A Novel Crowding Genetic Algorithm and Its Applications to Manufacturing Robots

Chiu-Hung Chen, Tung-Kuan Liu, and Jyh-Horng Chou, *Senior Member, IEEE*

***Abstract*—A niche genetic algorithm (GA) based on a novel twin-space crowding (TC) approach is proposed for solving multimodal manufacturing optimization problems. The proposed TC method is designed in a parameter-free paradigm. That is, when cooperatively exploring solutions with GAs, it does not require prior knowledge related to the solution space to design additional problem-dependent parameters in the evolutionary process. This feature makes the proposed TC method suitable for assisting GAs in solving real-world engineering optimization problems involving intractable solution landscapes. A set of numerical benchmark functions is used to compare effectiveness and efficiency in the proposed TCGA, in different niche GAs, and in several evolutionary computation methods. The TCGA is then used to solve multimodal joint-space inverse problems in serial-link robots to compare its convergence performance with that of conventional methods that apply the sharing function. Finally, the TCGA is used to solve iterative collision-free design problems for linkage-bar robotic hands to demonstrate its effectiveness for generating diverse solutions during the design process.**

***Index Terms*—Crowding method, joint-space, multimodal optimization, niche genetic algorithm (GA).**

I. Introduction

ENGINEERING problems usually contain multiple optima in the solution space. Particularly, joint-space inverse problems encountered when designing robots with multiple degrees of freedom (DOFs) have been validated as multimodal problems [1], [2]. In some multimodal optimization problems, a simple genetic algorithm (GA) [3] cannot efficiently and simultaneously maintain multiple global or local optima. Therefore, the population is easily trapped in a limited number of solutions, which is a condition that results in premature solutions with no capability to obtain better results. Therefore, niche methods [4] have been developed to reduce genetic drift effects resulting from the replacement operator in the simple GA.

By adding scaling fitness or by changing the fitness competence rule, niche methods modify the simple GA by guiding convergence so that multiple peak solutions can be maintained in the search space [5]. The capability to locate multiple *loci* often gives niche GAs the robustness and effectiveness needed to explore optima in various multimodal optimization problems [6]–[8]. When used to solve optimization problems, however, most niche methods require prior knowledge such as the niche radius or the distance threshold [9], [10].

To address these limitations, this study proposes twin-space crowding (TC), a novel niche method designed in a parameter-free paradigm; that is, it does not require prior knowledge of practical optimization problems, and the proposed crowding mechanism does not require additional customized parameters. This feature makes the proposed TC method suitable for combined use with GAs for solving real-world optimization problems, in which the solution landscapes are often intractable. To evaluate its performance in solving multimodal problems, the set of numerical benchmark functions presented in Ratnaweera *et al.* [11] and Bakwad *et al.* [12] is used to compare convergence performance between the proposed TC genetic algorithm (TCGA) and other methods.

Additionally, to evaluate its applicability for solving practical engineering problems, the TCGA is used to solve joint-space inverse kinematics problems in two different mechanisms currently used in robotic manufacturing systems: serial-link robots (SLRs) and linkage-bar robotic hands. An SLR has an open-loop structure with multiple DOFs, whereas a linkage-bar robotic hand has a closed-loop structure with only a single DOF. Since these mechanisms have very different kinematical characteristics, solving their inverse problems provides a convincing validation of the solution capabilities of the proposed approach.

Compared with path planning problems, which have been intensively studied in the past [13]–[15], joint-space inverse problems in serial-link or linkage-bar robots are very different and harder to solve because the inverse problems specified in the world coordinates must be solved in the joint space where the mapping between the joint space and Cartesian space is nonlinear, singular, and multimodal [16], [17]. However, a literature review [2] shows only a few studies of the evolutionary solution multiplicity of inverse problems originating from multimodal mapping.

Therefore, the proposed TCGA is used to explore multiple inverse solutions for SLRs, and its convergence performance is compared with that of the algorithm developed in Kalra *et al.* [2], in which the sharing function [5] is applied. This study also considers the engineering design problem of a six-bar robotic

Manuscript received August 28, 2013; revised February 14, 2014; accepted March 27, 2014. Date of publication April 11, 2014; date of current version August 05, 2014. This work was supported by the National Science Council of Taiwan under Grant NSC 102-2221-E-244-013, Grant NSC 102-2221-E-151-021-MY3, and Grant NSC 102-2217-E-151-001-MY3. Paper no. TII-13-0578.

C.-H. Chen is with the Department of Information Technology, Kao Yuan University, Kaohsiung 82151, Taiwan.

T.-K. Liu is with the Institute of Engineering Science and Technology, National Kaohsiung First University of Science and Technology, Kaohsiung 824, Taiwan.

J.-H. Chou is with the Department of Electrical Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 807, Taiwan; with the Institute of Electrical Engineering, Department of Mechanical and Automation Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung 824, Taiwan; and also with the Department of Healthcare Administration and Medical Informatics, Kaohsiung Medical University, Kaohsiung 807, Taiwan (e-mail: choujh@nkfust.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2014.2316638

hand. The problem is to obtain a synthesis design for a six-bar structure that can consider obstacles in the manufacturing environment. In contrast with the relatively simple synthesis design for low-order four-bar linkage structures [18]–[20], the design problem considered here is to synthesize a high-order, highly constrained, and collision-free coupler for a single-DOF mechanism. Such problems are usually solved using classic precise-point methods [21], [22] to guide the coupler along the desired path, which limits design flexibility. The iterative design method proposed here uses TCGA to enable iterative exploration of multiple inverse collision-free solutions.

II. Twin-Space Crowding Genetic Algorithm

The literature showed many improvements in evolutionary optimization and the development of efficient solvers for engineering problems [23]–[27]. In the niche family of GAs, population diversity is improved using a set of algorithms to change the genetic schema in the population replacement phase.

Like a simple GA, a general niche GA first initializes and evaluates the population to select the best fitting chromosomes for the mating pool. It then applies crossover and mutation operators to generate and evaluate the new offspring. When it evolves a new generation, the general niche GA does not merely reserve individuals with the best fitness. It usually applies a specific replacement operator to assist in selecting new parents.

A. Niche Family

One of the earliest niche mechanisms is preselection [28], in which an offspring can only replace one of its parents. A crowding schema [29] called niche crowding (NC) was then proposed for improving the preselection process during which an offspring replaces the most similar individual taken from a randomly selected subpopulation based on the crowding factor (CF) value. In this schema, the measurement of similarity between two real-code chromosomes is usually based on the Euclidean distance. The NC tends to cause a large genetic shift in multimodal functions, especially when the selection subpopulation size is small.

Another well-known niche method is the sharing function, which reduces the fitness of highly similar individuals within the population based on the specified niche radius [5]. A practical difficulty of the sharing function method is choosing an adequate radius value, which is usually problem-dependent and implies that the numbers and shapes of peaks must be known. A similar condition applies to the selection of specific parameter values in various niche algorithms, e.g., the selection of clearing radius (distance) in the clearing method [30], [31] and the selection of species distance in the species conservation method [32]–[34]. However, in most industrial applications, very little *a priori* knowledge about the fitness landscape is available during parameter setting. Therefore, some crowding methods apply a parameter-free paradigm in algorithm design.

Mahfoud [35] proposed a deterministic crowding (DC) method for designing a competence rule in Algorithm 1.

Algorithm 1: DC Algorithm (Mahfoud [35])

Step 1: Randomly select two different parents, $p1$ and $p2$.

Step 2: Apply crossover, mutation and other operations to generate two offspring, $c1$ and $c2$.

Step 3: If $(|p1, c1| + |p2, c2|) \leq (|p1, c2| + |p2, c1|)$ Then

  If Fitness($c1$) is better than Fitness($p1$) Then replace $p1$ with $c1$

  If Fitness($c2$) is better than Fitness($p2$) Then replace $p2$ with $c2$

Else

  If Fitness($c2$) is better than Fitness($p1$) Then replace $p1$ with $c2$

  If Fitness($c1$) is better than Fitness($p2$) Then replace $p2$ with $c1$

End.

In Algorithm 1, the DC method uses a deterministic replacement operator to group the closest offspring and parent, and a parent can be replaced only by better offspring that are grouped together. However, although the DC reduces selection errors, genetic shift problems can still prevent the algorithm from efficiently locating multiple *loci*. Additionally, DC rarely outperforms NC when $\mathrm{CF} = 1$. These issues are further addressed by the experiments discussed in Section III below.

Ling *et al.* [7] proposed a cluster crowding (CC) method in which a temporary tree structure is used to maintain the clustering relationship between parents and offspring. The CC method sets the parents as the roots and then classifies and inserts all offspring under the parent root according to the Euclidean-distance measurement. After completing the classification, the CC treats the coverage of each sub-tree as a niche range and uses the best individual as the niche center. When it enters a new generation, the CC sorts all cluster centers and then adds them to a parent pool one at a time according to the following rule: if the centers of the newly added niches are not located within the niche range formed by the individuals in the parent pool, add them to the pool; otherwise, hold another competence tournament to determine whether the new center can be added; if the final parent pool size is still insufficient to form the new parent population, generate and initialize chromosomes to obtain the complete parent population.

