Noname manuscript No.
(will be inserted by the editor)

A Directed Search Strategy for Evolutionary Dynamic Multiobjective Optimization

Yan Wu · Yaochu Jin · Xiaoxiong Liu

the date of receipt and acceptance should be inserted later

Abstract Many real-world multiobjective optimization problems are dynamic, requiring an optimization algorithm that is able to continuously track the moving Pareto front over time. In this paper, we propose a directed search strategy (DSS) consisting of two mechanisms for improving the performance of multiobjective evolutionary algorithms in changing environments. The first mechanism reinitializes the population based on the predicted moving direction as well as the directions that are orthogonal to the moving direction of the Pareto set, when a change is detected. The second mechanism aims to accelerate the convergence by generating solutions in predicted regions of the Pareto set according to the moving direction of the non-dominated solutions between two consecutive generations. The two mechanisms, when combined together, are able to achieve a good balance between exploration and exploitation for evolutionary algorithms to solve dynamic multiobjective optimization problems. We compare DSS with two existing prediction strategies on a variety of test instances having different changing dynamics. Empirical results show that DSS is powerful for evolutionary algorithms to deal with dynamic multiobjective optimization problems.

Yan Wu
School of Mathematics and Statistics, Xidian University, Xian 710071, China
E-mail: wuyan@mail.xidian.end.cn

Yaochu Jin
Department of Computing, University of Surrey, Guildford, GU2 7XH, U.K. and College of Information Sciences and Technology, Donghua University, Shanghai 201620, China
E-mail: yaochu.jin@surrey.ac.uk

Xiaoxiong Liu
School of Automation, Northwestern Polytechnical University, Xian 710072, China

Keywords dynamic multiobjective optimization · evolutionary algorithm · prediction · local search

1 Introduction

In many real-world optimization problems, the fitness functions to be optimized may change over time, which are known as dynamic optimization problems (DOPs). Like stationary optimization problems, DOPs can also be categorized into dynamic single-objective optimization problems (DSOPs) and dynamic multi-objective optimization problems (DMOPs), see detailed discussions in Farina et al (2004); Jin and Branke (2005); Nguyen et al (2012). In recent years, solving dynamic multi-objective optimization problems using evolutionary algorithms (EAs) has attracted increasing attention (Farina et al (2004); Jin and Branke (2005); Nguyen et al (2012)) due to its practical significance in a wide range of applications, such as scheduling (Abello et al (2011a,b); Deb et al (2007)), control (Zhang (2008)), airspace sectorization (Tang et al (2012)), vehicle motion planning (Wu et al (2011)) and wireless network design (Martins et al (2009)), although many challenges remain (Nguyen et al (2012)).

In this paper, we focus on the following class of DMOPs:

$$\begin{aligned} &\min f(x,t) = (f_1(x,t), f_2(x,t), \cdots, f_m(x,t)), \\ &\text{subject to } x \in [L,U], \end{aligned} \tag{1}$$

where $t = 0, 1, 2, \cdots \in T$ is the time instant, $x = (x_1, \cdots, x_n)$ is the decision vector.
$[L,U] = \{x = (x_1, \cdots, x_n) | l_i \leq x_i \leq u_i, i = 1, 2, \cdots, n\}$ defines the decision space, where $L = (l_1, \cdots, l_n)$, $U = (u_1, \cdots, u_n)$ are

the lower and upper bounds, respectively. The objective vector consists of m time varying objective functions $f_i(x,t), (i = 1, 2, \cdots, m)$.

For DMOPs defined in (1), the Pareto front (PF) and/or Pareto set (PS) may change over time. Typically, dynamic multiobjective optimization evolutionary algorithms (DMOEAs) aim to track moving PF (PS) as closely as possible when environment changes.

