Evolutionary Dynamic Multiobjective Optimization: Benchmarks and Algorithm Comparisons

Shouyong Jiang and Shengxiang Yang, *Senior Member, IEEE*

***Abstract*—Dynamic multiobjective optimization (DMO) has received growing research interest in recent years since many real-world optimization problems appear to not only have multiple objectives that conflict with each other but also change over time. The time-varying characteristics of these DMO problems (DMOPs) pose new challenges to evolutionary algorithms. Considering the importance of a representative and diverse set of benchmark functions for DMO, in this paper, we propose a new benchmark generator that is able to tune a number of challenging characteristics, including mixed Pareto-optimal front (convexity–concavity), nonmonotonic and time-varying variable-linkages, mixed types of changes, and randomness in type change, which have rarely or not been considered or tested in the literature. A test suite of ten instances with different dynamic features is produced from the generator in this paper. Additionally, a few new performance measures are proposed to evaluate algorithms for DMOPs with different characteristics. Six representative multiobjective evolutionary algorithms from the literature are investigated based on the proposed DMO test suite and performance measures. The experimental results facilitate a better understanding of strengths and weaknesses of these compared algorithms for DMOPs.**

***Index Terms*—Benchmark, dynamic multiobjective optimization (DMO), evolutionary algorithm, performance metric.**

I. Introduction

MANY real-world multiobjective optimization problems (MOPs) appear to change over time in a dynamic environment, such as planning [47], scheduling [7], design [40], and control [55]. Dynamic MOPs (DMOPs) have gained increasing attention in recent years. Due to the dynamics of these problems, the optimization of DMOPs is much more challenging than that of static MOPs as it has to deal with not only the conflicting objectives, but also the changes in objective functions or constraints. In other words, dynamic multiobjective optimization algorithms (DMOAs) must be capable of tracking the changing Pareto-optimal front (POF) to provide a diverse set of solutions that approximates the new POF over time.

Artificial benchmark problems have played a fundamental role in determining whether a DMOA has the ability to solve DMOPs. Furthermore, benchmark problems contribute to analyzing and identifying the strengths and weaknesses of a DMOA in order to modify it and improve its performance. However, one of the main issues in the field of dynamic multiobjective optimization (DMO) is that there are no standard test functions, and a few publications [10], [17], [18], [37], [49] pointed out that there should be more investigation of DMOPs to promote the research of DMO. Such issue is, however, yet to gain as much attention as the fields of static MOPs [28], [35], [39] and dynamic single-objective optimization problems [2], [5], [6], [15], [25], [32], [33], [38], [43], [45], [49].

The first research on constructing DMOPs was conducted by Jin and Sendhoff [25], who proposed to aggregate different objectives of the existing static MOPs and vary the weights dynamically. Later, Farina *et al.* [10] made a clear classification of DMOPs according to the possible effects of environmental changes on the POF and the Pareto-optimal set (POS), and they also considered several dynamic scenarios that may appear in dynamic environments. Following those scenarios, they further suggested a test suite of five FDA test functions based on the existing DTLZ [9] and ZDT [50] test suites of static MOPs. Since then, various dynamic benchmarks have sprung up in [1], [3], [4], [14], [20], [23], and [37].

Recalling those existing DMOPs, we can observe that most of commonly used DMOPs are adapted from the DTLZ [9] and ZDT [50] test suites. In other words, they are the variants of the FDA [10] test problems. As a consequence, they share more or less same or similar properties (e.g., same objective functions). Furthermore, few have taken into account the following characteristics: 1) mixed POFs in terms of convexity and concavity that change over time; 2) complicated diversity-resistant schemes that hinder a set of diverse solutions; 3) problems that can change between different types during the evolution; and 4) nonmonotonic and time-varying relationship between variables instead of static monotonic variable-linkage used in the literature. The lack of the above-mentioned characteristics in DMOPs implies that the DMOPs currently used in the literature are not sufficiently diverse and challenging. Therefore, developing a new set of

Manuscript received September 2, 2015; revised December 2, 2015; accepted December 12, 2015. Date of publication January 13, 2016; date of current version December 14, 2016. This work was supported in part by the Engineering and Physical Sciences Research Council of U.K. under Grant EP/K001310/1, and in part by the National Natural Science Foundation of China under Grant 61273031. This paper was recommended by Associate Editor G. G. Yen. *(Corresponding author: Shengxiang Yang.)*

The authors are with the Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U.K. (e-mail: syang@dmu.ac.uk).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org provided by the authors. This supplementary material includes detailed experimental results together with statistical comparison of tested algorithms on the test problems, a discussion on the difference between the variable-linkage used in JY3 and the one in the existing ZJZ problem, and the approximated Pareto fronts of JY8 obtained by compared algorithms after each environmental change. This material is 542 KB in size.

Digital Object Identifier 10.1109/TCYB.2015.2510698

test functions that covers those characteristics to compare the performance of DMOAs becomes meaningful and essential, which is greatly needed for the domain of DMO.

Based on the understanding of the limitations of current DMOP test problems and inspired by the works of [9], [10], [21], and [50], we have recently proposed a new benchmark generator [22] for DMOPs that can generate DMOPs with several complex characteristics, e.g., changing POFs of mixed types featuring both convexity and concavity, which are rarely tested in the literature. Following this line, in this paper, we further develop the proposed benchmark generator and produce a general class of DMOPs with more dynamisms and characteristics for facilitating theoretical analysis in DMO.

Besides suggesting a number of test instances derived from the benchmark generator, in this paper, we also discuss performance metrics used for DMO. Two new performance measures are thus proposed, providing more information about the performance of multiobjective evolutionary algorithms (MOEAs). We also test six representative MOEAs due to their recency and reported success. A fair and comprehensive comparison helps to compare the performance of algorithms and assess the nature and difficulty of the proposed benchmark functions. For this purpose, we also examine some performance metrics, including two new ones proposed in this paper, for assessing DMO algorithms for DMOPs.

The remainder of this paper is organized as follows. Section II presents a brief review of related work on DMO. Section III introduces the proposed benchmark generator and a set of test cases derived from the generator. Performance metrics are discussed in Section IV. Section V presents a fair and comprehensive comparison of MOEAs on the proposed test suite. Finally, Section VI concludes this paper.

II. Related Work

There are many dynamic characteristics involved in DMOPs, and different DMOPs may have different mathematical definitions. This paper focuses on the DMOPs defined as

$$\begin{aligned} \min \quad & F(x,t) = (f_1(x,t), \ldots, f_M(x,t))^T \\ \text{s.t.} \quad & \begin{cases} h_i(x,t) = 0, & i = 1, \ldots, n_h \\ g_i(x,t) \geq 0, & i = 1, \ldots, n_g \\ x \in \Omega_x, \quad t \in \Omega_t \end{cases} \end{aligned} \tag{1}$$

where M is the number of objectives, and n_h and n_g are the number of equity and inequity constraints, respectively. $\Omega_x \subseteq R^n$ is the decision space, t is the discrete time instance defined as $t = (1/n_t)\lfloor(\tau/\tau_t)\rfloor$ (where n_t, τ_t, and τ represent the severity of change, the frequency of change, and the iteration counter, respectively), and $\Omega_t \subseteq R$ is the time space. $F(x,t) : \Omega_x \times \Omega_t \to R^M$ is the objective function vector that evaluates the solution x at time t.

Most artificial DMOPs developed by researchers in the literature conform to Eq. (1). A distinct characteristic of Eq. (1) is that the POF and POS are susceptible to change, which challenges the tracking ability of DMOAs. Farina *et al.* [10] classified DMOPs into four types according to the dynamics of POF and POS.

1) *Type I:* The POS changes over time while the POF remains stationary.
2) *Type II:* Both POF and POS change over time.
3) *Type III:* The POF changes over time while the POS remains stationary.
4) *Type IV:* Both POF and POS remain stationary, although the objective function or constraints may change over time.

Farina *et al.* [10] also created some FDA DMOPs by adapting the static problems from the DTLZ [9] and ZDT [50] test suites. The FDA test suite has been widely used and further modified by researchers to test algorithms' performance. It should be noted that there is a misunderstanding of the FDA2 test problem. Helbig and Engelbrecht [16] studied the performance of several algorithms on FDA2, finding that all the tested algorithms lose track of the changing POF. However, the phenomenon of losing track of the changing POF they observed does not exist in FDA2 because the approximated POF would never be better than the true POF. Actually, their misleading results may come from a misunderstanding of the exact POS of FDA2 after a change.

Jin and Sendhoff [25] suggested a method for constructing DMOPs by dynamically changing the weights that aggregate the different objectives of static MOPs. But, their method does not provide clear defined problems. Guan *et al.* [14] studied DMOPs with objective replacement, where some objectives may be replaced with new objectives during the evolution. Mehnen *et al.* [37] argued that the DTLZ and ZDT test suites are already challenging in their static version, and simpler test functions are needed to analyze the effect of dynamics in DMOPs. Hence, they suggested the DSW functions for DMOPs. Furthermore, they proposed a new generic scheme DTF that is a generalized FDA function and allows the scaling of the complexity of the dynamic properties. They also added scalable and dynamic constraints to DMOPs by moving circular obstacles in the objective space.

The ZJZ problem defined by Zhou *et al.* [58] is the first DMOP with variable linkages, which is a modified version of FDA1 but more challenging. Zhou *et al.* [57] further argued that most DMOPs derived from the FDA test suite are too simple, and the correlation between decision variables should be enhanced. Thus, they gave four new DMOP test instances that have nonlinear correlation between the decision variables. Helbig and Engelbrecht [18] made a sound investigation into the current DMOPs used in the literature, and identified their shortcomings: none of them had deceptive and isolated features in the POF. Then, they developed DMOPs with either an isolated or deceptive POF that follows the concept of the static WFG [21] test suite. In addition, they proposed some HE problems that have complicated POSs based on the MOPs of [34].

Most existing DMOPs that have been discussed in the literature are included in the first three types of change mentioned earlier, but none of them is a type IV problem. Recently, Huang *et al.* [20] created several type IV problems by implying

that the current found POS may further affect the following POS or POF. Furthermore, they introduced two DMOPs where the number of decision variables or objective functions changes over time.