Although the CC can locate multiple loci, its limitation is its slow convergence speed. Therefore, its solution quality is substantially poorer compared with those of both the NC and DC methods under the same evolutionary conditions. This issue is discussed further in Section III.

B. Twin-Space Crowding

Generally, two key operations are implicitly implemented in various crowding methods: the first operation ensures that individuals with the best fitness in the population are usually

invested with the highest survival probabilities; the second operation is a competence tournament among the population individuals located in the same niche. However, due to the insufficient knowledge of the real solution space, erroneous judgments about niche locations can prevent perfect implementation of the two operations in the crowding algorithms.

To overcome the difficulty of locating niches, this study proposes a new mechanism for dynamically tracking niche coverage by performing several competence tournaments to increase the survival probabilities of high-fitness individuals and to remove redundant individuals within a pair of niches. The pseudo steps of Algorithm 2 are specified as follows.

Algorithm 2: TC Algorithm

Definition:

$\mathbf{x} \succ \mathbf{y}$: x dominates y (x has better fitness compared to y).

Input:

(1) The whole parent population $\mathbf{P} = \{P_1, \ldots, P_m\}$ with size m.

(2) The whole offspring population $\mathbf{O} = \{O_1, \ldots, O_n\}$ with size n.

(3) Dimension number d of any population member. (or number of genes in any chromosome).

Output: Updated parent population $\mathbf{P}$.

Initialization: Define a temporary variable i, where $i \leftarrow 1$.

[Phase One]

Step 1: Find $P' \in \mathbf{P}$ where $|P'O_i| = min_{j=1}^{m}\{|P_jO_i|\}$.

Step 2: If $O_i \succ P'$ then $P' \leftarrow O_i$ and go to step 9.

[Phase Two]

Step 3: Compute the set
$\mathbf{T} = \{T'|T' \in \mathbf{P}, (|T'P'| \leq |P'O_i|)\Lambda O_i \succ T'\}$.

Step 4: If $\mathbf{T} = \emptyset$ then go to step 9 (i.e. ignoring this offspring).

Step 5: Define a temporary member Q.
Let $Q_j = \frac{O_{i,j}+P'_j}{2}$, $1 \leq j \leq d$.

Step 6: If $Q \succ P'$, then $P' \leftarrow Q$, and go to step 9.

Step 7: If $Q \succ O_i$, then go to step 9 (*i.e.* ignore this offspring).

[Phase Three]

Step 8: Randomly choose a member $\bar{T}$ from $\mathbf{T}$, and let $O_i \leftarrow \bar{T}$.

Step 9: Let $i = i + 1$. If $i \leq n$, then go to step 1 (next offspring). Otherwise, stop the algorithm.

The proposed TC algorithm has three main phases. For each offspring member, e.g., O_i, the first phase finds the closest parent, which is designated P'. If O_i dominates P', P' is directly replaced by O_i. Notably, the competence rule used in this phase resembles that used in NC method. However, the offspring that are lost in the first phase but have better fitness values than the other parents, have another opportunity to enter the later phases.

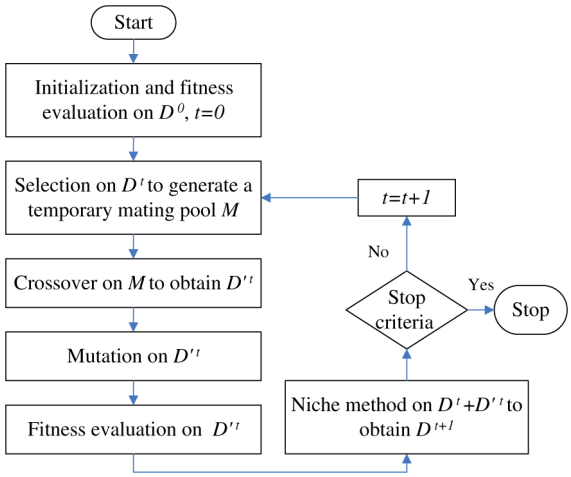


Fig. 1. Flowchart of a general niche-based genetic algorithm. The tth parent set is denoted by D^t, and the tth offspring set is denoted by D'^t.

The second phase uses a modified version of the *hill–valley* (HV) function [36]. The HV function is first performed to obtain a temporary member Q and to judge whether members O_i and P' are located at different *loci* (niches). The modified algorithm replaces P' with Q if Q is better than P' (since Q is also better than O_i in this situation). If Q is not better than P' and O_i, the third phase is performed.

The third phase determines the competence between O_i and the parents located in the circled area where the circle center is P' and the radius is $|P'O_i|$. In the area, one parent that is poorer than O_i is randomly chosen and replaced by O_i. This step is required because, under this condition, the chosen parent is not only poorer but also closer (more crowding) to P' in comparison with O_i. The third phase can be viewed as a self-adaptive niche exploration method, and the competence is held in a dynamically circled area. That is, the proposed algorithm dynamically selects the crowding space range for the survival competence and does not require specific niche radius information.

Fig. 1 shows the classic genetic flow of a general niche-based GA. That is, when incorporated in the GA, the proposed niche method (TC) is inserted into the flow as the replacement operator to generate a new parent population before a new generation is executed.

III. Comparative Study Solutions for Multimodal Benchmark Functions

Multimodal optimization problems have been intensively studied in the literature (e.g., [5]–[12], [37]). Although niche methods were originally designed for use in combination with GAs, some studies have applied them in other evolutionary algorithms (EAs). Examples include differential evolution (DE) [38] series containing the crowding DE (CDE) [39], the sharing DE (SDE) [39], and the species-based (SBDE) DE [40]. Notably, the CDE method is also a parameter-free approach. Its extension of DE with a crowding scheme enables it to track and maintain multiple optima. Furthermore, some hybrid approaches have also extended EAs by including local search methods (e.g., gradient-descent search [41]) or the

locality principle [42] for solving multimodal optimization problems. However, these hybrid approaches usually include new parameters to cooperate with the locality mechanisms (for a detailed review, see [41]). Additionally, some studies [e.g., particle swarm optimization (PSO)-based [43] approaches] have applied self-tuning techniques automatically adjusting specific parameters during the evolutionary loops [11], [44]. Even though these approaches may still need to design customized constants, for example, the precision in distance [44], these constants can be properly chosen in most practical applications. These self-tuning approaches thus approximate the parameter-free paradigm. Our study also compares convergence in various parameter-free and self-tuning evolutionary methods for solving multimodal optimization problems.

The hardware and software specifications for the development and execution environment in the experiments were an Intel P4 3.2G CPU (single-core) with 1G memory, XP SP2 OS, and Visual C++ 6.0 compiler. For each benchmark problem, optimization results were obtained for each crowding method using the same call numbers and the same genetic operators, including selection, crossover, and mutation. The customized genetic parameters for all test functions were set as follows: population size 50, offspring size 40, crossover rate 1.0, and mutation rate 0.05. The call numbers in different dimensional problems were set as suggested in the literature. In the sections, the comparison results were listed in tables for 100 independent runs of each method. Some poor solutions (larger than 1000) were indicated by dashes, and best solutions were indicated by bold type. Furthermore, a minimum value lower than 0.000001 was recorded as 0.

A. Genetic Operators and Parameters

The genetic operators applied in our experiments can be summarized as follows.

1) Selection and Constraint Handling: Tournament selection with size 2 is used in all experiments. A binary tournament with the following constraint domination relation developed by Deb [45] is used to handle constraints in the evolutionary schema: for two chromosomes, $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A}$ dominates $\mathbf{B}$ if the following conditions are satisfied:

1) $\mathbf{A}$ is feasible, but $\mathbf{B}$ is infeasible;
2) both $\mathbf{A}$ and $\mathbf{B}$ are infeasible, but $\mathbf{A}$ has smaller constraint value;
3) both $\mathbf{A}$ and $\mathbf{B}$ are feasible, but $\mathbf{A}$ has better fitness.

2) Crossover and Mutation: Here, a one-cut-point (1X) crossover operator is integrated with an arithmetical operator. The proposed method generates new offspring in the following steps: randomly choose two parents, select one cut-point, swap the genes before or after the cut-point for probability, and compute the linear combination of the gene at the cut-point. For example, the selected parents are specified as $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\mathbf{y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$. Suppose the cut-point is $\boldsymbol{i}$. The probabilistically generated offspring are

$$x' = \{y_1, y_2, \ldots, x'_i, x_{i+1}, \ldots, x_m\}$$
$$y' = \{x_1, x_2, \ldots, y'_i, y_{i+1}, \ldots, y_m\} \quad (1)$$

where $x'_i = x_i + \beta(y_i - x_i)$, $y'_i = y_i \pm \beta(y_i - x_i)$, and $\beta \in [0,1]$. To increase diversity, both interpolation and extrapolation methods are used to obtain the linear combinations used in the crossover.