However, EAs usually lose the ability to find a new optimum once the population has converged. A straightforward idea to address this problem is to increase the population diversity so that the population will not fully converge even if the current optimum has been found. Based on this idea, many approaches have been proposed to adapt EAs to solving DMOPs. Hyper mutation methods (Deb et al (2007); Zhou et al (2007); Zheng (2007); Liu et al (2011)) promote diversity by increasing the mutation rate drastically after a change is detected. Random immigrants and other immigrants methods (Deb et al (2007); Aragon et al (2005); Azevedo and Araujo (2011)) maintain diversity by inserting new immigrants throughout the evolutionary optimization. There are also other methods to increase diversity, such as employing multiple populations and parallel computing (Camara et al (2009, 2010)) and applying different crossover and mutation operator (Yang et al (2008); Jin and Sendhoff (2004)) after a change.

In addition to enhancing population diversity, accelerating convergence is another important issue for dynamic MOEAs to trace the moving PF(PS), as the time for relocating the changed optimum is typically rather short. Unfortunately, most methods for increasing population diversity are likely to disrupt convergence, e.g., the random initialization method (Deb et al (2007); Zhang (2008); Greeff and Engelbrecht (2008); Liu and Wang (2006, 2009)). To alleviate this problem, ideas of using history information about previous optimums have been proposed to speed up convergence, such as the memory based methods (Zhang (2008); Hatzakis and Wallace (2006b,a); Wang and Li (2009); Manriquez et al (2010); Vinek et al (2011); Helbig and Engelbrecht (2012)), among others. However, memory based approaches are most effective only when the changes are periodic.

History information of the previous optimums can be used to predict their future behavior, more or less, so long as the changes in the fitness functions are not fully random. Various prediction strategies have been suggested to demonstrate that they can accelerate convergence of dynamic MOEAs (Zhou et al (2007); Liu et al (2011); Hatzakis and Wallace (2006b,a); Wei and Zhang (2011); Ma et al (2011); Wei and Wang (2012); Zhou et al (2014)). A feed-forward prediction strategy (FPS) was studied in (Hatzakis and Wallace (2006b)), where an autoregressive model was used to estimate the new position of an individual once a change occurs based on the position of a number of previous optimums that are considered to be related to this individual. The difficulty is that it is non-trivial to identify the right previous optimums that are really associated with the current individual. In Zhou et al (2014), a population prediction strategy (PPS) has been investigated, which was shown to be able to enhance the performance of MOEAs in dynamic environments thanks to the predicted initial population after a change. Comparative studies indicated that PPS(Zhou et al (2014)) can converge faster than FPS(Hatzakis and Wallace (2006b)). However, as a linear regression model was used in PPS, the performance of PPS will severely degrade if the changes are irregular or nonlinear.

In solving DOPs, it is essential for EAs to achieve a good balance between maintaining a high degree of population diversity and accelerating convergence. To this end, we design a new prediction based method to take both diversity and convergence into account. The proposed method consists of two mechanisms. The first mechanism reinitializes the population based on the predicted moving direction and the orthogonal directions of the moving Pareto set. The second mechanism guides the search by adding a small number of individuals generated according to the moving direction of the non-dominated solutions between two consecutive generations. Among these mechanisms, generating candidate solutions along the predicted moving direction of the Pareto or non-dominated set is meant for speeding up convergence, whereas the solutions generated along the orthogonal directions of the moving direction can enhance the diversity in the beginning of an environmental change. It is noted that if the change severity is large, the new problem is less relevant to the previous problem and a restart strategy may be better.

The paper is structured as follows. Section 2 presents the proposed algorithm in detail. The test instances and performance metric are presented in Section 3. In Section 4, experimental results are reported. Finally, conclusions are drawn in Section 5.

2 The proposed algorithm

2.1 Directed search strategy (DSS)

The basic idea of the proposed method is to predict the possible regions in decision space where new PS may be located once an environmental change is detected. The main idea here is to take advantage of the predicted information about the moving directions of

non-dominated solutions between two consecutive environments and between two consecutive generations. For the sake of convenience, we term the proposed method directed search strategy (DSS). DSS contains two mechanisms, one used when an environmental change is detected (DSS1 for short), and the other used in each generation (DSS2).

