A Population Prediction Strategy for Evolutionary Dynamic Multiobjective Optimization

Aimin Zhou, *Member, IEEE,* Yaochu Jin, *Senior Member, IEEE,* and Qingfu Zhang, *Senior Member, IEEE*

***Abstract*—This paper investigates how to use prediction strategies to improve the performance of multiobjective evolutionary optimization algorithms in dealing with dynamic environments. Prediction-based methods have been applied to predict some isolated points in both dynamic single objective optimization and dynamic multiobjective optimization. We extend this idea to predict a whole population by considering the properties of continuous dynamic multiobjective optimization problems. In our approach, called population prediction strategy (PPS), a Pareto set is divided into two parts: a center point and a manifold. A sequence of center points is maintained to predict the next center, and the previous manifolds are used to estimate the next manifold. Thus, PPS could initialize a whole population by combining the predicted center and estimated manifold when a change is detected. We systematically compare PPS with a random initialization strategy and a hybrid initialization strategy on a variety of test instances with linear or nonlinear correlation between design variables. The statistical results show that PPS is promising for dealing with dynamic environments.**

***Index Terms*—Dynamic multiobjective optimization, evolutionary algorithm, prediction, time series.**

I. Introduction

In BOTH industrial applications and scientific research, there exists a large class of optimization problems having multiple objectives that change over time. This kind of problem is usually known as a dynamic multiobjective optimization problem (DMOP). On one hand, recent literature has shown that there is a rapid increase in research work dealing with DMOPs from various application areas, including scheduling [1]–[3], management [4], [5], planning [6]–[9], design [10], [11], and control [12], [13]. On the other hand, some scientific problems, such as machine learning [14], [15], bi-level optimization [16], and constrained optimization [17], [18], can be tackled in terms of DMOPs.

Manuscript received July 16, 2012; revised January 27, 2013; accepted January 31, 2013. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2011CB707104 and the National Natural Science Foundation of China under Grant 61273313. This paper was recommended by Associate Editor H. Ishibuchi.

A. Zhou is with the Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China (e-mail: amzhou@cs.ecnu.edu.cn).

Y. Jin is with the Department of Computing, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: yaochu.jin@surrey.ac.uk).

Q. Zhang is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: qzhang@essex.ac.uk).

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/TCYB.2013.2245892

According to the nature of the uncertainties in optimization, dynamic optimization problems can be classified into different categories [19], [20]. In this paper, we focus on the following class of DMOPs:

$$\begin{array}{ll} \text{minimize} & F(x,t) = (f_1(x,t), \ldots, f_m(x,t))^T \\ \text{subject to} & x \in \prod_{i=1}^{n}[a_i, b_i] \end{array} \tag{1}$$

where $t = 0, 1, 2, \ldots \in T$ represents time instants. $-\infty < a_i < b_i < +\infty$ for all $i = 1, \cdots, n$, $\prod_{i=1}^{n}[a_i, b_i] \subset R^n$ defines the feasible region of the decision space, and $x = (x_1, \cdots, x_n)^T$ is the decision variable vector. The objective vector $F(x,t) : R^n \times T \to R^m$ consists of m time varying objective functions: $f_i(x,t)$, $i = 1, \cdots, m$. R^m is the objective space. The DMOP defined in (1) can be considered a sequence of (stationary) multiobjective optimization problems (MOPs). An optimal tradeoff set, which is called Pareto set (PS) in the decision space and Pareto front (PF) in the objective space, defines the optimality of an MOP [21], [22]. Therefore, many algorithms target to trace the moving PF (PS).

Using evolutionary algorithms to solve dynamic optimization problems has attracted increasing attention in recent years [20], [23], [24]. Due to the nature of a DMOP, a dynamic multiobjective optimization evolutionary algorithm (DMOEA) could be regarded as to use a sequence of multiobjective optimization evolutionary algorithms (MOEAs) to tackle a sequence of correlated MOPs.

However, most of the current DMOEAs directly borrow the techniques from dynamic single objective optimization to enhance the performances of MOEAs in dynamic environments. The properties of DMOPs are thus neglected. In dynamic single objective optimization, the target is to trace a single moving optimal point, while in dynamic multiobjective optimization, we aim to trace the movement of the PF (PS). The main motivation of this paper is to design a prediction strategy for evolutionary dynamic multiobjective optimization by taking the properties of DMOPs into account. We assume that: 1) a DMOP consists of a sequence of stationary MOPs, and 2) the PFs (PSs) of consecutive MOPs are similar to each other in most cases, as done in most current DMOEAs. Based on the two assumptions, we propose a population prediction strategy (PPS) to reinitialize the population when the environment changes, substantially extending our previous work reported in [25]. Unlike our previous work in which a time series model is built for each point in the population [26],

Algorithm 1: A General DMOEA Framework

```
1  Set time step $t = 0$;
2  Initialize a population $P^t$;
3  while not terminate do
4      if Change() then
5          Set $t = t + 1$;
6          Maintain memory, tune algorithm parameters
           and/or reinitialize the population $P^t$;
7      else
8          Optimize the $t$-th MOP by using an MOEA for
           one generation;
9      end
10 end
```

PPS maintains only a single time series model to predict the whole population. The population (the approximation to the PF and PS) is divided into two parts: a center point and a manifold. A sequence of center points is maintained to predict the next center, and the previous manifolds are used to approximate the next manifold. The new population is therefore obtained by combining the predicted center and the estimated manifold.

The rest of this paper is organized as follows. Section II presents some related work on DMOEAs. Section III describes the population prediction strategy in detail. Section IV introduces the test instances, including four newly designed problems, and performance metrics used in the experiments. Sections V and VI give the experimental results and analysis. Finally, Section VII outlines some conclusions and suggests a few future research topics.

II. Related Work

This section presents a DMOEA framework and summarizes some related work on its major components.

Adapting search behavior quickly to the environmental changes is a key issue in dealing with dynamic problems. Therefore, how to balance the population diversity and convergence is extremely important for dynamic optimization. Change detection, change reaction, and MOP optimization are three major components of most existing DMOEAs. Algorithm 1 shows a general DMOEA framework.

Techniques from both multiobjective optimization [22], [27], [28] and dynamic optimization [20], [23], [24] have been incorporated into Alogrithm 1. In the following, we discuss a few main techniques used in each component of the DMOEA framework.

1) *Change Detection (Line 4):* This step detects whether a change has occurred, and if yes, finds out the degrees of similarity between current problem and previous ones. A change is usually detected by: a) reevaluating solutions [1], [6], [26], [29], or b) checking population statistical information [13], [19], [30]. The first approach is easy to implement but it assumes that there is no noise in function evaluations. The second one could overcome this shortcoming, but the algorithm may need some additional problem-dependent parameters. Most existing work still focuses on detecting whether a change has happened or not. A good strategy should further estimate the degree of changes, which might help the next two steps [1]–[3], [31].

2) *Change Reaction (Lines 5 and 6):* This step performs some actions in response to the detected change. Possible actions include the following.

Memory Maintenance: Add some individual points or information extracted from the current population into the memory and/or delete old information [26], [32], [33].

Parameter Tuning: Adapt algorithm parameters, such as the mutation probability [1].

Population Reinitialization: For some algorithms, the population needs to be reinitialized after an environmental change. The following techniques are widely utilized: a) reuse the previous population or the nondominated solutions from the previous populations [1], [6], [29], [32]–[34]; b) hyper mutate the previous population [1], [26], [35], [36]; c) randomly generate new solutions [1], [13], [29], [30], [37]; d) reuse recorded history populations [13]; e) predict the new population [26], [32], [33], [36], [38], [39] or the search boundaries [40]; and f) apply different crossover and mutation operators before/after a change has happened [41].

3) *MOP Optimization (Line 8):* This step tackles the current MOP for one generation. An MOEA developed for solving stationary MOPs is usually applied directly or after some modifications. The purpose of modifications is often to enhance the diversity of the population using various techniques, including: a) random immigrants, i.e., randomly generating some points into the population in each generation [42], [43]; b) recrudescence, i.e., generating solutions with high crossover and mutation probabilities [42]; c) memory mechanisms, e.g., maintaining a portion of dominated solutions [32], [33], [44]–[47]; and d) multiple populations or parallel computing [48], [49].

In practice, a DMOEA usually utilizes a combination of some of the above-mentioned techniques [50], [51]. It should also be noted that not all DMOEAs contain all the three steps. In some DMOEAs, MOEAs are applied to DMOPs directly or with minor modifications [52]–[57]; thus, only *Line 8* is implemented in these methods. Other DMOEAs assume that a DMOP changes constantly [32], [33], [41]; thus, *Line 4* is skipped.

It is worth mentioning that while most existing evolutionary algorithms for solving dynamic optimization problems aim to track the moving optimum or moving PS/PF, it is argued in [58] that a more practical alternative is to find an acceptable optimal/suboptimal solution that changes the most slowly over time, taking into account the cost to redesign new solutions in addition to the difficulty in closely tracking the moving optimum or PS/PF. More recently, a framework for achieving the robust optimal solutions over time has been proposed in [59].

This paper focuses on *Line 6* and introduces a strategy, which will be discussed with details in the following section to reinitialize the population after a change.

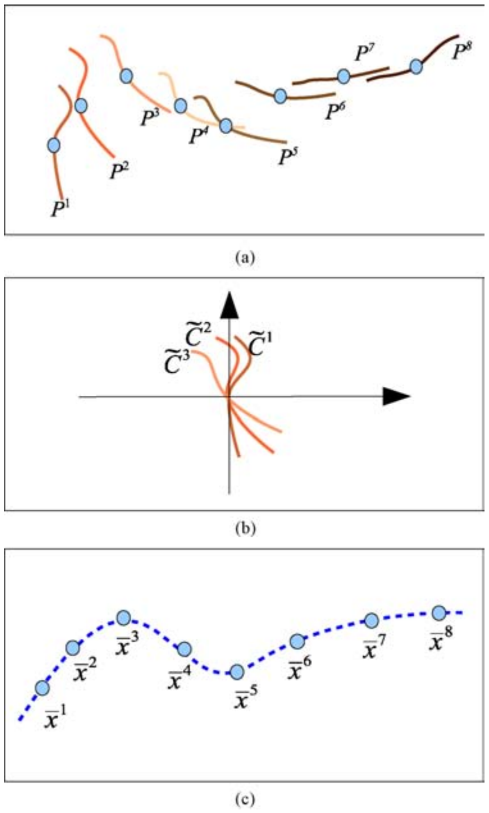

Fig. 1. Movement of PSs for a biobjective problem in a 2-D decision space. (a) Movement of PSs. (b) Movement of PS manifolds. (c) Movement of PS centers.

III. Population Prediction Strategy (PPS)

In this section, we address how to deal with environmental changes by using PPS for reinitializing a population when a change is detected. The basic idea of PPS is to utilize history information to predict an initial population that is close to the new PS. To achieve this goal, we first divide a population into two parts: a center point and a manifold, and then estimate each part to get a new population.