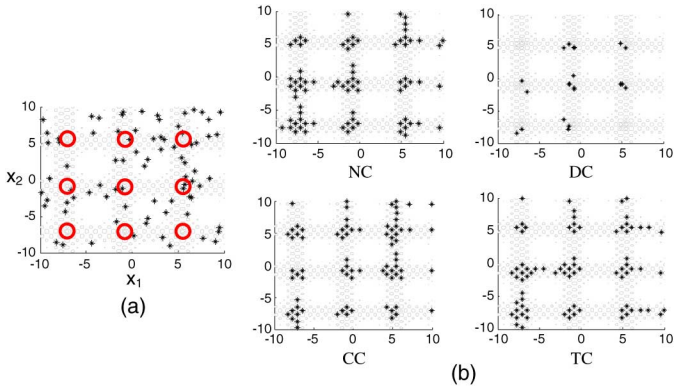


Fig. 2. Results obtained by niche GAs for Shubert function. The red circles indicate the locations of the optima, and the black asterisks indicate the locations of the population members. (a) Initialized population. (b) Population of 300th generation.

TABLE I
BENCHMARK FUNCTION LIST

Benchmark function		Search range	Initial range
$f1$ Sphere	$f_1(x) = \sum_{i=1}^{d} X_i^2$	$[-100,100]^d$	$[50,100]^d$
$f2$ Rosenbrock	$f_2(x) = \sum_{i=1}^{d} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-100,100]^d$	$[15,30]^d$
$f3$ Rastrigin	$f_3(x) = \sum_{i=1}^{d} (x_i^2 - 10\cos 2\pi x_i + 10)$	$[-10,10]^d$	$[2.56,5.12]^d$
$f4$ Griewank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\frac{x_i}{\sqrt{i}} + 1$	$[-600,600]^d$	$[300,600]^d$
$f5$ Ackley	$f_5(x) = -20e^{-0.2\sqrt{\left(\frac{1}{d}\sum_{i=1}^{d} x_i^2\right)}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos 2\pi x_i} + 20 + e$	$[-32,32]^d$	$[15,32]^d$
$f6$ Schaffer	$f_6(x) = 0.5 - \frac{(\sin\sqrt{x^2+y^2})^2 - 0.5}{(1.0 + 0.001(x^2+y^2))^2}$	$[-100,100]^2$	$[15,30]^2$

Mutation is performed by randomly selecting two genes and then normalizing and swapping them.

B. Numerical Benchmark Functions

To evaluate its solution capability, the proposed TCGA is compared with different methods using a set of benchmark functions.

1) Locating Multiple Loci: The first experiment evaluated the capability to locate optima in the solution space of the Shubert function, which is specified as follows:

$$f(x_1, x_2) = \sum_{i=1}^{5} [i\cos[(i+1)x_1 + i]] \times \sum_{i=1}^{5} [i\cos[(i+1)x_2 + i]]. \quad (2)$$

For the specified range, which includes nine optima, Fig. 2 shows the population distribution after several loops are executed. This figure shows that only the NC, CC, and TC methods can locate all optima. Thus, the DC method is a relatively unsuitable solver for exploring solution multiplicity in multimodal problems.

TABLE II
PERFORMANCE COMPARISON OF CROWDING GA METHODS IN TERMS OF SIX BENCHMARK FUNCTIONS. (A) CAPABILITY TO EXPLORE GLOBAL OPTIMA. (B) EXECUTION TIME OF EACH INDEPENDENT RUN

(a)

No	Dim	Func. calls	Mean minimum value (standard deviation)			
			NCGA	DCGA	CCGA	TCGA
*f*1	10	120000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
	20	160000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
	30	200000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
*f*2	10	120000	1.2162 (2.0377)	5.0497 (5.9544)	–	**1.1993** (1.9354)
	20	160000	10.5556 (8.0707)	30.9998 (35.0445)	–	**4.1573** (7.0146)
	30	200000	12.2826 (13.089)	50.463 (42.2775)	–	**1.0571** (5.1578)
*f*3	10	120000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
	20	160000	**0 (0)**	0.1989 (1.9799)	**0 (0)**	**0 (0)**
	30	200000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
*f*4	10	120000	**0 (0)**	0.0002 (0.0014)	5.6820 (3.8347)	**0 (0)**
	20	160000	**0 (0)**	**0 (0)**	36.9019 (19.3537)	**0 (0)**
	30	200000	**0 (0)**	**0 (0)**	46.5773 (25.3716)	**0 (0)**
*f*5	10	120000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
	20	160000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
	30	200000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
*f*6	2	200000	0.002646 (0.001892)	**0.002456 (0)**	0.002456 (0.0001)	**0.002456 (0)**

(b)

No	Dim	Func. calls	Mean execution time in one run (s)			
			NCGA	DCGA	CCGA	TCGA
*f*1	30	200000	3.099	2.536	3.614	3.69
*f*2	30	200000	1.142	0.603	3.456	1.747
*f*3	30	200000	1.226	0.686	1.932	1.847
*f*4	30	200000	1.179	0.552	3.623	1.776
*f*5	30	200000	0.316	0.215	1.248	0.477
*f*6	2	200000	0. 291	0.191	1.059	0.427

2) Solving Numerical Benchmark Functions: In many engineering applications, the computational cost of objective functions is much higher than that of niche operations. Therefore, the efficiency comparison of the studied methods was performed using the same number of fitness function calls in each method. Table I shows the studied benchmark functions containing the equations, the dimensions of input parameters, and the initialization and domain ranges.

In this table, the Sphere and Rosenbrock functions are unimodal, and the other four functions are multimodal. The Sphere function is a simple, smooth, and convex function. The global optima of the Rosenbrock function are within a very narrow, flat, parabola-shaped valley. Converging to the global minimum is very difficult when the parameter dimension is high. The Rastrigin function is a very difficult problem because of its numerous local minima. In the Griewank function, local minima increase exponentially as the number of dimensions increases. The Ackley function also has many local minima but only one global minimum. The Schaffer is a two-dimension function with many circular valleys surrounding the global minimum. In these difficult search spaces with numerous local optima, good optimization performance cannot be ensured if the proposed methods cannot explore and preserve multiple stable niches to assist the algorithms in escaping from the local optima.

In the literature, these functions are widely used to characterize optimization algorithms. The parameters were set according to relevant works reported in the literature (see [11] and [12]).

For these benchmark functions, Table II compares the solutions obtained by different crowding methods implemented by the authors. The cells show the mean minimum value and the standard deviation after 100 independent runs.

Comparisons of the results in Table II show that compared with all other crowding methods, the solution quality of the proposed TCGA is superior or at least comparable. However, when compared with the DC method, although the NC method obtains superior solutions in several test instances, it gives an inferior solution to the DC method for test function f6. From the above comparison, obtaining a solution clearly superior to those obtained by other solvers is extremely challenging.

The experimental results clearly show that the proposed method obtains the best solutions when used to explore global optima in different solution landscapes. However, regardless of the solution quality, the comparisons in Table II(b) show that DC is the fastest method followed by, in order of speed, NC, TC, and CC.

However, since the criterion for the performance comparisons is the same fitness call number, the main performance gaps are overheads except for the fitness evaluations. In the studied crowding algorithms, suppose the parent number is N and the offspring number is M. The DC method performs a competence tournament between two randomly selected parents for each offspring; hence, its complexity is $O(2M)$. The NC method must search the closest parent for each offspring; hence, it is a $O(MN)$ algorithm; in the worst case, the TC method must search the parent population twice for each offspring in order to find the closest parent and the poorest parent at the specified distance from the (estimated) niche range; hence, it is a $O(2MN)$ algorithm. The CC method involves more steps. It first finds the tree root (closest parent) and makes the tree for each offspring. The $O(MN)$ algorithm in this part reserves N simple trees. It then explores the tree center and radius by estimating the tree members. In the worst case, this part is a $O(M+N)$ algorithm. Finally, it sorts the tree centers $(O(N\log N))$ and inserts and compares each tree center into the new parent pool one by one $(O(1/2N^2-1/2N)O(1/2N^2))$; to sum up, the CC is a $O(MN+N\log N+1/2N^2)$ algorithm. In terms of complexity, different algorithms shown in Table II(b) have reasonable time costs.

However, the fitness evaluations in many practical evolutionary applications are very time-consuming (compared with other genetic operators). For example, the iterative design case in this paper requires approximately 546 s in 10 000 generations but the genetic operators (excluding fitness calls) take approximately 6.6 s. If the niche operator is replaced by another, the overall execution time improves by only 1–3 s (i.e., less than 1%) and the accuracy may be sacrificed. The overhead of the genetic operators is not a major consideration because of their relatively short execution time. For most practical applications, the additional execution time is acceptable if the solutions are sufficiently accurate.

3) Comparison With Other Evolutionary Computation Methods: The same benchmark functions were then used to compare solution capability with that in a set of evolutionary computation methods, including DE [38], PSO [43], CDE [39], *self-organizing hierarchical particle swarm optimizer with*

TABLE III
PERFORMANCE COMPARISON OF VARIOUS EVOLUTIONARY COMPUTATION METHODS IN TERMS OF SIX BENCHMARK FUNCTIONS

No	Dim	Mean minimum value (standard deviation)							
		G-PSO	PSO	HPSO-TVAC	AS	DE	CDE	SBFO	TCGA
$f1$	30	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
$f2$	30	2.46 (10.14)	75.30 (131.269)	7.14 (17.25)	161.921 (212.328)	34.35 (26.62)	21.6592 (2.6301)	28.7845 (0.0406)	**1.0571** (5.1578)
$f3$	30	0.13 (0.36)	28.25 (7.75)	1.59 (3.63)	140.045 (16.95)	27.43 (16.4193)	0.1299 (0.4155)	**0 (0)**	**0 (0)**
$f4$	30	0.066 (0.05)	0.0150 (0.016)	0.0157 (0.018)	**0 (0)**	0.0035 (0.0057)	0.0004 (0.0017)	**0 (0)**	**0 (0)**
$f5$	30	0.037 (0.205)	3.423 (7.564)	19.381 (0.287)	20.559 (2.04)	0.0008 (0.0047)	**0 (0)**	0.0002 (0.0698)	**0 (0)**
$f6$	2	0.002 (0.0039)	0[a] (0)	0.00793 (0.00588)	0.0034 (0.0046)	0.0488 (0.089)	0.002456 (0)	0.0025 (0.0004)	0.002456 (0)

[a]The minimal value of the specified Schaffer function ($f6$) should be 0.002456.

