the method of OGA/Q, we further use

factor analysis of OED to estimate main effects of all parameters with Q levels and generate an additional point as an individual for each subspace using a reasoned combination of all parameters with the best level. Therefore, IEA use $S(Q^2+1)$ fitness evaluations in generating an initial population. IEA uses $N_{\text{pop}} = 50 + S$, where $S = 20$ for g_1, g_4, g_{11},g_{14}, g_{15}, and $S = 10$ for the others according to the domain size as those of OGA/Q. A parameter is encoded using 14 bits. The stopping condition is that when the given optimal value is reached or the best solution cannot be effectively improved in successive 30 iterations. The performance comparisons of IEA and OGA/Q using 50 independent runs are given in Table VII, where the reported results of OGA/Q are obtained from [8].

Let $P_{\text{OGA/Q}}$ and P_{IEA} be the initial populations generated using the methods of OGA/Q and IEA, respectively. Let I_{opt} be an individual with an optimal solution. All test functions can be categorized into four classes by carefully examining the simulation results as follows:

1) Functions g_2, g_4, g_{11}, g_{13}, g_{14}, g_{15}: I_{opt} exists in both $P_{\text{OGA/Q}}$ and P_{IEA}. Therefore, IEA and OGA/Q can obtain the optimal solution with a zero standard deviation.
2) Functions g_3, g_{10}, g_{12}: I_{opt} exists in P_{IEA} but not in $P_{\text{OGA/Q}}$. This means that IGC using factor analysis can efficiently collect optimal values of individual parameters, which are distributed in the population, to generate an I_{opt}.
3) Functions g_1, g_7, g_9: I_{opt} does not exist in either $P_{\text{OGA/Q}}$ or P_{IEA}, and there exists no interaction among parameters. IEA performs better than OGA/Q in terms of both solution quality and computation cost.
4) Functions g_5, g_6, g_8: I_{opt} does not exist in either $P_{\text{OGA/Q}}$ or P_{IEA}, and there exists interactions among parameters. IEA performs better than OGA/Q. For reducing the degree of epistasis of g_8, IEA replaces the square operation with an absolute one using the following fitness function such that IEA can easily obtain the optimal solution to g_8, where $x_j = \omega_j$, $j = 1, \ldots, D$, and $D = 100$:

$$\min .\ G_8 = \sum_{i=1}^{D}\sum_{j=1}^{D}[|\chi_{ij}(\sin x_j - \sin \varpi_j)| + |\phi_{ij}(\cos x_j - \cos \varpi_j)|]. \quad (7)$$

VI. Performance Comparisons of IMOEA

The performance of IMOEA is evaluated by the multiobjective 0/1 knapsack problems and test functions. There are several state-of-the-art methods in the MOEA literature [27], [34]–[45]; we compared against the following subset of these: SPEA [27], SPEA2 [34], NSGA [35], NSGA2 [36], NPGA [37], and VEGA [38]. The cover metric of two nondominated solution sets obtained by two algorithms A and B is used for performance comparison [27]

$$C(A,B) = \frac{\text{the number of individuals in } B \text{ weakly dominated by } A}{\text{the number of individuals in } B}. \quad (8)$$

The value $C(A,B) = 1$ means that all individuals in B are weakly dominated by A. On the contrary, $C(A,B) = 0$ denotes that none of individuals in B is weakly dominated by A. Because the cover metric considers weak dominance relationship between two sets of algorithms A and B, $C(A,B)$ is not necessarily equal to $1 - C(B,A)$.

A. Multiobjective 0/1 Knapsack Problems

The multiobjective 0/1 knapsack problems have been fully described in [27], and the comparison results demonstrated that SPEA outperforms NSGA, VEGA, NPGA, HLGA [44], and FFGA [45]. Generally, each investigated problem consists of I knapsacks, a set of items, weight and profit associated with each item, and an upper bound for the capacity of each knapsack. The task of the multiobjective 0/1 knapsack problem is to find a subset of items, which can maximize the profits of I knapsacks.

