Robust Multiobjective Optimization via Evolutionary Algorithms

Zhenan He, Gary G. Yen, *Fellow, IEEE*, and Zhang Yi, *Fellow, IEEE*

***Abstract*—Uncertainty inadvertently exists in most real-world applications. In the optimization process, uncertainty poses a very important issue and it directly affects the optimization performance. Nowadays, evolutionary algorithms (EAs) have been successfully applied to various multiobjective optimization problems (MOPs). However, current researches on EAs rarely consider uncertainty in the optimization process and existing algorithms often fail to handle the uncertainty, which have limited EAs' applications in real-world problems. When MOPs come with uncertainty, they are referred to as robust MOPs (RMOPs). In this paper, we aim at solving RMOPs using EA-based optimization search. We propose a novel robust multiobjective optimization EA (RMOEA) with two distinct, yet complement, parts: 1) multiobjective optimization finding global Pareto optimal front ignoring disturbance at first and 2) robust optimization searching for the robust optimal front afterward. Furthermore, a comprehensive performance evaluation method is proposed to quantify the performance of RMOEA in solving RMOPs. Experimental results on a group of benchmark functions demonstrate the superiority of the proposed design in terms of both solutions' quality under the disturbance and computational efficiency in solving RMOPs.**

***Index Terms*—Evolutionary algorithms (EAs), multiobjective optimization, robust optimization.**

I. Introduction

OVER the years evolutionary algorithms (EAs) have demonstrated the success in effectively and efficiently solving multiobjective optimization problems (MOPs) [1]–[3]. However, most of these works do not consider *uncertainty* in the problem formulation. Yet, uncertainty exists in most real-world applications, e.g., measurement of overflow rate in water resource management [4], the variability of nominal geometry in aerodynamic design [5], deviations from the predicted processing time in scheduling [6], disturbed thickness control in car side impact design [7], the inevitable small amount of production errors in manufacturing [8], the varied material temperature in electromagnet experiment [9], to name a few. During the real-world optimization process, uncertainty is an unavoidable issue and it directly affects the optimization performance. Based on the previous definition in [5] and [9]–[13], in this paper we classify the uncertainty in optimization problems into three classes. The first type of uncertainty is the disturbance added on decision variables, which imposes perturbations on decision variables. For instance, the imperfection of a machine's actuator belongs to this type [5], [11]. The second type involves the noise affecting the objective (fitness) evaluations [12], where the noise can be sensory measurement errors from different sources [11] or approximation errors due to inaccurate modeling [5], [10], [11]. In general, this type of uncertainty generates errors in performance estimation [13]. The last type of uncertainty comes from the fluctuation of environmental parameters subject to varying environmental and operational condition, which causes the specifications of optimization problem change over time [10], e.g., the optimization problem contains time-variant Pareto optimal front and time-dependent fitness functions.

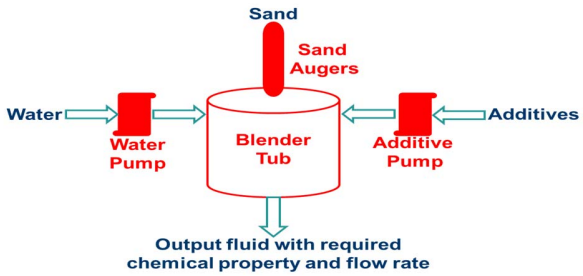

Fig. 1. Simplified example of the blender system.

Let us look at a real-world example in oil and gas industry. The blender system shown in Fig. 1 is an important part of fracturing stimulation [14], which plays a crucial role in oil and gas production. The blender system equipped with several different types of sensors mixes water, chemical additives, and sand together and outputs fracturing fluid with required chemical property and flow rate. The amount of these materials is determined by a calculation model which is generated from the ad-hoc experience of engineers. Here, the inaccuracy of the pump and augers creates disturbance on each machine's

Manuscript received May 24, 2017; revised June 3, 2018; accepted July 20, 2018. Date of publication July 25, 2018; date of current version March 29, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61602328, Grant 61625204, Grant 61432012, and Grant 61525302, in part by State Key Laboratory of Synthetical Automation for Process Industries under Grant PAL-N201707, and in part by the Fundamental Research Funds for the Central Universities under Grant 20822041B4106. *(Corresponding author: Gary G. Yen.)*

Z. He and Z. Yi are with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: zhenan@scu.edu.cn; zhangyi@scu.edu.cn).

G. G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74075 USA (e-mail: gyen@okstate.edu).

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/TEVC.2018.2859638

rate, which belongs to the first type of uncertainty. Meanwhile, the sensory readings from each sensor instrument add noise to the system and the approximation of calculation model incorporates erroneousness into the model, both of which contribute to the second type of uncertainty. Finally, the varying downhole environment and field condition change over time, which brings to the third type of uncertainty.

As stated in [10], [11], and [15], if one optimal solution is quite sensitive to perturbations, its implementation into real problems may be different from it and the implemented solution cannot achieve the same optimal performance as the theoretical optimal one, e.g., in the last example, the deviation of pump's rate from the optimal setting makes the blender loss of discharge pressure to the pumps, which leads to forever mechanical damage to fluid ends and power ends. Therefore, in real practice, experienced engineers always prefer the most conservative design to ensure the safety, while sacrificing the optimality. However, this is a huge waste of resource and it ultimately leads to a miss in the target.

Therefore, in order to fully exploit the value of available resource and achieve the possible best optimum while minimize the influence of uncertainty, there is a need to develop efficient approaches for optimization under uncertainty. Previous works have made some attempts, especially the robust optimization is a process of searching for robust optimal solutions which achieves the best tradeoff between optimization and robustness. Robustness implies some insensitivity degree of a solution to perturbations [11], [13], [15], [16]. Under this spirit, a solution is robust if it is insensitive to slight disturbance [11], [13], [15], [16].

In literature, a major part of robust optimization approaches focuses on the first type of uncertainty and tries to search for robust solutions suffering perturbations on decision variables. Especially in [15] and [17], the concept "robust optimization" directly refers to searching for robust solutions subject to perturbations on decision space. Under the second type of uncertainty, optimization problem with objective functions affected by noise is often characterized by the interval optimization problem [18]–[21], where uncertainty can be quantified as intervals in the objective space. The set-based EAs are applied [18]–[23] to solve these problems. Other approaches dealing with this type of uncertainty can also be found in [12]. Finally, for the third type of uncertainty, in the varying environmental condition, the dynamic optimization [24]–[28] tries to detect the change in the dynamic environment and reoptimize the new varied Pareto front when the change is detected [24]–[28]. The new search exploits information from the previous search space and the diversity of population is maintained in order to quickly approach the new optimum [10]. Recently, robust optimization over time (ROOT) is proposed in [29] in order to handle dynamic optimization problems more practically, where an optimization algorithm finds an acceptable solution and updates it slowly over time, rather than search for the global optimal solution frequently. In ROOT, the change of environment is assumed to be not random and further change of environment can be predicted and estimated based on the analysis of previous environmental condition [29].

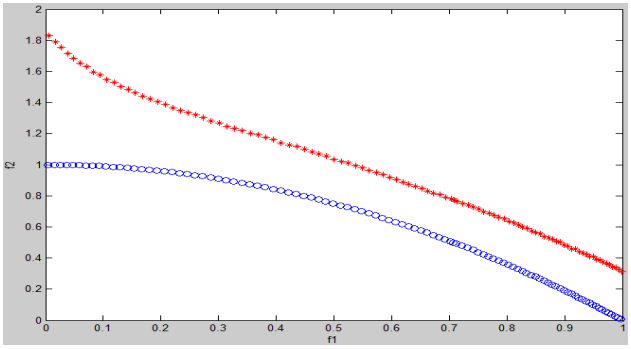

Fig. 2. Pareto front changes under disturbance.

In this paper, we primarily focus on developing robust optimization algorithm to solve MOPs with uncertainty on decision variables. Here, we call this problem robust MOP (RMOP). The goal of robust optimization algorithm is to find robust optimal solutions, which can achieve the best optimality with certain degree of perturbations on decision variables, rather than the global Pareto optimal front.

Fig. 2 shows by adding slightly disturbance on each decision variable, how the original Pareto front of 2-D TP1 [15] changes. The line in "blue circle" represents the original Pareto front, while the line in "red star" represents the new front changed from the original Pareto front after the disturbance is imposed upon each decision variable, where the maximum disturbance degree is set as 0.01 while the range of decision variables is [0, 1] for the first one and $[-1, 1]$ for others. From Fig. 2, the performance of Pareto front deteriorates appreciably and each solution on the front is no longer converged.

Generally, there are three issues about optimization of RMOP. First, the representation of each solution's robust optimal degree in an MOP remains difficult, especially for those approaches in single-objective optimization problems. Without a valid representation, it is difficult to measure the quality of solutions in terms of both optimality and robustness. Therefore, an efficient way for the representation of robust optimal degree is required, where both the change of convergence and diversity of solutions under uncertainty need to be considered. Second, the final robust optimal front should be balanced between robustness and optimality while many existing methods either sacrifices optimality to achieve robustness or ignores robustness to preserve optimality. Therefore, it is necessary to develop an effective algorithm to balancing robustness and optimality so that the final robust optimal front can maintain convergence and diversity under disturbance. Third, in the performance evaluation, whether the robust optimal performance of a solution or the whole robust Pareto front should be measured based on the worst case scenario or average case of all scenarios under the disturbance is still questionable. Actually, either of them merely reflect part of robust performance and a comprehensive robust optimal evaluation approach integrating all scenarios is absolutely needed.

To overcome these deficiencies and effectively solve RMOPs, in this paper we have developed a novel robust multiobjective optimization EA (RMOEA). This algorithm includes two parts: 1) multiobjective optimization finding global Pareto optimal front ignoring the disturbance and 2) robust optimization searching for the robust optimal front. In multiobjective optimization, besides the obtained Pareto optimal solutions, all visited solutions in the evolutionary process are recorded and organized in both decision space and objective space. In the robust optimization, an advanced detection method is proposed to effectively measure the robustness of the region and directly search for the robust region. Then, a group of robust optimal regions are picked up among all robust regions so that these selected regions together constructing a robust optimal front which achieves the best convergence and diversity performance under the disturbance. Finally, a comprehensive performance evaluation method is developed. The main contributions of this paper is summarized as follows.