Most recently, Biswas *et al.* [1] discussed some ways of designing DMOPs and proposed some general techniques for introducing dynamisms into the POS and POF through shifting, shape variation, phase variation, and several other types. By extending the bound-constrained multiobjective testbed developed by Li and Zhang [34], they suggested nine benchmark functions for DMO.

III. Proposed Benchmark Generator and Test Instances

A. Proposed Benchmark Generator

The proposed benchmark generator is based on the following framework:

$$JY: \begin{cases} \min \quad F(\mathbf{x},t) = (f_1(\mathbf{x},t), f_2(\mathbf{x},t))^T \\ f_1(\mathbf{x},t) = (1+g(\mathbf{x_{II}},t))(h(\mathbf{x_I}) + A_t \sin(W_t \pi h(\mathbf{x_I})))^{\alpha_t} \\ f_2(\mathbf{x},t) = (1+g(\mathbf{x_{II}},t))(1-h(\mathbf{x_I}) + A_t \sin(W_t \pi h(\mathbf{x_I})))^{\beta_t} \end{cases} \tag{2}$$

where $0 \le h(\mathbf{x_I}) \le 1$, and $\mathbf{x_I}$ and $\mathbf{x_{II}}$ are subvectors of the decision vector $\mathbf{x}$. A_t and W_t are two parameters to control the local shape of the POF, with A_t adjusting the curvature and W_t controlling the number of mixed convex and concave segments on the POF. A large value of W_t causes the POF to have disconnected regions, while a small value produces a continuous POF. Here, W_t is recommended to be an integer. α_t and β_t ($\alpha_t > 0$, $\beta_t > 0$) are parameters that control the overall shape of the POF: when $\alpha_t > 1$ and $\beta_t > 1$ or $\alpha_t < 1$ and $\beta_t < 1$, the overall shape is convex or concave, respectively; when $\alpha_t = \beta_t = 1$, the overall shape is linear; otherwise, the overall shape is mixed. $g(\mathbf{x_{II}}, t)$ is a non-negative function, hindering algorithms from converging toward the true POF. The minimum of $g(\mathbf{x_{II}}, t)$ is zero. Thus, Eq. (2) can produce various POF geometries by properly configuring relevant parameters. Generally, the mathematical description of the continuous POF for Eq. (2) is as follows:

$$f_1^{\frac{1}{\alpha_t}} + f_2^{\frac{1}{\beta_t}} = 1 + 2A_t \sin\left(W_t \pi \frac{f_1^{\frac{1}{\alpha_t}} - f_2^{\frac{1}{\beta_t}} + 1}{2} \right) \tag{3}$$

where the values of A_t and W_t must enable Eq. (2) to be a continuous POF. To have a better understanding of the proposed generator, we denote $F_1 = \sqrt[\alpha_t]{f_1} - \sqrt[\beta_t]{f_2}$ and $F_2 = \sqrt[\alpha_t]{f_1} + \sqrt[\beta_t]{f_2}$. This means that a clockwise rotation with an angle $\pi/4$ is made from the current coordinate axis. Then, Eq. (3) can be rewritten as

$$F_2 = 1 + 2A_t \sin\left(W_t \pi \frac{F_1 + 1}{2} \right) \tag{4}$$

where a sine wave is described if $W_t \neq 0$ and $A_t \neq 0$. Thus, the proposed generator has a wave-like geometry, containing both concave and convex regions. Fig. 1 illustrates examples of POFs of *JY* with linear and nonlinear overall shapes.

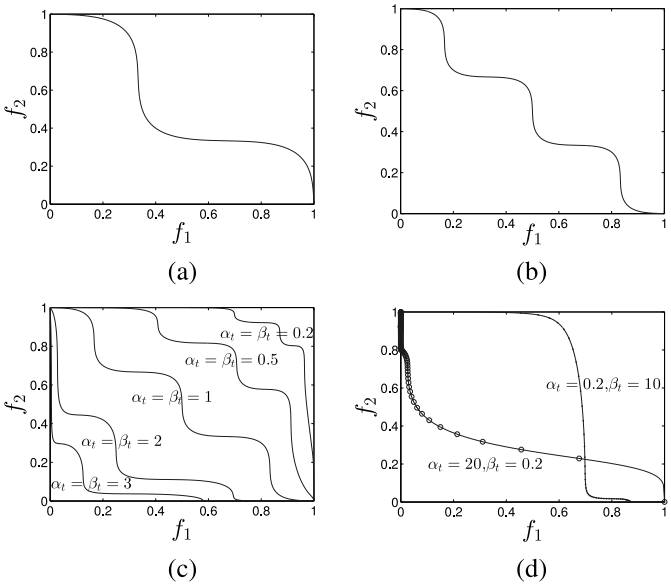


Fig. 1. POFs of *JY* with different overall shapes. (a) $\alpha_t = \beta_t = 1$, $A_t = 0.1$, and $W_t = 3$. (b) $\alpha_t = \beta_t = 1$, $A_t = 0.05$, and $W_t = 6$. (c) Convex or concave overall shapes with $A_t = 0.05$ and $W_t = 6$. (d) Mixed overall shapes with $A_t = 0.05$ and $W_t = 6$.

B. Test Instances

In this paper, we concentrate on $h(\mathbf{x_I}) = x_1$, although we recognize that movement across the POF can be achieved by adjusting a number of variables, i.e., the use of rotation matrices for $h(\mathbf{x_I})$ and the normalization of $h(\mathbf{x_I})$. Below, we provide ten benchmark instances with detailed information for a number of types of changes in the environment.

*JY*1, as shown in Eq. (5), is a type I problem, where the POS changes over time in a regular pattern, with $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$. It mainly tests the convergence speed and reactivity of an algorithm, and fast-convergence algorithms can easily solve this problem. The control of processing plant [20], where the POS varies in different time-dependent scenarios and controllers are required to have a fast response speed, can be a corresponding real-world application of *JY*1.

$$JY1: \begin{cases} \min \quad F(\mathbf{x},t) = (f_1(\mathbf{x},t), f_2(\mathbf{x},t))^T \\ f_1(\mathbf{x},t) = (1+g(\mathbf{x_{II}},t))(x_1 + A_t \sin(W_t \pi x_1)) \\ f_2(\mathbf{x},t) = (1+g(\mathbf{x_{II}},t))(1-x_1 + A_t \sin(W_t \pi x_1)) \\ g(\mathbf{x_{II}},t) = \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2, G(t) = \sin(0.5\pi t) \\ A(t) = 0.05, \quad W(t) = 6 \\ \mathbf{x_I} = (x_1) \in [0,1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1,1]^{n-1}. \end{cases} \tag{5}$$

*JY*2, as shown in Eq. (6), is a type II problem with dynamic POFs and POSs. The POS changes over time, and the objective vector oscillates among several modes. As a result, the POF, as illustrated in Fig. 2, changes its shape over time [refer to Eq. (3)]. A similar real-world application is the electric power supply problem [29], where the objective functions oscillate among several optimization modes, resulting in the

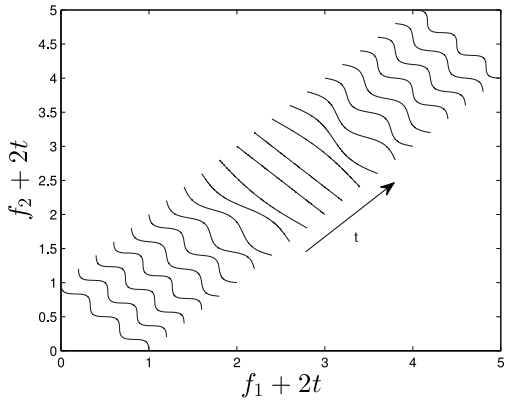


Fig. 2. POF of *JY*2 with 21 time windows varying from 0 to 2. For a better visualization, $f_1 + 2t$ and $f_2 + 2t$ are shown on the x- and y-axis, respectively.

change of optimal solutions in real time.

$$JY2: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1)) \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1)) \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2, G(t) = \sin(0.5\pi t) \\ A(t) = 0.05, W(t) = \lfloor 6\sin(0.5\pi(t-1)) \rfloor \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{6}$$

*JY*3, as shown in Eq. (7), introduces time-varying nonmonotonic dependencies between any two decision variables, which is close to real-world problems like the greenhouse system [48]. The POF of *JY*3 is similar to that of *JY*2, and the POS is $y_1 = |x_1 \sin((2\alpha + 0.5)\pi x_1)|$, $\alpha = \lfloor 100\sin^2(0.5\pi t) \rfloor$, and $y_i = \sqrt{y_{i-1}}$, $i = 2, \ldots, n$, meaning that each variable has different amount of change. The POS for variables x_1 and x_2 is shown in Fig. 3, where the dependency between these two variables is nonmonotonic and becomes increasingly complicated as the time increases. Furthermore, the density of solutions also changes over time. Therefore, *JY*3 not only assesses the effect of variable-linkage but also tests the diversity performance of an algorithm in a dynamic environment.

$$JY3: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(y_1 + A_t \sin(W_t \pi y_1)) \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - y_1 + A_t \sin(W_t \pi y_1)) \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} (y_i^2 - y_{i-1})^2, A(t) = 0.05 \\ W(t) = \lfloor 6\sin(0.5\pi(t-1)) \rfloor, \alpha = \lfloor 100 \sin^2(0.5\pi t) \rfloor \\ y_1 = |x_1 \sin((2\alpha + 0.5)\pi x_1)|, y_i = x_i, i = 2, \ldots, n \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{7}$$

*JY*4, as shown in Eq. (8), is constructed to have a time-changing number of disconnected POF segments, which is the case with hydro-thermal power scheduling [7], where the POF is discontinuous. This problem may pose challenges to some algorithms to find all the POF components. As illustrated in Fig. 4, the POF of *JY*4 is subject to the definition of Eq. (3), but has a number of disconnected segments. The POS

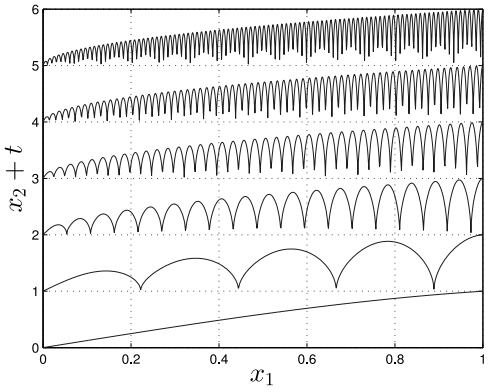


Fig. 3. POS of *JY*3 for the first two variables with six time windows varying from 0 to 0.5. For a better visualization, x_1 and $x_2 + t$ are shown on the x- and y-axis, respectively.