Under mild conditions, the PS of a continuous MOP with m objectives forms an $(m-1)$-dimensional piecewise continuous manifold [60]. Thus, in the t-th environment, we can divide PS^t into two parts: a center of mass $\bar{x}^t$, and a manifold $\widetilde{C}^t$ of which the center of mass is in the origin point, that is

$$PS^t \cong \bar{x}^t + \widetilde{C}^t. \tag{2}$$

Let $P^t = \{x^t\}$ be the output of the t-th stationary MOP, which is an approximation to PS^t. The center of mass of PS^t could be estimated by

$$\bar{x}^t = \frac{1}{|P^t|}\sum_{x^t \in P^t} x^t$$

where $|P^t|$ is the cardinality of P^t. Each point $x^t \in P^t$ can thus be formulated as

$$x^t = \bar{x}^t + \tilde{x}^t. \tag{3}$$

Then

$$\widetilde{C}^t = \{\tilde{x}^t\}$$

approximates the manifold of PS^t with the center of mass in the origin point.

In the case of biobjective problems, the PSs form piecewise 1-D curves. Fig. 1 illustrates, in a 2-D decision space, the movements of PSs, the PS manifolds, and the PS centers, respectively.

We deal with the centers and manifolds by different strategies. For the former, the center points $\{\bar{x}^0, \bar{x}^1, \cdots, \bar{x}^t\}$ form a time series naturally. Thus, we can use the history centers to predict the next center $\bar{x}^{t+1}$. For the latter, in Section I it is assumed that two consecutive PSs should be similar to each other in a sense. Thus, we could estimate $\widetilde{C}^{t+1}$ by the previous PS manifolds. With the estimated two parts: $\bar{x}^{t+1}$ and $\widetilde{C}^{t+1}$, we can get an initial population to approximate PS^{t+1}.

The following sections give the details to estimate $\bar{x}^{t+1}$ and $\widetilde{C}^{t+1}$, and deduce a method to generate a new population for the next time step.

A. PS Center Prediction

At time $t+1$, PPS maintains a sequence of history center points, $\bar{x}^{t-k}, k = 0, 1, \cdots, M-1$, where M is the maximum length of the sequence. This sequence forms a time series. Therefore, a time series prediction method could be applied here to forecast the next location of the center $\bar{x}^{t+1}$. Since we focus on the strategy to initialize a population, we choose a univariate autoregression (AR) model to do so as suggested in [33]. The properties and more details of the AR model are referred to in [61]. In the following, we briefly introduce the AR model and the calculation of the model parameters.

Let $\bar{x}^t = (\bar{x}^t_1, \cdots, \bar{x}^t_n)^T$, then $\bar{x}^{t+1}$ can be estimated by an AR(p) model for each component as

$$\bar{x}^{t+1}_i = \sum_{j=0}^{p-1} \lambda_{j,i}\bar{x}^{t-j}_i + \varepsilon^c_i \tag{4}$$

where $\lambda_{j,i}$ are parameters of the AR(p) model, $\varepsilon^c_i \sim N(0, \sigma^c_i)$ is white noise with variance σ^c_i, $i = 1, 2, \ldots, n$, $j = 0, 1, \cdots, p-1$, and p is the order of the AR model. Let

$$\Psi_i = (\bar{x}^t_i, \bar{x}^{t-1}_i, \cdots, \bar{x}^{t-M+p}_i)^T$$

$$\Phi_i = \begin{pmatrix} \bar{x}^{t-1}_i & \cdots & \bar{x}^{t-p+1}_i \\ \bar{x}^{t-2}_i & \cdots & \bar{x}^{t-p}_i \\ \vdots & \vdots & \vdots \\ \bar{x}^{t-M+p-1}_i & \cdots & \bar{x}^{t-M+1}_i \end{pmatrix}$$

$$\Lambda_i = (\lambda_{0,i}, \lambda_{1,i}, \cdots, \lambda_{p-1,i})^T.$$

Substituting the history points $\bar{x}^{t-k}, k = 0, 1, \cdots, M-1$ into (4), we get a matrix formulation

$$\Psi_i = \Phi_i\Lambda_i$$

where $i = 1, 2, \ldots, n$. The coefficient vector, Λ_i ($i = 1, 2, \ldots, n$), of the AR(p) model can be calculated by using the least squares regression method as

$$\Lambda_i = (\Phi_i^T\Phi_i)^{-1}\Phi_i^T\Psi_i.$$

The variance σ_i^c $(i = 1, 2, \ldots, n)$ is estimated as the average squared error

$$\sigma_i^c = \frac{1}{M-p} \sum_{k=t-M+p}^{t} [\overline{x}_i^j - \sum_{j=0}^{p-1} a_{j,i}\overline{x}_i^{k-j}]^2. \qquad (5)$$

B. PS Manifold Estimation

PPS records the last two approximated manifolds $\widetilde{C}^t$ and $\widetilde{C}^{t-1}$ to estimate the PS manifold $\widetilde{C}^{t+1}$. More specifically, each point $\widetilde{x}^t \in \widetilde{C}^t$ is applied to estimate a new point as

$$\widetilde{x}_i^{t+1} = \widetilde{x}_i^t + \varepsilon_i^m \qquad (6)$$

where $i = 1, \cdots, n$, $\varepsilon_i^m \sim N(0, \sigma_i^m)$. The variance σ_i^m is calculated as

$$\sigma_i^m = \frac{1}{n} D(\widetilde{C}^t, \widetilde{C}^{t-1})^2$$

and $D(A, B)$ measures the distance between manifolds A and B, which is defined as

$$D(A, B) = \frac{1}{|A|} \sum_{x \in A} \min_{y \in B} ||x - y||$$

where $|A|$ is the cardinality of A, and $||x-y||$ is the Euclidean distance between x and y.

It is clear that the variance is the same for each point and each dimension. There might be other ways to measure the similarity between two PS manifolds and to estimate the variance. We use (6) due to its simplicity in implementation.

C. Solution Generation

Suppose the noises in (4) and (6) are independent of each other. Substituting (4) and (6) into (3), we can deduce an equation to predict a new position for each $x^t \in P^t$ as

$$\begin{aligned} x_i^{t+1} &= \overline{x}_i^{t+1} + \widetilde{x}_i^{t+1} \\ &= (\sum_{j=0}^{p-1} \lambda_{j,i}\overline{x}_i^{t-j} + \varepsilon_i^c) + (\widetilde{x}_i^t + \varepsilon_i^m) \\ &= \sum_{j=0}^{p-1} \lambda_{j,i}\overline{x}_i^{t-j} + \widetilde{x}_i^t + \varepsilon_i \end{aligned} \qquad (7)$$

where $i = 1, 2, \cdots, n$, $\varepsilon_i \sim N(0, \sigma_i^c + \sigma_i^m)$, and $\sigma_i^c + \sigma_i^m$ denotes the variance of the white noise.

Hopefully, the initial population $\{x^{t+1}\}$ generated by (7) is close to PS^{t+1}.

D. PPS Procedure

The PPS procedure is integrated in the DMOEA framework. It targets to initialize a new population P^t when a change happens, i.e., at the beginning of time t. The PPS procedure is summarized as follows.

```
Step 1: If $t \leq p$, randomly sample half of $P^t$, randomly
        select the other half from $P^{t-1}$, and return.
Step 2: Estimate the AR($p$) model coefficients $\lambda_{j,i}$, the
        variances $\sigma_i^c$ and $\sigma_i^m$, $i = 1, 2, \cdots, n$, $j =
        0, 1, \cdots, p-1$, by using history information.
Step 3: For each point $x^{t-1}$ in $P^{t-1}$,
    3.1: generate a new point $y$ as (7),
    3.2: set
         $x_i^t = \begin{cases} y_i & \text{if } a_i \leq y_i \leq b_i \\ 0.5(a_i + x_i^{t-1}) & \text{if } y_i < a_i \\ 0.5(b_i + x_i^{t-1}) & \text{if } y_i > b_i \end{cases}$
         where $i = 1, 2, \cdots, n$, and put $x^t$ into $P^t$.
```

In ***Step 1***, we randomly sample half of the population and reuse half from the previous search when the history information is not enough to build an AR(p) model. In ***Step 3.1***, a new solution is predicted by the proposed strategy. Each new point is repaired to be inside the boundary if it is outside the boundary in ***Step 3.2***.

Compared to the feed-forward prediction strategy (FPS) proposed in [32] and [33], PPS has the following advantages: 1) a whole population, instead of some isolated points, is predicted by considering the properties of continuous DMOPs; and 2) in the running process, a time series model is built and the history centers and two manifolds are maintained in a memory. Thus both the time complexity and the space complexity of PPS are smaller than those of FPS.

IV. Test Instances and Performance Metrics

This section introduces the test instances and performance metrics used in the experiments.

A. Test Instances

Benchmark problems play important roles in assessing the performance of an algorithm and guiding the algorithm design. Some DMOPs have been proposed in [19] and [62], and variants of these test problems can be found in [26] and [54]. In the FDA test suite [19], the PF and/or the PS change with time, while the number of decision variables, the number of objectives and the boundaries of the search space keep fixed throughout the run. Another characteristic with the FDA test suite is that there is linear correlation between the decision variables. Furthermore, the geometric shapes of the PSs are 1-D line segments for biobjective problems and 2-D rectangles for triobjective problems. On one hand, there is no reason that the PS of a real world problem is so simple. On the other hand, this property may favor some offspring reproduction procedures as discussed in [63]. For this reason, we propose four new DMOP test instances, which have nonlinear correlation between the decision variables.

Table I lists all of the eight test problems and their PFs and PSs with details. Among them, F1 and F4 are from the FDA test suite [19], F2 and F3 are from [54], and F5–F8 are newly proposed. Figs. 2 and 3 plot their PFs and the projections of their PSs onto lower dimensional spaces. Fig. 4 plots the mean points of the PSs and shows the order of the moving PSs.

B. Performance Metrics

Some metrics have been designed for dynamic optimization [64], [65]. In this section, we first introduce the inverted generational distance (IGD) metric [63], [66] for stationary MOPs, and then propose a modified version of the IGD (MIGD) metric for DMOPs.

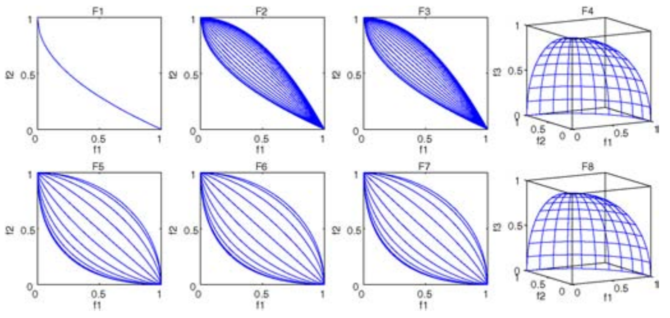

Fig. 2. PFs for F1–F8 with $t = 120, 121, 122, \cdots, 160$. For F1, F4, and F8, their PFs keep fixed throughout the run and for the others, their PFs change over time. The parameters are $n_T = 10$ and $n = 20$.