1) The proposed decomposition framework draw appreciable improvements in representation of each solution's (region's) convergence and diversity performance for multiobjective optimization under the disturbance.
2) A delicate balance in solutions' robustness and optimality performance has been achieved. The process of searching for Pareto optimal solutions and robust optimal solutions are separated and each process run independently without any constraints imposed from the other.
3) Assisted by organization of all visited solutions at both decision space and objective space in multiobjective optimization as well as archive updating in robust optimization, a fast and efficient means to directly search for robust regions in the decision space rather than the single solution is obtained.
4) One important advantage of EA has been intensely exploited: population-based EA with distributed nature makes solutions with different robust degrees located in the same neighborhood. That is, there may exist nonrobust optimal solution next to robust solutions. Under some scenarios, nonrobust optimal solution can perform better and in other cases, it is replaced by robust optimal solutions in the same neighborhood without degrading performance. Therefore, the total performance is improved among all scenarios.
5) A comprehensive performance evaluation method to quantify both optimality and robustness of an approximate front has been constructed. Besides the front, we also consider about the performance of each solution. In literature, as we know, there is no such a method on performance evaluation in the RMOPs.

The remaining sections complete the presentation of this paper. Section II provides a literature review for the general formulation of RMOP as well as existing research on robust optimization. Our proposed selection strategy and its details are explained in Section III. In Section IV, we elaborate on the experimental results given selected benchmark problems. Finally, the conclusion is given in Section V along with pertinent observations.

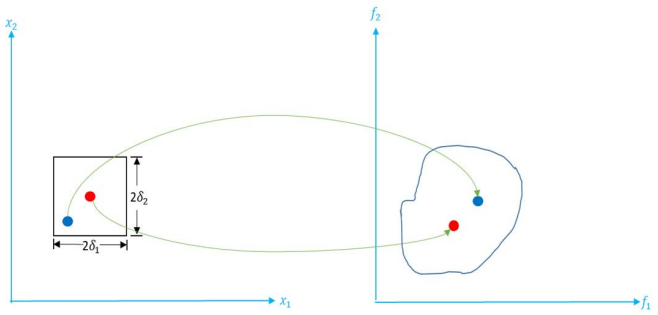

Fig. 3. Illustration of a solution in both decision space and objective space.

II. Literature Review

This section introduces the general formulation of RMOPs and summarizes existing researches on robust optimization.

A. Robust Multiobjective Optimization Problems

First, let us look at the general form of MOP under all three types of uncertainty. From [5], [10], and [11], the general form can be represented as

$$\begin{aligned} &\text{minimize } F(x) = \left(f_1'(x', c'), \ldots, f_M'(x', c')\right)^T \\ &\text{with} \quad x' = (x_1 + \delta_1, \ldots, x_n + \delta_n)^T \\ &\qquad\; c' = (c_1 + v_1, \ldots, c_k + v_k)^T \\ &\qquad\; f_{1:M}'(x', c') = f_{1:M}(x', c') + e_{1:M} \\ &\text{subject to } x \in \Omega \end{aligned} \tag{1}$$

where Ω is the decision space and $x = (x_1, \ldots, x_n)^T$ is the vector of decision variables with dimension n. $\delta = (\delta_1, \ldots, \delta_n)^T$ is the vector of the disturbances added on each decision variable which represents the first type of uncertainty. M denotes the number of objectives and $e = (e_1, \ldots, e_M)^T$ refers to the noise on function evaluation for each objective and belongs to the second type of uncertainty. Finally, $c = (c_1, \ldots, c_k)^T$ is the environmental parameters and k is the number of parameters in the model. The perturbation under this type of uncertainty is represented as $v = (v_1, \ldots, v_k)^T$.

For RMOP in this paper, only the disturbance imposed upon each decision variable is considered. Then, as shown in Fig. 3, a solution is represented as a region in both decision space and objective space. The size of region in decision space is consistent for each solution but varied in the objective space, e.g., from Fig. 3, for both solutions containing the equal size of region in the decision space, the red one with the smaller size of region in the objective space is more robust than the blue one.

The general form of RMOP of our interest is

$$\begin{aligned} &\text{minimize } F(x) = \left(f_1(x'), \ldots, f_M(x')\right)^T \\ &\text{with} \quad x' = (x_1 + \delta_1, \ldots, x_n + \delta_n)^T \\ &\text{subject to } x \in \Omega \end{aligned} \tag{2}$$

given the maximum disturbance degree $\delta^{\max} = (\delta_1^{\max}, \ldots, \delta_n^{\max})^T$. For each $x \in \Omega$, there exists a region $\rho(x) \subset \Omega$ centered at x. For $\rho(x)$, among all points $\bar{x} \in \rho(x)$, under a defined performance measurement, the worst and mean performances are defined as $S^w(\rho(x)) = \text{worst}_{\bar{x}\in\rho(x)} S(F(\bar{x}))$ and $S^m(\rho(x)) = \text{mean}_{\bar{x}\in\rho(x)} S(F(\bar{x}))$, respectively. Performance function S can be Pareto dominance or any aggregation functions, such as weighted sum [3] or penalty-based boundary intersection (PBI) [3]. According to [10] and [15], based on the general form of RMOP in (2), the definition for multiobjective robust optimal solution is given as below.

1) Multiobjective Robust Optimal Solution: A solution $x \in \Omega$ is a multiobjective robust optimal solution under a defined performance measurement S, if $\forall y \in \Omega$, $y \neq x$, $\rho(y) \subset \Omega$ centered at y, such that $S^w(\rho(x))$ is no worse than $S^w(\rho(y))$ and $S^m(\rho(x))$ is no worse than $S^m(\rho(y))$.

This definition considers both the worst case scenario and average case of all scenarios for a solution's performance under the disturbance. According to this definition, the proposed robust algorithm design attempts to find the robust optimal solutions with good performance in both the worst case scenario and average case of all scenarios.

B. Existing Researches on Robust Optimization

In literature, in solving robust optimization under the first type of uncertainty, previous research works have made some attempts on the design of sampling methods, quantification of robustness, and development of optimization algorithms.

First, the most directly and simplest sampling approach is randomly generating several solutions around neighborhood of the current solution. However, this strategy creates randomness by visiting the same solution several times [15], [30]. There are three main types of statistical sampling methods which aim at eliminating randomness and efficiently applying to robust optimization problems. The first type of methods divides the neighbors of the solution to be evaluated (the size of the neighbors is determined by the disturbance degree on each decision variable) into several equal-sized grids and do the sampling in each grid, so that the variance among each sampling time is reduced [16], e.g., stratified sampling [31] and Latin hypercube sampling (LHS) [32]. The second type of methods, such as importance sampling [33] and methods evaluating important solutions more frequently [16], [34], tries to evaluate some important solutions (e.g., solutions with better fitness) more often than others. From [16], since these better solutions are more likely to survive, more frequently evaluation of them can provide a more accurate estimate. The strategy that dynamically adjusting the sample size for each solution [34] also belongs to this type, e.g., a larger sample size is applied on better solutions [35], [36]. The last type tries to avoid the expensive computation of fitness evaluation and uses historical solutions in the neighbor generated during the evolutionary process, including history record method [16], [34] and computational efficiency model [16]. From [10] and [34], the expected fitness of the current solution can also be estimated by its similar solutions evaluated in the past. All sampling methods devote to provide a group of solutions with less number of function evaluations for determining true robustness of solutions.

In researches related to quantification of robustness, there are three different categories. First, the explicit averaging method with a large number of function evaluations estimates expectation and variance values of one solution by integrating all solutions' fitness values in its neighborhood. Monte Carlo integration in [8], [10], and [16] is of this type. Similarly, in [34], weights are assigned to each solution according to probability distribution of the disturbance and then fitness values of all solutions are accumulated together. Thus, this weighted average fitness values is applied as the expectation of solutions [16]. Then, the relation among several statistical indexes, e.g., mean value, standard deviation, and variance, are investigated and these indexes are combined to reflect the robustness degree. For instance, the ratio between mean value and variance [13], the integration of mean value and standard deviation [37], [38], six-sigma quality measures [13], and the difference between the current solution and its mean effective objective functions [15] belong to this type. The second category consists of strategies widely used in reliability-based optimization, where the optimization problem involves constraints. The reliability of a solution is based on the distance between the solution and the closest constraint boundary in the objective space. The smaller the distance, the less robust the solution is, e.g., most probable point (MPP) approach [39]. Variants of MPP includes performance measure approach and fast performance measure approach [24]. The first order reliability method and the second order reliability method [13], [39], [40] also belong to this type. Then, sensitive analysis is conducted, e.g., sensitivity analysis based on gradient information and computation of sensitive region based on acceptable performance variation region. Furthermore, the probability of success in satisfying design requirement or constraints over several generations is calculated to measure the robustness of the solution [9], [41]. Third, the modification of Pareto dominance considers both optimality and robustness. In [41] and [42], probabilistic dominance classifies solutions into different ranking. Besides, the r-dominace based on set coverage metric makes a comparison between two solutions [43]. Furthermore, the dominance robustness is defined as solution's ability to retain the Pareto-optimal front under perturbations [17].

Finally, there are three different types of algorithms developed for robust optimization. First, EAs come with surrogate model for fitness approximation. In [16], there are single, nearest, and ensemble models applied for fitness approximation. Also, a max–min surrogate-assisted EA is designed for robust engineering [5]. Recently, a regularity model to handle noisy MOP is introduced in [44]. Second, the multiobjective framework is used to achieve both optimality and robustness [5], [9], [10], [45]–[47], e.g., the variance is used as an additional objectives [47] and modified Pareto dominance considers both optimality and robustness [30]. Then, the original RMOP is extended to many-objective optimization since more objectives are added [13]. Third, the robust constraints are added into the process of EAs, e.g., the prescribed robustness EA [9] with robust constraint-based fitness function. In

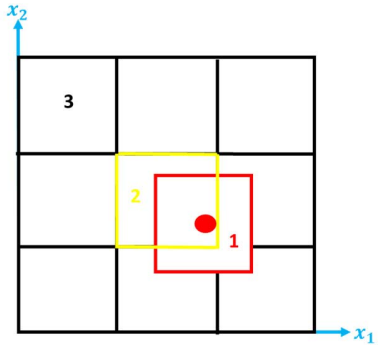

Fig. 4. Grid setting in the decision space.