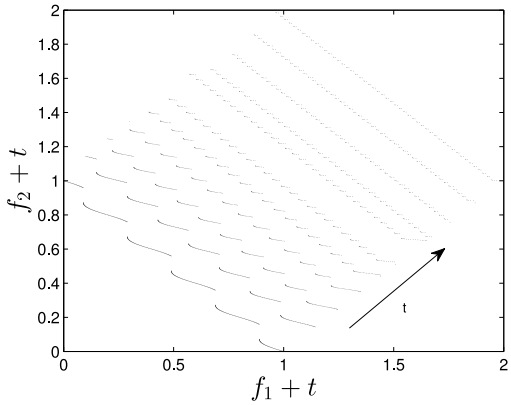


Fig. 4. POF of *JY*4 with 11 time windows varying from 0 to 2. For a better visualization, $f_1 + t$ and $f_2 + t$ are shown on the x- and y-axis, respectively.

is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$.

$$JY4: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1)) \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1)) \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2, G(t) = \sin(0.5\pi t) \\ A(t) = 0.05, \quad W(t) = 10^{1+|G(t)|} \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{8}$$

Contrary to the above problems, *JY*5 [as shown in Eq. (9)] does not have a mixed POF and is a type III problem. Its POF is very simple and changes from convex geometry to concave geometry. The POF is defined in Eq. (3) and illustrated in Fig. 5. A similar real-life problem that has changing POF shapes can be referred to the route guidance system [44].

$$JY5: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1)) \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1)) \\ g(\mathbf{x_{II}}, t) = \sum\limits_{x_i \in \mathbf{x_{II}}} x_i^2, A_t = 0.3 \sin(0.5\pi(t-1)), W_t = 1 \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{9}$$

The above-generated instances are all unimodal, and they are not sufficiently challenging. In contrast, *JY*6 is a

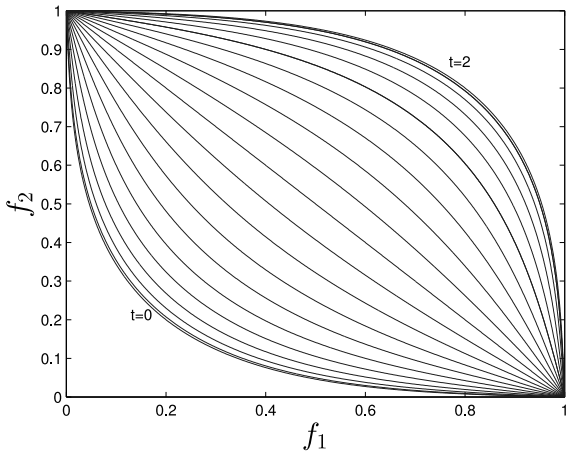


Fig. 5. POF of *JY*5 with 21 time windows varying from 0 to 2.

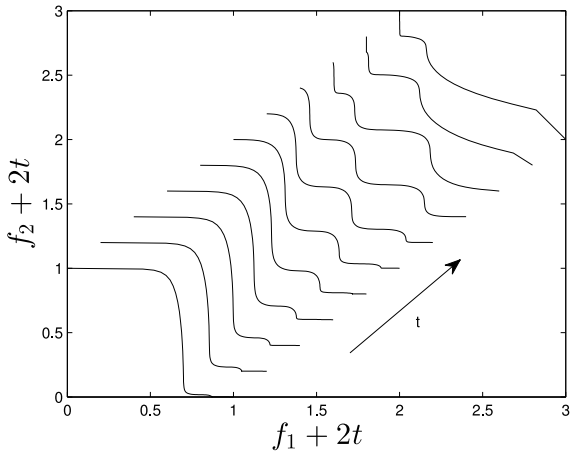


Fig. 6. POF of *JY*8 with 11 time windows varying from 0 to 1. For a better visualization, $f_1 + 2t$ and $f_2 + 2t$ are shown on the x- and y-axis, respectively.

multimodal problem, where not only the number of local optima changes over time, but also the POS is dynamically shifted. The POF of *JY*6 remains stationary, and its POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$.

$$JY6: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1)) \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1)) \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} \left(4y_i^2 - \cos(K_t \pi y_i) + 1\right) \\ A_t = 0.1, \quad W_t = 3, K_t = 2 * \lfloor 10 * |G(t)| \rfloor \\ G(t) = \sin(0.5\pi t), y_i = x_i - G(t), \quad i = 2, \ldots, n \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{10}$$

*JY*7 takes into account the shift of POS, multimodality, and the overall shape of the POF. Different from *JY*6, the number of local optima in *JY*7 remains fixed, and the overall POF shape can be concave or convex due to environmental changes. The POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, and the POF is mathematically described in Eq. (3) and similarly illustrated in Fig. 1(c). A multimodal real-world problem similar to *JY*7 can be the dynamic speed reducer design [46], [56].

$$JY7: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1))^{\alpha_t} \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1))^{\beta_t} \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} \left(y_i^2 - 10\cos(2\pi y_i) + 10\right) \\ A_t = 0.1, W_t = 3, \alpha_t = \beta_t = 0.2 + 2.8 * |G(t)| \\ G(t) = \sin(0.5\pi t), y_i = x_i - G(t), \quad i = 2, \ldots, n \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{11}$$

In *JY*8, the POS remains static, but the POF changes over time. the dynamism of *JY*8 lies in the change of its overall POF shape, in which the geometry and the number of mixed segments of the POF vary over time. The POF is defined in Eq. (3) and illustrated in Fig. 6.

$$JY8: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1))^{\alpha_t} \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1))^{\beta_t} \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} x_i^2, G(t) = \sin(0.5\pi t) \\ A_t = 0.05, W_t = 6, \alpha_t = \frac{2}{\beta_t}, \beta_t = 10 - 9.8 * |G(t)| \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1}. \end{cases} \tag{12}$$

The first three types of change can be easily realized when constructing test functions, and they have been commonly reported in the literature. However, many real-world optimization problems with dynamic characteristics, e.g., the controller selection for dynamic plants [20], may jump between types. To the best of our knowledge, none of this kind of test function has been introduced into the family of DMOPs. In this paper, we propose such a problem that cyclically switches from types I to II, then to type III. Technically, this kind of problem is macroscopically a type II problem from the perspective of the whole period of changes. Despite that, we would refer to this kind of change as the mixed type from a microscopic angle, which can help us to analyze the performance of algorithms on a problem with changing types. This type of problem can be formulated as follows:

$$JY9: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1)) \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1)) \\ g(\mathbf{x_{II}}, t) = \sum\limits_{x_i \in \mathbf{x_{II}}} (x_i + \sigma - G(t))^2, G(t) = |\sin(0.5\pi t)| \\ A_t = 0.05, \quad W_t = \lfloor 6 \sin^\sigma(0.5\pi(t-1)) \rfloor \\ \sigma \equiv \left\lfloor \frac{\tau}{\tau_t \rho_t} \right\rfloor \pmod 3 \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1} \end{cases} \tag{13}$$

where ρ_t represents the frequency of type change, and is suggested as $\rho_t = 5$, meaning that the current type lasts five

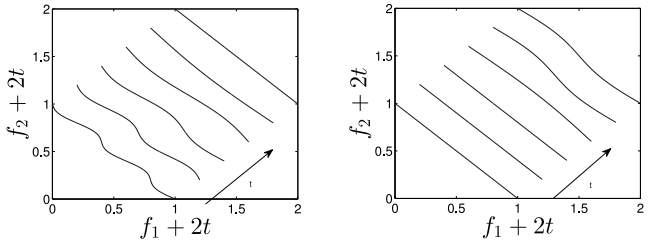


Fig. 7. POF of *JY*9 with 12 time windows varying from (a) 0.5 to 1 and (b) 1 to 1.5. For a better visualization, $f_1 + 2t$ and $f_2 + 2t$ are shown on the x- and y-axis, respectively.

time windows. If $\sigma = 0$, *JY*9 is a type I problem, and the POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, the POF is referred to Eq. (3) and similar to Fig. 1(b). If $\sigma = 1$, *JY*9 belongs to type II, where the POS is $x_i = G(t) - 1$, $\forall x_i \in \mathbf{x_{II}}$, and the POF is referred to Eq. (3) and illustrated in Fig. 7(a). If $\sigma = 2$, *JY*9 is a type III problem with the POS being $x_i = -1$, $\forall x_i \in \mathbf{x_{II}}$, and the POF not being (3) since $g(\mathbf{x_{II}}, t) \neq 0$. In this case, the minimum of $g(\mathbf{x_{II}}, t)$ is $g^*(t) = (n-1)(1 - G(t))^2$. Thus, the POF is

$$f_1 + f_2 = \big(1 + g^*(t)\big) \times \left(1 + 2A_t \sin\left(W_t \pi \left(\frac{f_1 - f_2}{2(1 + g^*(t))} + \frac{1}{2}\right)\right)\right) \tag{14}$$

where the POF is illustrated in Fig. 7(b).