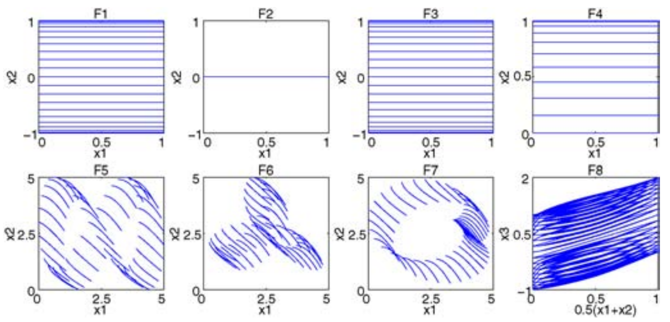

Fig. 3. Projections of the PSs onto the lower dimensional space for F1–F8 with $t = 120, 121, 122, \cdots, 160$. For F2, its PS keeps fixed throughout the run and for the others, their PSs change over time. The parameters are $n_T = 10$ and $n = 20$.

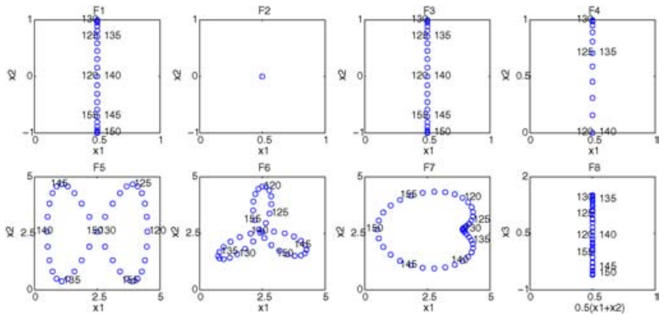

Fig. 4. Projections of the mean points of the PSs onto the lower dimensional space for F1–F8 with $t = 120, 121, 122, \cdots, 160$. The numbers beside the points denote time steps. The parameters are $n_T = 10$ and $n = 20$.

Let P^{t*} be a set of uniformly distributed Pareto optimal points in the PF^t. Let P^t be an approximation of PF^t. The IGD metric is defined as

$$IGD(P^{t*}, P^t) = \frac{\sum_{v \in P^{t*}} d(v, P^t)}{|P^{t*}|}$$

where $d(v, P^t) = \min_{u \in P^t} ||F(v) - F(u)||$ is the distance between v and P^t and $|P^{t*}|$ is the cardinality of P^{t*}. The IGD metric can measure both diversity and convergence. To have a low IGD value, P^t must be very close to PF^t and cannot miss any part of the whole PF^t.

The MIGD metric is then defined as the average of the IGD values in some time steps over a run

$$MIGD = \frac{1}{|T|} \sum_{t \in T} IGD(P^{t*}, P^t)$$

where T is a set of discrete time points in a run and $|T|$ is the cardinality of T.

In our experiments, 1000 points, in which f_1 or s taking 1000 equidistant values from their lower bounds to their upper bounds, are selected from the PFs of biobjective problems to be P^{t*} for computing the IGD metrics. $50 \times 50 = 2500$ points in the PFs of F4 and F8 with $u, v = \frac{0}{49} \times \frac{\pi}{2}, \frac{1}{49} \times \frac{\pi}{2}, \ldots, \frac{49}{49} \times \frac{\pi}{2}$, are taken to form P^{t*} for computing the IGD metric for experiments on triobjective problems.

V. Experiments

A. Compared Strategies and Parameter Settings

The DMOEA framework introduced in Section II is used in the experiments and its three major components are as follows.

Change Detection: Some randomly selected solutions are reevaluated to detect environmental changes.

Change Reaction: After a change, the population will be reinitialized by one of the following three strategies: 1) randomly initialize strategy (RIS), which randomly generates a new population in the feasible region of the decision space; 2) feed-forward prediction strategy (FPS) [26], [32], [33], which initializes a population by a hybrid strategy; and 3) PPS, which is proposed in this paper.

MOP Optimization: We choose RM-MEDA [63] as the MOEA optimizer. At time t, RM-MEDA evolves for one generation as follows.

```
Step 1: Build a probability model to capture the structure
        of the solutions in $P^t$ in the decision space.
Step 2: Sample a set of new solutions $Q$ which satisfies
        $|Q| = |P^t|$.
Step 3: Repair the infeasible solutions in $Q$ and evaluate
        the new solutions.
Step 4: Select the new population $P^t$ for the next genera-
        tion from $Q \cup P^t$.
```

In ***Step 2***, the regularity property [60] of continuous MOPs is considered for model building, which differentiates RM-MEDA from other probability model based MOEAs. In ***Step 4***, the nondominated sorting scheme [67] is used to do selection and more details about RM-MEDA are referred to [63].

The problem and algorithm parameters are as follows.

1) The problem parameters: The severity of changes is $n_T = 10$. The dimensions of the test problems are all set to be $n = 20$.
2) The population size: In the comparison, the population size is set to be 100 for biobjective problems and 200 for triobjective problems.
3) Parameters in PPS: The AR(p) model order is $p = 3$, and the length of history mean point series is $M = 23$.
4) Parameters in FPS: The FPS version proposed in [33] is applied in this paper. It uses the same AR(p) model as in PPS and the order is $p = 3$. The length of history mean point series is $M = 23$. The probability in prediction model is $p = 0.9$. In the initial population, there are $3(m + 1)$ predicted points; 70% of the rest points are

TABLE I
TEST INSTANCES USED IN OUR EXPERIMENTS: F1–F4 ARE CITED FROM [19] AND [54], F5–F8 ARE NEWLY DESIGNED TEST INSTANCES

Instance	Search Space	Objectives, PS and PF	Remarks
$F1$	$[0,1]\times[-1,1]^{n-1}$	$f_1(x,t)=x_1,\ f_2(x,t)=g(1-\sqrt{\frac{f_1}{g}})$,	FDA1 in [19].
		$g=1+\sum_{i=2}^{n}(x_i-G(t))^2,\ G=\sin(0.5\pi\frac{t}{n_T})$.	PF is fixed.
		PS(t): $0\le x_1\le 1,\ x_i=G$, for $i=2,\dots,n$.	PS changes.
		PF(t): $f_2=1-\sqrt{f_1},\ 0\le f_1\le 1$.	Two objectives.
$F2$	$[0,1]\times[-1,1]^{n-1}$	$f_1(x,t)=x_1,\ f_2(x,t)=g(1-(\frac{f_1}{g})^H)$,	dMOP1 in [54].
		$g=1+9\sum_{i=2}^{n}x_i^2,\ H=1.25+0.75\sin(0.5\pi\frac{t}{n_T})$.	PF changes.
		PS(t): $0\le x_1\le 1,\ x_i=0$, for $i=2,\dots,n$.	PS is fixed.
		PF(t): $f_2=1-f_1^H,\ 0\le f_1\le 1$.	Two objectives.
$F3$	$[0,1]\times[-1,1]^{n-1}$	$f_1(x,t)=x_1,\ f_2(x,t)=g(1-(\frac{f_1}{g})^H)$,	dMOP2 in [54].
		$g=1+\sum_{i=2}^{n}(x_i-G)^2$,	PF changes.
		$G=\sin(0.5\pi\frac{t}{n_T}),\ H=1.25+0.75\sin(0.5\pi\frac{t}{n_T})$.	PS changes.
		PS(t): $0\le x_1\le 1,\ x_i=G$, for $i=2,\dots,n$.	Two objectives.
		PF(t): $f_2=1-f_1^{H(t)},\ 0\le f_1\le 1$.	
$F4$	$[0,1]^n$	$f_1(x,t)=(1+g)\cos(0.5\pi x_2)\cos(0.5\pi x_1)$,	FDA4 in [19].
		$f_2(x,t)=(1+g)\cos(0.5\pi x_2)\sin(0.5\pi x_1)$,	PF is fixed.
		$f_3(x,t)=(1+g)\sin(0.5\pi x_2)$,	PS changes.
		$g(x,t)=\lvert\sum_{i=3}^{n}(x_i-G)^2\rvert,\ G=\sin(0.5\pi\frac{t}{n_T})$.	Three objectives.
		PS(t): $0\le x_1,x_2\le 1,\ x_i=G$, for $i=3,\dots,n$.	
		PF(t): $f_1=\cos(u)\cos(v),\ f_2=\cos(u)\sin(v),\ f_3=\sin(u),\ 0\le u,v\le\pi/2$.	
$F5$	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$,	PF changes.
		$f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$,	PS changes.
		$y_i=x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}},\ H=1.25+0.75\sin(\pi\frac{t}{n_T})$,	Two objectives.
		$a=2\cos(\pi\frac{t}{n_T})+2,\ b=2\sin(2\pi\frac{t}{n_T})+2$,	
		$I_1=\{i\mid 1\le i\le n, i \text{ is odd}\},\ I_2=\{i\mid 1\le i\le n, i \text{ is even}\}$.	
		PS(t): $a\le x_1\le a+1,\ x_i=b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}}$, for $i=2,\dots,n$.	
		PF(t): $f_1=s^H,\ f_2=(1-s)^H,\ 0\le s\le 1$.	
$F6$	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$,	PF changes.
		$f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$,	PS changes.
		$y_i=x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}},\ H=1.25+0.75\sin(\pi\frac{t}{n_T})$,	Two objectives.
		$a=2\cos(1.5\pi\frac{t}{n_T})\sin(0.5\pi\frac{t}{n_T})+2,\ b=2\cos(1.5\pi\frac{t}{n_T})\cos(0.5\pi\frac{t}{n_T})+2$,	
		$I_1=\{i\mid 1\le i\le n, i \text{ is odd}\},\ I_2=\{i\mid 1\le i\le n, i \text{ is even}\}$.	
		PS(t): $a\le x_1\le a+1,\ x_i=b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}}$, for $i=2,\dots,n$.	
		PF(t): $f_1=s^H,\ f_2=(1-s)^H,\ 0\le s\le 1$.	
$F7$	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$,	PF changes.
		$f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$,	PS changes.
		$y_i=x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}},\ H=1.25+0.75\sin(\pi\frac{t}{n_T})$,	Two objectives.
		$a=1.7(1-\sin(\pi\frac{t}{n_T}))\sin(\pi\frac{t}{n_T})+3.4,\ b=1.4(1-\sin(\pi\frac{t}{n_T}))\cos(\pi\frac{t}{n_T})+2.1$,	
		$I_1=\{i\mid 1\le i\le n, i \text{ is odd}\},\ I_2=\{i\mid 1\le i\le n, i \text{ is even}\}$.	
		PS(t): $a\le x_1\le a+1,\ x_i=b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}}$, for $i=2,\dots,n$.	
		PF(t): $f_1=s^H,\ f_2=(1-s)^H,\ 0\le s\le 1$.	
$F8$	$[0,1]^2\times[-1,2]^{n-2}$	$f_1(x,t)=(1+g)\cos(0.5\pi x_2)\cos(0.5\pi x_1)$,	PF changes.
		$f_2(x,t)=(1+g)\cos(0.5\pi x_2)\sin(0.5\pi x_1)$,	PS changes.
		$f_3(x,t)=(1+g)\sin(0.5\pi x_2)$,	Three objectives.
		$g=\sum_{i=3}^{n}(x_i-(\frac{x_1+x_2}{2})^H-G)^2$,	
		$H=1.25+0.75\sin(\pi\frac{t}{n_T}),\ G=\sin(0.5\pi\frac{t}{n_T})$.	
		PS(t): $0\le x_1,x_2\le 1,\ x_i=(\frac{x_1+x_2}{2})^H+G(t)$, for $i=3,\dots,n$.	
		PF(t): $f_1=\cos(u)\cos(v),\ f_2=\cos(u)\sin(v),\ f_3=\sin(u),\ 0\le u,v\le\pi/2$.	