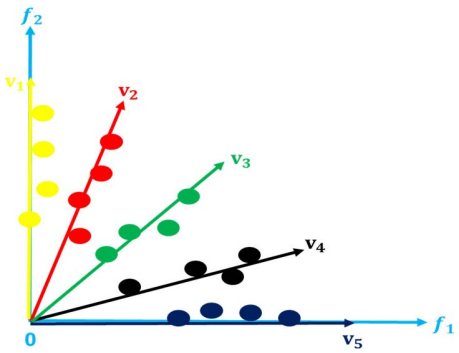

Fig. 5. Decomposition in the objective space.

addition to the above algorithms, traditional MOEAs with effective robustness quantification are also used in solving the robust optimization problems [13], [15], e.g., NSGA-II [1] was tested on a group of benchmark problems in [15], and MOEA/D [3] combined with six-sigma quality-measures proposed in [13] was applied to solve real robust engineering design problems.

In the next section, the proposed RMOEA present innovations on these three steps.

III. Proposed Method

There are two distinct, yet complement, parts in the design of RMOEA. First, the multiobjective optimization contains three steps: 1) searching for Pareto optimal front while ignoring disturbance; 2) performing grid-setting in the decision space; and 3) constructing decomposition in the objective space. Second, robust optimization involves performance estimation of optimal solutions under disturbance, archive update, robust region detection, and selection of robust solutions to construct the robust optimal front. Finally, a comprehensive performance evaluation method is proposed to measure both optimality and robustness of the approximate robust optimal front.

A. Multiobjective Optimization

First, an MOEA is applied to search for the global Pareto optimal front, where the number of optimal solutions is N. This MOEA can be any state-of-the-art design which has been successfully applied in solving MOPs, such as NSGA-II [1], SPEA2 [2], and MOEA/D [3], to name a few. In this paper, NSGA-II is chosen as a case study.

During the evolutionary process, in order to improve the efficiency of the algorithm, beside the original steps of NSGA-II, two additional steps are added to organize all visited solutions in both decision space and objective space, which help to accelerate the search process in the next robust optimization. First, by grid-setting method, the decision space is divided into multiple equal-sized hyperboxes, the size of which in each dimension equals to two times of disturbance degree on that dimension. Then, for each visited solution as shown in Fig. 4, there are three types of hyperboxes: 1) Hyperbox 1 (in red) is the hyperbox with the center to be each solution; 2) Hyperbox 2 (in yellow) is the hyperbox, where the solution is inside; and 3) Hyperbox 3 (in black) is the hyperbox in the neighborhood of Hyperbox 2. If a hyperbox intersects with Hyperbox 2, then this hyperbox is inside the neighborhood of Hyperbox 2. Here, the intersection part between Hyperboxes 2 and 3 can be a point, a line, or a hyper-plane.

The objective space is decomposed by a set of predefined well distributed weight vectors, the size of which is set as N and equals to the number of optimal solutions. Each visited solution is associated with one weight vector which is closest to this solution among all weight vectors. Each vector has its own archive to store all solutions associated with it. The closeness is determined by the angle between the vector and solution. Fig. 5. shows an example of decomposed 2-D objective space, where each color represents one vector. For instance, each solution in red is associated to the vector v_2 and the archive of vector v_2 contains four red solutions. Then, the optimal solution along each vector is calculated by the PBI method [3], [48], which measures the performance of solutions to the single-objective problem specified by each vector. From [3], the general form of PBI is written as

$$\begin{aligned} &\text{minimize} \quad g\left(x|\lambda, z^*\right) = d_1 + \theta d_2 \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{3}$$

where the value of penalty parameter θ is set to five according to [49], $d_1 = \|(z^* - f(x))^T\lambda\|/\|\lambda\|$ and $d_2 = \|f(x) - (z^* - d_1\lambda)\|$, λ represents one weight vector. For each λ, the solution with the minimum PBI is considered the optimal solution along λ. The whole process of multiobjective optimization step is summarized in Algorithm 1.

B. Robust Optimization

After the first step, for each vector i in the objective space, its archive, $Archive_i$, and the optimal solution s_i along it are obtained. The set of optimal solutions S consists of all optimal solutions along each vector. In the beginning, each solution in $Archive_{1:N}$ is normalized through subtracting its objective values by the obtained ideal point z^*, where each dimension of z^* is the best value found among all solutions on this dimension (objective). After that, the performance of each optimal solution under the disturbance is tested, which determines how to update each archive in the next step. Then, the robust

Algorithm 1 Multiobjective Optimization

Input:

N: population size
n: the number of decision variables
$v_{1:N}$: predefined weight vectors with size N
$\delta_{1:n}$: the maximum disturbance degree on each decision variable

1) Generate Pareto optimal front PF ignoring disturbance

P: the set of all visited solutions in the evolutionary process

2) Grid-setting in the decision space

for $k = 1{:}n$
The length of the grid in ith dimension= $2\delta_k$
end

3) Decomposition in the objective space

for $i = 1{:}N$
For each vector v_i, construct its archive:
Archive$_i$ = $\{p \in P | v_i$ is the closest vector for $p\}$
The optimal solution along v_i:

$$s_i = \left\{ p \in Archive_i \middle| p = \underset{t \in Archive_i}{\operatorname{argmin}} PBI_t \right\}$$

end

Output:

The set of optimal solutions: $S = \{s_1, s_2, \ldots, s_N\}$
Each vector's archive: $Archive_{1:N}$

region is detected and assigned to different vectors. Finally, the robust optimal front is constructed. The framework of robust optimization step is shown in Algorithm 2.

Please note, the search for robust optimal solution is along every weight vector, whose archive contains all visited solutions generated at every generations during the evolutionary process with different convergence degrees. Even though the robust optimal front locates far from the Pareto optimal front, the combination of all archives can cover the position of the true robust optimal front. Thus, the search process can still find the robust optimal front.

1) Performance Estimation of S Under the Disturbance: The performance of S is tested by sampling around the neighborhood of each solution on S, which is a hyperbox with the size of each side equals to the maximum disturbance degree δ on the corresponding dimension. LHS method [32] is applied to generate 50 solutions, all of which are sorted by their PBI values under the same associated vector. The worst solution with the largest PBI values is labeled as the worst case of the optimal solution for this vector while the mean value of them is recorded as the average case of this solution, e.g., in Algorithm 3, Ws_i is the worst case of s_i and Ms_i is the average case of s_i, where $s_i \in S,\ i \in \{1, \ldots, N\}$.

2) Archive Updating: The archive updating is conducted based on the worst cases of all solutions in S (e.g., $Ws_{1:N}$ in Algorithm 2). In each vector's own archive, the solution with its PBI values larger than the worst case of the optimal solution is eliminated from the archive while solutions with smaller PBI values are kept. For each vector, searching its associated robust regions starts from each solution in the updated archive. After this step, the size of each updated archive is controlled to a small number so that the computational costs of the following steps, "robust region detection" and "construct the robust optimal front" are not as expensive.

3) Robust Region Detection: Along every weight vector, for each solution in the archive, its associated three types of hyperboxes in the decision space are generated. The first two types of hyperboxes can be directly determined by the location

Algorithm 2 Robust Optimization

Input:

N, n, $v_{1:N}$, $\delta_{1:n}$
$S = \{s_1, s_2, \ldots, s_N\}$: the set of optimal solutions
$Archive_{1:N}$: each vector's archive
T: the size of neighborhood

1) Performance estimation of S under the disturbance

$i = 1{:}N$
Around the neighborhood of s_i, sample 50 solutions: $s_i^{t=1:50}$
Record the PBI value of each s_i^t along vector v_i,
Calculate s_i's worst performance: $Ws_i = \max_{t=1:50} PBI_{s_i^t}$
Calculate s_i's average performance: $Ms_i = \operatorname{mean}_{t=1:50} PBI_{s_i^t}$
end

2) Archive updating

for $i = 1{:}N$
$New_{Archive\,i} = \left\{ p \in Archive_i | PBI_p \le Ws_i \right\}$
end

3) Robust region detection

for $i = 1{:}N$
for $j1 = 1{:}$size$(New_{Archive\,i})$
$H =$ collection of its 12 hyperboxes
for $j2 = 1{:}12$
$a_{1:50} =$ the sampled 50 solutions inside the hyperbox
for $j3 = 1{:}T$
for $j4 = 1{:}50$
Calculate a_{j4}'s PBI value under v_{j3}: PBI_{j4}
end
$MPBI_{j3} = \text{mean}\{PBI_1, \ldots, PBI_{50}\}$
$WPBI_{j3} = \max\{PBI_1, \ldots, PBI_{50}\}$
if $MPBI_{j3} \le Ms_{j3}$ and $WPBI_{j3} \le Ws_{j3}$
Hyperbox $j2$ is robust to vector $j3$.
end
end
Assign this hyperbox to the mth vector in neighbor:
$m = \operatorname{argmin}_m MPBI_{m=1:T}$
This hyperbox $j2$ is robust to vector m.
end
end
end

4) Construct the robust optimal front

for $i = 1{:}N$
$R =$ all robust regions associated with $\mathrm{v_i}$
if size $(R) \ge 1$
$r_i = \operatorname{argmin}_r MPBI_{r \in R}$
else
$A_1 =$ collected robust regions associated with $v_{1:T}$
$A_2 = \{c \in A_1 | WPBI_c \le Ws_i\}$
if $A_2 \notin \emptyset$
$r_i = \operatorname{argmin}_r MPBI_{r=1:size(A_2)}$
else
keep the Pareto optimal solution or no solution is selected
end
end
end

Output:

The robust optimal front $RF = \{r_1, \ldots, r_N\}$

of the solution. For the third type, since the number of hyperboxes in the neighborhood grows drastically as the number of decision space's dimensions grows, we randomly pick up ten hyperboxes in its neighborhood. So the total number of associated hyperboxes is 12 for each solution and LHS method [32] does sampling in each hyperbox.