The multiobjective 0/1 knapsack problems are formulated as follows. Given a set of I knapsacks and a set of J items, with the following:

$p_{i,j}$ profit of item j according to knapsack i;
$w_{i,j}$ weight of item j according to knapsack i;
c_i capacity of knapsack i;

find a vector $X = [x_1, x_2, \ldots, x_J]^T \in \{0,1\}^J$, that

$$\max .f_i(x) = \sum_{j=1}^{J} p_{i,j} \cdot x_j, \quad i = 1, \ldots, I$$
$$\text{s.t.} \quad \sum_{j=1}^{J} w_{i,j} \cdot x_j \le c_i, \quad i = 1, \ldots, I \quad (9)$$

where $p_{i,j}$ and $w_{i,j}$ are random integers in the interval [10, 100], and $c_i = (1/2)\sum_{j=1}^{J} w_{i,j}$.

The test data sets are available from the authors [27]. The parameter settings of NSGA2 and SPEA2 are the same as those in [27]. The parameter settings of IMOEA are listed in Table VIII. The chromosomes of IMOEA are encoded uses a binary string of J bits, which are divided into $N = 15$ gene segments. A greedy repair method adopted in [27] is applied that the repair algorithm removes items from the infeasible solutions step by step until all capacity constraints are fulfilled. The order in which the items are deleted is determined by the maximum profit/weight ratio per item. Thirty independent runs of IMOEA are performed per test problem using the same number N_{eval} as that of SPEA. The test problems and reported results of SPEA are directly gleaned from the authors' website.

The direct comparisons of each independent run between IMOEA and all compared EAs based on the cover metric from 30 runs are depicted using box plots, as shown in Fig. 4. A box plot provides an excellent visual result of a distribution. The box stretches from the lower hinge (defined as the 25th percentile) to the upper hinge (the 75th percentile) and, therefore, contains the middle half of the scores in the distribution. The median is shown as a line across the box. For a typical example, Pareto fronts of the problem with two knapsacks and 750 items merged from 30 runs for IMOEA, SPEA, SPEA2, and NSGA2

TABLE VIII
PARAMETER SETTINGS OF MULTIOBJECTIVE EAS. FOR THE DETAIL PARAMETER SETTINGS OF THE COMPARED EAS, PLEASE REFER TO [26] AND [27]

Problems	Algorithm	N_{pop}	N_{Emax}	N	parameters	N_{eval} ($\times 10^3$)
0/1 knapsack problems	IMOEA	50	50	15	p_s=0.2, p_c=0.8, p_m=0.01	J | 250 | 500 | 750 I=2 | 75 | 100 | 125 I=3 | 100 | 125 | 150 I=4 | 125 | 150 | 175
	NSGA SPEA NSGA2 SPEA2	[27]	[27]	NA	[27]	
ZDT1-6	IMOEA	30	30	63	p_s=0.2, p_c=0.6, p_m=0.01	25
	VEGA NSGA NPGA SPEA NSGA2 SPEA2	[26]	[26]	NA	[26]	

are shown in Fig. 5. Figs. 4 and 5 reveal the superiority of IMOEA over the compared EAs in terms of robustness and nondominated solution quality. The experimental results reveal that IMOEA outperforms SPEA, SPEA2, and NSGA2, especially for problems with a large number of items (parameters). IMOEA also performs well for various numbers of knapsacks (objectives).

B. Test Functions With Various Problem Features

Deb [46] has identified several problem features that may cause difficulties for multiobjective algorithms in converging to the Pareto-optimal front and maintaining population diversity in the current Pareto front. These features are multimodality, deception, isolated optima and collateral noise, which also cause difficulties in single-objective EAs. Following the guidelines, Zitzler *et al.* [26] constructed six test problems involving these features, and investigated the performance of various popular multiobjective EAs. The empirical results demonstrated that SPEA outperforms NSGA, VEGA, NPGA, HLGA, and FFGA in small-scale problems.