$x=(x_1,\dots,x_n)$ is decision vector, $t=0,1,\cdots$ indexes discrete time, and n_T is a problem parameter.

inherited from the previous population and the other 30% are randomly sampled from the search space.

5) Parameter in RM-MEDA: The number of clusters is 5.
6) Algorithm cost: It is assigned to be 3000 for biobjective problems and 6000 for triobjective problems, i.e., the problems change every 30 generations.
7) Change detection: 5% randomly selected points from the population are reevaluated to detect the environment changes for all strategies.
8) Number of runs and stopping condition: We run each algorithm 20 times for each test instance independently. The algorithms stop when $t > 160$, i.e., there are 160 environmental changes in each run.

The statistical results of MIGD values for RM-MEDA with RIS, FPS, and PPS on F1–F8 over 20 runs are shown in Table II. The t-test results between RIS, FPS, and PPS are shown as well.

TABLE II

STATISTICAL RESULTS OF MIGD METRIC FOR RIS, FPS, AND PPS ON F1–F8 OVER 20 RUNS

		$0 \le t \le 20$					$20 < t \le 160$				
Instance	Strategy	Mean	First Q	Median	Third Q	t-test	Mean	First Q	Median	Third Q	t-test
	RIS	0.6208	0.5849	0.6106	0.6616	+	0.6262	0.6141	0.6275	0.6307	+
F1	FPS	**0.0140**	**0.0135**	**0.0140**	**0.0145**	–	0.0076	0.0075	0.0076	0.0076	+
	PPS	0.0560	0.0144	0.0354	0.0686		**0.0053**	**0.0052**	**0.0053**	**0.0053**	
	RIS	0.9493	0.9229	0.9444	0.9709	+	0.8773	0.8613	0.8786	0.8849	+
F2	FPS	**0.1452**	**0.0927**	0.1550	**0.1858**	–	**0.0046**	**0.0046**	**0.0046**	**0.0046**	–
	PPS	0.1996	0.0979	**0.1303**	0.2058		0.0052	0.0052	0.0052	0.0052	
	RIS	1.1935	1.1365	1.2126	1.2387	+	0.9009	0.8810	0.8935	0.9144	+
F3	FPS	**0.0226**	**0.0208**	**0.0227**	**0.0244**	–	0.0085	0.0084	0.0085	0.0085	+
	PPS	0.1536	0.0219	0.0547	0.1595		**0.0054**	**0.0053**	**0.0054**	**0.0054**	
	RIS	0.3169	0.3093	0.3180	0.3252	+	0.3166	0.3099	0.3154	0.3207	+
F4	FPS	**0.1193**	**0.1164**	**0.1198**	**0.1225**	–	0.0873	0.0869	0.0874	0.0877	+
	PPS	0.1211	0.1168	0.1202	0.1262		**0.0833**	**0.0828**	**0.0831**	**0.0835**	
	RIS	1.2390	1.1778	1.2387	1.2696	+	1.3093	1.2801	1.3157	1.3309	+
F5	FPS	**0.1661**	0.1230	0.1447	**0.1567**	–	0.0796	0.0670	0.0772	0.0891	+
	PPS	0.2111	**0.0855**	**0.1353**	0.3052		**0.0103**	**0.0085**	**0.0092**	**0.0112**	
	RIS	0.8559	0.8234	0.8586	0.8735	+	0.7111	0.7044	0.7111	0.7206	+
F6	FPS	0.3778	0.3191	0.3899	0.4265	–	0.0273	0.0255	0.0268	0.0278	+
	PPS	**0.3369**	**0.2116**	**0.3354**	**0.3877**		**0.0086**	**0.0080**	**0.0085**	**0.0087**	
	RIS	0.5376	0.5268	0.5377	0.5482	+	0.7462	0.7323	0.7457	0.7538	+
F7	FPS	**0.1418**	**0.1163**	**0.1359**	**0.1736**	–	0.0192	0.0186	0.0192	0.0197	+
	PPS	0.1809	0.1369	0.1673	0.1925		**0.0081**	**0.0077**	**0.0079**	**0.0081**	
	RIS	1.5403	1.5128	1.5345	1.5674	+	1.7568	1.7330	1.7547	1.7613	+
F8	FPS	**0.3289**	**0.3165**	**0.3277**	**0.3364**	–	**0.1535**	**0.1512**	**0.1540**	**0.1556**	–
	PPS	0.5216	0.4986	0.5187	0.5413		0.3691	0.3606	0.3678	0.3740	

+ (-) supports (fails to support) the hypothesis that PPS performs better than the compared strategy at the 95% significance level under t-test.

B. Results on F1–F4

We first test RIS, FPS, and PPS on F1–F4, which have linear correlation between decision variables. As shown in Fig. 3, the PSs of F1–F3 are line segments and the PSs of F4 are 2-D rectangles.

The statistical results of MIGD over 20 runs can be found in Table II. The mean, median, first, and third quartile values for $0 \le t \le 20$ and $20 < t \le 160$ are presented, respectively. To show the runtime performance, Fig. 5 plots the average IGD values versus the time.

It is clear from Table II that when $0 \le t \le 20$, FPS performs better than RIS and PPS on the four problems for most of the metric values. However, when $20 < t \le 160$, PPS shows better performance than both FPS and RIS on F1, F3 and F4, and FPS is better than PPS on F2. For all metric values on the four test instances, RIS is dominated by both FPS and PPS. The t-test with a 95% significance level in Table II and the run time performances in Fig. 5 also confirm these conclusions.

To visualize the quality of the obtained populations, we also draw the initial and final populations with the lowest MIGD metric values at time $t = 130, 135, 140, 145, 150$ on F2 and F3 in Figs. 6 and 7, respectively. The figures indicate that RIS fails to approximate the PFs at the beginning and after 30 generations of run on F2 and F3. Although after 30 generations, FPS and PPS can approximate the PFs very well. The quality of the initial predicted population of PPS is higher than that of FPS, as shown in Fig. 6.

It is understandable that RIS performs the worst among the three strategies because: 1) the given computational cost is not enough to tackle a stationary MOP, and 2) RIS does not take any effort to improve the algorithm performance as FPS and PPS do.

It is not difficult to explain why FPS performs better than PPS at the beginning stages. When $0 < t \le 20$, on one hand the quality of history information stored by PPS is not high; on the other hand, the number of stored mean points is small. The limitation of history information at the beginning stages deteriorates performance of PPS. On the contrary in FPS, only a small part of population is predicted, and a large part of the population is inherited from the previous population. When a change is detected, the population generated by FPS is more close to the PS than that of PPS at the beginning stages. While at later stages when $t > 20$, as the quality of history information stored by PPS improves, the quality of predicted population increases as well. However, the high quality history information does not help to improve the performance of FPS.

F2 has a fixed PS throughout the run. Since FPS always reuses a large part of the previous population to initialize a new population when a change is detected, it could approximate the PS and PF very quickly once it finds the PS of F2.

C. Results on F5–F8

In this section, we consider the newly proposed problems F5–F8, which have nonlinear correlation between decision variables. As shown in Fig. 3, the PSs of F5–F7 are 1-D curves and the PSs of F8 are 2-D surfaces.

The statistical results of MIGD on F5–F8 over 20 runs can be found in Table II. To show the runtime performance, Fig. 8

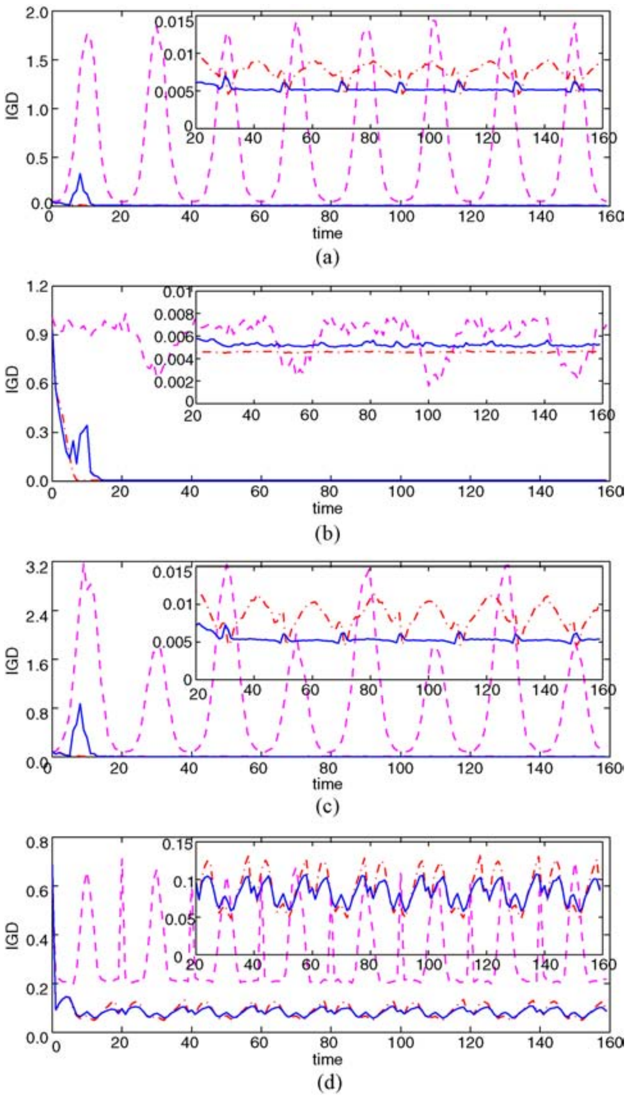

Fig. 5. Average IGD values over 20 runs versus time for RIS, FPS, and PPS on (a) F1, (b) F2, (c) F3, and (d) F4. The dashed lines are with RIS, dash-dot lines are with FPS, and solid lines are with PPS.