For each hyperbox, there are two important questions to answer. First, how to quantify the hyperboxes' robustness and detect whether it is a robust region? Second, which vector this hyperbox should be assigned to? If one hyperbox is assigned to multiple vectors, the diversity of final robust optimal front will be degraded. For the first issue, we will check the PBI value of all solutions inside the same hyperbox along the current

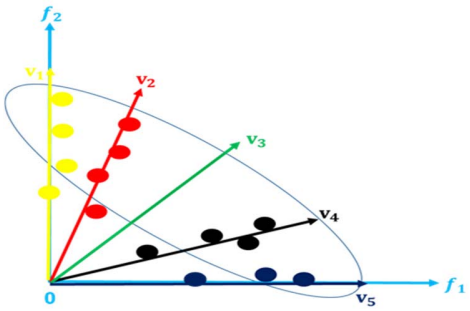

Fig. 6. Choose other nonselected robust solutions.

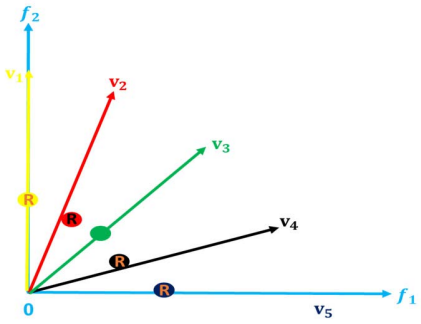

Fig. 7. Nonrobust Pareto optimal solution is kept.

weight vector and vectors in its neighborhood, which consists of T closest vectors for the current vector in the objective space. For each vector, the average and the worst of their PBI values are compared with the average case and the worst case of the optimal solution associated with the same vector, e.g., consider a region j_2 under vector v_i, MPBI_{j_2} is compared with Ms_i and WPBI_{j2} is compared with Ws_i. If MPBI_{j2} is smaller than or equal to Ms_i and WPBI_{j2} is smaller than or equal to Ws_i, then this region is considered as the *robust region* of this vector. When the center point of robust region is mapped to the objective space, the mapped solution is called *robust solution* of this vector. This detection rule ensures the robust region achieves better "mean performance" and "worst performance" than the optimal solution associated with this vector. For the second issue, we find all vectors in the neighborhood for which this hyperbox is robust. Among all these vectors, the vector in which the hyperbox achieves the best MPBI value is selected and this hyperbox is assigned to this vector. If this hyperbox is not robust for any vectors in the neighborhood, then it is discarded. After this step, each robust region has been assigned to a vector. However, some vectors may be assigned to several regions while others contain no robust region. So we need another step to rearrange the robust regions.

4) Construct the Robust Optimal Front: In this step, we first scan each vector. If one vector contains several robust solutions, the solution whose corresponding region in decision space contains the best (smallest) MPBI is selected as the *robust optimal solution*. After the first scan, if one vector does not contain a robust optimal solution, we check other nonselected robust solutions associated with vectors in its neighborhood. For example, as shown in Fig. 6, the vector v_3 does not contain a robust solution, then all nonselected robust solutions (which are enclosed by the blue circle in the figure) are considered. Then, all these solutions whose regions' WPBI is smaller than the worst case of the optimal solution associated with this vector are retained. Among these retained solutions, the solution whose region with the best MPBI is selected. Similar to the robust region detection step, here we also consider both mean performance and the worst performance of each region. If still no robust region can be picked up, its original optimal solution is kept when any of its three closest neighbor vectors contain robust region (solution). Otherwise, no solution is selected for this vector and the direction associated with this vector does not contribute any solutions to final robust optimal front. Solutions within these closest neighbor contain the most similar preference information, which can replace the optimal, but nonrobust solution in most uncertainty cases. In some other cases, the nonrobust solution can still perform better and improve the performance in these special scenarios. Therefore, the switch between the nonrobust solution and robust solution in its neighbor can ensure a better performance of population. As shown in Fig. 7, the nonrobust optimal solution of v_3 (the solid ball with green color) is kept since robust solutions (solid balls with label R) exist in its neighborhood. Thus, the *robust optimal front* is a front which consists of two parts of solutions, a large part of them is robust optimal solutions and a small part is nonrobust optimal solutions whose three closest neighbor vectors contain robust solutions.

In summary, for a region R in the decision space and a weight vector v in the objective space, after all points sampled in R are mapped to the objective space, if their worst and mean PBI values corresponding to v are smaller than or equal to the worst and mean performance values of the optimal solution associated with v, respectively, then R is identified as the *robust region of* v. When the center point of R is mapped to the objective space, the mapped solution is called *robust solution of* v. Among all robust solutions of v, the solution whose corresponding robust region in decision space contains the best (smallest) mean PBI performance value is selected as its *robust optimal solution of* v. Finally, a *robust optimal front* is a front containing two types of solutions, a large number of robust optimal solutions and a small number of nonrobust optimal solutions whose three closest neighbor vectors contain robust solutions.

C. Performance Evaluation

In order to evaluate the performance of a robust optimal front, we propose three new metrics to evaluate its performance under the disturbance as shown in Algorithm 3: S_IGD considers the change of robust optimal front in each scenario, S_PBI provides the optimization degree along each vector during the test, and P_change measures the percentage of each robust optimal solution retaining nondominated to all other solutions among all scenarios.

First, the performance of the robust optimal front is tested 1000 times under the disturbance. At the end of each testing time, a new front is generated by the change of each

Algorithm 3 Performance Evaluation

Input:
$RF = \{r_1, \ldots, r_N\}$: the robust optimal front
$\delta_{1:n}$: disturbance degree on each decision variable
tt: the number of testing times under the disturbance
1) Performance estimation of RF under the disturbance $\delta_{1:n}$
for $t = 1{:}tt$
for $i = 1{:}N$
r_i changes to r_i^t and record the norm of r_i^t : $\|r_i^t\|$
end
Calculate the IGD value of $RF' = \{r_1^t, \ldots, r_N^t\}$: IGD_t
Calculate the average PBI value among $v_{1:N}$ under RF' : PBI_t
end
2) *S*_IGD metric: get the average and worst IGD value
$IGD_{mean} = \underset{t=1:tt}{\text{mean}} IGD_t$ and $IGD_{worst} = \underset{t=1:tt}{\max} IGD_t$
3) *S*_PBI metric: get the average value
$PBI_{mean} = \underset{t=1:tt}{\text{mean}} PBI_t$
4) P_c*hange* metric
for $i = 1{:}N$
Among all tt generations,
Calculate the percentage of r_i to be non-dominated
end
Output:
S_IGD, *S_PBI*, *P_change*

solution under the disturbance. Then, the inverted generational distance (IGD) [50] value of this front is recorded. When all testing are finished, *S*_IGD consists of the average IGD value, and the worst IGD value over all 1000 trials. This aspect measures the statistical performance of the whole front. Compared with the similar application of IGD metric in dynamic optimization problem [24]–[28], where the reference front also changes in each generation, *S*_IGD uses a fixed reference front consists of optimal solutions for the original optimization problem without the disturbance and records the standard deviation of IGD value, worst IGD value among all generations. Second, after each testing, for each vector, find its best (smallest) PBI value any solution can achieve, which reflects how each single-objective problem specified by this vector is optimized at this time, or what the optimization degree along this direction is. Since each single-objective problem can represent specific preference information, this aspect reveals how each different decision makers' preference can be satisfied. Then, the mean PBI value among all solutions is recorded at the end of each testing. We collect the average PBI value, and the worst PBI value over all 1000 tests into *S*_PBI. Finally, among all generations, the percentage of each robust optimal solution remaining nondominated to all other solutions is measured by *P*_change, which evaluates how each solution stays on the current Pareto optimal front among all scenarios.

In summary, all three metrics break the limitation of existing evaluation methods which mainly focus on mean and variance of performance. Here, we place more emphasis on the performance at each scenario under the special degree of perturbation. Then, the comprehensive evaluation is the integration of performance at each scenario. Furthermore, each metric plays a complement role to others and all of them together achieve a comprehensive evaluation. *S*_IGD concerns the performance of the whole front, while *S*_PBI focuses on how each direction in the objective space is optimized. Finally, *P*_change provides information about how each solution remains nondominated to others. Thus, the evaluation process contains three different elements: 1) direction in the objective space; 2) the whole Pareto front; and 3) each single solution.

IV. Experiment Results

A. Benchmark Problems

Nine 2-D multiobjective benchmark functions TP1-9, where the first four functions provided by Deb and Gupta [15] and the latter five by Gaspar-Cunha *et al.* [30], are chosen for empirical studies and both of them have been applied to test the algorithms' robust performance in literature. In addition to robustness testing, these problems also contain a variety of problem characteristics presenting various degrees of difficulties to test the performance of algorithms, e.g., TP1-2 are multimodal problems while TP3-4 involve the local fronts.

B. Experimental Setting

In RMOEA, the size of uniformly distributed weight vectors is set as the same as the population size 100. Initial populations are generated randomly from the search space in NSGA-II. The simulated binary crossover (SBX) and polynomial mutation are used. The distribution indexes in SBX is set as 0 in TP1-4 and 20 in TP5-9 while it in the polynomial mutation is set to be 20. The crossover rate is 1.00, while the mutation rate is $1/n$, where n is the number of decision variables. At the multiobjective optimization part, the number of generations for is 3000 for TP1-4 and 2000 for TP5-9. In the robust optimization step, the size of neighborhood is set as 10.

C. Experiment Findings With Varied Disturbance Degrees

In this experiment, RMOEA is tested in TP1-4 under three different disturbance degrees, where the maximum degrees on each decision variable have been chosen as 0.015, 0.025, and 0.035, respectively. The number of testing times in each benchmark function is set as 1000. As shown in Fig. 8, three plots are generated for each benchmark function. In each plot, front 1, front 2, and front 3 correspond to maximum disturbance degree 0.015, 0.025, and 0.035, respectively. The left plot shows different robust optimal fronts with the original Pareto optima front, the center plot exhibits the worst case of robust Pareto front during 1000 trials (which achieves the largest IGD_{worst} values among all 1000 testings), and the right plot shows the mean PBI value for each weight vector (PBI_{mean}) during the test. Fig. 9 presents the *P*_change result on each benchmark functions, which considers the ability of each solution remaining nondominated during the test.