Each of the test problems is structured in the same manner and consists of three functions f_1, g, h [41]

$$\begin{aligned} &\text{minimize } T(X) = (f_1(x_1), f_2(X)) \\ &\text{s.t.} \quad f_2(X) = g(x_2, \ldots, x_m) \\ &\qquad\qquad \cdot h\left(f_1(x_1), g(x_2, \ldots, x_m)\right), \text{ where} \\ &X = [x_1, x_2, \ldots, x_m]^T \end{aligned} \tag{10}$$

where f_1 is a function consisted of the first decision variable x_1 only, g is a function of the remaining $m-1$ parameters, and the two parameters of the function h are the function values of f_1 and g. Six test problems $\text{ZDT}_1, \ldots, \text{ZDT}_6$ can be retrieved from [41].

There are m parameters in each test problem. Each parameter in chromosomes is represented by 30 bits, except the parameters $x_2, \ldots, x_m$ are represented by 5 bits for the deceptive problem ZDT_5.

The experiments in Zitzler's study indicate that the test problems ZDT_4, ZDT_5, and ZDT_6 cause difficulties to evolve a well-distributed Pareto-optimal front. In their experiments, the reports are absent about the test problems with a large number of parameters. As a result, the extended test problems with a large number of parameters ($m = 63$) are further tested in order to compare the performance of various algorithms in solving large MOOPs. Thirty independent runs were performed using the same stopping criterion as that of SPEA, where $N_{\text{eval}} = 25\,000$. The parameter settings of various algorithms are listed in Table VIII. VEGA, NPGA, NSGA, SPEA, NSGA2, and SPEA2 are the same as those in [26], summarized as follows: $p_c = 0.8$, $p_m = 0.1$, $t_{\text{dom}} = 10$, $\sigma_{\text{share}} = 0.488\,62$, and $N_{\text{pop}} = 100$. The population size and the external population size of SPEA are 80 and 20, respectively. For the test problem ZDT_5, the sharing factor σ_{share} of NSGA is 34. The parameter settings of IMOEA are: $N_{\text{pop}} = 30$, $N_{\text{Emax}} = 30$, $p_s = 0.2$, $p_c = 0.6$, $p_m = 0.01$, and $N = m = 63$.

The nondominated solutions merged from 30 runs of VEGA, NPGA, NSGA, NSGA2, SPEA, SPEA2, and IMOEA in the objective space are reported, as shown in Figs. 6–11, and the curve in each figure is the Pareto-optimal front of each test problem except ZDT_3. The direct comparisons of each independent run between IMOEA and all compared EAs based on the cover metric for 30 runs are depicted in Fig. 12.

For test problems ZDT_1, ZDT_2, and ZDT_3, IMOEA, SPEA2, and NSGA2 can obtain well-distributed Pareto fronts, and the Pareto fronts of IMOEA for ZDT_1 and ZDT_2 are very close to the Pareto-optimal fronts. For the multimodal test problem ZDT_4, only IMOEA can obtain a better Pareto front which is much closer to the Pareto-optimal front, compared with the other algorithms. The well-distributed nondominated solutions resulted from that IGC has well-distributed by-products which are nondominated solutions at that time. For ZDT_5, IMOEA, SPEA2, and NSGA2 obtained widely distributed

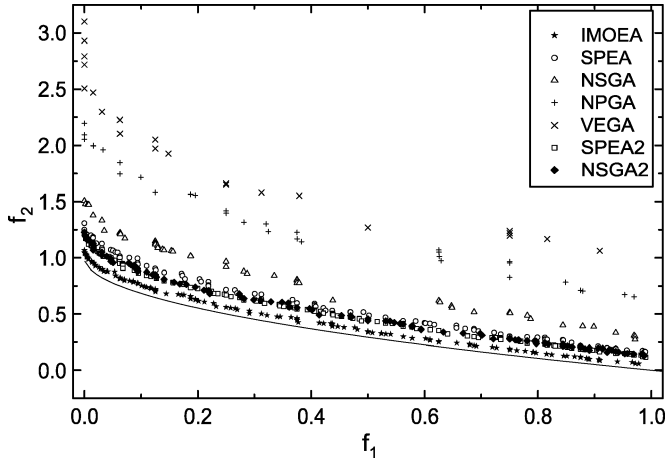

Fig. 6. Convex test problem ZDT_1.