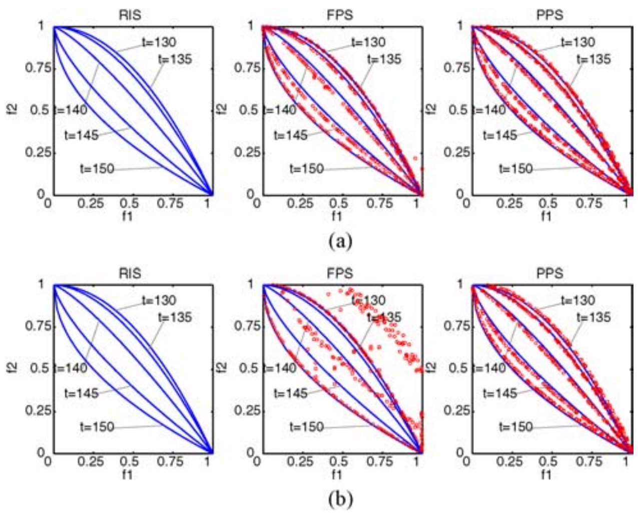

Fig. 6. Initial populations obtained by RIS, FPS, and PPS at t = 130, 135, 140, 145, and 150 with lowest MIGD values among 20 runs on (a) F2 and (b) F3.

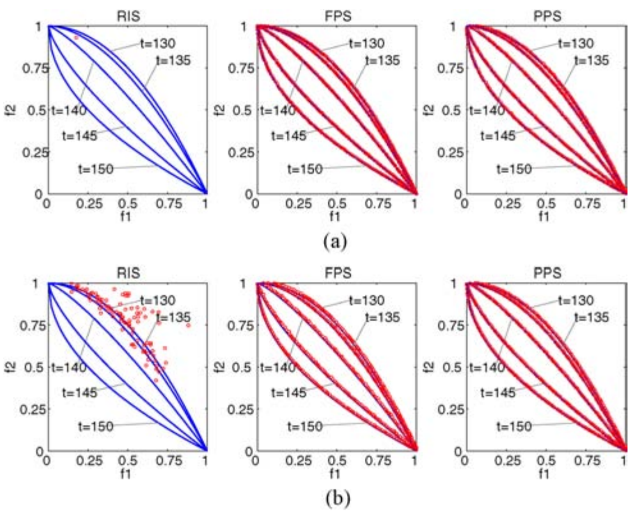

Fig. 7. Final populations obtained by RIS, FPS, and PPS at t = 130, 135, 140, 145, and 150 with lowest MIGD values among 20 runs on (a) F2 and (b) F3.

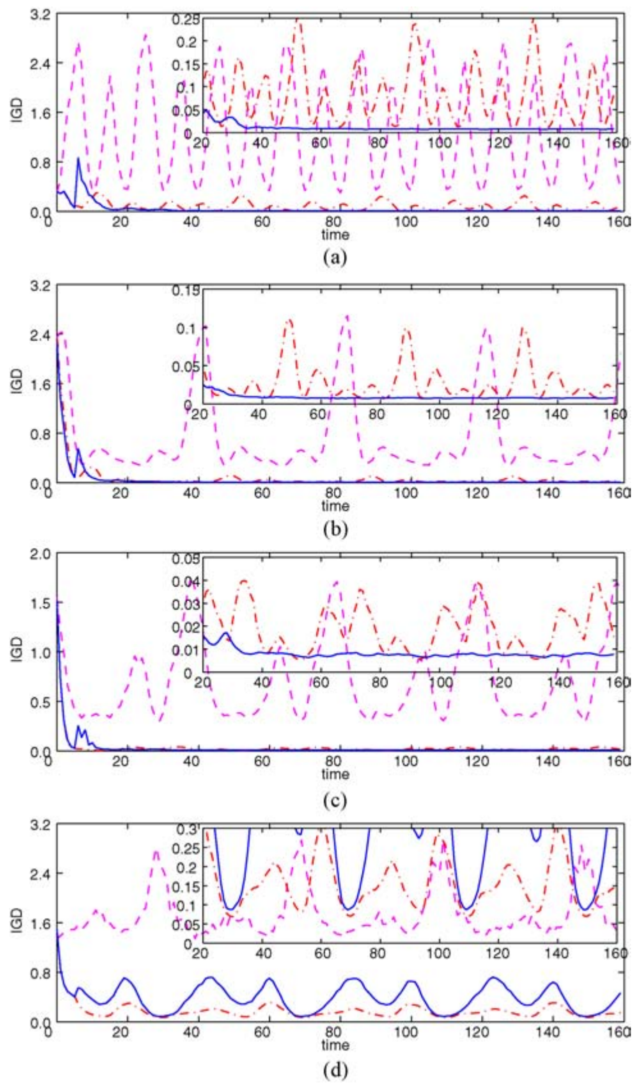

Fig. 8. Average IGD values over 20 runs versus time for three strategies on (a) F5, (b) F6, (c) F7, and (d) F8. The dashed lines are with RIS, dash-dot lines are with FPS, and solid lines are with PPS.

plots the average IGD metric values over 20 independent runs versus the time. The initial and final populations obtained by RIS, FPS, and PPS with the lowest MIGD values at $t = 145, 147, 149, 151, 153$ on F5 and F6 are plotted in Figs. 9 and 10, respectively.

The experimental results are similar to those in the previous section. It is clear from Table II that at the early stages, FPS dominates both RIS and PPS on all the four test instances for most of the MIGD metric values, while at later stages, PPS shows much better performances than RIS and FPS on F5–F7, and FPS outperforms PPS on F8, which are confirmed by the t-test. Take the mean of MIGD values as an example, the mean values of PPS are 12.94%, 31.50%, 42.19% and 240.45% of those of FPS, and 0.79%, 1.21%, 1.09% and 21.01% of those of RIS on F5–F8, respectively.

From Fig. 8, we can see that the performance curves of PPS oscillate at the beginning of the runs for all the four test problems, and they become stable when $20 < t \leq 160$ especially on F5, F6, and F7. It is clear that the performance curves of PPS are below those of FPS in most of times when $20 < t \leq 160$. The initial populations in Fig. 9 show that RIS could not find good initial solutions, FPS could find some good initial solutions but they are far away from the PFs, and only PPS could initialize solutions close to the PFs on F6 and F7. The final populations in Fig. 10 show that although FPS could find good approximations on F6, the final populations obtained by FPS are worse than those found by PPS.

The final results could be explained as the same in the previous section. F5–F8 are more difficult than F1–F4 because of the nonlinear correlation between decision variables. Since the same overhead is assigned to all the algorithms on all the test problems, the statistical results on F5–F8 are worse than those on F1–F4. However, the influence of the two groups of test problems to PPS is less than the influence to FPS.

There might be some reasons why PPS outperforms FPS in the later stages for most of the test problems: 1) in FPS, only a small proportion of the new population is predicted and most of solutions are inherited from the previous population, while in PPS, the whole new population is predicted: when the PS moves in the search space, the initial populations obtained by FPS might be worse than those obtained by PPS; and 2) in PPS, the time series is for the center points of the populations, which might be more statistically stable than the isolated points in FPS, and therefore, the quality of the predicted population of PPS is higher than that of FPS.

VI. More Discussions

In this section, we investigate the performance of the proposed strategy in more angles.

A. Influence of the MOEA Cost and Severity of Changes

The cost assigned to each time step and the severity of changes, n_T, are two important parameters that may influence the algorithm performance. For this reason, we test FPS and PPS on F6 with $cost = 1,000, 2,000, 3,000, 4,000$ function evaluations, and $n_T = 5, 10, 15, 20$, respectively. The other parameters are the same as in Section V.

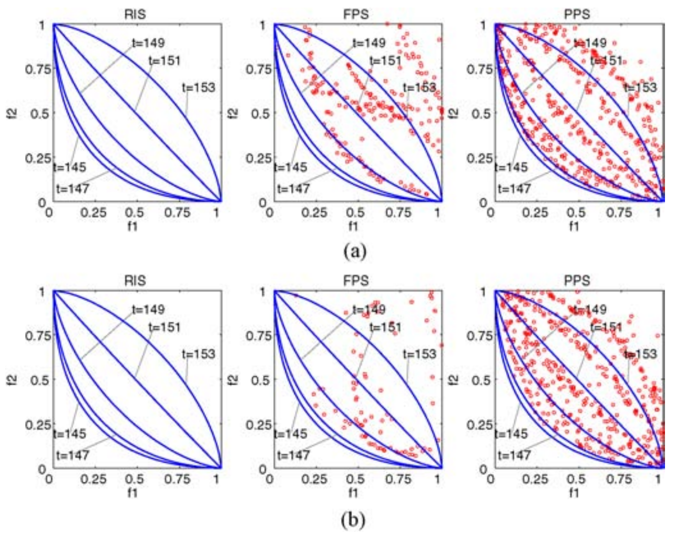

Fig. 9. Initial populations obtained by the three strategies at t = 145, 147, 149, 151, and 153 with lowest MIGD values among 20 runs on (a) F6 and (b) F7.

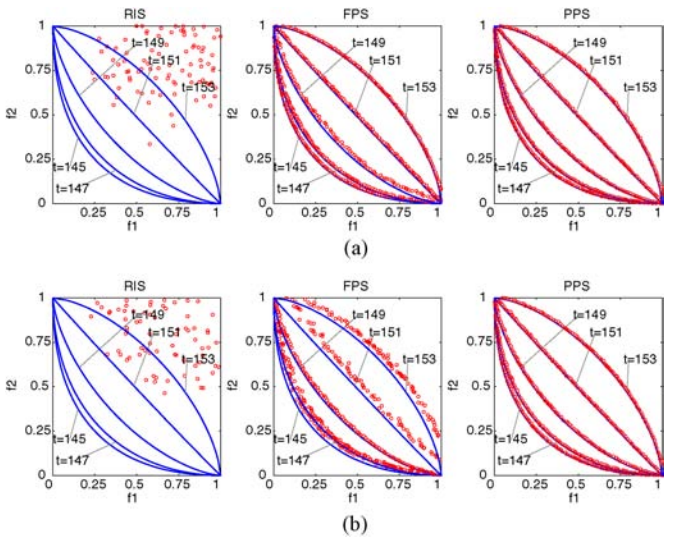

Fig. 10. Final populations obtained by the three strategies at t = 145, 147, 149, 151, and 153 with lowest MIGD values among 20 runs on (a) F6 and (b) F7.

Fig. 11 shows the box plot of the MIGD values over 20 runs for FPS and PPS with $20 < t \leq 160$ and different settings of $cost$ and n_T. It is clear that: 1) as the computational cost increases, the performances of both FPS and PPS increase, and 2) as n_T increases (severity becomes smaller), the performances of both FPS and PPS increase as well. These conclusions are obvious. Let us consider the strategies with low cost, i.e., $cost = 1000$. PPS works unstable and performs worse or similar to FPS. The reason might be that with the given cost (ten generations), RM-MEDA is not able to find good approximations to the PSs. The center points of the populations are not correct, and this may mislead the prediction directions. When $cost \geq 2000$ and $n_T \geq 10$, PPS performs much better than FPS and the statistical results are more stable than those of FPS.