From Fig. 8, as the disturbance degree increases, objective values for the worst case of robust front and mean PBI values for each weight vector are also increased while the percentage of each solution remaining nondominated is decreased. In Fig. 9, the larger the disturbance degree, the worse the ability of most solutions to retain nondominated. However, even if the disturbance degree varies, the comparison is conducted among the same solutions. It seems that the larger disturbance makes

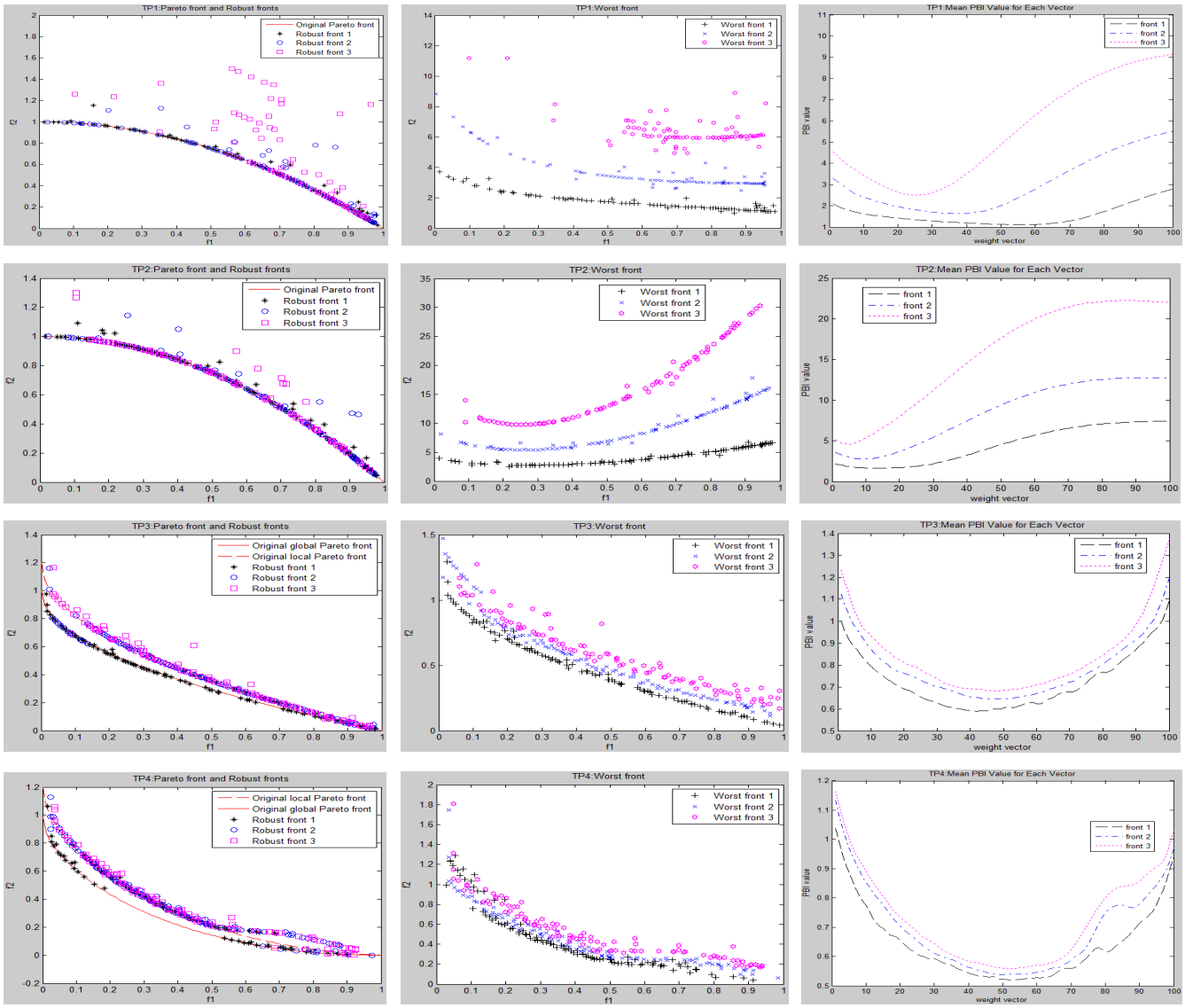

Fig. 8. Experiment results on S_IGD and S_PBI in TP1-4 under three different maximum disturbance degrees of 0.015, 0.025, and 0.035.

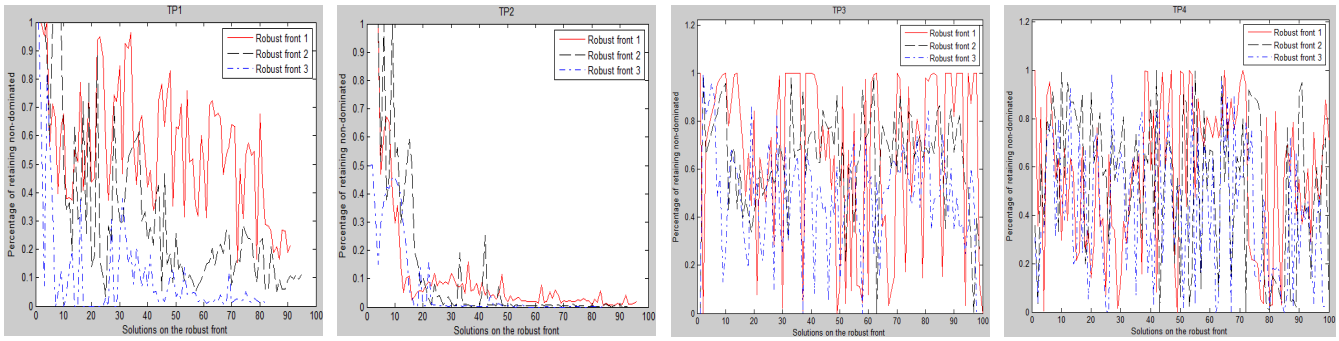

Fig. 9. Experiment results on P_change in TP1-4 under three different disturbance degrees.

the situation more and more unpredictable. Therefore, the larger disturbance degree causes more serious perturbations on the average performance. On the other hand, in each problem the robust front remains in nearly the same position under different disturbance degrees, which means robust regions are still consistent even though the disturbance degree is changed.

In TP3-4, as the disturbance degree increases, the robust front moves from global Pareto optimal front toward local Pareto optimal front, which was also reported in [15] and [30]. In TP2, Fig. 9 presents that most solutions are dominated by others during the whole test which corresponds to the results in Fig. 8, where most solutions on the worst case of robust

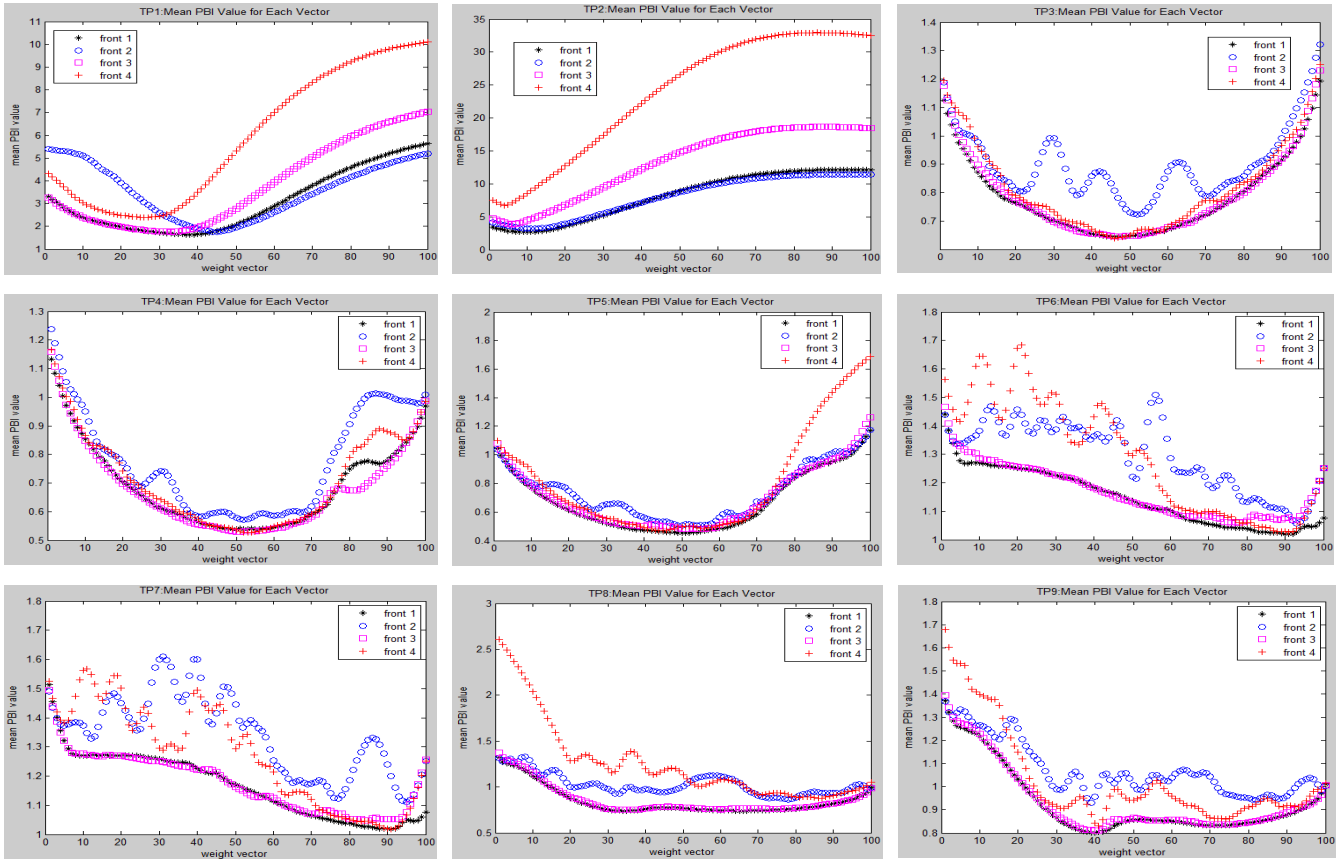

Fig. 10. Comparison results on S_PBI in TP1-9.

front are dominated by a very small part of solutions on the left. This observation is similarly as the results documented in [15]. From Fig. 9, every solution can become nondominated to others in some special scenario and only a small part of solutions can remain nondominated among all scenarios. This is why we retain some nonrobust solutions in the step of constructing the robust optimal front. In some scenarios, these nonrobust solutions can still perform better than some robust solutions while in other scenarios robust solutions in their neighborhood can replace them.