- **DSS1:** Reinitialize the population based on the predicted moving direction of the Pareto set and a local search along the directions orthogonal to the moving direction of PS between two consecutive environments once a change occurs.
- **DSS2:** Guide the search by introducing the promising individuals generated in the regions predicted based on relative positions of the non-dominated solutions between two consecutive generations.

The first mechanism, DSS1, is used to reinitialize the population after an environmental change. After a change occurs, part of the population is reinitialized with individuals generated in the predicted regions where the new PS may be located. In addition, the rest of the individuals are generated by performing a local search along the orthogonal directions of the predicted Pareto set, aiming to enhance the diversity of the population and to deal with inaccurate prediction and sudden irregular changes. By contrast, the second mechanism, DSS2, aims to improve the speed of convergence to PS. At every generation, some individuals generated around the PS region predicted using the history information of the PSs are inserted into the population to speed up the convergence.

2.2 DSS1

2.2.1 Prediction upon an Environmental Change

The prediction in this work plays a similar role as other methods (Hatzakis and Wallace (2006b); Zhou et al (2014)), which aims to make a good guess of the location of the new optimum so that solutions close to the new PS can be generated and the new PS can be obtained more quickly than random initialization. The main assumption in predicting the location of the new PS is that the moving direction of PS at time $t+1$ can be estimated based on the changes of the non-dominated solutions in $t-1$ and t. The prediction we used here is very coarse, which however, offers the benefit of a more explorative search than the finer prediction in FPS and PPS, as illustrated in Fig. 1.

Let C^t be the centroid of PS^t, PS^t is the non-dominated solutions obtained by the algorithm at the end of time step t, then C^t can be calculated as follows:

$$C^t = \frac{1}{|PS^t|} \sum_{x \in PS^t} x, \tag{2}$$

where $|PS^t|$ is the cardinality of PS^t, $x = (x_1, \cdots, x_n) \in PS^t$. Then the moving direction of the non-dominated set at time t, denoted by D_1^t can be estimated as follows:

$$D_1^t = C^t - C^{t-1}. \tag{3}$$

Then the predicted moving direction D_1^t is used to generate individuals according to the following formulas for initializing the population after the environmental change:

$$S^t = \text{sgn}(D_1^t), \tag{4}$$

$$y = x + D_1^t + N(0, d) \times S^t, \tag{5}$$

where x is any individual in the population before the environmental change, $\text{sgn}(\cdot)$ is the sign function, d is the Euclidian distance between centroid C^t and C^{t-1}, $N(0, d)$ denotes a normally distributed one-dimensional random number with mean of zero and standard deviation d. In the above equation, the noise is added for compensating possible errors in the prediction. Noted that S^t results in a bias towards the original moving direction, refer to Fig. 1.

2.2.2 Directed local search

The predicted individuals focus on generating solutions in the predicted new PS area after an environmental change. As the population might have converged before the environmental change, the diversity of the new population may be insufficient if all individuals are reinitialized with individuals based on prediction. For this reason, part of the population will be reinitialized with individuals generated in the regions orthogonal to D_1^t. We will show in the empirical studies that the proposed local search along the direction orthogonal to the moving direction of the non-dominated sets in two consecutive environments works better than a random local search.

There are many directions which are orthogonal to D_1^t and here we choose the orthogonal basis of the null space of D_1^t as the orthogonal directions to D_1^t. Let $D_1^t = (v_1, \cdots, v_n)$, the null space of D_1^t is defined as

$$\text{Null}(D_1^t) = \{w \in R^n : D_1^t w = 0\}.$$

According to singular value decomposition, the orthogonal basis of $\text{Null}(D_1^t)$ are as follows

$$D_i^t = (-\frac{v_i}{v_1}, 0, \cdots, 0, 1, 0, \cdots, 0), i \in \{2, \cdots, n\}.$$