To increase the flexibility of type changes, that is, the problem can be any type after a type change, we introduce a more challenging problem as follows:

$$JY10: \begin{cases} \min \quad F(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))^T \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(x_1 + A_t \sin(W_t \pi x_1))^{\alpha_t} \\ f_2(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}}, t))(1 - x_1 + A_t \sin(W_t \pi x_1))^{\beta_t} \\ g(\mathbf{x_{II}}, t) = \sum_{x_i \in \mathbf{x_{II}}} (x_i + \sigma - G(t))^2, G(t) = |\sin(0.5\pi t)| \\ A(t) = 0.05, \quad W(t) = 6 \\ \alpha_t = \beta_t = 1 + \sigma G(t), \sigma \equiv \left(\left\lfloor \frac{\tau}{\tau_t \rho_t} \right\rfloor + R\right) \pmod 3 \\ \mathbf{x_I} = (x_1) \in [0, 1], \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1} \end{cases} \tag{15}$$

where *JY*10 is almost defined the same as *JY*9, however, in addition to two time-dependent parameters α_t and β_t that control the simple overall POF shape, *JY*10 introduces a random integer $R \in [1, 3]$ to switch the problem into a random type of change every ρ_t time windows. This randomness makes the test problem hard to optimize. The POS of *JY*10 is the same as that of *JY*9, but the POF is a little different, which is defined as

$$f_1^{\frac{1}{\alpha_t}} + f_2^{\frac{1}{\beta_t}} = \big(1 + g^*(t)\big) \times \left(1 + 2A_t \sin\left(W_t \pi \left(\frac{f_1^{\frac{1}{\alpha_t}} - f_2^{\frac{1}{\beta_t}}}{2(1 + g^*(t))} + \frac{1}{2}\right)\right)\right) \tag{16}$$

where $g^*(t)$ is defined the same as in *JY*9, and the POF is similar to that of *JY*9 [as illustrated in Eq. (14)] except that there are two more time-varying parameters, i.e., α_t and β_t, in Eq. (16). When *JY*10 is in type III ($\sigma = 2$), the POF not only has a time-varying overall shape, but also shifts over time because of the time-changing value of $g^*(t)$. Due to the stochastic nature of change, it is not possible to draw the actual time-changing POFs of *JY*10.

In total, ten test instances are developed in this paper. In practice, more complicated benchmark problems can be generated by further varying parameters A_t, W_t, α_t, or β_t over time.

C. Comparison With Other Benchmarks

Table I presents a comparison between some existing test suites and our proposed one, where the main characteristics of each test suite are briefly tabulated. On the basis of the comparison, we would like to highlight the following features of the *JY* test suite.

1) The *JY* test suite introduces a new type of change, i.e., mixed type, to help classify DMOPs. Correspondingly, two mixed type instances are included in this test suite.
2) In addition to sharing some common POF properties, most of the *JY* instances have mixed convex/concave components on the POF, and the number of mixed components is controllable and can be time-changing.
3) There is a *JY* problem with nonlinear, time-varying, and nonmonotonic variable linkages, while variable linkages in other test suites, e.g., UDF and ZJZ, are monotonic, which may be easy to crack by hill-climbing methods.
4) *JY* develops a problem with random changes on the problem type, that is, the problem can be any type when a change occurs.
5) *JY* includes a problem with time-varying multimodality. This means the difficulty of the problem changes over time due to the changing number of local optima.
6) *JY* problems are newly developed whereas many existing DMOPs are adapted from their static counterparts or from the FDA test suite.

D. Discussion

Changing factors in real-world applications vary from problem to problem. Generally speaking, changing factors can be objective functions, decision variables, time-linkage, constraints, switch-mode changes, as revealed by the survey in [38]. As a consequence, real-life problems present various dynamic characteristics, such as time-changing POS and/or POF, disconnectivity, multimodality, undetectability, periodicity, and predictability. On the other hand, it is often hard or even impossible to use only a few artificial benchmarks to simulate all dynamic real-life problems as problems have a variety of mathematical definitions. Therefore, in this paper, the *JY* test problems are mainly aimed to imitate several representative dynamics of real-world applications instead of defining similar mathematical formulations, and these test problems have some dynamic properties in common with the referenced real-world examples.

TABLE I
COMPARISON OF CHARACTERISTICS INVOLVED IN SOME EXISTING TEST SUITES FOR DMO

Test Suite	Problem Type	POF Geometry	Variable-linkage	Randomness	Other Features	Origination
FDA [10]	Type I, Type II, Type III	continuous, concave, convex, time-varying curvature and/or location	no	no	unimodal	ZDT [50], DTLZ [9]
dMOP [13]	Type I, Type II, Type III	continuous, convex, time-varying curvature	no	random selection of diversity-related variables for dMOP3	unimodal	FDA [10]
DSW [37]	Type II	continuous, convex, time-varying location	no	no	unimodal, disconnected POS	Schaffer [41]
DIMP [30]	Type I	continuous, concave, convex	no	no	unimodal, multimodal, variables have different amount of change	ZDT [50]
T [20]	Type III, Type IV	continuous, concave, convex	no	no	unimodal, multimodal, time-varing number of objectives/variables	DTLZ [9]
HE [18]	Type III	continuous, discontinous, time-varying curvature	static and nonlinear linkage	no	unimodal, multimodal, isolated, deceptive	ZDT [50], LZ [34], WFG [21]
UDF [1]	Type II, Type III	continuous, discontinuous, time-varying curvature and/or location	nonlinear, time-varying and monotonic linkage	random selection of POF variations for UDF9 and UDF10	unimodal, multimodal, variables have different amount of change	UF [52]
ZJZ [58]	Type II	continuous, time-varying curvature	nonlinear, time-varying and monotonic linkage	no	unimodal	FDA [10]
JY	Type I, Type II, Type III, Mixed Type	mixed POF components, continuous, discontinuous, time-varying curvature and/or location	nonlinear, time-varying and non-monotonic linkage	random selection of problem types for JY10	unimdal, multimodal, time-varying multimodality, variables have different amount of change	New

Although there is a lack of clear link that to what extent test problems reflect real-world scenarios, it is not trivial to use these test problems to test and improve the performance of DMOAs. After all, what if these test problems appear in real-world applications and algorithms struggle to handle them? In this sense, the test problems can serve as a useful tool to identify strengths and weaknesses of algorithms before these algorithms can be applied to real-world DMOPs. Nevertheless, the test problems are far from being realistic, and it is necessary to further investigate their relation to real-world applications.

IV. PERFORMANCE METRICS

In this section, we present four performance metrics for DMO, which are spacing (S) [42], maximum spread (MS) [42], inverted generational distance (IGD) [24], [54], and our proposed robustness (R). The S and MS metrics are used to analyze algorithms' distribution and coverage, respectively. The IGD metric measures both distribution and proximity of an obtained approximation POF, thus a good IGD value implies that an algorithm performs well on spacing, coverage, and convergence simultaneously. To this end, we can use S, MS, and IGD to reveal algorithms' specific performance on different indicators. Besides, the robustness metric R is introduced to express the variability of an algorithm's performance over a number of changes in dynamic environments.

A. Schott's Spacing Metric

Schott [42] developed a metric with regard to the distribution of the discovered Pareto front, called the S metric. It measures how evenly the members in an approximated POF (denoted POF*) obtained by an algorithm are distributed, and is computed as

$$S = \sqrt{\frac{1}{n_{\text{POF}^*} - 1} \sum_{i=1}^{n_{\text{POF}^*}} (D_i - \overline{D})^2}$$
$$\overline{D} = \frac{1}{n_{\text{POF}^*}} \sum_{i=1}^{n_{\text{POF}^*}} D_i \tag{17}$$

where D_i is the Euclidean distance between the ith member in POF* and its nearest member in POF*.

A method of extending the spacing metric to examine the performance of algorithms for DMOPs is to define the average time steps in a run, as follows:

$$\overline{S} = \frac{1}{T_s} \sum_{t=1}^{T_s} S(t) \tag{18}$$

where $S(t)$ refers to the Schott's spacing metric at time instance t and is calculated just before the next change occurs, and T_s is the number of time steps.

B. Maximum Spread

The MS, first introduced by Zitzler *et al.* [50], measures to what extent the extreme members in POF* has been reached. Goh and Tan [12] proposed a modified version of MS by taking into account the proximity of POF* toward POF, as follows:

$$\text{MS}' = \sqrt{\frac{1}{M} \sum_{k=1}^{M} \left[\frac{\min[\overline{\text{POF}_k}, \overline{\text{POF}_k^*}] - \max[\underline{\text{POF}_k}, \underline{\text{POF}_k^*}]}{\overline{\text{POF}_k^*} - \underline{\text{POF}_k^*}} \right]^2} \tag{19}$$

where $\overline{\text{POF}_k}$ and $\underline{\text{POF}_k}$ is the maximum and minimum of the kth objective in POF, respectively. Similarly, $\overline{\text{POF}_k^*}$ and $\underline{\text{POF}_k^*}$

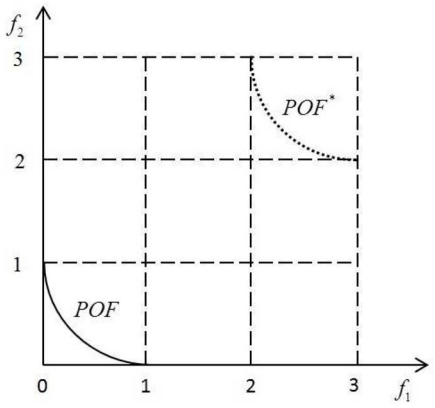


Fig. 8. Example of the obtained POF* far from the POF.

is the maximum and minimum of the kth objective in POF*, respectively.

A large value of MS′ indicates a good spread of POF*, and MS′ will have a value of one when POF* covers the whole POF. Sometimes, however, a high MS′ value can be deceptive. Fig. 8 gives such an example, where POF* is far from POF, and MS′ equals one since $\min[\overline{\text{POF}_k}, \overline{\text{POF}_k^*}]$ is smaller than $\max[\underline{\text{POF}_k}, \underline{\text{POF}_k^*}]$, $k = 1, 2$. In this case, the MS is not justifiable and may cause a misleading understanding of the approximated POF*. For this reason, we propose a revised MS (RMS) as follows:

$$\text{RMS} = \sqrt{\frac{1}{M}\sum_{k=1}^{M}\left[\frac{\mu_L\left(\left[\underline{\text{POF}_k^*}, \overline{\text{POF}_k^*}\right] \cap \left[\underline{\text{POF}_k}, \overline{\text{POF}_k}\right]\right)}{\mu_L\left(\left[\underline{\text{POF}_k^*}, \overline{\text{POF}_k^*}\right]\right)}\right]^2} \tag{20}$$

where $\mu_L(A)$ represents the Lebesgue measure [27] of the set A. Let us consider again the example illustrated in Fig. 8, by computing RMS, we can get RMS $= 0$. This is reasonable since the approximated POF* does not converge well, not to mention spread widely over the POF.