From the above discussion, starts from Pareto optimal solutions, the robust optimization step has successfully searched for the robust solutions so as to improve the total robust quality of the front. In the next section, we will compare the performance of RMOEA with other three algorithms in TP1-9.

D. Comparison Results in TP1-9

In this part, besides RMOEA which is represented as "front 1" in Figs. 10 and 11, three other robust optimization approaches are applied for comparison, including coevolutionary robust MOEA/D (C-RMOEA/D) [51], Deb's Type-I robust multiobjective optimization procedure [15], and decomposition-based EA-r (DBEA-r) [13], each of them in Figs. 10 and 11 is represented as "front 2," "front 3," and "front 4," respectively. All comparisons are conducted under the maximum perturbation degree of 0.025 in TP1-9. Among these four algorithms, different from those involving fitness sampling around each solution, C-RMOEA/D does not apply any sampling steps. Given the number of objectives, M, and the population size, N, the total number of function evaluations for RMOEA is $GMN + SMN + 12SMNN_1$, where G is the number of generations for RMOEA's multiobjective optimization process in Algorithm 1, S is the sampling size which is set to 50, and N_1 is the size of each updated archive after archive updating step in Algorithm 2. From the observation of experiment, N_1 is approximately equal to 10. Therefore, the number of function evaluations for RMOEA is around $(G + 6,050)MN$. The number of function evaluations for DBEA-r [13] and Deb's Type-I robust multiobjective optimization procedure [15] are $G(S + 1)MN$ and $GSMN$, respectively. G is the number of generations for the evolutionary process. For C-RMOEA/D, $2GMN$ function evaluations are used [51]. In order to make a fair comparison among all competing algorithms, in principle the numbers of function evaluations for all algorithms should be kept as close to each other as possible. However, due to the complexity involved in each unique design, this comparison can only be done as fair as possible. In this spirit, for the stopping criteria, the number of generations of all algorithms' evolutionary process is 3000 for TP1-4 and 2000 for TP5-9 as suggested in [15] and [30], except for C-RMOEA/D, which is 5000 for all TP1-9. In doing so, the total number of function evaluations for RMOEA is around 1 810 000 for TP1-4 [i.e., $(G+6,050)MN = (3000+6050) \times 2 \times 100$] and 1 610 000 for TP5-9 [i.e., $(2000+6050) \times 2 \times 100$]. For C-RMOEA/D, the

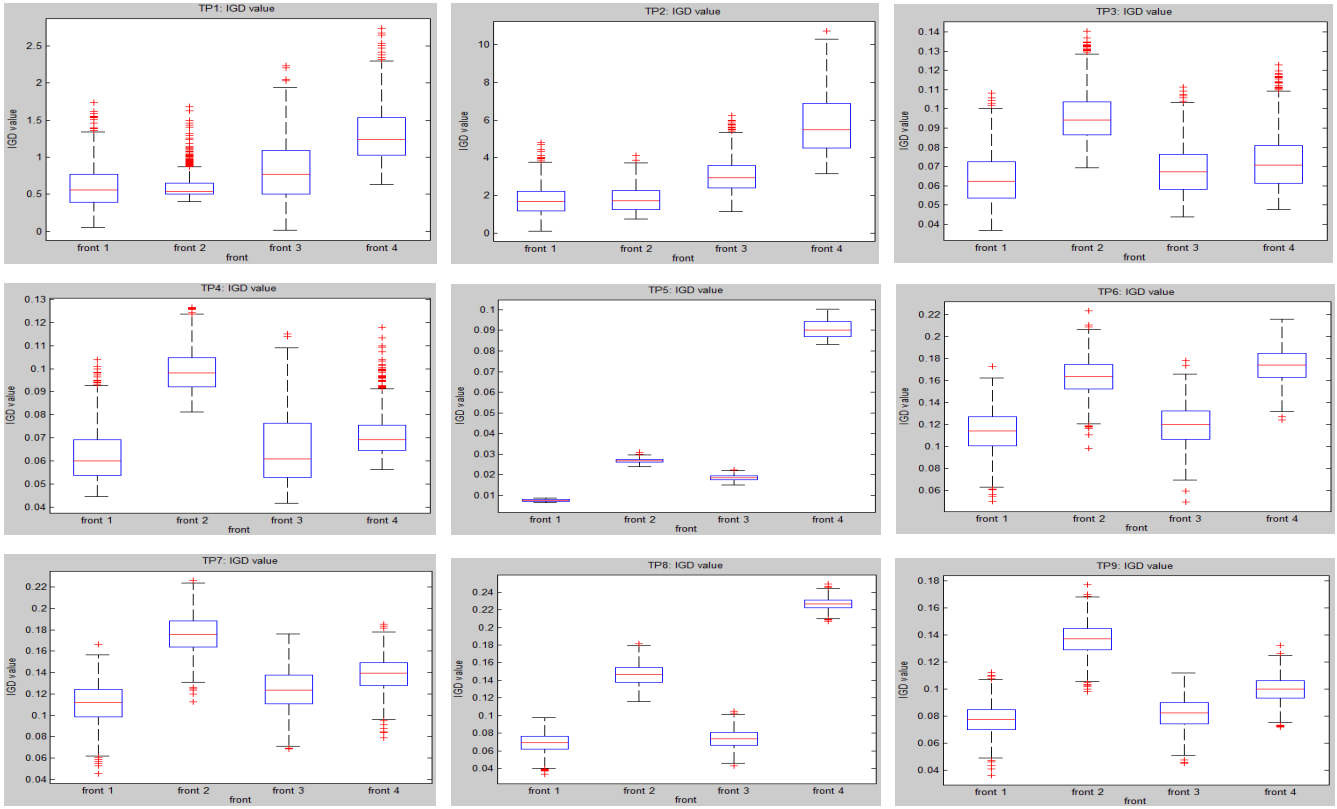

Fig. 11. Comparison results on S_IGD in TP1-9.

total number of function evaluations is 2 000 000 for TP1-9 (i.e., $2GMN = 2 \times 5000 \times 2 \times 100$). As far as DBEA-r and Deb's Type-I robust multiobjective optimization procedure are concerned, the numbers of function evaluations [i.e., $G(S+1)$MN and GSMN, respectively], are increased to around 30 000 000 for TP1-4 and 20 000 000 for TP5-9. These numbers are much larger than those of RMOEA and C-RMOEA/D due to the sampling process involved in each generation of their evolutionary processes. Without deteriorating the performance, S is set at 50 as suggested in its original publication. It is worth noting that the proposed design, RMOEA, consumes the least computational complexity while delivering the best performance. During each independent testing out of 1000 trials, the disturbance degree is a random number between $[-0.025, 0.025]$ and all robust fronts are subjected by the same disturbance degree.

Fig. 10 exhibits the mean PBI value for each weight vector among 1000 test generations. For each vector, the smaller the mean PBI value, the better performance along this vector, thus the better optimal degree is gained in this direction. From the figure, RMOEA nearly achieves the smallest PBI value in all problems (shown as "black star" in each figure) while Deb's Type-I robust multiobjective optimization procedure gets the second best results. In TP1, C-RMOEA/D achieves better performance than Deb's Type-I robust multiobjective optimization procedure and DBEA-r. In TP2, both RMOEA and C-RMOEA/D share nearly the same performance. Please note, for some test problems, there are intersections between different lines, e.g., front 2 and front 3 intersect in both TP1 and TP2. Therefore, even in the same problem, one algorithm may perform better along some direction vectors but worse in others. In order to make all directions optimized well under the disturbance, a combination of several algorithms may be a better choice.

Fig. 11 represents the boxplot results for comparison of S_IGD values in each problem, where the label "*" represents the IGD_{mean} value. The smaller the S_IGD value, the better the performance of the whole front is. From the figure, RMOEA achieves the best IGD results (e.g., IGD_{mean} and $\text{IGD}_{\text{worst}}$) among all scenarios. In TP1-2, RMOEA and C-RMOEA/D show nearly equal performance. In TP4 and TP6, RMOEA and Deb's Type-I robust multiobjective optimization procedure achieve the similar performance. Similar to Fig. 10, among all benchmark problems, Deb's Type-I robust multiobjective optimization procedure obtains the second best results. Every algorithm contains more extreme points in TP1-4 than those in TP5-9. In each problem, the variance of IGD values among all scenarios for each algorithm is not much different from each other. For each algorithm the difference between the best case and the worst case of its approximate front is similar as others. This difference value is mainly determined by the perturbation degree.