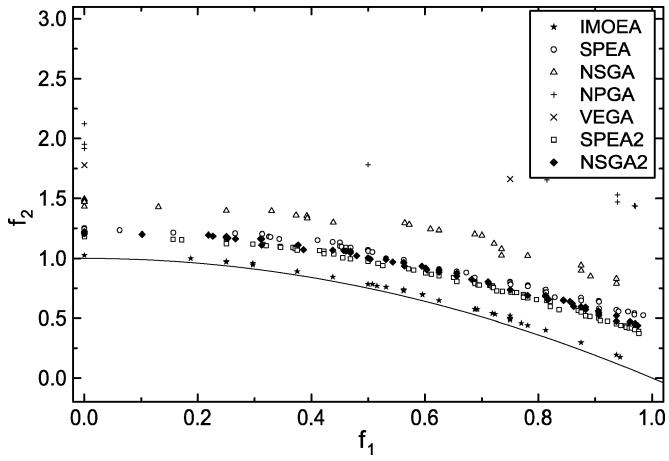

Fig. 7. Nonconvex test problem ZDT_2.

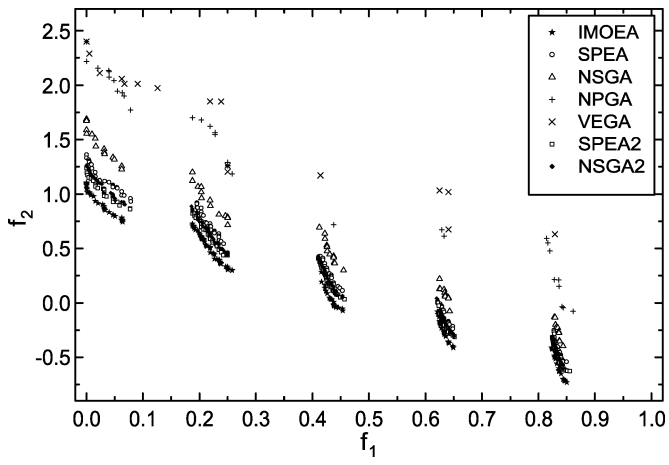

Fig. 8. Discrete test problem ZDT_3.

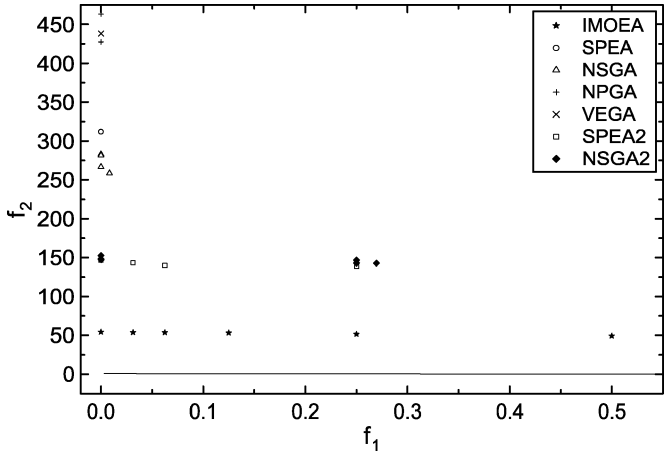

Fig. 9. Multimodal test problem ZDT_4.