In order to apply the RMS metric to evaluate the performance of algorithms for DMOPs, an alternative method is to calculate the average of the RMS values over T_s time steps, as follows:

$$\overline{\text{RMS}} = \frac{1}{T_s}\sum_{t=1}^{T_s}\text{RMS}(t) \tag{21}$$

where RMS(t) represents the RMS value at time instance t.

C. Inverted Generational Distance

The IGD metric in [49] and [54] measures both convergence and diversity of found solutions by an algorithm. Let POF be a set of uniformly distributed points in the true POF, and POF* be an approximation of the POF. The IGD is calculated as follows:

$$\text{IGD} = \frac{\sum_{i=1}^{n_{\text{POF}}} d_i}{n_{\text{POF}}} \tag{22}$$

where $n_{\text{POF}} = \|\text{POF}\|$, d_i is the Euclidean distance between the ith member in POF and its nearest member in POF*.

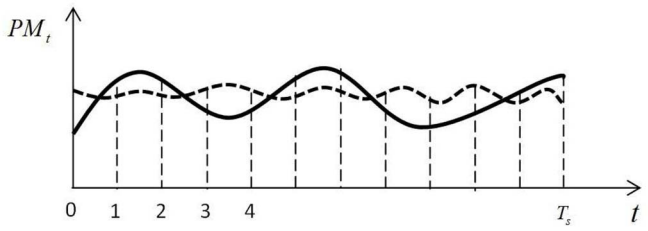


Fig. 9. Illustration of performance measure against time.

To have a low IGD value, POF* must be very close to POF and cannot miss any part of the whole POF.

Zhou *et al.* [57] modified the IGD metric and adopted the average of the IGD values in some time steps over a run as a performance indicator for DMO, which is computed as

$$\overline{\text{IGD}} = \frac{1}{T_s}\sum_{t=1}^{T_s}\text{IGD}(t) \tag{23}$$

where IGD(t) refers to the IGD metric at time instance t and is calculated just before the next change occurs.

D. Robustness

In dynamic environments, it is desirable to make algorithms as robust as possible. In other words, algorithms must be able to resist changes and be immune to an amount of uncertainty and perturbation. There have been some studies on robustness performance for dynamic optimization [11], [26], showing that robustness is a quite important goal in dynamic environments. Besides, many existing performance measures are adapted from their static counterparts and may not be suitable for dynamic environments. For example, suppose that PM_t (the smaller, the better) is a performance measure of population P_t at time t, the average value of PM_t over some time steps in a run is commonly used in dynamic optimization [13], [36], [57], [58]. As a compact form of assessment, average values are helpful for measuring algorithms' performance, but it cannot reflect the robustness performance. As illustrated in Fig. 9, the dashed and solid curves, respectively, represent the performance of two algorithms. Both algorithms have the same or similar average values in terms of PM_t, but the dashed is more robust than the other one, thus it is clear that the dashed achieves better performance on PM_t.

In this paper, the robustness of an algorithm on PM_t can be defined as

$$R(\text{PM}) = \sqrt{\frac{1}{T_s - 1}\sum_{t=1}^{T_s}\left(\text{PM}_t - \overline{\text{PM}}\right)^2} \tag{24}$$

where R(PM) denotes the robustness of the metric PM over T_s time steps and $\overline{\text{PM}}$ is the average of PM values over T_s time steps, i.e., $\overline{\text{PM}} = \sum_{t=1}^{T_s}\text{PM}_t/T_s$. A smaller value of R(PM) indicates a better robustness performance on the PM metric. Note that, PM can be any unary performance metric pertinent to the multiobjective optimization goals of proximity, diversity and distribution. With this definition of robustness, in Fig. 9, the dashed algorithm will achieve better performance than the solid one in terms of robustness on the performance measure PM_t.

V. Experimental Study

A. Compared Algorithms and Parameter Settings

We consider six multiobjective optimization algorithms (MOEAs) from the literature and compare their performance on our proposed test suite to assess the property and difficulty of these benchmark functions. These algorithms are classified into two groups. The first group includes four well-known dynamic MOEAs: 1) dynamic multiobjective particle swarm optimization (DMOPSO) algorithm [31]; 2) dynamic nondominated sorting genetic algorithm II (DNSGA-II) [7]; 3) dynamic competitive-cooperative coevolutionary algorithm (dCOEA) [13]; and 4) population prediction strategy (PPS) [57]. Each algorithm in this group has a mechanism of dealing with dynamism for dynamic optimization. Note that, there are two versions of DNSGA-II and DNSGA-II with randomly created solutions whenever a change occurs is adopted here. The second group includes two classic MOEAs: 1) strength Pareto evolutionary algorithm II (SPEA2) [51] and 2) MOEA based on decomposition (MOEA/D) [53], and they are high-performance algorithms for static multiobjective optimization. To handle dynamic environments, the algorithms of this group adopt the following strategy in our experiments: 10% randomly selected population members is re-evaluated for change detection, and the restart scheme is employed for change response. The parameter settings for all the tested algorithms are inherited from the referenced papers.

The experiments were conducted at different combinations of change severity levels and frequencies, i.e., $(n_t, \tau_t) = (5, 10)$, $(10, 10)$, and $(10, 5)$. To guarantee the fairness for all the tested algorithms, the total number of changes for problems $JY1$–$JY10$ was set to 20 during the evolution. Besides, 100 more generations were given to each algorithm before the first change to minimize the potential effect of static optimization. Thus, the total number of generations for running an algorithm was $100 + 20\tau_t$. For each problem, the number of decision variables was set to 10, and each tested algorithm, with a population size of 100, was executed 30 runs, and the experimental results were recorded. The experimental comparison employs the four performance metrics mentioned in Section IV, and the IGD metric is selected as the indicator PM for the robustness metric. That is, we use R(IGD) to reflect the robustness performance of MOEAs for DMO. Furthermore, for the computation of the IGD metric, 500 uniformly-distributed points (using the kth nearest neighbor truncation method [51]) were sampled from the true POF at each time step.

B. Experimental Results

For each combination (n_t, τ_t) of a test problem, we conducted the Wilcoxon rank-sum test [19] at the 0.05 significance level to judge whether the results of an algorithm are significantly different from those of another algorithm on each considered performance metric. An algorithm that outperforms most of the competitors will rank the first, and the one that outperforms the least will be assigned the worst rank. In case that several algorithms have the same number of outperformed algorithms, they share the same rank. After that, the average rank of an algorithm on the three combinations of a problem is calculated. As a result, for each problem, the algorithm with the smallest average rank value is further considered as the best, and the one with the second smallest average rank value takes the second place, and so on. Using this ranking method, the final rank of each algorithm under each performance metric for every test problem is indicated in Table II, where ordering in the rank column is purely alphabetical. For a close inspection of differences between the compared algorithms on each metric, readers are referred to our supplementary material.

It can be observed from the table that, on most of the test problems, dCOEA and MOEA/D obtain the best population distribution in their approximations. This performance might

TABLE II
Performance Rankings on Four Metrics for Benchmarks $JY1$–$JY10$

Prob.	Rank	Ranking by $\overline{S}$	Ranking by $\overline{RMS}$	Ranking by $\overline{IGD}$	Ranking by $R(IGD)$
JY1	1th	dCOEA	DMOPSO	MOEA/D dCOEA	MOEA/D
	2nd	MOEA/D	DNSGA-II	PPS	dCOEA
	3rd	PPS	PPS	DMOPSO	PPS
	4th	DNSGA-II	SPEA2	SPEA2	DMOPSO
	5th	DMOPSO	MOEA/D	DNSGA-II	SPEA2
	6th	SPEA2	dCOEA		DNSGA-II
JY2	1th	dCOEA	DMOPSO	dCOEA	dCOEA
	2nd	MOEA/D	DNSGA-II	MOEA/D	MOEA/D
	3rd	PPS	PPS	PPS	PPS
	4th	DNSGA-II	SPEA2	DMOPSO	DMOPSO
	5th	DMOPSO	MOEA/D	DNSGA-II	SPEA2
	6th	SPEA2	dCOEA	SPEA2	DNSGA-II
JY3	1th	DNSGA-II	DMOPSO	DNSGA-II	DNSGA-II
	2nd	MOEA/D	MOEA/D	dCOEA	dCOEA
	3rd	dCOEA	DNSGA-II SPEA2	PPS	MOEA/D
	4th	PPS	PPS	MOEA/D	PPS
	5th	SPEA2	dCOEA	SPEA2	SPEA2
	6th	DMOPSO		DMOPSO	DMOPSO
JY4	1th	MOEA/D	PPS SPEA2	MOEA/D	MOEA/D
	2nd	DNSGA-II	DMOPSO	dCOEA	dCOEA
	3rd	PPS	DNSGA-II	PPS	PPS
	4th	dCOEA	MOEA/D	DMOPSO	SPEA2
	5th	SPEA2	dCOEA	DNSGA-II SPEA2	DNSGA-II
	6th	DMOPSO			DMOPSO
JY5	1th	DNSGA-II	DMOPSO	DNSGA-II	DNSGA-II
	2nd	dCOEA	DNSGA-II	dCOEA	dCOEA
	3rd	MOEA/D	PPS	MOEA/D	DMOPSO
	4th	PPS	SPEA2	PPS	MOEA/D
	5th	DMOPSO	MOEA/D	DMOPSO	PPS
	6th	SPEA2	dCOEA	SPEA2	SPEA2
JY6	1th	MOEA/D	DMOPSO	dCOEA	DNSGA-II
	2nd	dCOEA	DNSGA-II	MOEA/D	dCOEA
	3rd	PPS	PPS	PPS SPEA2	DMOPSO
	4th	SPEA2	SPEA2	DMOPSO	MOEA/D
	5th	DMOPSO	MOEA/D	DNSGA-II	PPS
	6th	DNSGA-II	dCOEA		SPEA2
JY7	1th	dCOEA	DMOPSO	dCOEA	dCOEA
	2nd	MOEA/D	DNSGA-II	SPEA2	MOEA/D
	3rd	SPEA2	SPEA2	MOEA/D	SPEA2
	4th	DNSGA-II	PPS	DNSGA-II	DNSGA-II
	5th	PPS	dCOEA	PPS	PPS
	6th	DMOPSO	MOEA/D	DMOPSO	DMOPSO
JY8	1th	DNSGA-II	DMOPSO	DNSGA-II	DNSGA-II
	2nd	dCOEA	SPEA2	dCOEA	DMOPSO
	3rd	MOEA/D	DNSGA-II	PPS MOEA/D	PPS
	4th	PPS	PPS	DMOPSO	dCOEA
	5th	DMOPSO	MOEA/D	SPEA2	SPEA2 MOEA/D
	6th	SPEA2	dCOEA		
JY9	1th	MOEA/D	DMOPSO	MOEA/D	MOEA/D
	2nd	dCOEA	DNSGA-II	dCOEA	dCOEA
	3rd	PPS SPEA2	PPS	SPEA2	SPEA2
	4th	DNSGA-II	SPEA2	PPS	DMOPSO PPS
	5th	DMOPSO	MOEA/D	DMOPSO	DNSGA-II
	6th		dCOEA	DNGA-II	
JY10	1th	dCOEA	SPEA2	dCOEA	dCOEA
	2nd	PPS	DNSGA-II	SPEA2	SPEA2
	3rd	SPEA2	DMOPSO	MOEA/D	DMOPSO
	4th	DNSGA-II	PPS	DMOPSO	DNSGA-II MOEA/D
	5th	MOEA/D	MOEA/D	DNSGA-II	PPS
	6th	DMOPSO	dCOEA	PPS	