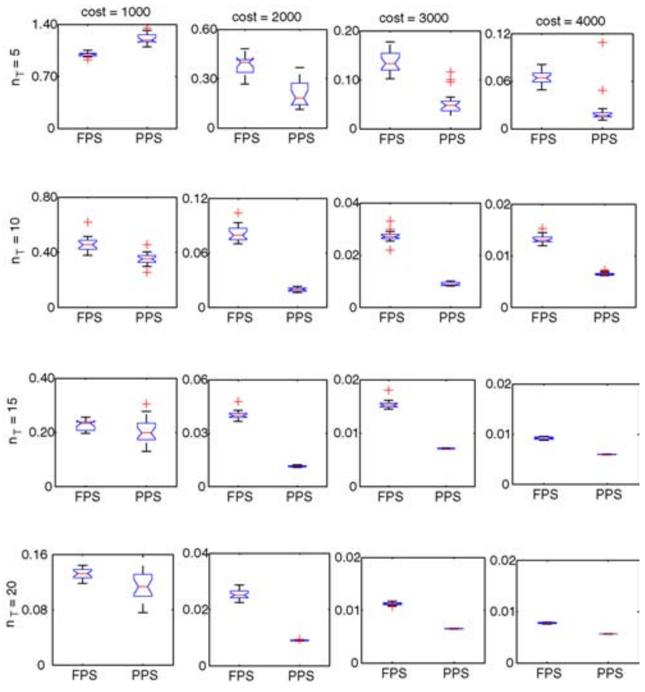

Fig. 11. Influence of $cost$ and n_T to FPS and PPS on F6. The MIGD values are with $20 < t \leq 160$.

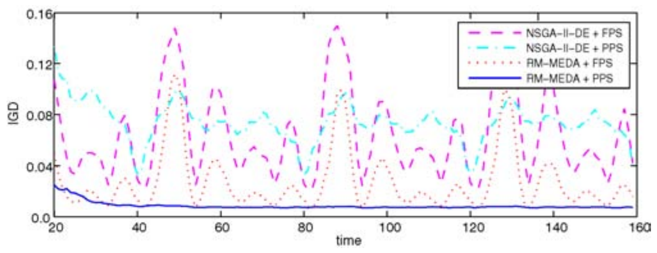

Fig. 12. Average IGD values over 20 runs versus time for FPS and PPS with RM-MEDA and NSGA-II-DE on F6 with $20 < t \leq 160$.

B. Influence of Multiobjective Optimizers

The multiobjective optimizers also play an important role in tackling DMOPs. In this section, we empirically study the performance of RM-MEDA [63] and NSGA-II-DE [68], a modified version NSGA-II [67], by replacing the SBX crossover operator by a DE operator. The only difference between RM-MEDA and NSGA-II-DE is that they use different offspring reproduction operators. The two MOEAs are combined with FPS and PPS, and the four combinations are applied to F6.

In NSGA-II-DE, the mutation probability is $P_m = 0.05$, the parameter in polynomial mutation is $\eta_m = 20$, the DE parameters are $CR = 1.0$ and $F = 0.5$. The other algorithm and problem parameters are the same as in Section V.

Fig. 12 plots the average IGD values versus time for FPS and PPS with optimizers RM-MEDA and NSGA-II-DE on F6.

On F6, both NSGA-II-DE + PPS and RM-MEDA + PPS are more stable than NSGA-II-DE + FPS and RM-MEDA + FPS, respectively; furthermore, RM-MEDA + PPS is much better than NSGA-II-DE + PPS.

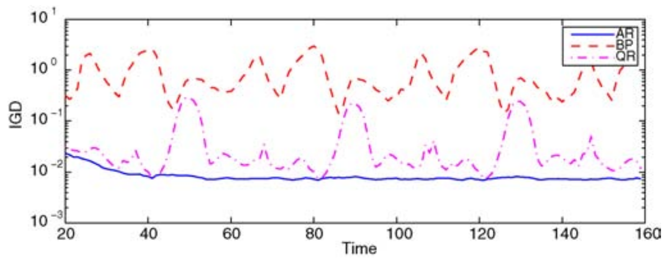

Fig. 13. Average IGD values over 20 runs versus time for PPS with AR, QR, and BP methods on F6 with $20 < t \leq 160$.

The statistical results indicate that RM-MEDA is better than NSGA-II-DE. The reason might be that RM-MEDA utilizes the problem specific knowledge while NSGA-II-DE does not. Therefore, RM-MEDA could recover from environmental changes faster than NSGA-II-DE.

C. Influence of Time Series Predictor

This section considers the influence of different time series prediction methods. Since the AR model used in our method is actually a linear prediction model, we choose a quadratic regression (QR) model and a backpropagation (BP) network for comparison. For the ith element in the PS center, the QR model is defined as

$$\begin{aligned}\overline{x}_i^{t+1} &= \lambda_{0,i} + \lambda_{1,i}\overline{x}_i^t + \lambda_{2,i}\overline{x}_i^{t-1}\\ &\quad + \lambda_{3,i}\overline{x}_i^t\overline{x}_i^{t-1} + \lambda_{4,i}(\overline{x}_i^t)^2 + \lambda_{5,i}(\overline{x}_i^{t-1})^2\\ &\quad + \varepsilon_i^c\end{aligned} \tag{8}$$

where $\lambda_{j,i}$, $j = 0, 1, \cdots, 5$ are parameters of the QR model, $\varepsilon_i^c \sim N(0, \sigma_i^c)$ is white noise with variance σ_i^c, $i = 1, 2, \ldots, n$. The BP model is defined as

$$\overline{x}_i^{t+1} = BP(\overline{x}_i^t, \overline{x}_i^{t-1}) + \varepsilon_i^c \tag{9}$$

where $\varepsilon_i^c \sim N(0, \sigma_i^c)$ is white noise with variance σ_i^c, $i = 1, 2, \ldots, n$.

We use the above two equations to replace the AR model in (4) and the variance is estimated as the same in (5). The history center points are used to estimate the model parameters as in the case of AR model. In particular, the parameters of the QR model are obtained by using the least square error method, and for the BP model, the numbers of input and hidden neurons are 2 and 5, respectively. The learning rates are $\alpha = 0.05$ and $\eta = 0.01$, and the training process stops after 1000 iterations based on empirical studies.[1] The three prediction methods are applied to F6.

Fig. 13 plots the average IGD values versus time for AR, QR, and BP models on F6. For clarity, we use log scale on the vertical coordinate. It is clear that the AR model obtains the best performance, while the BP model performs worst among them. Actually, the result obtained by the QR model are similar to that of the AR model. However, the QR model is not stable on several points. The BP model fails to predict the PS centers. The experimental results indicate that a simple linear model may be more suitable than nonlinear models for online prediction in dynamic environments considered in this work.

[1]The source code is from http://www.ip-atlas.com/pub/nap/nn-src/bpn.txt.

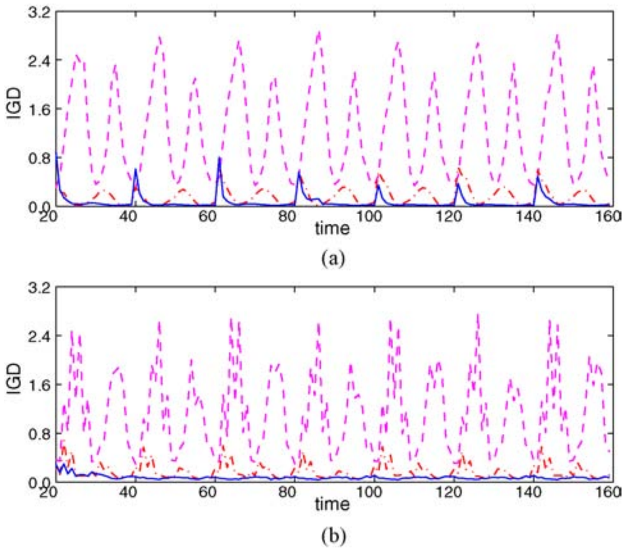

Fig. 14. Average IGD values over 20 runs versus time for RIS, FPS, and PPS on (a) F9, and (b) F10. The IGD values are with $20 < t \leq 160$. The dashed lines are with RIS, dash-dot lines are with FPS, and solid lines are with PPS.

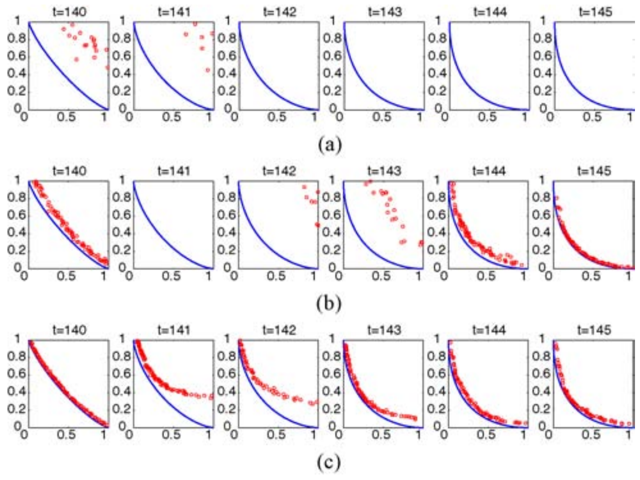

Fig. 15. Final populations obtained by (a) RIS, (b) FPS, and (c) PPS at t = 140, 141, 142, 143, 144, and 145 with lowest MIGD values among 20 runs on F9. A jump happens between t = 140 and t = 141.

The reasons might be: 1) nonlinear models usually need more parameters, which are problem dependent and should be tuned carefully; 2) it is usually easier to train a simple linear model than a nonlinear model online; and 3) simple linear models might be more stable than nonlinear models in some cases.

The comparative study reported in this section is very preliminary. How to choose a proper prediction method in the PPS framework is worth further investigation.

D. Performance on Complicated Problems

In the problems discussed in the previous sections, the environment changes smoothly, and the geometric shapes of two consecutive Pareto sets are similar to each other in a sense. It is interesting to study the performance of the strategies on more complicated problems.