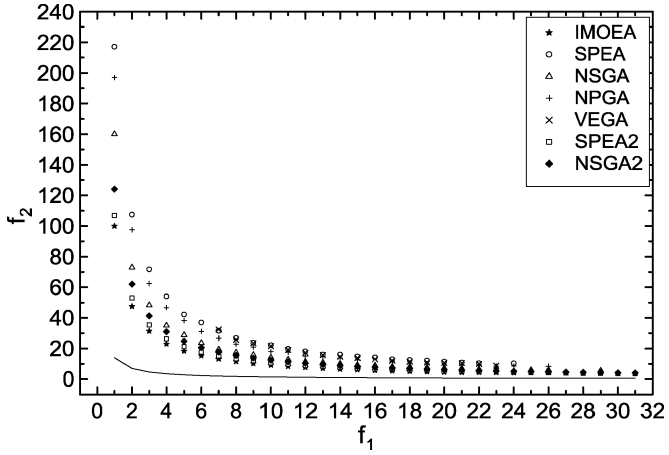

Fig. 10. Deceptive test problem ZDT_5.

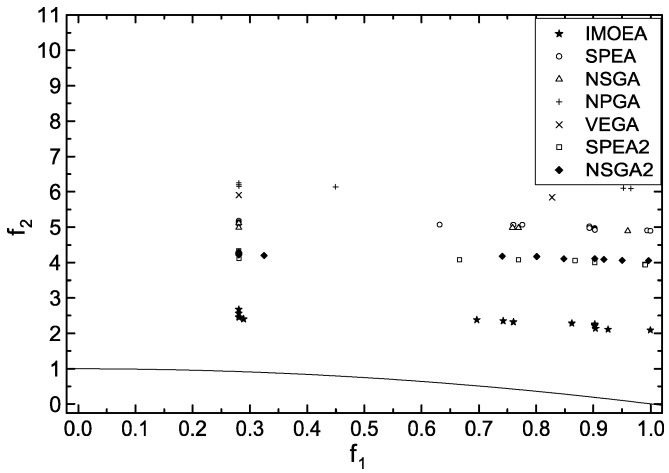

Fig. 11. Nonuniform test problem ZDT_6.

Pareto fronts, while VEGA failed. The comparisons for ZDT_5 indicate that IMOEA is the best algorithm; the next ones are SPEA2, NSGA2, NSGA, and NPGA, while SPEA and VEGA are the worst ones. For ZDT_6, IMOEA also obtained a widely distributed front and IMOEA's solutions dominate all the solutions obtained by the other algorithms. Finally, it can be observed from [26] and our experiments that when the number of parameters increases, difficulties may arise in evolving a well-distributed nondominated front. Moreover, it is observed that VEGA obtained some excel solutions in the objective f_1 in some runs of ZDT_2, ZDT_3, and ZDT_6, as shown in Fig. 12. This phenomenon agrees with [8] and [47] that VEGA may converge to individual champion solutions only.

As shown in Figs. 6–12, the quality of nondominated solutions obtained by IMOEA is superior to those of all compared