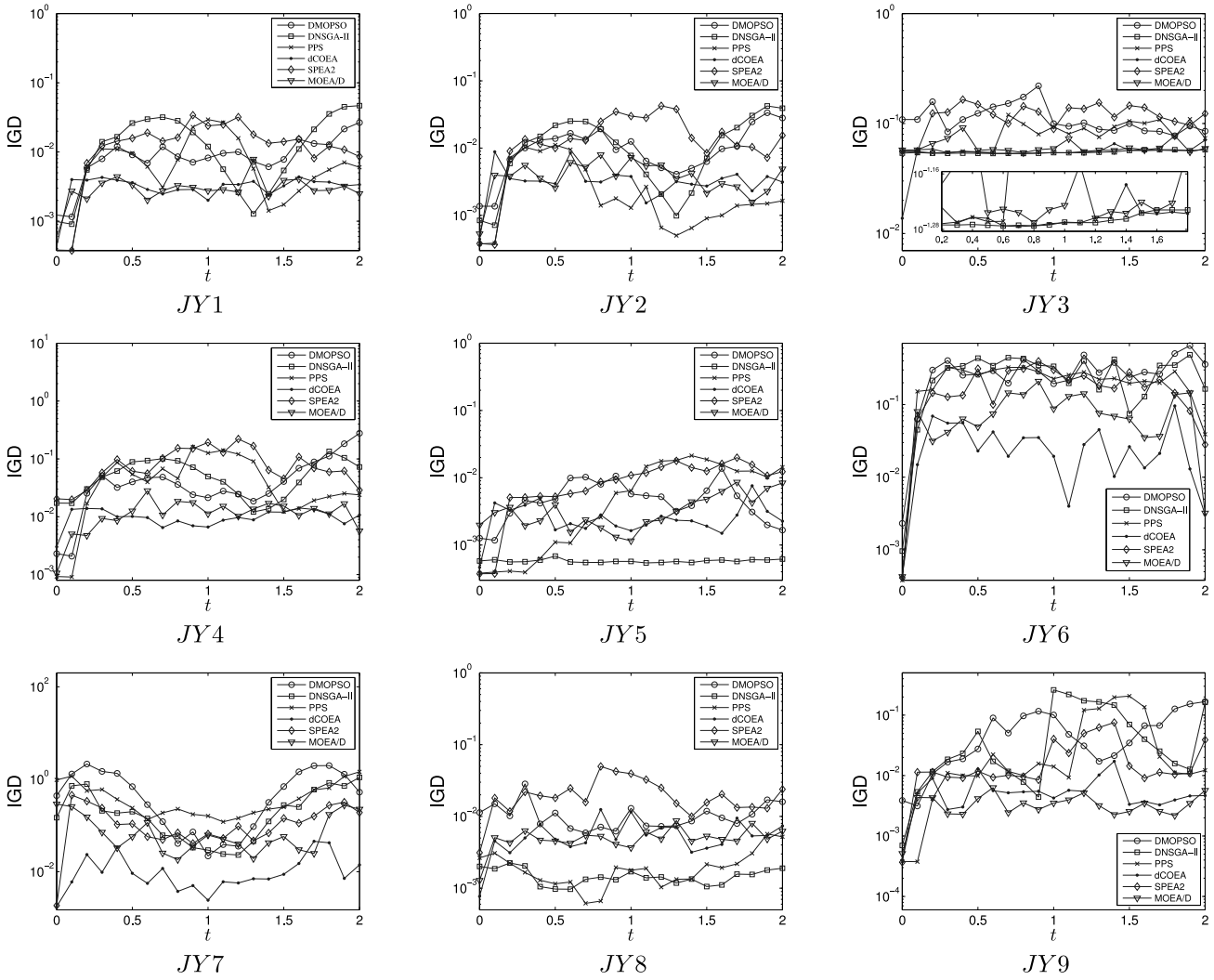


Fig. 10. Tracking of the IGD values obtained by six algorithms for time t from 0 to 2.

be attributed to the fact that, dCOEA employs a multipopulation strategy to keep diversity during the search and MOEA/D always keeps a set of well-diversified subproblems regardless of environmental changes. However, due to the use of the niche-sharing method to preserve diversity, dCOEA faces difficulties in distributing the population uniformly along the discontinuous POF components of *JY*4, and for the same reason, DMOPSO also faces this problem. MOEA/D has poor distribution performance on *JY*10 as it may require a long time to adapt to the change of problem type. It is not surprising that DNSGA-II achieves good distribution performance for *JY*5 and *JY*8 because their POS remains stationary, and when the POS is static, the optimization task is to adjust the distribution of solutions on the time-varying POFs, which is easy for crowding-distance [8] based DNSGA-II. It is understandable that DMOPSO and SPEA2 perform poorly on the majority of the test problems regarding the $\overline{S}$ metric due to the lack of diversity maintenance or diversity increase methods in the event of change.

Considering the $\overline{\text{RMS}}$ metric, we can see that dCOEA and MOEA/D are outperformed by the other algorithms for almost all the problems although they can achieve good distribution mentioned above. This can be explained by the fact that all the algorithms except dCOEA and MOEA/D explicitly or implicitly tend to reward boundary solutions, which may favor the coverage of an approximation. Notably, DMOPSO is always the top-performing algorithm because it prefers to choose boundary solutions as leaders since they are located at less crowded areas, thus naturally driving more solutions toward boundary regions.

The $\overline{\text{IGD}}$ metric reveals both diversity and convergence performance of an algorithm. It can be clearly seen that, dCOEA and MOEA/D are high-performing algorithms on the majority of the problems although their $\overline{\text{RMS}}$ values are not very competitive. $\overline{S}$ and $\overline{\text{IGD}}$ jointly indicate that dCOEA and MOEA/D track changing environments better than the other algorithms on these problems. Despite the fast convergence, MOEA/D seems ineffective for handling the time-varying nonmonotonic variable linkages of *JY*3, on which DNSGA-II performs the best. Besides, DNSGA-II also wins on *JY*5 and *JY*8 as both problems have static POS. *JY*2 and *JY*9 have very similar definitions except that *JY*9 involves a mixed type of change. On *JY*2, SPEA2 is the bottom performer, as indicated by its $\overline{\text{IGD}}$ rank. However, for the harder problem *JY*9,

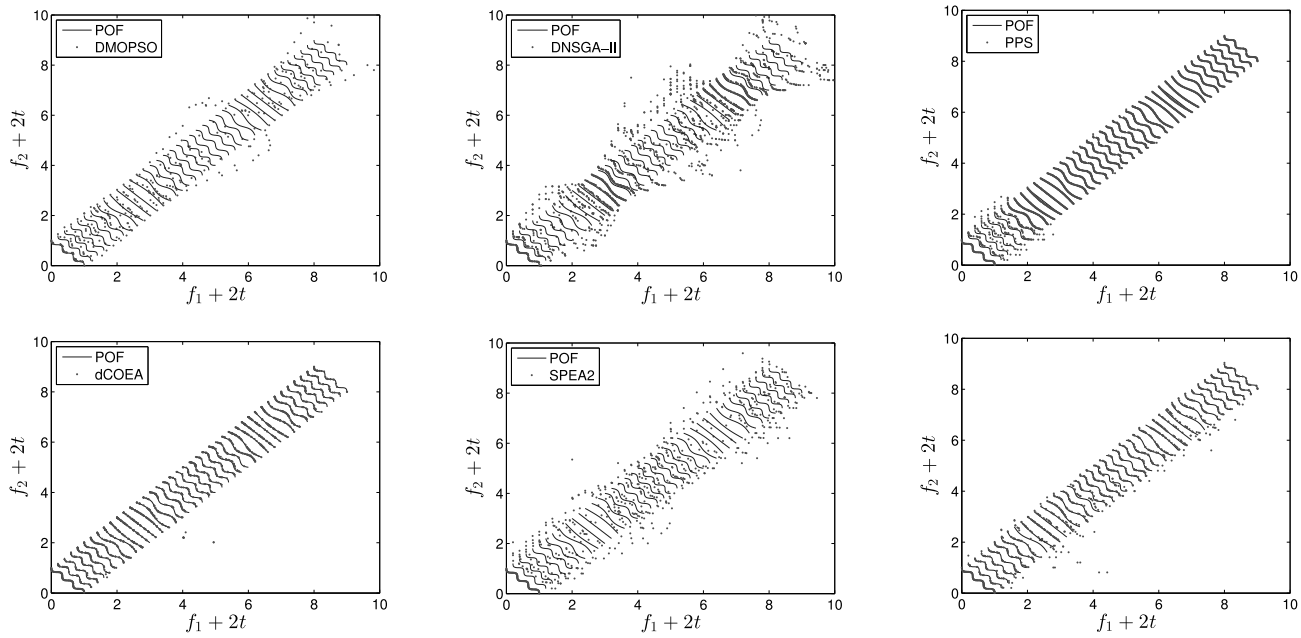


Fig. 11. POFs of *JY*2 with the lowest $\overline{\text{IGD}}$ values obtained by six algorithms for time t from 0 to 4.