TABLE IV
COMPARISON OF THE TCGA METHOD AND THE VPPNM METHOD

No	Dim	Func. calls	Mean minimum value (standard deviation)	
			VPPNM	TCGA
$f1$	10	120 000	**0(0)**	**0(0)**
	20	160 000	**0(0)**	**0(0)**
	30	200 000	**0(0)**	**0(0)**
$f2$	10	120 000	–	**1.1993 (1.9354)**
	20	160 000	–	**4.1573 (7.0146)**
	30	200 000	–	**1.0571 (5.1578)**
$f3$	10	120 000	26.4958 (7.6295)	**0 (0)**
	20	160 000	–	**0 (0)**
	30	200 000	–	**0 (0)**
$f4$	10	120 000	0.963 (0.309)	**0 (0)**
	20	160 000	1.0371 (0.5832)	**0 (0)**
	30	200 000	1.14645 (0.5554)	**0 (0)**
$f5$	10	120 000	1.0934 (0.6328)	**0 (0)**
	20	160 000	2.4614 (0.8211)	**0 (0)**
	30	200 000	36.668 (14.5133)	**0 (0)**
$f6$	2	200 000	0.0026 (0.0007)	**0.002456 (0)**

time-varying acceleration coefficients (HPSO-TVAC) [11], *gregarious particle swarm optimizer* (GPSO) [44], *affine shaker* (AS) [46], and *synchronous bacterial foraging optimization* (SBFO) [12]. These methods are either the original versions (in this case, DE, PSO, and AS) or the extended versions, which are parameter-free or have self-tuning parameters. In addition to the CDE implemented by the authors, the results of the other algorithms are obtained directly from works by [11], [12], and [44].

Table III shows the computation results obtained by these methods for all test functions under the same execution criteria. Again, the solution quality of the proposed TCGA is superior or at least comparable.

4) Comparison With Global Simplex Methods: Earlier, [47] proposed a Globalized Bounded Nelder–Mead (GBNM) algorithm for solving engineering multimodal problems. In the GBNM method, globalization is achieved by probabilistic restart in which the spatial probability of starting a local search is determined by an improved Nelder–Mead (simplex) algorithm. In [48], the probability density schema used in the GBNM method was replaced with a Variable Variance Probability (VVP). The resulting VVPNM method obtained better performance and solution quality compared with the GBNM method. To realize global capability, the VPPNM method is used to solve the benchmark functions in Table I. Table IV shows the comparison results for the VPPNM method and the proposed TC method, which revealed that the proposed TCGA method outperforms the VPPNM method in all test instances and that when the number of dimensions reaches 20 or above, the efficiency of the VPPNM method in solving the benchmark problems substantially decreases.

5) Comparison of Different Crossover Patterns: This study compared the effects of three different crossover patterns: one-cut-point (1X), two-cut-point (2X), and uniform (U). Except for their mating patterns, the 2X and uniform crossovers are similar to the 1X crossover. Similar to the definitions of (1) and (2), the 2X crossover probabilistically generates two offspring for two selected parents x and y

$$x' = \left\{y_1, y_2, \ldots, x'_i, \ldots, x'_j, \ldots, x_m\right\}$$
$$y' = \left\{x_1, x_2, \ldots, y'_i, \ldots, y'_j, \ldots, y_m\right\}. \quad (3)$$

For the same parents, the uniform crossover generates

$$x' = \left\{x'_1, x'_2, \ldots, x'_m\right\} \quad \text{and} \quad y' = \left\{y'_1, y'_2, \ldots, y'_m\right\}. \quad (4)$$

The relationship between (x'_i, y') and (x_i, y_i) is defined by (1).

Table V shows the computation results obtained by the studied crowding methods for different crossovers where each combined method is designated by crowding method-crossover pattern. The computation results show that TC-1X and TC-2X dominate TC-U in all test instances, but TC-1X and TC-2X cannot dominate each other. That is, the uniform-based crossover is less suitable for the proposed TC crowding, which is also the case in the NC and DC methods. However, when combined with 1X and 2X crossovers, the TC method is superior or at least comparable in all test instances compared with the crowding methods. The TC method also obtains superior or at least very close results in all test instances when combined with the uniform crossovers. Generally, compared with the crowding methods, the proposed TC method generates superior or at least very close results when combined with various crossovers.

Furthermore, for functions in which the number of dimensions is high ($d = 45$ or 60), the TC-1X method dominates all others despite the crossover patterns.

IV. PERFORMANCE OF TCGA IN SOLVING JOINT-SPACE INVERSE PROBLEMS

Since the proposed TC method does not need additional parameters, use of the TCGA is identical to that of the simple GA. To verify its use in industrial applications, the proposed TCGA was used to solve joint-space inverse problems in two manufacturing robots, including an inverse problem in an SLR and the design problem of collision-free six-bar robotic hands with a planar linkage structure.

A. Joint-Space Inverse Problems in SLRs

In SLRs, a set of links is connected by various revolute or prismatic joints [49]. By assigning coordinate frames to each link of an SLR, joint motion can be analyzed and obtained through a coordinate transformation. The Denavit–Hartenberg (D–H)

TABLE V
PERFORMANCE COMPARISON OF CROWDING GA METHODS WITH DIFFERENT CROSSOVERS

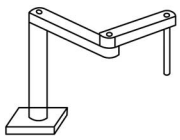
No	Dim	Func. calls	Mean minimum value (standard deviation)											
			NC-1X	NC-2X	NC-U	DC-1X	DC-2X	DC-U	CC-1X	CC-2X	CC-U	TC-1X	TC-2X	TC-U
*f*1	10	120000	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
	20	160000	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
	30	200000	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
	45	500000	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
	60	1000000	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
*f*2	10	120000	1.2162 (2.0377)	0.1189 (0.5409)	7.4159 (7.1083)	5.0497 (5.9544)	4.0128 (4.7570)	6.3224 (3.1512)	–	–	–	1.1993 (1.9354)	**0.0825 (0.1448)**	1.6245 (2.6956)
	20	160000	10.5556 (8.0707)	0.0825 (0.232)	14.655 (7.4284)	30.9998 (35.0445)	3.3854 (6.1393)	8.7179 (4.5327)	–	–	–	4.1573 (7.0146)	**0.0462 (0.1759)**	15.9566 (10.4996)
	30	200000	12.2826 (13.089)	0.1411 (0.5591)	8.4376 (4.6715)	50.463 (42.2775)	5.839 (24.6349)	33.1274 (71.7085)	–	–	–	1.0571 (5.1578)	**0.0974 (0.3831)**	7.666 (6.5771)
	45	300000	17.6375 (20.5919)	0.1375 (0.6233)	4.3059 (4.8434)	82.9448 (249.7714)	0.4998 (0.9777)	27.1321 (101.7102)	–	–	–	**0.0249 (0.1663)**	0.0314 (0.0674)	3.1742 (2.8588)
	60	400000	28.4257 (36.547)	0.0531 (0.1498)	2.6299 (5.0008)	60.5166 (213.5005)	1.5129 (7.227)	4.5529 (34.7635)	–	–	–	**0.0169 (0.124)**	0.0442 (0.2647)	2.4782 (2.5568)
*f*3	10	120000	**0 (0)**	**0 (0)**	5.2503 (1.4135)	**0 (0)**	**0 (0)**	6.5299 (2.7848)	**0 (0)**	**0 (0)**	6.8126 (1.3693)	**0 (0)**	**0 (0)**	1.2943 (2.01352)
	20	160000	**0 (0)**	**0 (0)**	22.1768 (10.5421)	0.1989 (1.9799)	**0 (0)**	60.028 (17.3718)	**0 (0)**	**0 (0)**	27.0227 (14.3267)	**0 (0)**	**0 (0)**	24.6928 (6.2708)
	30	200000	**0 (0)**	**0 (0)**	85.3382 (37.7926)	**0 (0)**	**0 (0)**	134.7291 (53.5652)	**0 (0)**	**0 (0)**	100.6060 (37.4597)	**0 (0)**	**0 (0)**	45.5546 (11.1701)
	45	300000	**0 (0)**	**0 (0)**	241.2003 (88.8189)	6.2683 (15.5357)	**0 (0)**	369.4807 (39.2758)	**0 (0)**	**0 (0)**	247.0732 (87.4227)	**0 (0)**	**0 (0)**	141.8709 (80.7849)
	60	400000	**0 (0)**	**0 (0)**	285.6501 (107.2883)	2.9848 (13.0108)	**0 (0)**	489.3491 (62.0632)	**0 (0)**	**0 (0)**	291.3558 (110.1985)	**0 (0)**	**0 (0)**	163.3896 (119.678)
*f*4	10	120000	**0 (0)**	0.0009 (0.0025)	0.0382 (0.0247)	0.0002 (0.0014)	0.0005 (0.002)	0.0126 (0.0218)	5.6820 (3.8347)	22.2466 (9.6406)	46.4782 (12.9045)	**0 (0)**	0.0003 (0.0015)	0.0309 (0.0302)
	20	160000	**0 (0)**	0.0001 (0.001)	0.0005 (0.002)	**0 (0)**	**0 (0)**	0.0018 (0.0033)	36.9019 (19.3537)	135.6619 (33.0296)	177.7703 (34.8983)	**0 (0)**	0.0000 (0.0007)	0.0016 (0.0057)
	30	200000	**0 (0)**	**0 (0)**	0.0001 (0.001)	**0 (0)**	**0 (0)**	0.0002 (0.0012)	46.5773 (25.3716)	276.1670 (44.3308)	347.2386 (44.7080)	**0 (0)**	**0 (0)**	0.0001 (0.0009)
	45	300000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	0.0001 (0.0007)	106.9910 (56.1246)	424.6612 (52.0912)	526.4681 (57.9239)	**0 (0)**	**0 (0)**	0.0001 (0.0007)
	60	400000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	197.9409 (91.8268)	790.0793 (72.9096)	905.9507 (96.952)	**0 (0)**	**0 (0)**	**0 (0)**
*f*5	10	120000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	13.5381 (1.4550)	**0 (0)**	**0 (0)**	**0 (0)**
	20	160000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	0.0714 (0.5004)	**0 (0)**	2.1014 (2.1514)	17.1346 (0.6578)	**0 (0)**	**0 (0)**	**0 (0)**
	30	200000	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	3.0983 (0.7334)	**0 (0)**	11.4598 (2.1716)	17.8172 (0.6556)	**0 (0)**	**0 (0)**	**0 (0)**
	45	300000	**0 (0)**	**0 (0)**	3.3586 (0.0442)	**0 (0)**	**0 (0)**	4.2968 (1.2032)	**0 (0)**	13.6812 (1.1049)	17.4746 (0.7648)	**0 (0)**	**0 (0)**	2.4006 (1.4966)
	60	400000	**0 (0)**	**0 (0)**	3.4265 (0.0405)	0.0005 (0.0055)	**0 (0)**	4.5201 (1.329)	**0 (0)**	16.9927 (0.8251)	17.0984 (0.8497)	**0 (0)**	**0 (0)**	2.9757 (1.0933)
*f*6	2	200000	0.0026 (0.0018)	0.0034 (0.0041)	**0.002456 (0)**	**0.002456 (0)**	**0.002456 (0)**	**0.002456 (0)**	0.002456 (0.0001)	0.0035 (0.0045)	**0.002456 (0)**	**0.002456 (0)**	**0.002456 (0)**	**0.002456 (0)**