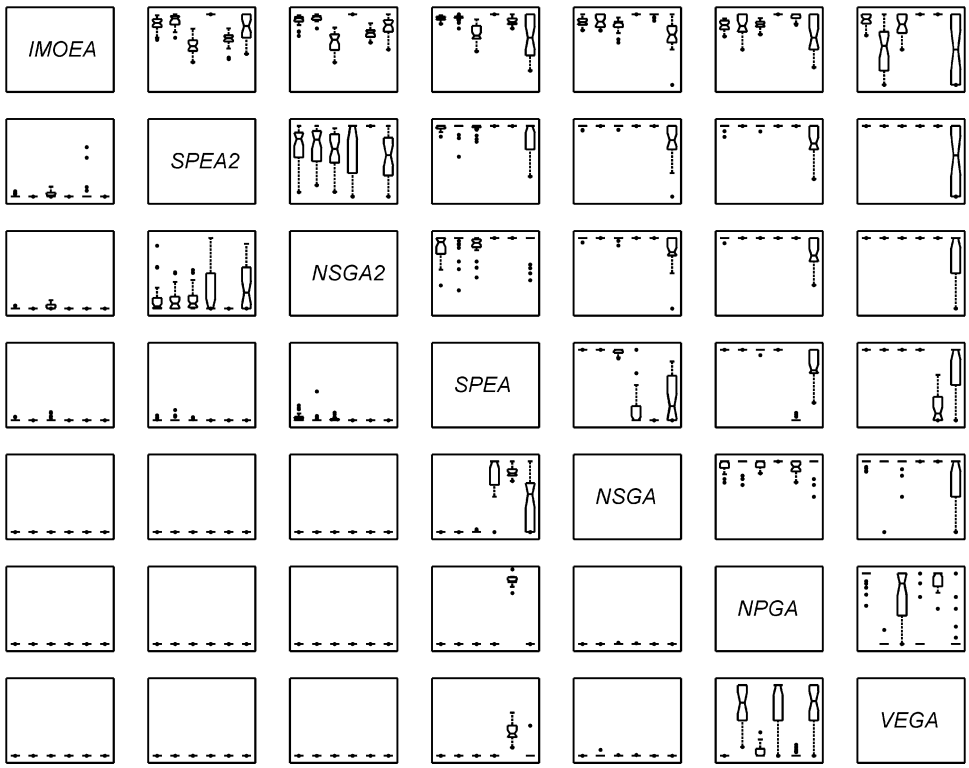

Fig. 12. Box plots based on the cover metric for multiobjective parametric problems. The leftmost box plot relates to ZDT1, the rightmost to ZDT6. Each rectangle refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives six box plots representing the distribution of the cover metric $C(A, B)$. The scale is 0 at the bottom and 1 at the top per rectangle.

EAs in terms of the number of nondominated solutions, the distance between the obtained Pareto front and Pareto-optimal front, and the distribution of nondominated solutions. Fig. 12 reveals that the nondominated solutions obtained by IMOEA dominate most of the nondominated solutions obtained by all compared EAs. On the other hand, only few nondominated solutions of IMOEA are dominated by the nondominated solutions of all compared EAs.

VII. Conclusion

In this paper, we have proposed two intelligent evolutionary algorithms IEA and IMOEA using a novel intelligent gene collector IGC to solve large parameter optimization problems (LPOPs) with single and multiobjective functions, respectively. IGC uses a divide-and-conquer mechanism based on orthogonal experimental design (OED) to solve LPOPs. Since OED is advantageous for the problems with weak interactions among parameters and IEA/IMOEA works without using linkage identification, it is essential to encode parameters into chromosomes such that the degree of epistasis can be minimized. Generally, the problem's domain knowledge is necessarily incorporated to effectively achieve this goal. Furthermore, maintaining feasibility of all candidate solutions corresponding to the conducted and reasoned combinations of OED can enhance the accuracy of factor analysis for constrained problems. Modifying genetic operator and repairing strategies can be conveniently used for maintaining feasibility. IMOEA works well for efficiently finding a set of Pareto-optimal solutions to a generalized multiobjective optimization problem with a large number of parameters. IMOEA is powerful based on the abilities of the proposed GPSIFF and IGC. We believe that the auxiliary techniques, which can improve performance of conventional EAs, can also improve performances of IEA and IMOEA. It is shown empirically that IEA and IMOEA have high performance in solving benchmark functions comprising many parameters, as compared with some existing EAs. Due to its simplicity, theoretical elegance, generality, and superiority, IEA and IMOEA can be most widely used for solving real-world applications of LPOPs.

Acknowledgment

The authors would like to thank the editor and the anonymous reviewers for their valuable remarks and comments that helped us to improve the contents of this paper.

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. New York: Oxford, 1997.

[2] K. KrishnaKumar, S. Narayanaswamy, and S. Garg, "Solving large parameter optimization problems using a genetic algorithm with stochastic coding," in *Genetic Algorithms in Engineering and Computer Science*, G. Winter, J. Périaux, M. Galán, and P. Cuesta, Eds. New York: Wiley, 1995.

[3] D. Thierens, D. E. Goldberg, and Â. G. Pereira, "Domino convergence, drift, and the temporal-salience structure of problems," in *Proc. 1998 IEEE Int. Conf. Evolutionary Computation*, 1998, pp. 535–540.