E. Parameter Analysis

In this section, we investigate how the number of associated hyperboxes (H) along each vector affects the performance of robust region detection. Besides original selected number 12,

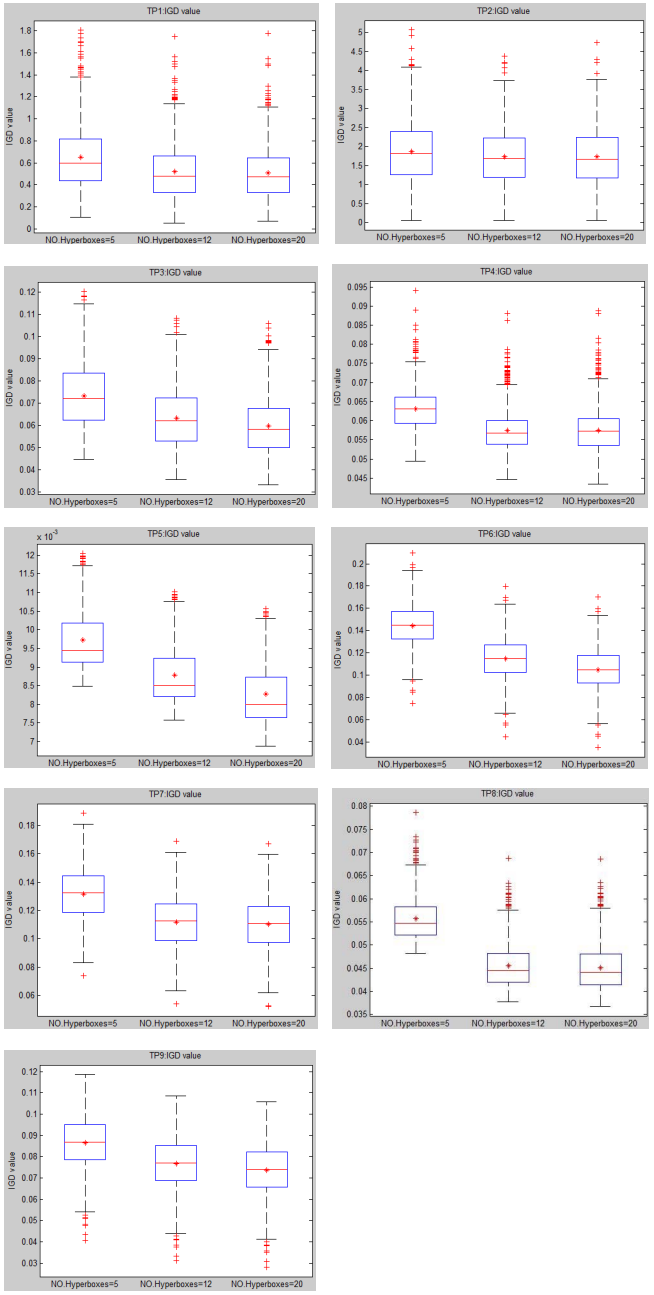

Fig. 12. Comparison results on S_IGD in TP1-9 under three different setting.

we also choose H as 5 and 20 for comparison. Fig. 12 shows the comparison results under the set of three different parameters in all nine benchmark problems.

From Fig. 12, the performance of robust region detection is improved significantly when the number of hyperboxes increases from 5 to 12. However, when this number continues increasing to 20, incurring much more computational cost, the performance of robust region detection only improves slightly compared with the performance when the number of hyperboxes is 12. Therefore, in these 2-D objective spaces, the number of hyperboxes set to 12 achieves the best tradeoff between the performance of robust region detection and the computational cost.

F. Comparison on Different Variants

In this section, we evaluate the effects of two components in RMOEA by comparing it with two variants. In variant I, quantification of a solution's robustness is based on the Euclidean distance between it and the original point rather than PBI value to its associated vector in our proposed design. Now the robustness degree of a solution only covers its convergence performance while the robustness degree of a solution along its closest weight vector covers both convergence and diversity performance. This variant aims to validate whether the robustness degree based on PBI covering both convergence and diversity performance can generate the robust optimal front with better quality than that generated by the robustness degree covering only convergence performance.

For variant II, one step of constructing the robust optimal front in Algorithm 2 is changed. In our proposed design, after the first scan, if one vector does not contain a robust solution, we check other nonselected robust solutions associated with vectors in its neighborhood. Then, among all these solutions whose regions' WPBI is smaller than the worst case of the optimal solution associated with this vector, the solution whose region with the best MPBI is selected. In this variant, this step is skipped and its original optimal solution is kept when any of its three closest neighboring vectors contain robust region. The comparison between this variant and RMOEA can determine whether the final performance of the whole robust front can be enhanced by this selection step in constructing the robust optimal front. These two variants are tested on TP1-9 under the same parameter setting as in Section IV-D, The experiment results are shown in Fig. 13, where RMOEA is represented as front 1, variant I as front 2, and variant II as front 3.

From Fig. 13, the performance of RMOEA is better than two variants in all problems, especially it is much better in problems TP4-5. Meanwhile, the performance difference between RMOEA and variant I is larger than that between RMOEA and variant II. Therefore, the effect of robustness degree based on PBI covering both convergence and diversity performance seems more significant than that of the selection step in constructing the robust optimal front. However, the latter one still enhances the final performance of the whole robust front.

G. Computational Complexity Analysis

In this section, given the number of objectives to be M and the population size N, we show an upper bound of the computational complexity for one generation of RMOEA. In multiobjective optimization, in order to generate Pareto optimal front ignoring disturbance, as stated in [1], the total computation complexity of NSGA-II is $O(MN^2)$. After that, grid-setting step and decomposition step are conducted only once so that their computation complexity is equal to $O(1)$. Thus, the total computational load in multiobjective optimization is $O(MN^2)$. In the robust optimization part, since all steps are only conducted once, the final computation complexity is $O(1)$. In summary, the overall computation complexity of RMOEA is $O(MN^2)$. Therefore, compared with the process of nonrobust optimization using NSGA-II, our

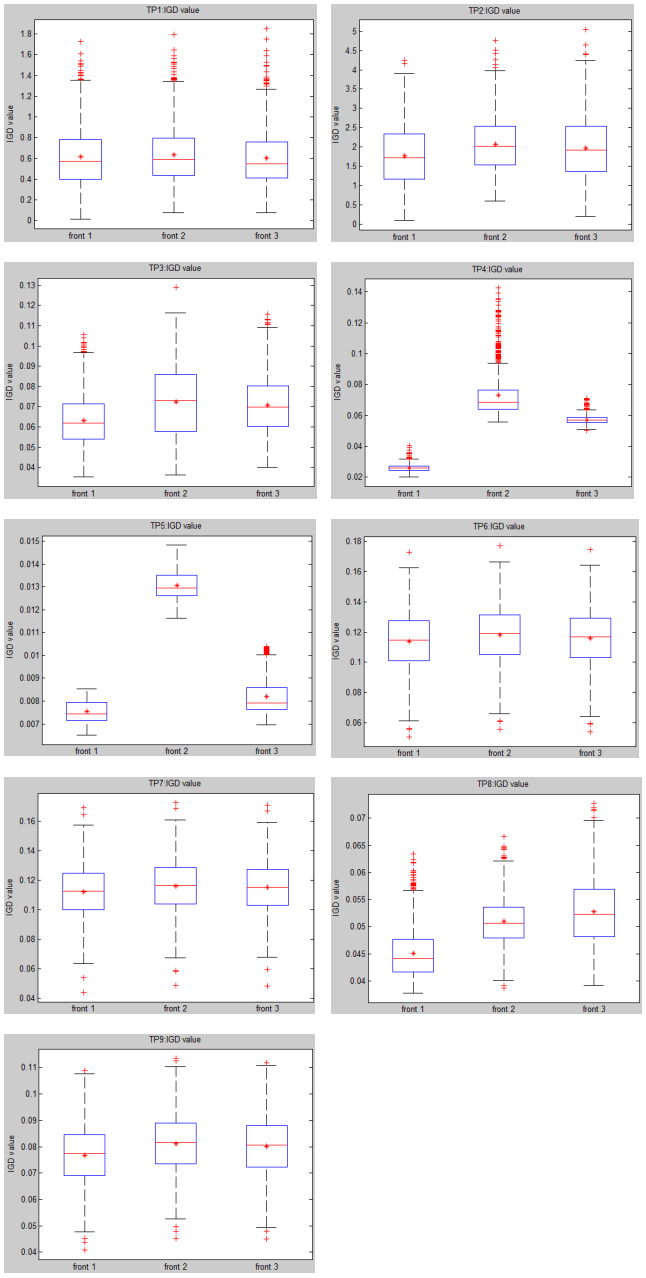

Fig. 13. Comparison of RMOEA and its two different variants in TP1-9.

proposed RMOEA exerts the same computational complexity as NSGA-II and all these robust optimization steps do not increase the time complexity.

V. Conclusion

In order to efficiently and effectively solve RMOPs, a novel RMOEA is developed. First, in multiobjective optimization, besides the obtained global Pareto optimal solutions, all visited solutions in the evolutionary process are organized in both decision space and objective space. In the robust optimization, an advanced robust detection method is proposed to effectively measure the robust degree of the region and directly search for the robust region. Afterward, a part of regions are picked up among all robust regions so that these selected regions together constructing a robust optimal front which achieves the best convergence and diversity performance. Finally, a comprehensive performance evaluation method is developed to faithfully quantify the quality of the algorithms.

Compared with research works in literature, we have made innovations in five aspects in order to directly handle the three issues of robust optimization we have mentioned at the beginning. First, we successfully balance solution's robustness and optimality performance by separating the search process of Pareto optimal solutions and robust solutions. Second, assisted by organization of all visited solutions at both decision space and objective space in multiobjective optimization as well as archive updating in robust optimization, we provide a fast and efficient means to directly search for the robust regions. Third, the new proposed decomposition framework offers appreciable improvements in representation of each solution (region)'s convergence and diversity performance under disturbance. Fourth, by fully exploiting EA's distributed nature, the average performance of the robust front is improved by allowing solutions with different robustness located in the same neighborhood. Finally, a comprehensive performance evaluation method is designed to quantify both optimality and robustness of the approximate front and individual solution. From the experimental results and analysis, the proposed RMOEA ensures a quality performance in both optimality and robustness under the disturbance given a number of benchmark problems.

In our future research, this advanced robust optimization strategy will be further extended in solving constrained and dynamic many-objective optimization problems [52]–[55], where the number of objectives is larger than three. Furthermore, in order to address real-world problems, our newly proposed robust optimization strategy with decision maker's preference [56] incorporated can be used to search for a subset of robust optimal solutions mandated by the decision makers.

References

[1] K. Deb, A. Pratab, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[2] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Dept. Elect. Eng., Swiss Federal Inst. Technol., Zürich, Switzerland, Rep. TIK-Report103, 2001.

[3] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[4] K. Shimoyama, A. Oyama, and K. Fujii, "Development of multi-objective six-sigma approach for robust design optimization," *J. Aerosp. Comput. Inf. Commun.*, vol. 5, no. 8, pp. 215–233, Aug. 2008.

[5] Y.-S. Ong, P. B. Nair, and K. Y. Lum, "Max–min surrogate-assisted evolutionary algorithm for robust design," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 392–404, Aug. 2006.

[6] M. T. Jensen, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 275–288, Jun. 2003.

[7] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2013.

[8] D. Wiesmann, U. Hammel, and T. Bäck, "Robust design of multilayer optical coatings by means of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 2, no. 4, pp. 162–167, Nov. 1998.