SPEA2 outperforms PPS, DMOPSO and DNSGA-II, and dCEOA and MOEA/D swap their ranks. This means, the increase of difficulty caused by the mixed type of change significantly influences those algorithms' performance. On the other hand, by comparing the $\overline{\text{IGD}}$ ranks between *JY*9 and *JY*10, randomness brings about great challenges to MOEA/D and PPS. Besides, the $\overline{S}$, $\overline{\text{RMS}}$, and $\overline{\text{IGD}}$ metrics together help compare the convergence ability of algorithms. For example, PPS is better than MOEA/D on *JY*3, and dCOEA is better than DNSGA-II on *JY*4.

The last indicator, i.e., R(IGD), illustrates the variability of the performance of the six algorithms under a number of changes. Generally, the R(IGD) rank of each algorithm is roughly consistent with the $\overline{\text{IGD}}$ rank. This is understandable because the computation of the robustness metric is based on the $\overline{\text{IGD}}$ metric. Despite that, robustness provides a way to break ties between algorithms when they perform similarly in terms of a metric, e.g., MOEA/D and dCOEA for *JY*1, DNSGA-II and SPEA2 for *JY*4, PPS and SPEA2 for *JY*6, and PPS and MOEA/D for *JY*8. In addition, R(IGD) gives an insight to the stability of response to environmental changes during the evolution. For example, for *JY*2, *JY*3, and *JY*6, although DNSGA-II on average tracks their changing POFs better than SPEA2, as indicated by $\overline{\text{IGD}}$, R(IGD) shows that SPEA2 responses to changes more stably than DNSGA-II. Other examples of inconsistency between $\overline{\text{IGD}}$ and R(IGD) are MOEA/D and PPS for *JY*3 and *JY*8, DMOPSO and SPEA2 for *JY*4, DNSGA-II and dCOEA for *JY*6, MOEA/D and SPEA2 for *JY*7, and DMOPSO, PPS, and MOEA/D for *JY*5 and *JY*8.

Fig. 10 shows the tracking of IGD values obtained by the six algorithms over the period from $t = 0$ to $t = 2$ for the first nine test problems. The IGD curve of *JY*10 is not plotted here because of its stochastic nature of change. The figure gives a close inspection of algorithms' tracking ability and robustness. More specifically, DNSGA-II tracks the environmental changes stably for *JY*3, *JY*5, and *JY*8, dCOEA shows fairly robust performance for *JY*1–*JY*4 and *JY*7, while MOEA/D performs well in terms of robustness for *JY*1, *JY*4, and *JY*9. This is consistent with the results of the R(IGD) ranking list shown in Table II.

To have a better understanding of the tracking ability of these algorithms, we also plot the approximated POFs of *JY*2 and *JY*5 over a number of time steps in Figs. 11 and 12, respectively. In Fig. 11, dCOEA can approximate the POFs of *JY*2 very well, and at the beginning time steps PPS does not track the changing POFs well because the quality of history information stored by PPS is not high. Fig. 12 further shows the good tracking performance of several algorithms (i.e., DNSGA-II, PPS, and dCOEA for *JY*5).

Due to the page limit, we only list the impacts of time-changing multimodality, nonmonotonic variable-linkage, and frequency τ_t and severity n_t of change, which can be found in the supplementary material, in the following.

1) Generally, the performance of algorithms deteriorates when environments become tough (i.e., n_t and/or τ_t become smaller), and the effect of τ_t is more obvious and severe than that of n_t in this paper.
2) The time-varying multimodality significantly aggravates the performance of algorithms, resulting in a notable degradation in their tracking ability.
3) A comparison between *JY*3 and the ZJZ problem [58] reveals that nonmonotonic variable linkages pose great challenges to some algorithms so that the population coverage and convergence are negatively influenced.

C. Limitations

In the DMO community, the mean of performance measure over particular time steps is widely adopted as it can present the performance of algorithms in a compact form. Despite its

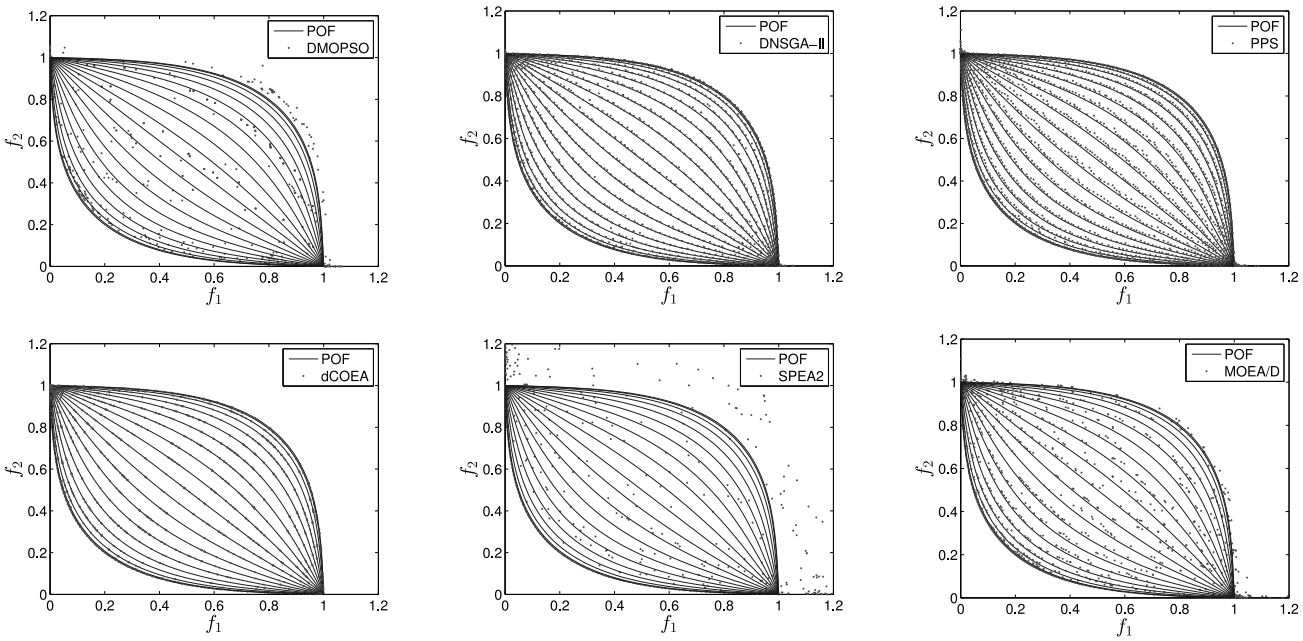


Fig. 12. POFs of *JY*5 with the lowest $\overline{\text{IGD}}$ values obtained by six algorithms for time t from 0 to 2.

significance and success, the mean value for experimental analysis might have limitations if the considered performance measure contains outliers at a time step [16]. For this reason, our used performance measures, i.e., $\overline{S}$, $\overline{\text{RMS}}$, and $\overline{\text{IGD}}$, may be inaccurate, and other approaches, if exist, are needed to further verify conclusions based on these measures. However, besides measuring algorithms' stability over a number of changes, our proposed robustness can be also used as a tool to embody the impact of outliers on a performance measure, because it takes into account the variability of the performance measure values over different time steps. If there are outliers at a specific time step influencing the performance measure, the robust value will be large. This way, robustness provides additional information to justify the performance of algorithms, increasing the reliability of performance analysis. On the other hand, no performance measure is perfect, and standard performance metrics are urgently needed in the field of DMO.

VI. Conclusion

An extensive study of the current DMOPs used to assess the performance of algorithms showed that there is a lack of standard test functions that could occur in real-world life for the DMO domain. The commonly used DMOPs are so simple that some characteristics are excluded, such as mixed convex/concave POF components, a more powerful diversity-resistant structure in the problem, a nonmonotonic correlation between variables, and a mixed type of problem that can jump between different types in a regular or random pattern.

To address the above shortcomings, this paper presents a generic scheme to generate desired benchmark functions that can compare the performance of different DMOAs. Furthermore, six representative MOEAs were tested on ten test instances generated by the benchmark generator and the results were evaluated by several performance metrics, including two new performance metrics proposed in this paper. The comparison among these algorithms shows that the proposed test instances are effective and can help to clearly distinguish the performance of each algorithm through proper statistical testing.

The key findings from the empirical study are summarized as follows.

1) Fast converging algorithms, i.e., MOEA/D and dCOEA, can adapt quickly to changing environments, thus they may have advantages in dealing with DMOPs.
2) The lack of diversity maintenance brings about severe consequences to algorithms like DMOPSO and SPEA2 in dynamic environments. DNSGA-II, dCOEA, and MOEA/D performs very well regarding the spacing metric because they successfully maintain good population diversity during the evolution.
3) The modified MS helps assess algorithms' coverage over the POF. Niche sharing (DMOPSO) and crowding distance (DNSGA-II) tend to reward boundary solutions, thus the MS of algorithms with those schemes is roughly good.
4) Nonmonotonic variable linkages are significantly harder than monotonic ones. Time-changing multimodality (*JY*6) is more challenging than static multimodality. Besides, the mixed type of change (*JY*9) complicates dynamic environments and randomness (*JY*10) in type of change further challenges algorithms' performance.
5) Robustness is another important performance indicator for DMO, which helps to have a comprehensive assessment of algorithms' performance. The experimental results have shown that, when several algorithms achieve similar values on a performance metric and there

is no statistically significant difference among them, the proposed robustness performance measure can provide additional information to distinguish their performance. This is really helpful for a better understanding and comparison of algorithms' performance. The significance of this new indicator can be even clearer when a stable and high-performance transient response to an environmental change is pursued.

Despite that our proposed benchmark generator can produce a series of features that are rarely tested in the literature, further research is needed regarding how to extend the generator to many-objective problems where the number of objectives is easy to scale up. Besides, this paper focuses mainly on comparing the performance of existing different metaheuristics for DMO. It is greatly needed to design new and robust MOEAs that can handle various environmental changes. These issues will be left for further discussions in our future work.