[4] S. H. Park, *Robust Design and Analysis for Quality Engineering*. London, U.K.: Chapman & Hall, 1996.
[5] T. P. Bagchi, *Taguchi Methods Explained: Practical Steps to Robust Design*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[6] A. S. Hedayat, N. J. A. Sloane, and J. Stufken, *Orthogonal Arrays: Theory and Applications*. New York: Springer-Verlag, 1999.
[7] A. Dey, *Orthogonal Fractional Factorial Designs*. New York: Wiley, 1985.
[8] Y. W. Leung and Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 41–53, Feb. 2001.
[9] G. Taguchi, *Introduction to Quality Engineering: Designing Quality Into Products and Processes*. Tokyo, Japan: Asian Productivity Organization, 1986.
[10] M. S. Phadke, *Quality Engineering Using Robust Design*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[11] A. Kunjur and S. Krishnamurty, "A robust multicriteria optimization approach," *Mech. Mach. Theory*, vol. 32, no. 7, pp. 797–810, 1997.
[12] D. M. Osborne and R. L. Armacost, "Review of techniques for optimizing multiple quality characteristics in product development," *Comput. Ind. Eng.*, vol. 31, no. 1–2, pp. 107–110, 1996.
[13] J. Antony, "Multiresponse optimization in industrial experiments using Taguchi's quality loss function and principal component analysis," *Int. J. Qual. Reliab. Eng.*, vol. 16, no. 1, pp. 3–8, 2000.
[14] E. A. Elsayed and A. Chen, "Optimal levels of process parameters for products with multiple characteristics," *Int. J. Prod. Res.*, vol. 31, no. 5, pp. 1117–1132, 1993.
[15] A. A. Song, A. Mathur, and K. R. Pattipati, "Design of process parameters using robust design techniques and multiple criteria optimization," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 1437–1446, Dec. 1995.
[16] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple Criteria Decision Making, Theory and Applications*, A. P. Fandel and A. P. Gal, Eds. New York: Springer-Verlag, 1980.
[17] S. Rochet, "Epistasis in genetic algorithms revisited," *Inform. Sci.*, vol. 102, pp. 133–155, 1997.
[18] S.-Y. Ho and Y.-C. Chen, "An efficient evolutionary algorithm for accurate polygonal approximation," *Pattern Recognit.*, vol. 34, no. 12, pp. 2305–2317, 2001.
[19] H.-L. Huang and S.-Y. Ho, "Mesh optimization for surface approximation using an efficient coarse-to-fine evolutionary algorithm," *Pattern Recognit.*, vol. 36, no. 5, pp. 1065–1081, 2003.
[20] S.-Y. Ho, C.-C. Liu, and S. Liu, "Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm," *Pattern Recognit. Lett.*, vol. 23, no. 13, pp. 1495–1503, Nov. 2002.
[21] Q. Wu, "On the optimality of orthogonal experimental design," *Acta Math. Appl. Sinica*, vol. 1, no. 4, pp. 283–299, 1978.
[22] Z. Michalewicz, D. Dasgupta, R. G. Le Riche, and M. Schoenauer, "Evolutionary algorithms for constrained engineering problems," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 851–870, Sept. 1996.
[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[24] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. 1st Conf. GECCO-99*, 1999, pp. 525–532.
[25] S.-Y. Ho, H.-M. Chen, S.-J. Ho, and T.-K. Chen, "Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space," *IEEE Trans. Systems, Man, Cybern. B*, vol. 34, pp. 1031–1044, Apr. 2004.
[26] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjecctive evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[27] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strengthen Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257–271, Nov. 1999.
[28] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*. New York: Wiley, 2001.
[29] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evolut. Comput.*, vol. 8, no. 2, pp. 125–147, 2000.
[30] H. Esbensen and P. Mazumder, "SAGA: A unification of the genetic algorithm with simulated annealing and its application to macro-cell placement," in *Proc. IEEE Int. Conf. VLSI Design*, 1994, pp. 211–214.
[31] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Systems, Man, Cybern.*, vol. 24, pp. 656–667, Apr. 1994.
[32] L. J. Eshelman and J. D. Schaffer, *Real-Coded Genetic Algorithms and Interval Schemata. Foundation of Genetic Algorithm-2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993.
[33] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routine," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 53–62, Apr. 1999.
[34] E. Zitzler, M. Laumanns, and L. Thiele *et al.*, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Proc. EUROGEN 2001—Evolutionary Methods for Design, Optimization and Control With Applications to Industrial Problems*, K. C. Giannakoglou *et al.*, Eds., 2001, pp. 95–100.
[35] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, Apr. 2002.
[37] J. Horn, N. Nafplotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. IEEE Int. Conf. Evolutionary Computation*, 1994, pp. 82–87.
[38] J. D. Schaffer, "Multiobjective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, J. J. Grefenstette, Ed., Hillsdale, NJ, 1985, pp. 93–100.
[39] H. Ishibuchi and T. Murata, "A multiobjective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Systems, Man, and Cybern. C: Applications Reviews*, vol. 28, pp. 392–403, Aug. 1998.
[40] A. Jaszkiewicz, "Genetic local search for multiple objective combinatorial optimization," Inst. Comput. Sci., Poznan Univ. Technol., Poznan, Poland, Res. Rep., RA-014/98, Dec. 1998.
[41] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto-envelope based selection algorithm for multiobjective optimization," in *Proc. 6th Int. Conf. Parallel Problem Solving From Nature (PPSN VI)*, 1998, pp. 839–848.
[42] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, 2000.
[43] ——, "M-PAES: A memetic algorithm for multiobjective optimization," in *Proc. Congr. Evolutionary Computation*, San Diego, CA, July 16–19, 2000, pp. 325–332.
[44] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," *Struct. Opt.*, no. 4, pp. 99–107, 1992.
[45] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., San Mateo, CA, 1993, pp. 416–423.
[46] K. Deb, "Multiobjective genetic algorithms: Problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, 1999.
[47] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Int. J. Knowl. Inform. Syst.*, vol. 1, no. 3, pp. 269–308, 1999.