convention is typically used to select transformations in robot links [49]. In the D–H convention, each transformation between links i and $i-1$ is represented as a 4×4 composite homogeneous matrix ${}_{i}^{i-1}\boldsymbol{T}$. The matrix is a product of four basic transformations typically designated link length, link skew (twist) angle, link offset, and joint angle (α, a, d, and θ, respectively). Using the chain product of successive coordinate transforms, the homogenous matrix ${}_{\mathrm{i}}^{0}\boldsymbol{T}$ represents the forward (kinematic) transform of the ith coordinate system with respect to the base coordinate system. For the end-effector of a robot with the ith joint variable, the position relative to the base coordinate system can be obtained from the last column of the transform matrix, and the orientation can be converted from the first three columns.

1) Optimal Manipulation Problem in SLRs: Although industrial robots generally operate in the joint-variable space, manipulated objects are expressed in the world coordinate system. To reach an object, inverse kinematic solutions are needed to control the position and orientation of the end effectors. However, when operating SLRs, the position errors are accumulated and nonlinearly amplified from link to link [49]. That is, a small difference in joint values may have a large impact on the positioning task of an end-effector. Therefore, sufficiently accurate values are essential when setting joint angles in robotic manipulation applications [1], [50]. As in previous evolutionary approaches [2], this study optimized a general position-error goal function that obtains inverse kinematic solutions for SLRs by including the above forward kinematics in the optimization process. For an industrial manufacturing robot with n joints with at least three intersecting axes, the DOF can be reduced to "$n-3$" by decoupling the wrist position solution from the variables used to represent hand joint orientation. Therefore, this and previous works find the optimal solution for the inverse problem mainly by improving the capability to locate joints 1 to $n-3$. This problem is formulated as follows:

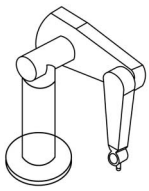
$$\begin{aligned} &\text{minimize } \|p' - p\| \\ &\text{subject to } \theta_i^L \le \theta_i \le \theta_i^U, 1 \le i \le n-3 \end{aligned} \tag{5}$$

where p is the expected locating position of the $(n-3)$th joint and where $\| * \|$ is the Euclidean norm function, p' is the position after completing the evaluation of the coordinate transform



α (deg)	a (mm)	d (mm)	θ (deg)	Joint range (deg)
0	0	0	θ_1	[−115,115]
0	580	0	θ_2	[−145,145]
0	470	0	0	

Fig. 3. SCARAB and its D–H form for the first two joints.



α (deg)	a (mm)	d (mm)	θ (deg)	Joint range (deg)
0	0	0	θ_1	[−160, 160]
−90	0	0	θ_2	[−225, 45]
0	431.8	149.09	θ_3	[−45, 225]
−90	20.32	433.07	0	

Fig. 4. PUMA560 and its D–H form for the first three joints.

function $f(\alpha, a, d, \theta)$ according to the regular D–H formulation, and θ_i^L, θ_i^U are the lower and upper bounds, respectively, of the ith joint value.

2) Experiments and Comparative Results: The same manipulation problems studied in Kalra *et al.* [2] were used for experimental evaluation of solution performance in a GA embedded with the sharing function method in two different SLRs, a PUMA 560 and a SCARA. Figs. 3 and 4 show the draft structures and the D–H convention parameters for the two robots.

The experiment compared the results obtained by different approaches when using the same genetic parameters except for the niche mechanisms. In terms of chromosome representation, the proposed TCGA method solved the optimal inverse problem by including the position-error objective in (4) in the fitness evaluation. Figs. 3 and 4 show that, because of the variables in the D–H forms, the chromosomes are encoded to contain two and three real-coded genes in each of the two robots. That is, the gene number is equal to the number of variables in the D–H forms.

For a fair comparison, the same genetic operators used in Kalra *et al.* [2] were applied, including tournament selection, simulated binary crossover (SBX) crossover [51], and mutation. Mutation was formulated as the following equation:

$$\theta' = \theta + \delta(\theta^U - \theta^L) \tag{6}$$

where θ' is the mutation of variable θ, θ^U, and θ^L are the upper and lower bounds, respectively, and δ is a perturbance factor derived as described in Deb and Goyal [52]. This study also applied the same manufacturing points used in Kalra *et al.* [2].

As in previous works, population size and call number were set to 80 and 7200 in the SCARA case and to 150 and 40 500 in the PUMA case, respectively. Tables VI and VII show the experimental results, which confirm that the proposed TCGA method accurately located all configurations, and its position location accuracy was much higher than methods reported in previous works within the same number of fitness function calls.

B. Inverse Problems for Six-Bar Robotic Transferring Hands

The following problems were motivated by the need for a practical manufacturing system (http://www.techbasecorp.com/) in which a six-bar robotic transferring hand can be combined with other equipment to perform a transferring task of liquid metal. When such a system is operated in a practical industrial environment, related equipment located in the manufacturing area may become obstacles. Fig. 5 shows the real-world manufacturing environment and a conceptualization of the six-bar structure. A manufacturing curve is also given for the example of an essential equipment item (here, a boiler) that becomes an obstacle.

1) Iterative Design Method: The classic design problem is to arrange several precise-points such that the coupler can be guided to avoid collisions with obstacles. This study applies a collision-free design method [22] that formulates the positions of obstacles and can evolutionarily explore the design solutions. Fig. 6 compares the performance of the two methods. Fig. 6(a) shows that the precise-point method is clearly inadequate because it does not include all solutions obtained by the collision-free design method shown in Fig. 6(b). The diagram also shows that the collision-free design problem is multimodal and highly constrained.

In the previous work, the authors solved the collision-free design problem by multiobjective solution approach, in which the objectives must be revealed in the initial stage. The design approach proposed here, however, is applicable if some objectives and constraints must be explored through iterative discussion and analysis, i.e., if only the problem itself must be specified in the initial stage. The proposed TCGA method is then used in subsequent stages to solve solution multiplicity under several progressively strict design rules.