Two test instances, F9 and F10, are designed for this purpose. The details of the two problems can be found in

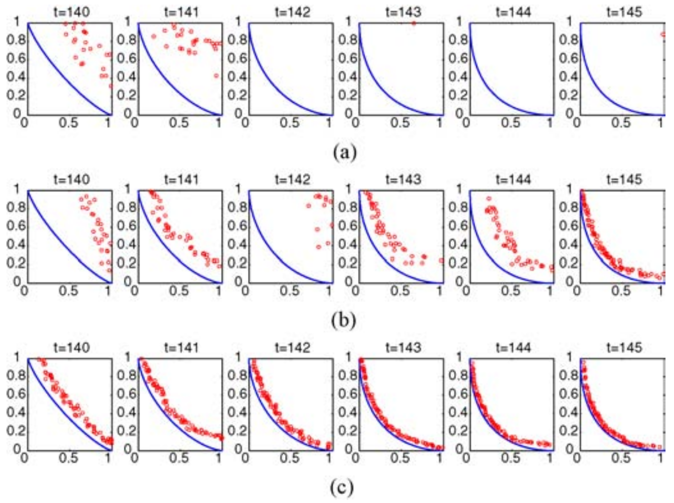

Fig. 16. Final populations obtained by (a) RIS, (b) FPS, and (c) PPS, at t = 140, 141, 142, 143, 144, and 145 with lowest MIGD values among 20 runs on F10.

Table III, and their characteristics are as follows: 1) in F9, the environment changes smoothly in most of the cases, and occasionally, the Pareto set jumps from one area to another area; and 2) in F10, the geometric shapes of two consecutive PFs are totally different from each other.

The experimental parameters are the same as in Section V.

First, we consider F9. There are eight jumps in the time interval [0, 160]. Fig. 14(a) plots the average IGD values versus time. It shows that the performances of RIS, FPS, and PPS are all influenced by the jumps. However, PPS can recover from the jumps and increase the performance, while RIS and FPS could not although the jumps influence PPS more than FPS. Take t = 140, for example; the environment jumps between time t = 140 and t = 141. As shown in Fig. 15, FPS and PPS could approximate the PF at t = 140 while RIS could not. After the change, RIS could not recover from the change, but both FPS and PPS could. However FPS needs more time than PPS.

Second, we consider F10. Both the runtime performance in Fig. 14(b) and the obtained final populations in Fig. 16 show that PPS dominates both RIS and FPS on F10. Although PPS could not approximate the PFs very well all the time, the populations obtained by PPS are much closer to the PFs than those obtained by RIS and FPS. The reason why PPS still works on this problem might be that, in this case, PPS could detect the dissimilarity between two consecutive PSs because the variances from center point prediction and shape estimation are big, and thus the predicted population can cover the PF in a sense.

VII. Conclusions and Future Work

In this paper, we proposed PPS to enhance the performance of MOEAs in dealing with dynamic environments. In PPS, a population was divided into two parts: a center point and a PS manifold. Based on the analysis of the two parts, a method was deduced to predict a whole new population when an environmental change was detected. The main advantages of the proposed PPS strategy over other learning and prediction

TABLE III
TWO NEW TEST INSTANCES WITH COMPLICATED CHARACTERISTICS

Instance	Search Space	Objectives, PS and PF	Remarks
$F9$	$[0, 5]^n$	$f_1(x, t) = \|x_1 - a\|^H + \sum_{i \in I_1} y_i^2$,	PF changes.
		$f_2(x, t) = \|x_1 - a - 1\|^H + \sum_{i \in I_2} y_i^2$,	PS changes.
		$y_i = x_i - b - 1 + \|x_1 - a\|^{H + \frac{i}{n}}$, $H = 1.25 + 0.75 \sin(\pi \frac{t}{n_T})$,	Two objectives.
		$a = 2\cos((\frac{t}{n_T} - \lfloor \frac{t}{n_T} \rfloor)\pi) + 2$, $b = 2\sin(2(\frac{t}{n_T} - \lfloor \frac{t}{n_T} \rfloor)\pi) + 2$,	
		$I_1 = \{i \| 1 \le i \le n, i \text{ is odd}\}$, $I_2 = \{i \| 1 \le i \le n, i \text{ is even}\}$.	
		PS(t): $a \le x_1 \le a + 1$, $x_i = b + 1 - \|x_1 - a\|^{H + \frac{i}{n}}$, for $i = 2, \ldots, n$.	
		PF(t): $f_1 = s^H$, $f_2 = (1 - s)^H$, $0 \le s \le 1$.	
$F10$	$[0, 5]^n$	$f_1(x, t) = \|x_1 - a\|^H + \sum_{i \in I_1} y_i^2$,	PF changes.
		$f_2(x, t) = \|x_1 - a - 1\|^H + \sum_{i \in I_2} y_i^2$,	PS changes.
		$y_i = \begin{cases} x_i - b - \|x_1 - a\|^{H + \frac{i}{n}} & \text{if } t \text{ is odd} \\ x_i - b - 1 + \|x_1 - a\|^{H + \frac{i}{n}} & \text{otherwise} \end{cases}$,	Two objectives.
		$a = 2\cos(\pi \frac{t}{n_T}) + 2$, $b = 2\sin(2\pi \frac{t}{n_T}) + 2$,	
		$H = 1.25 + 0.75 \sin(\pi \frac{t}{n_T})$,	
		$I_1 = \{i \| 1 \le i \le n, i \text{ is odd}\}$, $I_2 = \{i \| 1 \le i \le n, i \text{ is even}\}$.	
		PS(t): $a \le x_1 \le a + 1$, $x_i = b + 1 - \|x_1 - a\|^{H + \frac{i}{n}}$, for $i = 2, \ldots, n$.	
		PF(t): $f_1 = s^H$, $f_2 = (1 - s)^H$, $0 \le s \le 1$.	

$x = (x_1, \ldots, x_n)$ is decision vector, $t = 0, 1, \cdots$ indexes discrete time, and n_T is a problem parameter.

strategies were as follows.

1) PPS maintained only one time series of the centers and recorded the last two PS manifolds for prediction. Thus, both the time complexity and space complexity were small.
2) PPS not only detected environmental changes but also studied the similarity between two consecutive PSs, and used this information to initialize populations.

PPS was compared with a random initialization strategy and a hybrid strategy on a variety of DMOPs with linear or nonlinear correlation between decision variables. The statistical results indicated that PPS was very promising for dealing with dynamic environments. We also studied the influences of some algorithm and problem parameters and the influences of different MOEA optimizers. Since RM-MEDA utilized the regularity property of MOPs, it performed better than NSGA-II-DE in the DMOEA framework. We then investigated the influence of different time series predictors and found that a linear model worked better than nonlinear models in PPS. Finally, we applied PPS to two DMOPs with complicated properties, and the results shown that PPS could tackle these problems in a sense.

The work presented in this paper is preliminary, and there are some possible directions for future work.

1) A potential weakness with PPS is that its prediction quality is not high at the early evolutionary search stages due to limited history information. Hybridization with other strategies might be a possible way to improve its performance at the beginning stages.
2) The prediction method plays an key role in PPS. Combing PPS with other prediction methods, such as Kalman-based prediction strategy [69], and studying their influences are worth further investigating.
3) The similarity measurement of PS manifolds is an important issue in PPS. How to estimate the next PS manifold with the reordered information more efficiently is still an open question.
4) It is also interesting to combine PPS with other algorithms like MOEA/D [70] or an MOEA with unfixed population or archive.
5) It is worth testing PPS on more problems with different types of changes, with or without constraints.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the six anonymous reviewers for their very helpful and constructive comments.

REFERENCES

[1] K. Deb, U. V. Rao, and S. Karthik, "Dynamic multiobjective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling," in *Proc. EMO*, LNCS 4403, 2007, pp. 803–817.

[2] M. B. Abello, L. T. Bui, and Z. Michalewicz, "An adaptive approach for solving dynamic scheduling with time-varying number of tasks: Part II," in *Proc. IEEE CEC*, Jun. 2011, pp. 1711–1718.

[3] M. B. Abello, L. T. Bui, and Z. Michalewicz, "An adaptive approach for solving dynamic scheduling with time-varying number of tasks: Part I," in *Proc. IEEE CEC*, Jun. 2011, pp. 1703–1710.

[4] A. K. Hutzschenreuter, P. A. Bosman, and H. Poutré, "Evolutionary multiobjective optimization for dynamic hospital resource management," in *Proc. EMO*, LNCS 5467, 2009, pp. 320–334.

[5] J. Tang, S. Alam, C. Lokan, and H. A. Abbass, "A multiobjective evolutionary method for dynamic airspace re-sectorization using sectors clipping and similarities," in *Proc. IEEE CEC*, Jun. 2012, pp. 3565–3572.

[6] A. Isaacs, V. Puttige, T. Ray, W. Smith, and S. Anavatti, "Development of a memetic algorithm for dynamic multiobjective optimization and its applications for online neural network modeling of UAVs," in *Proc. IJCNN*, 2008, pp. 548–554.

[7] L. T. Bui and Z. Michalewicz, "An evolutionary multiobjective approach for dynamic mission planning," in *Proc. IEEE CEC*, July 2010, pp. 1–8.

[8] A. R. da Cruz, R. T. N. Cardoso, and R. H. C. Takahashi, "Multiobjective dynamic optimization of vaccination campaigns using convex quadratic approximation local search," in *Proc. EMO*, LNCS 6576, 2011, pp. 404–417.

[9] P. P.-Y. Wu, D. Campbell, and T. Merz, "Multiobjective 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.

[10] P. D. Barba, "Dynamic multiobjective optimization: A way to the shape design with transient magnetic fields," *IEEE Trans. Magn.*, vol. 44, no. 6, pp. 962–965, Jun. 2008.

[11] F. V. C. Martins, E. G. Carrano, E. F. Wanner, R. H. C. Takahashi, and G. R. Mateus, "A dynamic multiobjective hybrid approach for designing wireless sensor networks," in *Proc. IEEE CEC*, May 2009, pp. 1145–1152.

[12] B. Andrés Toro, J. M. Girón Sierra, P. Fernández Blanco, J. A. López Orozco, and E. Besada Portas, "Multiobjective optimization and multivariable control of the beer fermentation process with the use of evolutionary algorithms," *J. Zhejiang Univ. (Sci.)*, vol. 5, no. 4, pp. 378–389, May 2004.

[13] 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.

[14] K. Kim, R. B. McKay, and B.-R. Moon, "Multiobjective evolutionary algorithms for dynamic social network clustering," in *Proc. GECCO*, 2010, pp. 1179–1186.

[15] B. S. Rabil, R. Sabourin, and E. Granger, "Watermarking stack of grayscale face images as a dynamic multiobjective optimization problem," in *Proc. MDA*, 2011, pp. 63–77.

[16] M. Linnala, E. Madetoja, H. Ruotsalainen, and J. Hämäläinen, "Bilevel optimization for a dynamic multiobjective problem," *Eng. Optimiz.*, vol. 44, no. 2, pp. 195–207, 2012.

[17] S. Zeng, S. Chen, J. Zhao, A. Zhou, Z. Li, and H. Jing, "Dynamic constrained multiobjective model for solving constrained optimization problem," in *Proc. IEEE CEC*, Jun. 2011, pp. 2041–2046.

[18] S. Zeng, S. Chen, J. Zhao, A. Zhou, Z. Li, and H. Jing, "Dynamic constrained multiobjective model for solving constrained optimization problem," in *Proc. IEEE CEC*, Jun. 2011, pp. 2649–2654.