ACKNOWLEDGMENT

The authors would like to thank Dr. A. Zhou, Dr. M. Helbig, and Prof. A. P. Engelbrecht for providing their kind help and source codes for the experiments.

REFERENCES

[1] S. Biswas, S. Das, P. N. Suganthan, and C. A. Coello Coello, "Evolutionary multiobjective optimization in dynamic environments: A set of novel benchmark functions," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 3192–3199.

[2] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Proc. EvoWorkshops Appl. Evol. Comput.*, Coimbra, Portugal, 2004, pp. 489–500.

[3] M. Cámara, J. Ortega, and F. de Toro, "Approaching dynamic multi-objective optimization problems by using parallel evolutionary algorithms," in *Advances in Multi-Objective Nature Inspired Computing*, vol. 272, C. A. Coello Coello, C. Dhaenens, and L. Jourdan, Eds. Berlin, Germany: Springer, 2010, pp. 63–86.

[4] C. A. Coello Coello, C. Dhaenens, and L. Jourdan, *Advances in Multi-Objective Nature Inspired Computing* (Studies in Computational Intelligence), vol. 272. Berlin, Germany: Springer, 2010.

[5] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: A survey on problems, methods and measures," *Soft Comput.*, vol. 15, no. 7, pp. 1427–1448, 2011.

[6] 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. 2013.

[7] K. Deb and S. Karthik, "Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling," in *Proc. 4th Int. Conf. Evol. Multi-Criterion Optim.*, Matsushima, Japan, 2007, pp. 803–817.

[8] K. Deb, A. Pratap, S. Agarwal, 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.

[9] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*, A. Abraham, L. Jain, and R. Goldberg, Eds. London, U.K.: Springer, 2005, pp. 105–145.

[10] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 425–442, Oct. 2004.

[11] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Finding robust solutions to dynamic optimization problems," in *Proc. 16th Eur. Conf. Appl. Evol. Comput.*, Vienna, Austria, 2013, pp. 616–625.

[12] C.-K. Goh and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 354–381, Jun. 2007.

[13] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.

[14] S.-U. Guan, Q. Chen, and W. Mo, "Evolving dynamic multi-objective optimization problems with objective replacement," *Artif. Intell. Rev.*, vol. 23, no. 3, pp. 267–293, 2005.

[15] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 881–897, Jun. 2013.

[16] M. Helbig and A. P. Engelbrecht, "Performance measures for dynamic multi-objective optimisation algorithms," *Inf. Sci.*, vol. 250, pp. 61–81, Nov. 2013.

[17] M. Helbig and A. P. Engelbrecht, "Benchmarks for dynamic multi-objective optimisation," in *Proc. IEEE Symp. Comput. Intell. Dyn. Uncertain Environ.*, Singapore, 2013, pp. 84–91.

[18] M. Helbig and A. P. Engelbrecht, "Benchmarks for dynamic multi-objective optimisation algorithms," *ACM Comput. Surveys*, vol. 46, no. 3, pp. 1–39, Jan. 2014.

[19] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. New York, NY, USA: Wiley-Interscience, 1999.

[20] L. Huang, I. H. Suh, and A. Abraham, "Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants," *Inf. Sci.*, vol. 181, no. 11, pp. 2370–2391, 2011.

[21] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.

[22] S. Jiang and S. Yang, "A benchmark generator for dynamic multi-objective optimization problems," in *Proc. U.K. Workshop Comput. Intell. (UKCI)*, Bradford, U.K., 2014, pp. 1–8.

[23] S. Jiang and S. Yang, "A framework of scalable dynamic test problems for dynamic multi-objective optimization," in *Proc. IEEE Symp. Comput. Intell. Dyn. Uncertain Environ.*, Orlando, FL, USA, 2014, pp. 32–39.

[24] S. Jiang and S. Yang, "An improved multiobjective optimization evolutionary algorithm based on decomposition for complex Pareto fronts," *IEEE Trans. Cybern.* [Online]. Available: http://dx.doi.org/10.1109/TCYB.2015.2403131

[25] Y. Jin and B. Sendhoff, "Constructing dynamic optimization test problems using the multi-objective optimization concept," in *Proc. EvoWorkshops Appl. Evol. Comput.*, Coimbra, Portugal, 2004, pp. 525–536.

[26] 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. 3–8, 2013.

[27] F. Jones, *Lebesgue Integration on Euclidean Space*. London, U.K.: Jones Bartlett Learn., 2001.

[28] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1808–1820, Oct. 2014.

[29] W. Kong, T. Chai, S. Yang, and J. Ding, "A hybrid evolutionary multi-objective optimization strategy for the dynamic power supply problem in magnesia grain manufacturing," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2960–2969, 2013.

[30] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Comput.*, vol. 2, no. 2, pp. 87–110, 2010.

[31] M. S. Lechuga, "Multi-objective optimisation using sharing in swarm optimisation algorithms," Ph.D. dissertation, School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2009.

[32] J. Lepagnot, A. Nakib, H. Oulhadj, and P. Siarry, "A new multi-agent algorithm for dynamic continuous optimization," *Int. J. Appl. Metaheuristic Comput.*, vol. 1, no. 1, pp. 16–38, 2010.

[33] C. Li, S. Yang, and D. Pelta, "Benchmark generator for CEC'2012 competition on evolutionary computation for dynamic optimization problems," School Comput. Sci., China Univ. Geosci., Wuhan, China, Tech. Rep., 2011.

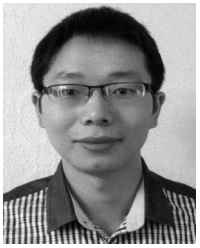
[34] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[35] M. Li, S. Yang, K. Li, and X. Liu, "Evolutionary algorithms with segment-based search for multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1295–1313, Aug. 2014.

[36] R. Liu, Y. Chen, W. Ma, C. Mu, and L. Jiao, "A novel cooperative coevolutionary dynamic multi-objective optimization algorithm using a new predictive model," *Soft Comput.*, vol. 18, no. 10, pp. 1913–1929, 2013.

[37] J. Mehnen, G. Rudolph, and T. Wagner, "Evolutionary optimization of dynamic multiobjective functions," in *Proc. 2nd Italian Workshop Evol. Comput.*, 2006, pp. 1–10.

[38] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1–24, Oct. 2012.
[39] P. C. Roy, M. M. Islam, K. Murase, and X. Yao, "Evolutionary path control strategy for solving many-objective optimization problem," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 702–715, Apr. 2015.
[40] S. Salomon, G. Avigad, P. J. Fleming, and R. C. Purshouse, "Active robust optimization: Enhancing robustness to uncertain environments," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2221–2231, Nov. 2014.
[41] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genet. Algorithms*, Pittsburgh, PA, USA, 1985, pp. 93–100.
[42] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," M.S. thesis, Dept. Aeronaut. Astronaut., Massachusetts Technol., Cambridge, MA, USA, 1995.
[43] R. Tinós and S. Yang, "A self-organizing random immigrants genetic algorithm for dynamic optimization problems," *Genet. Program. Evol. Mach.*, vol. 8, no. 3, pp. 255–286, 2007.
[44] J. Wahle, O. Annen, C. Schuster, L. Neubert, and M. Schreckenberg, "A dynamic route guidance system based on real traffic data," *Eur. J. Oper. Res.*, vol. 131, no. 2, pp. 302–308, 2001.
[45] H. Wang, D. Wang, and S. Yang, "A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems," *Soft Comput.*, vol. 13, nos. 8–9, pp. 763–780, 2009.
[46] L. Wismans, E. van Berkum, and M. Bliemer, "Handling multiple objectives in optimization of externalities as objectives for dynamic traffic management," *Eur. J. Transp. Infrastruct. Res.*, vol. 14, no. 2, pp. 159–177, 2014.
[47] P. P.-Y. Wu, D. Campbell, and T. Merz, "Multi-objective four-dimensional vehicle motion planning in large dynamic environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 3, pp. 621–634, Jun. 2011.
[48] R. K. Ursem, T. Krink, and B. Filipic, "A numerical simulator of a crop-producing greenhouse," Dept. Comput. Sci., Univ. Aarhus, Aarhus, Denmark, Tech. Rep. EVALife 2002-01, 2002.
[49] S. Yang and X. Yao, *Evolutionary Computation for Dynamic Optimization Problems* (Studies in Computational Intelligence), vol. 490. Berlin, Germany: Springer, 2013.
[50] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[51] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Probl. (EUROGEN)*, Athens, Greece, 2001, pp. 95–100.
[52] Q. Zhang *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School Comput. Sci. Elec. Eng., Univ. Essex, Colchester, U.K., Nanyang Technol. Univ., Singapore, Tech. Rep. CES-487, 2008, pp. 1–30.
[53] 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.
[54] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.
[55] Z. Zhang, "Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control," *Appl. Soft Comput.*, vol. 8, no. 2, pp. 959–971, 2008.
[56] Z. Zhang and S. Qian, "Artificial immune system in dynamic environments solving time-varying non-linear constrained multi-objective problems," *Soft Comput.*, vol. 15, no. 7, pp. 1333–1349, 2011.
[57] 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.
[58] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization," in *Proc. 4th Int. Conf. Evol. Multi-Criterion Optim.*, Matsushima, Japan, 2007, pp. 832–846.



Shouyong Jiang received the B.Sc. degree in information and computation science and the M.Sc. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2011 and 2013, respectively. He is currently pursuing the Ph.D. degree with the School of Computer Science and Informatics, De Montfort University, Leicester, U.K.

His current research interests include evolutionary computation, multiobjective optimization, and dynamic optimization.

Shengxiang Yang (M'00–SM'14) received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China, in 1993, 1996, and 1999, respectively.

He is currently a Professor of Computational Intelligence and the Director of the Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, U.K. He has over 200 publications. His current research interests include evolutionary and genetic algorithms, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling, and relevant real-world applications.

Prof. Yang is the Chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, under the Evolutionary Computation Technical Committee of the IEEE Computational Intelligence Society and the Founding Chair of the Task Force on Intelligent Network Systems, under the Intelligent Systems Applications Technical Committee of the IEEE Computational Intelligence Society.