2) Iterative Design Flow: For the structural design problem, a draft design [see example in Fig. 5(b)] is conceptualized to include the loading and injecting positions in the basic structure. When the structure operates, input angle θ changes from 0° to θ_{last} while the end point F of the last bar changes from M_1 to M_{last}. The structure contains two pivot points and several linkage bars. The constant pivot point O is directly set at the origin of the coordinate system, but another pivot point C and the bar lengths denoted as a set $\text{BR} = \{\text{BR}_{\text{OU}}, \text{BR}_{\text{UB}}, \text{BR}_{\text{BC}}, \text{BR}_{\text{CD}}, \text{BR}_{\text{DE}}, \text{BR}_{\text{EF}}, \text{BR}_{\text{UE}}\}$ are reserved as design variables.

The optimization problem for the design problem can be formulated as follows:

$$\begin{aligned} &\text{minimize } \sigma(0^\circ, M_1, \varepsilon) + \sigma(\theta_{\text{last}}, M_{\text{last}}, \varepsilon) \\ &\text{subject to } \Phi(\theta, C, O, BR) = 0 \end{aligned} \tag{7}$$

where $\sigma(\theta_i, M_i, \varepsilon) = \begin{cases} 0, & |\varphi(\theta_i), M_i| \le \varepsilon \\ |\varphi(\theta_i), M_i|, & \text{otherwise} \end{cases}$, ε is a customized constant, and $\varphi(\cdot)$ is a function for computing the position of the end point F at the specified angle, which can be derived by the kinematic equations applied in earlier studies of four linkage bars [19], and $\Phi(\cdot)$ is a function for computing the collision values (for the derivation steps, see [22]). According to the kinematical characteristics of planar linkages, function $\Phi(\cdot)$ can be derived and entered into the equations by using only variables θ, C, and BR. Therefore, variables θ, C, and BR are encoded into chromosome genes. Since pivot point C contains two components Cx and Cy, and set BR contains seven components, the chromosome contains ten real-coded genes. This study also applies the genetic operators specified in Section III-A.

TABLE VI
COMPARISON OF KALRA *ET AL.* [2] AND THIS WORK IN TERMS OF SOLUTION PERFORMANCE IN SCARAB JOINT-SPACE INVERSE PROBLEM

No	Manufacturing points (mm)	Kalra *et al.*			This work		
		θ_1 (rad)	θ_2 (rad)	Position error (mm)	θ_1 (rad)	θ_2 (rad)	Position error (mm)
1	(600, 400, 130)	−0.1201	1.6380	0.69	−0.1194	1.6389	0.27
		1.2960	−1.6404	0.50	1.2960	−1.6397	0.18
2	(400, −600, 130)	−1.6907	1.6383	0.45	−1.6906	1.6394	0.13
		−0.2741	−1.6401	0.48	−0.2747	−1.6395	0.12
3	(350, 350, 130)	−0.1066	2.1804	0.14	−0.1065	2.1806	0.05
		1.6783	−2.1807	0.44	1.6774	−2.1803	0.21
4	(−100, 700, 130)	0.9903	1.6762	0.16	0.9906	1.6760	0.08
5	(650, −450, 130)	−1.2358	1.4449	0.49	−1.2360	1.4457	0.24
		0.0257	−1.4475	0.41	0.0256	−1.4464	0.13

TABLE VII
COMPARISON OF KALRA *ET AL.* [2] AND THIS WORK IN TERMS OF SOLUTION PERFORMANCE FOR PUMA 560 JOINT-SPACE INVERSE PROBLEM

No	Manufacturing points (mm)	Kalra *et al.*				This work			
		θ_1 (rad)	θ_2 (rad)	θ_3 (rad)	Position error (mm)	θ_1 (rad)	θ_2 (rad)	θ_3 (rad)	Position error (mm)
1	(600, 149.09, 200)	−0.0005	−1.0745	3.1210	0.67	−0.0003	−1.0752	3.1206	0.21
		−0.0014	0.4304	0.1152	1.01	0.0002	0.4325	0.1133	0.62
		−2.6560	−2.0659	0.1158	1.16	−2.6543	−2.0668	0.1148	0.25
		−2.6530	−3.5735	3.1233	1.29	−2.6543	3.5747	3.1225	0.80
2	(500, 240, 230)	0.1742	−1.2415	3.2852	0.93	0.1746	−1.2426	3.2866	0.66
		0.1756	0.4299	−0.0546	1.91	0.1755	0.4301	−0.0506	0.33
		−2.4217	−1.8985	−0.0498	0.36	−2.4221	−1.8994	−0.0499	0.26
		−2.4201	−3.5690	3.2827	1.45	−2.4223	−3.5709	3.2852	0.25
3	(540, 210, 260)	0.1113	−1.2124	3.1705	0.56	0.1108	−1.2126	3.1699	0.27
		0.1125	0.3420	0.0692	1.53	0.1107	0.3437	0.0645	0.12
		−2.5094	−1.9272	0.0639	0.72	−2.5100	−1.9287	0.0651	0.31
		−2.5115	−3.4856	3.1744	1.75	−2.5106	−3.4850	3.1707	0.22
4	(180, −400, 400)	−1.4937	−1.6174	3.3063	0.74	−1.4945	−1.6167	3.3065	0.15
		−1.4948	0.0780	−0.0729	0.51	−1.4949	0.0771	−0.0720	0.16
		2.3399	−1.5268	−0.0699	0.86	2.3408	−1.5246	−0.0716	0.14
		2.3403	−3.2198	3.3081	0.56	2.3415	−3.2185	3.3072	0.42
5	(−180, 400, −200)	1.6458	−0.5640	3.6446	0.90	1.6472	−0.5638	3.6416	0.36
		−0.8024	−2.5784	−0.4049	0.86	−0.8006	−2.5770	−0.4077	0.28

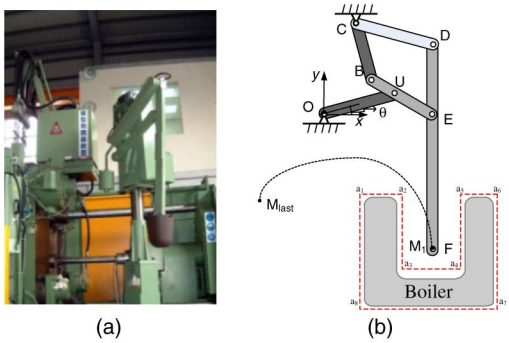


Fig. 5. Real-world manufacturing environment where the loading position is M_1 and the injecting position is M_{last}. (a) Real-world manufacturing environment. (b) Conceptual design under collision-free requirements.

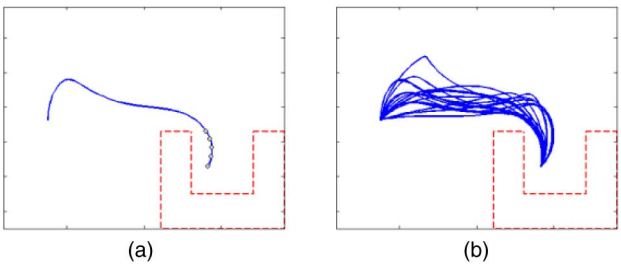


Fig. 6. Comparison of results obtained by two different design methods. (a) Results obtained by classic precise-point method. (b) Results obtained by collision-free design method.

TABLE VIII
PARAMETERS OF DESIGN PROBLEM OF ROBOTIC TRANSFERRING HAND. (A) SPATIAL AND GA PARAMETERS. (B) PARAMETERS OF DESIGN PROBLEM

(a)

Various spatial points				GA parameters
a_1	(220, −850)	a_5	(1420, −850)	Loop_count: 15000
a_2	(620, −850)	a_6	(1820, −850)	Parent_size: 100
a_3	(620, −1750)	a_7	(1820, −2250)	Offspring_size: 90
a_4	(1420, −1750)	a_8	(220, −2250)	Crossover_rate: 1.0
M_1	(838, −1350)	M_{last}	(−1247, −680)	Mutation_rate: 0.3

(b)

Parameters	Values
ε	5 (mm)
Range_of_C_x	[−300, 300]
Range_of_C_y	[450, 550]
Range_of_BR_{OU}_BR_{UB}_BR_{BC}_BR_{DE}_BR_{UE}	[0, 500]
Range_of_BR_{CD}_BR_{EF}	[0, 1000]
Range_of_θ_{last}	[120°, 170°]

When the TCGA method is used to explore the solutions, Table VIII lists the customized genetic parameters obtained in the design phase, and Table IX lists all solutions explored in the loops. Based on these solutions, Fig. 7(b)–(d) shows the examples of coupling curves for several structures. The designer can analyze and refine the first design by adding requirements and rules to obtain improved solutions in the subsequent design phase. For example, in the studied system, the practical controller must ensure that the liquid surface is parallel to the horizon during the