[19] 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.

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

[21] K. Miettinen, *Nonlinear Multiobjective Optimization*. Dordrecht, The Netherlands: Kluwer Academic, 1999.

[22] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, USA: Wiley, 2001.

[23] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Dordrecht, The Netherlands: Kluwer Academic, 2002.

[24] 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.

[25] A. Zhou, "Estimation of distribution algorithms for continuous multiobjective optimization," Ph.D. dissertation, Univ. Essex, 2009.

[26] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multiobjective optimization," in *Proc. EMO*, LNCS 4403, 2007, pp. 832–846.

[27] C. A. Coello Coello, D. A. van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, USA: Kluwer Academic, 2002.

[28] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjectiveevolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[29] M. Greeff and A. P. Engelbrecht, "Solving dynamic multiobjective problems with vector evaluatedparticle swarm optimisation," in *Proc. IEEE CEC*, July 2008, pp. 2922–2929.

[30] C. Liu and Y. Wang, "New evolutionary algorithm for dynamic multiobjective optimization problems," in *Proc. Evol. Comput. Theory Algorithms*, LNCS 4221, 2006, pp. 889–892.

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

[32] I. Hatzakis and D. Wallace, "Dynamic multiobjective optimization with evolutionary algorithms: A forward-looking approach," in *Proc. GECCO*, 2006, pp. 1201–1208.

[33] I. Hatzakis and D. Wallace, "Topology of anticipatory populations for evolutionary dynamic multiobjective optimization," in *Proc. 11th AIAA/ISSMO Multidisciplinary Anal. Optimization Conf.*, 2006, pp. 1944–1953.

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

[35] B. Zheng, "A new dynamic multiobjective optimization evolutionary algorithm," in *Proc. ICNC*, 2007, pp. 565–570.

[36] R. Liu, W. Zhang, L. Jiao, F. Liu, and J. Ma, "A sphere-dominance based preference immune-inspired algorithm for dynamic multiobjective optimization," in *Proc. GECCO*, 2011, pp. 423–430.

[37] C. Liu and Y. Wang, "Multiobjective evolutionary algorithm for dynamic nonlinear constrained optimization problems," *J. Syst. Eng. Electron.*, vol. 20, no. 1, pp. 204–210, 2009.

[38] J. Wei and M. Zhang, "Simplex model based evolutionary algorithm for dynamic multiobjective optimization," in *Proc. Adv. Artif. Intell.*, LNCS 7106, 2011, pp. 372–381.

[39] Y. Ma, R. Liu, and R. Shang, "A hybrid dynamic multiobjective immune optimization algorithm using prediction strategy and improved differential evolution crossover operator," in *Proc. Neural Inform. Process.*, LNCS 7063, 2011, pp. 435–444.

[40] J. Wei and Y. Wang, "Hyper rectangle search based particle swarm algorithm for dynamic constrained multiobjective optimization problems," in *Proc. IEEE CEC*, Jun. 2012, pp. 259–266.

[41] M. Yang, L. Kang, and J. Guan, "Multialgorithm co-evolution strategy for dynamic multiobjective TSP," in *Proc. IEEE CEC*, July 2008, pp. 466–471.

[42] V. Aragón, S. Esquivel, and C. A. Coello Coello, "Evolutionary multiobjective optimization in non-stationary environments," *J. Comput. Sci. Technol.*, vol. 5, no. 3, pp. 133–143, 2005.

[43] C. R. B. Azevedo and A. F. R. Araujo, "Generalized immigration schemes for dynamic evolutionary multiobjective optimization," in *Proc. IEEE CEC*, Jun. 2011, pp. 2033–2040.

[44] Y. Wang and B. Li, "Investigation of memory-based multiobjective optimization evolutionary algorithm in dynamic environment," in *Proc. IEEE CEC*, May 2009, pp. 630–637.

[45] A. D. Manriquez, G. T. Pulido, and J. G. R. Torres, "Handling dynamic multiobjective problems with particle swarm optimization," in *Proc. ICAART*, 2010, pp. 337–342.

[46] E. Vinek, P. P. Beran, and E. Schikuta, "A dynamic multiobjective optimization framework for selecting distributed deployments in a heterogeneous environment," *Procedia Comput. Sci.*, vol. 4, pp. 166–175, Jun. 2011.

[47] M. Helbig and A. P. Engelbrecht, "Analyses of guide update approaches for vector evaluated particle swarm optimisation on dynamic multiobjective optimisation problems," in *Proc. IEEE CEC*, Jun. 2012, pp. 2621–2628.

[48] M. Cámara, J. Ortega, and F. de Toro, "A single front genetic algorithm for parallel multiobjective optimization in dynamic environments," *Neurocomputing*, vol. 72, nos. 16–18, pp. 3570–3579, Oct. 2009.

[49] M. Cámara, J. Ortega, and F. de Toro, "Approaching dynamic multiobjective optimization problems by using parallel evolutionary algorithms," in *Proc. Advances Multi-Objective Nature Inspired Comput.*, vol. 272. 2010, pp. 63–86.

[50] Y. Wang and B. Li, "Multistrategy ensemble evolutionary algorithm for dynamic multiobjective optimization," *Memetic Comput.*, vol. 2, no. 1, pp. 3–24, Mar. 2010.

[51] M. Helbig and A. P. Engelbrecht, "Archive management for dynamic multiobjective optimisation problems using vector evaluated particle swarm optimisation," in *Proc. IEEE CEC*, Jun. 2011, pp. 2047–2054.

[52] R. Shang, L. Jiao, M. Gong, and B. Lu, "Clonal selection algorithm for dynamic multiobjective optimization," in *Proc. Comput. Intell. Security*, LNCS 3801, July 2005, pp. 846–851.

[53] A. K. M. Talukder and M. Kirley, "Inventory management and the impact of anticipation in evolutionary stochastic online dynamic optimization," in *Proc. IEEE CEC*, July 2008, pp. 2270–2277.

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

[55] X. Li, J. Branke, and M. Kirley, "On performance metrics and particle swarm methods for dynamic multiobjective optimization problems," in *Proc. IEEE CEC*, Sep. 2007, pp. 576–583.

[56] S. Zeng, G. Chen, L. Zhang, H. Shi, H. de Garis, L. Ding, and L. Kang, "A dynamic multiobjective evolutionary algorithm based on an orthogonal design," in *Proc. IEEE CEC*, July 2006, pp. 2588–2595.

[57] Y. Wang and C. Dang, "An evolutionary algorithm for dynamic multiobjective optimization," *Appl. Math. Comput.*, vol. 205, no. 1, pp. 6–18, 2008.

[58] X. Yu, Y. Jin, K. Tang, and X. Yao, "Robust optimization over time: A new perspective on dynamic optimization problems," in *Proc. IEEE CEC*, July 2010, pp. 3998–4003.

[59] 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.
[60] C. Hillermeier, *Nonlinear Multiobjective Optimization—A Generalized Homotopy Approach*. Birkhäuser, 2001.
[61] C. Chatfield, *The Analysis of Time Series: An Introduction*. Boca Raton, FL, USA: CRC Press, 2004.
[62] Y. Jin and B. Sendhoff, "Constructing dynamic optimization test problems using the multiobjective optimization concept," in *Proc. Appl. Evol. Comput.*, LNCS 3005, 2004, pp. 525–536.
[63] 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.
[64] M. Cámara, J. Ortega, and F. de Toro, "Performance measures for dynamic multiobjective optimization," in *Bio-Inspired Systems: Computational and Ambient Intelligence*, LNCS 5517, Berlin, Germany: Springer, 2009, pp. 760–767.
[65] E. Tantar, A.-A. Tantar, and P. Bouvry, "On dynamic multiobjective optimization, classification and performance measures," in *Proc. IEEE CEC*, Jun. 2011, pp. 2759–2766.
[66] M. R. Sierra and C. A. Coello Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance," in *Proc. EMO*, LNCS 3410, 2005, pp. 505–519.
[67] 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.
[68] H. Li and Q. Zhang, "Comparison between NSGA-II and MOEA/D on a set of multiobjective optimization problems with complicated Pareto sets," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.
[69] C. Rossi, M. Abderrahim, and J. C. Díaz, "Tracking moving optima using Kalman-based predictions," *Evol. Comput.*, vol. 16, no. 1, pp. 1–30, 2008.
[70] 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.

Aimin Zhou (S'08–M'10) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 2001 and 2003, respectively, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2009.

He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include evolutionary computation, machine learning, image processing, and their applications.

Yaochu Jin (M'98–SM'02) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr. Ing. degree from Ruhr-University Bochum, Bochum, Germany, in 2001.

He is currently a Professor and the Chair in computational intelligence with the Department of Computing, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. Before joining the University of Surrey, he was a Principal Scientist and Group Leader with the Honda Research Institute Europe, Offenbach am Main, Germany. He has (co-)edited five books and three conference proceedings, authored a monograph, and (co-)authored over 150 peer-reviewed journal and conference papers. He has been granted eight U.S., EU, and Japan patents. His current research is funded by EU FP7, U.K. EPSRC, and industry corporations, including Intellas U.K., Santander, Aero Optimal, Bosch U.K. and Honda. His current research interests include computational intelligence, computational neuroscience, and computational systems biology, with applications to complex engineering optimization, bioengineering, swarm robotics, and autonomous systems.

Dr. Jin is an Associate Editor of *BioSystems*, the *International Journal of Fuzzy Systems*, *Soft Computing*, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B: CYBERNETICS, the IEEE TRANSACTIONS ON NANOBIOSCIENCE, and the *IEEE Computational Intelligence Magazine*. He is a Distinguished Lecturer from 2013 to 2015 and an Elected AdCom Member from 2012 to 2014 of the IEEE Computational Intelligence Society. He was a recipient of the Best Paper Award at the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He has delivered over ten invited keynote speeches on morphogenetic robotics, developmental neural systems, modeling, analysis, synthesis of gene regulatory networks, evolutionary optimization, and multiobjective learning at international conferences.

Qingfu Zhang (M'01–SM'06) received the B.Sc. degree in mathematics from Shanxi University, Shanxi, China, in 1984, and the M.Sc. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xian, China, in 1991 and 1994, respectively.

He is currently a Professor with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K. From 1994 to 2000, he was with the National Laboratory of Parallel Processing and Computing, National University of Defense Science and Technology, Changsha, China, Hong Kong Polytechnic University, Hung Hom, Hong Kong, the German National Research Center for Information Technology (now Fraunhofer-Gesellschaft), Darmstadt, Germany, and University of Manchester Institute of Science and Technology, Manchester, U.K. He holds two patents and is the author of many research publications. His current research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE tRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART B: CYBERNETICS. He is also an Editorial Board Member of three other international journals. MOEA/D, a multiobjective optimization algorithm developed in his group, won the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation in 2009, and he was awarded the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award.