Shinn-Ying Ho (M'00) was born in Taiwan, R.O.C., in 1962. He received the B.S., M.S., and Ph.D. degrees in computer science and information engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1984, 1986, and 1992, respectively.

From 1992 to 2004, he was a Professor in the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, R.O.C. He is currently a Professor in the Department of Biological Science and Technology, Institute of Bioinformatics, National Chiao Tung University, Hsinchu, Taiwan, R.O.C. His research interests include evolutionary algorithms, image processing, pattern recognition, bioinformatics, data mining, virtual reality applications of computer vision, fuzzy classifier, large parameter optimization problems, and system optimization.

Li-Sun Shu (S'04) was born in Taiwan, R.O.C., in 1972. He received the B.A. degree in mathematics from Chung Yuan University, Taoyuan, Taiwan, in 1995 and the M.S. and Ph.D. degrees in information engineering and computer science from Feng Chia University, Taichung, Taiwan, R.O.C., in 1997 and 2004, respectively.

He is a Lecturer at the Overseas Chinese Institute of Technology, Taichung, Taiwan, R.O.C. His research interests include evolutionary computation, large parameter optimization problems, fuzzy systems, and system optimization.

Jian-Hung Chen (S'00) was born in Chai-Yi, Taiwan, R.O.C., in 1974. He received the B.A. degree in surveying engineering from National Cheng Kung University, Tainan, Taiwan, in 1997 and the Ph.D. degree in information engineering and computer science from Feng Chia University, Taichung, Taiwan, R.O.C., in 2004.

From 2001 to 2002, he was with the Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana–Champaign. His research interests include evolutionary computation, multiobjective optimization, pattern recognition, data mining, flexible manufacturing systems, resources planning, experimental design, and system optimization.

Dr. Chen was the recipient of the Taiwan Government Visiting Scholar Fellowship in 2000. He was the recipient of the IEEE Neural Networks Society Walter Karplus Research Grant (2003).