TABLE IX
SOLUTIONS EXPLORED IN THE FIRST DESIGN PHASE

Solutions	Pivot C (mm)		Lengths of bars (mm)							θ_{last} (deg.)
	CX	CY	OU	UB	BC	DE	CD	UE	EF	
#1	294.8	505.5	331.2	291.3	441.5	416.4	1032.9	855.6	1008.4	122.3
#2	181.9	520	326.4	279.5	357.8	451.4	1058.1	829.5	855.4	137
#3	181.9	520	326.4	279.5	357.8	451.4	1058.1	829.5	855.4	137.6
#4	147	507.9	322.3	292	335.8	459.9	1073.6	829.6	861.7	140.6
#5	127.1	489.6	306.1	306.1	335.8	342.8	1154.7	861.3	916.1	141.4
#6	144.1	508.1	322.3	280.8	335.6	350.8	1156.6	865.2	827.5	142.6
#7	119.6	480.8	312.7	283.2	335.8	446.8	1137	859.7	917.7	144.3
#8	84.3	492.1	341	259.1	327.7	541.5	1051.9	805.4	830.2	146.6
#9	96.3	483.5	326.8	259.1	327.7	541.5	1051.9	805.4	870.1	147.1
#10	49.5	513.9	341.4	273.5	356.4	527.8	860.2	614.2	975.2	151.8
#11	28.9	499.4	341.4	273.5	355.2	556.4	874	614.2	999	153.1
#12	49.5	513.9	341.4	273.5	356.4	527.8	860.2	614.2	975.2	153.9
#13	53	516.3	341.4	281.1	356.4	445.1	1195.9	829.5	885.7	154.3
#14	28.3	483.4	313.7	371.7	289.4	424.1	1230.3	860.2	917.7	155
#15	28.3	488.4	341.4	249.6	356.4	527.8	860.2	614.3	977.7	157.3
#16	4.1	513	319.9	410.1	408.9	467.7	1196.6	785.5	1152.7	158.2
#17	4.1	508.6	317.2	414.2	296.3	332.9	1280.7	853.8	926.2	160.5
#18	−92.3	487	332.3	414.2	263	444.7	1037.1	639.9	977.2	168
#19	−92.3	500.8	346.2	414.2	263	447.9	1037.1	643.7	938.8	169.8

TABLE X
ADDITIONAL PARAMETERS AND THE SOLUTIONS EXPLORED IN FINAL DESIGN PHASE. (A) ADDITIONAL PARAMETERS. (B) SOLUTIONS EXPLORED IN THE FINAL PHASE

(a)

Parameters	Values
Angle_velocity_of_θ	0.01 (rad/s)[a]
ε_τ	0.35 (mm/s^2)
ε_ρ	50 (mm)
Range_of_θ_{last}	[140°, 160°]

(b)

Solutions	Pivot C (mm)		Lengths of bars (mm)							θ_{last} (deg.)
	CX	CY	OU	UB	BC	DE	CD	UE	EF	
#1	130.3	488.6	308.4	331.8	397.8	455	1378.5	1016.4	1088.2	141.6
#2	131.5	488.6	308.4	331.8	397.8	455	1378.5	1016.4	1088.2	142.7
#3	115.7	489.8	309.4	343.1	393.3	450.7	1387.1	1012.3	1078.8	144.2
#4	31.7	518	327.1	367.9	378.2	531.1	1135.2	764.1	1071.9	154.3
#5	33.3	507.5	327.1	329.3	383.1	531.1	1135.2	764.2	1075.3	154.4
#6	33.3	488.3	315.5	320.9	389.2	515.8	1135.2	769.8	1127.8	154.5
#7	33.3	488.3	315.5	324.9	389.2	515.8	1135.2	769.8	1127.8	155.8
#8	-15.3	523.3	338.4	383.8	369.5	557.3	1194.6	764.1	1071.9	158.8

[a]During laboratory testing, θ was operated at a constant speed to verify the relative position acceleration of point F.

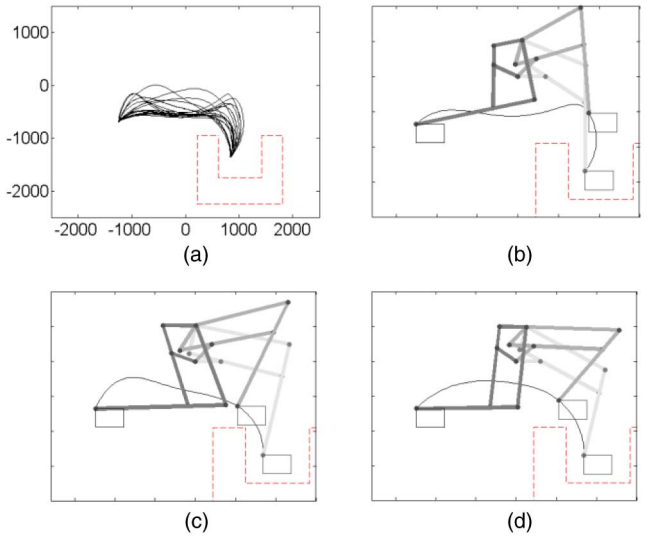


Fig. 7. Coupling curves for initial solutions: (a) All solutions, (b) Case #1, (c) Case #2, and (d) Case #3.

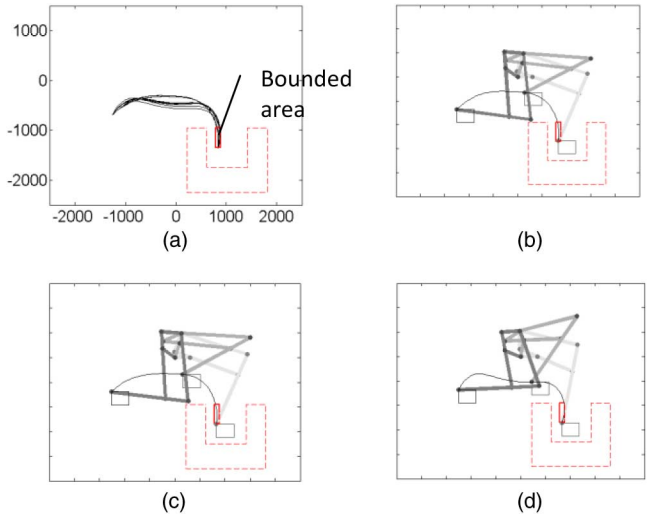


Fig. 8. Coupling curves for final solutions obtained by iterative design method. The rectangular areas indicate the bounds of the initial loading operation. (a) All solutions, (b) Case #1, (c) Case #2, and (d) Case #3.

transfer process. Therefore, the designers in the manufacturing company must maintain the vertical orientation of the initial loading motion (here, the loading is performed in liquid) in order to decrease the swing angle and to simplify the controller.

Table X(a) shows the additional customized parameters included in the final design requirements. Table X(b) lists all solutions explored in the final case, and Fig. 8(b)–(d) shows examples of structures and coupling curves decoded from some solutions.

The results show that the proposed iterative design first explores multiple diverse solutions that satisfy the collision-free constraints. The design framework then refines the design requirements by progressively adding acceleration value limitations and (initially) vertical loading requirements. Because of the multi-*loci* locating capabilities of the proposed TCGA, the conceptual design progressively improves and increasingly comprehensive design cases can be considered.

Furthermore, in our empirical study, the multiobjective method should take about 100 000 generations to explore the Pareto front in the design problem, whereas the TCGA can explore the expected partial Pareto front in about 10 000 generations. Although this reveals the potential time savings of a single objective scheme, the proposed approach still requires additional (iterative) design time to complete the task.

V. Conclusion

This work developed a new TC NC method of solving numerical multimodal optimization problems. Since the proposed algorithm does not require prior knowledge of the solution space to assist the crowding mechanism, the proposed TCGA is suitable for solving engineering problems in which the solution space is intractable.

The effectiveness and efficiency of the proposed TCGA were verified using a set of numerical benchmark functions, and the joint-space inverse problem in SLRs was used to compare solution performance between the TCGA and other methods. The experimental results show that the proposed TCGA obtains better solutions within the same number of evolutionary function calls. Another experimental application of the TCGA for solving an iterative collision-free design problem involving a six-bar robotic transferring hand further confirmed that, by effectively and progressively exploring diverse solutions, the TCGA can provide designers with a wide selection of solutions.

References

[1] D. Manocha and J. F. Canny, "Efficient inverse kinematics for general 6R manipulators," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 648–657, Oct. 1994.

[2] P. Kalra, P. B. Mahapatra, and D. K. Aggarwal, "An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots," *Mech. Mach. Theory*, vol. 41, pp. 1213–1229, 2006.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Anna Arbor, MI, USA: Univ. of Michigan Press, 1975.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison Wesley, 1989.

[5] D. E. Goldberg and J. J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genet. Algorithm Genet. Algorithm Their Appl.*, 1987, pp. 41–49.

[6] B. Sareni, L. Krahenbuhl, and A. Nicolas, "Niching genetic algorithms for optimization in electromagnetics. I. Fundamentals," *IEEE Trans. Magn.*, vol. 34, no. 5, pp. 2984–2987, Sep. 1998.

[7] Q. Ling, G. Wu, Z. Yang, and Q. Wang, "Crowding clustering genetic algorithm for multimodal function optimization," *Appl. Soft Comput.*, vol. 8, pp. 88–95, 2008.

[8] A. A. Alugongo, "Multimodal problems, premature convergence versus computation effort in dynamic design optimization," in *Proc. World Congr. Eng. (WCE)*, vol. 3, 2011, pp. 2537–2542.

[9] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Trans. Evolut. Comput.*, vol. 2, no. 3, pp. 97–106, Sep. 1998.

[10] J. E. Vitela and O. Castaños, "A sequential niching memetic algorithm for continuous multimodal function optimization," *Appl. Math. Comput.*, vol. 218, pp. 8242–8259, 2011.