[9] R. R. Chan and S. D. Sudhoff, "An evolutionary computing approach to robust design in the presence of uncertainties," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 900–912, Dec. 2010.

[10] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005

[11] H.-G. Beyer and B. Sendhoff, "Robust optimization—A comprehensive survey," *Comput. Methods Appl. Mech. Eng.*, vol. 196, nos. 33–34, pp. 3190–3218, Jul. 2007.

[12] R. Pratyusha, K. Amit, and S. Das, "Noisy evolutionary optimization algorithms—A comprehensive survey," *Swarm Evol. Comput.*, vol. 33, pp. 18–45, Apr. 2017.

[13] M. Asafuddoula, H. K. Singh, and T. Ray, "Six-sigma robust design optimization using a many-objective decomposition-based evolutionary algorithm," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 490–507, Aug. 2015.

[14] T. C. Huang, D. G. Wang, S. Z. Zhou, and X. M. Yuan, "Theoretical design research for stirring device of fracturing blender truck," *Appl. Mech. Mater.*, vols. 220–223, pp. 909–912, Nov. 2012.

[15] K. Deb and H. Gupta, "Introducing robustness in multi-objective optimization," *Evol. Comput.*, vol. 14, no. 4, pp. 463–494, Dec. 2006.

[16] I. Paenke, J. Branke, and Y. Jin, "Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 405–420, Aug. 2006.

[17] L. T. Bui, H. A. Abbass, M. Barlow, and A. Bender, "Robustness against the decision-maker's attitude to risk in problems with conflicting objectives," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 1–19, Feb. 2012.

[18] E. J. Hughes, "Evolutionary multi-objective ranking with uncertainty and noise," in *Proc. 1st Int. Conf. Evol. Multi Criterion Optim.*, Zürich, Switzerland, 2001, pp. 329–343.

[19] H. Karshenas, C. Bielza, and P. Larrañaga, "Interval-based ranking in noisy evolutionary multi-objective optimization," *Comput. Optim. Appl.*, vol. 61, no. 2, pp. 517–555, Jun. 2015.

[20] V. A. Shim, K. C. Tan, J. Y. Chia, and A. A. Mamum, "Multi-objective optimization with estimation of distribution algorithm in a noisy environment," *Evol. Comput.*, vol. 21, no. 1, pp. 149–177, Jan. 2013.

[21] D. Gong, J. Sun, and Z. Miao, "A set-based genetic algorithm for interval many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 47–60, Feb. 2018.

[22] E. Zitzler, L. Thiele, and J. Bader, "On set-based multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 58–79, Feb. 2010.

[23] J. Bader, D. Brockhoff, S. Welten, and E. Zitzler, "On using populations of sets in multiobjective optimization," in *Proc. 5th Int. Conf. Evol. Multi Criterion Optim.*, Nantes, France, 2009, pp. 140–154.

[24] S. B. Gee, K. C. Tan, and C. Alippi, "Solving multiobjective optimization problems in unknown dynamic environments: An inverse modeling approach," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4223–4234, Dec. 2017.

[25] S. Das, A. Mandal, and R. Mukherjee, "An adaptive differential evolution algorithm for global optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 966–978, Jun. 2014.

[26] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 40–53, Jan. 2014.

[27] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 556–577, Aug. 2012.

[28] S. Jiang and S. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 65–82, Feb. 2017.

[29] Y. Jin, K. Tang, X. Yu, B. Sendhoff, and X. Yao, "A framework for finding robust optimal solutions over time," *Memetic Comput.*, vol. 5, no. 1, pp. 3–18, Mar. 2013.

[30] A. Gaspar-Cunha, J. Ferreira, and G. Recio, "Evolutionary robustness analysis for multi-objective optimization: Benchmark problems," *Struct. Multidiscipl. Optim.*, vol. 49, no. 5, pp. 771–793, 2014.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[32] M. D. McKay, W. J. Conover, and R. J. Beckman, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[33] A. Harbitz, "An efficient sampling method for probability of failure calculation," *Struct. Safety*, vol. 3, no. 2, pp. 109–115, 1986.

[34] J. Branke, "Creating robust solutions by means of evolutionary algorithms," in *Proc. 5th Int. Conf. Parallel Problem Solving Nat.*, Amsterdam, The Netherlands, 1998, pp. 119–128.

[35] C. Qian *et al.*, "On the effectiveness of sampling for evolutionary optimization in noisy environments," in *Proc. 13th Int. Conf. Parallel Problem Solving Nat.*, 2014, pp. 302–311.

[36] P. Stagge, "Averaging efficiently in the presence of noise," in *Proc. 5th Int. Conf. Parallel Problem Solving Nat.*, Amsterdam, The Netherlands, 1998, pp. 188–197.

[37] G. Lei *et al.*, "Robust design optimization of PM-SMC motors for six sigma quality manufacturing," *IEEE Trans. Magn.*, vol. 49, no. 7, pp. 3953–3956, Jul. 2013.

[38] G. Sun *et al.*, "Crashworthiness design of vehicle by using multiobjective robust optimization," *Struct. Multidiscipl. Optim.*, vol. 44, no. 1, pp. 99–110, 2011.

[39] A. M. Hasofer and N. C. Lind, "Exact and invariant second-moment code format," *ASCE J. Eng. Mech. Div.*, vol. 100, no. 1, pp. 111–121, Feb. 1974.

[40] M. Hohenbichler, S. Gollwitzer, W. Kruse, and R. Rackwitz, "New light on first- and second-order reliability methods," *Struct. Safety*, vol. 4, no. 4, pp. 267–284, 1987.

[41] R. F. Coelho, "Probabilistic dominance in multiobjective reliability-based optimization: Theory and implementation," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 214–224, Apr. 2015.

[42] R. F. Coelho and P. Bouillard, "Multi-objective reliability-based optimization with stochastic metamodels," *Evol. Comput.*, vol. 19, no. 4, pp. 525–560, 2011.

[43] M. Li, R. Silva, F. Guimarães, and D. Lowther, "A new robust dominance criterion for multiobjective optimization," *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1–4, Mar. 2015.

[44] H. Wang, Q. Zhang, L. Jiao, and X. Yao, "Regularity model for noisy multiobjective optimization," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 1997–2009, Sep. 2016.

[45] W. Chen, J. Allen, K.-L. Tsui, and F. Mistree, "A procedure for robust design: Minimizing variations caused by noise factors and control factors," *ASME J. Mech. Design*, vol. 118, no. 4, pp. 478–485, 1996.

[46] I. Das, "Robustness optimization for constrained nonlinear programming problems," *Eng. Optim.*, vol. 32, no. 5, pp. 585–618, 2000.

[47] A. Gaspar-Cunha and J. A. Covas, "Robustness in multi-objective optimization using evolutionary algorithms," *Comput. Optim. Appl.*, vol. 39, no. 1, pp. 75–96, 2008.

[48] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating Pareto optimal points in multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, Aug. 1998.

[49] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.

[50] P. Czyżżak and A. Jaszkiewicz, "Pareto simulated annealing—A metaheuristic technique for multiple-objective combinatorial optimization," *J. Multi Criteria Decis. Anal.*, vol. 7, pp. 34–47, Dec. 1998.

[51] I. R. Meneghini, F. G. Guimarães, and A. Gaspar-Cunha, "Competitive coevolutionary algorithm for robust multi-objective optimization: The worst case minimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 586–593.

[52] Z. He and G. G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 145–160, Feb. 2016.

[53] J. Chen, H. Mao, Y. Sang, and Z. Yi, "Subspace clustering using a symmetric low-rank representation," *Knowl. Based Syst.*, vol. 127, pp. 46–57, Jul. 2017.

[54] Z. He and G. G. Yen, "Visualization and performance metric in many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 386–402, Jun. 2016.

[55] Z. He and G. G. Yen, "Many-objective evolutionary algorithms based on coordinated selection strategy," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 220–233, Apr. 2017.

[56] W. Y. Chiu, G. G. Yen, and T.-K. Juan, "Minimum Manhattan distance approach to multiple criteria decision making in multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 972–985, Dec. 2016.

Zhenan He received the B.E. degree in automation from the University of Science and Technology Beijing, Beijing, China, in 2008 and the M.S. degree and the Ph.D. degree in electrical and computer engineering from Oklahoma State University, Stillwater, OK, USA, in 2011 and 2014, respectively.

He is currently an Associate Professor with the College of Computer Science, Sichuan University, Chengdu, China. His current research interests include multiobjective optimization using evolutionary algorithms, intelligent optimization, and machine learning.

Gary G. Yen (S'87–M'88–SM'97–F'09) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1992.

He was with the Structure Control Division, U.S. Air Force Research Laboratory, Albuquerque, NM, USA. In 1997, he joined Oklahoma State University (OSU), Stillwater, OK, USA, where he is currently a Regents Professor with the School of Electrical and Computer Engineering. His current research interests include intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications.

Dr. Yen was a recipient of the Andrew P. Sage Best Transactions Paper Award from the IEEE Systems, Man and Cybernetics Society, in 2011 and the Meritorious Service Award from the IEEE Computational Intelligence Society in 2014. He was an Associate Editor of the *IEEE Control Systems Magazine*, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, *Automatica*, *Mechantronics*, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, and the IEEE TRANSACTIONS ON NEURAL NETWORKS. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS. He served as the General Chair for the 2003 IEEE International Symposium on Intelligent Control in Houston, TX, USA, and 2006 IEEE World Congress on Computational Intelligence in Vancouver, BC, Canada. He served as the Vice President for Technical Activities from 2005 to 2006 and the President from 2010 to 2011 of the IEEE Computational Intelligence Society, and was the Founding Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2006 to 2009. He is a fellow of IET.

Zhang Yi (F'16) received the Ph.D. degree in mathematics from the Institute of Mathematics, Chinese Academy of Science, Beijing, China, in 1994.

He is currently a Professor with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China. He has co-authored three books: *Convergence Analysis of Recurrent Neural Networks* (Kluwer Academic Publishers, 2004), *Neural Networks: Computational Models and Applications* (Springer, 2007), and *Subspace Learning of Neural Networks* (CRC Press, 2010). His current research interests include neural networks and big data.

Dr. Yi was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2009 to 2012, and he became an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS in 2014.