[11] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[12] K. M. Bakwad, S. S. Pattnaik, B. S. Sohi, S. Devi1, B. K. Panigrahi, and S. V. R. Gollapudi, "Multimodal function optimization using synchronous bacterial foraging optimization technique," *IETE J. Res.*, vol. 56, pp. 80–87, 2010.

[13] C. Hocaoglu and A. C. Sanderson, "Planning multiple paths with evolutionary speciation," *IEEE Trans. Evol. Comput.*, vol. 5, no. 3, pp. 169–191, Jun. 2001.

[14] C. T. Cheng, K. Fallahi, H. Leung, and C. K. Tse, "A genetic algorithm-inspired UUV path planner based on dynamic programming," *IEEE Trans. Syst. Man, Cybern. C., Appl. Rev.*, vol. 42, no. 6, pp. 1128–1134, Nov. 2012.

[15] C. C. Hsu, Y. J. Chen, M. C. Lu, and S. A. Li, "Optimal path planning incorporating global and local search for mobile robots," in *Proc. IEEE 1st Global Conf. Consum. Electron. (GCCE)*, 2012, pp. 668–671.

[16] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robot. Res.*, vol. 4, pp. 3–9, 1985.

[17] R. Muszynski, "A solution to the singular inverse kinematic problem for a manipulation robot mounted on a track," *Control Eng. Pract.*, vol. 10, pp. 35–43, 2002.

[18] F. Freudenstein, "An analytical approach to the design of four-link mechanisms," *Trans. ASME*, vol. 76, pp. 483–492, 1954.

[19] F. Freudenstein, "Approximate synthesis of four-bar linkages," *Trans. ASME*, vol. 77, p. 853, 1955.

[20] R. R. Bulatovic and S. R. Dorevic, "Control of the optimum synthesis process of a four-bar linkage whose point on the working member generates the given path," *Appl. Math. Comput.*, vol. 217, pp. 9765–9778, 2011.

[21] G. H. Martin, *Kinematics and Dynamics of Machines*. New York, NY, USA: McGraw-Hill, 1982.

[22] C. H. Chen, T. K. Liu, I. M. Huang, and J. H. Chou, "Multiobjective synthesis of six-bar mechanisms under manufacturing and collision-free constraints," *IEEE Comput. Intell. Mag.*, vol. 7, no. 1, pp. 36–48, Feb. 2012.

[23] J. T. Tsai, K. M. Lee, and J. H. Chou, "Robust evolutionary optimal tolerance design for machining variables of surface grinding process," *IEEE Trans. Ind. Inform.*, vol. 10, no. 1, pp. 301–312, Feb. 2014.

[24] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 132–141, Feb. 2013.

[25] S. H. Hur, R. Katebi, and A. Taylor, "Modeling and control of a plastic film manufacturing web process," *IEEE Trans. Ind. Inform.*, vol. 7, no. 2, pp. 171–178, May 2011.

[26] P. Siano, C. Cecati, H. Yu, and J. Kolbusz, "Real time operation of smart grids via FCN networks and optimal power flow," *IEEE Trans. Ind. Inform.*, vol. 8, no. 4, pp. 944–952, Nov. 2012.

[27] W. Zeng and M. Y. Chow, "Modeling and optimizing the performance-security tradeoff on D-NCS using the coevolutionary paradigm," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 394–402, Feb. 2013.

[28] D. J. Cavicchio, "Adaptive search using simulated evolution," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. of Michigan, Ann Arbor, MI, USA, 1970.

[29] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. of Michigan, Ann Arbor, MI, USA, 1975.

[30] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. IEEE Conf. Evol. Comput.*, Nagoya, Japan, 1996, pp. 798–803.

[31] A. Della Cioppa, C. De Stefano, and A. Marcelli, "On the role of population size and niche radius in fitness sharing," *IEEE Trans. Evol. Comput.*, vol. 8, no. 6, pp. 580–592, Dec. 2004.

[32] K. Deb, J. P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multi-modal function optimization," *Evol. Comput.*, vol. 10, pp. 207–234, 2002.

[33] J. P. Li and A. Wood, "Random search with species conservation for multimodal functions," in *Proc. 11th Conf. Congr. Evol. Comp.*, Trondheim, Norway, 2009, pp. 3164–3171.

[34] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, "Multimodal optimization by means of a topological species conservation algorithm," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp.842–864, Dec. 2010.

[35] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Dept. Gen. Eng., Univ. of Illinois, Urbana, IL, USA, 1995.

[36] R. Ursem," Multinational evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, Washington, DC, USA, 1999, pp. 1633–1640.

[37] K. Deb and A. Saha, "Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach," in *Proc. 12th Annu. Conf. Genet. Evol. Comput.*, 2010, pp. 447–454.

[38] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, pp. 341–359, 1997.

[39] R. Thomsen, "Multi-modal optimization using crowding-based differential evolution," in *Proc. Congr. Evol. Comput.*, 2004, vol. 2, pp. 1382–1389.

[40] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. Conf. Genet. Evol. Comput. (GECCO)*, Washington, DC, USA, 2005, pp. 873–880.

[41] J. Ronkkonen, "Continuous multimodal global optimization with differential evolution based methods," Ph.D. dissertation, Dept. Inf. Technol., Lappeenranta Univ. Technology, Finland, 2009.

[42] K. C. Wong, C. H. Wu, R. K. P. Mok, C. Peng, and Z. Zhang, "Evolutionary multimodal optimization using the principle of locality," *Inform. Sci.*, vol. 194, pp. 138–170, 2012.

[43] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, 1995, pp. 1942–1948.

[44] S. Pasupuleti and R. Battiti, "The gregarious particle swarm optimizer (G-PSO)," in *Proc. Genet. Evol. Comp. Conf.* (GECCO), Seattle, Washington, USA, 2006, pp. 67–74.

[45] K. Deb, "An efficient constraint-handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, pp. 311–338, 2000.

[46] R. Battiti and G. Tecchiolli, "Learning with first, second and no derivatives: A case study in high energy physics," *Neurocompt.*, vol. 6, pp. 181–206, 1994.
[47] M. A. Luersen and R. Le. Riche, "Globalized Nelder–Mead method for engineering optimization," *Comput. Struct.*, vol. 82, pp. 2251–2260, 2004.
[48] H. Ghiasi, D. Pasini, and L. Lessard, "Constrained globalized Nelder–Mead method for simultaneous structural and manufacturing optimization of a composite bracket," *J. Compos. Mater.*, vol. 42, no. 7, pp. 717–736, 2008.
[49] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York, NY, USA: McGraw-Hill, 1987.
[50] T. H. S. Li, Y. H. Wang, C. C. Chen, and C. J. Lin, "A fast color information setup using EP-like PSO for manipulator grasping color objects," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 645–654, Feb. 2014.
[51] K. Deb and A. Kumar, "Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems," *Complex Syst.*, vol. 9, pp. 431–454, 1995.
[52] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Comput. Sci. Inf.*, vol. 26, pp. 30–45, 1996.

Chiu-Hung Chen received the B.Sc. degree in computer engineering from National Chiao Tung University, Shinchu, Taiwan, in 1990; the M.Sc. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1992; and the Ph.D. degree in engineering science and technology from the National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, in 2009.

He is currently an Assistant Professor with Kao Yuan University, Kaohsiung. From 1992 to 1999, he was a Manager with the Institute of Information Industry, Taipei. From 2000 to 2002, he was a Senior Manager with Photon Computer, Taipei. From 2003 to 2006, he was a Senior Manager with ULead Inc., Taipei. His research interests include evolutionary computation, multiobjective optimization, intelligent manufacturing, game intelligence, and multimedia system.

Tung-Kuan Liu received the B.S. degree in mechanical engineering from National Akita University, Akita, Japan, in March 1992, and the M.S. and Ph.D. degrees in mechanical engineering and information science from National Tohoku University, Sendai, Japan, in March 1994 and March 1997, respectively.

Currently, he is a Professor with the Mechanical and Automation Engineering Department, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan. From October 1997 to July 1999, he was a Senior Manager with the Institute of Information Industry, Taipei, Taiwan. From August 1999 to July 2002, he was an Assistant Professor with the Department of Marketing and Distribution Management, National Kaohsiung First University of Science and Technology. His research interests include artificial intelligence, applications of multiobjective optimization genetic algorithms, and integrated manufacturing and business systems.

Jyh-Horng Chou (SM'04) received the B.S. and M.S. degrees in engineering science from National Cheng-Kung University, Tainan, Taiwan, in 1981 and 1983, respectively, and the Ph.D. degree in mechatronic engineering from National Sun Yat-Sen University, Kaohsiung, Taiwan, in 1988.

Currently, he is the Chair Professor with the Department of Electrical Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, as well as a Distinguished Professor with the Institute of Electrical Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung. He has coauthored three books and published more than 250 refereed journal papers. He also holds five patents. His research interests include intelligent systems and control, computational intelligence and methods, automation technology, robust control, and quality engineering.

Prof. Chou was the recipient of the 2011 Distinguished Research Award from the National Science Council of Taiwan, the 2012 IEEE Outstanding Technical Achievement Award from the IEEE Tainan Section, the Research Award and the Excellent Research Award from the National Science Council of Taiwan 14 times, and numerous academic awards from various societies.