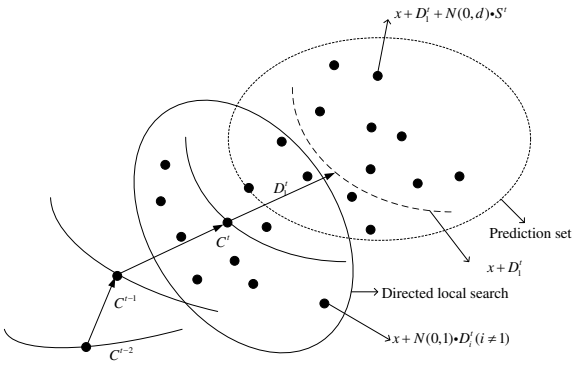


Fig. 1: Population initialization after a change.

Given $D_i^t, i \in \{2, \cdots, n\}$ as calculated above are orthogonal to D_1^t, new candidate solutions can be generated as follows from any individual (x) in the population before the environmental change:

$$y = x + N(0,1) \times D_i^t, i \in \{2, \cdots, n\}, \tag{6}$$

where $N(0,1)$ denotes a normally distributed random number with mean zero and standard deviation 1.

The main steps of DSS1 are summarized in Algorithm 1.

Algorithm 1 The pseudo code of the DSS1

1: Input PS^t, N and the previous centroid C^{t-1};
2: Calculate the centroid C^t of PS^t, and $D_i^t, i = 1, \cdots, n$ and S^t according to (2),(3) and (4);
3: **for** $i = 1, \cdots, r_1 \times N$ **do**
4: Randomly select a solution x from PS^t;
5: Generate a new point y according to (5);
6: Boundary check and put y into P^t;
7: **end for**
8: **for** $i = 1, \cdots, (1 - r_1) \times N$ **do**
9: Randomly select a solution x from PS^t;
10: Randomly select a direction from D_i^t, $i = 2, \cdots, n$;
11: Generate a new solution y according to (6);
12: Boundary check and put y into P^t;
13: **end for**

In line 1 of Algorithm 1, N is population size, set $C^0 = (0, \cdots, 0)$, assuming that the starting location of the centroid is at the origin. In line 3, $0 < r_1 < 1$ is a ratio. Typically, we set $r_1 = 0.5$ in this paper. In lines 6 and 12, P^t is the reinitialized population. Boundary check is to see if the generated solutions are within the given boundary of the decision variable and if not, a repairing operation will be performed as follows:

$$y_i = \begin{cases} x_i & \text{if } l_i \leq x_i \leq u_i \\ 0.5(l_i + x_i) & \text{if } y_i < l_i \\ 0.5(u_i + x_i) & \text{if } y_i > u_i \end{cases} \tag{7}$$

where $i = 1, \cdots, n$.

2.3 DSS2

In tracking a moving Pareto front, the convergence speed is central to the performance since the time period between two environmental changes can be very short. DSS1 aims to improve the tracking performance by generating an initial population close enough to and widely distributed along the Pareto set. Clearly, these one-shot measures based on a very rough prediction are inadequate for reliably and significantly improving the search performance in a changing environment. DSS2 is therefore designed to accelerate the convergence speed of DMOEAs by inserting promising individuals into the current population at every generation. These promising individuals are generated according to the moving direction of centroid of PS^{iter} of the non-dominated solutions between two consecutive generations, where iter is the generation index. Here, the assumption is that the moving direction of centroid between two consecutive generations indicates the direction of true PS to head for. The moving direction D_1^{iter} is estimated as in Equation (3) with t being replaced by iter. Thus, promising individuals are generated as follows:

$$y = x + D_1^{\text{iter}} + N(0, d^{'}) \times S^{\text{iter}}, \tag{8}$$

where $x \in PS^{\text{iter}}$ is a randomly selected non-dominated individual from the current population, $d^{'}$ is the Euclidian distance between centroid C^{iter} and $C^{\text{iter}-1}$, $N(0, d^{'})$ denotes a normally distributed random number with mean zero and standard deviation $d^{'}$, $d^{'}$ is used to control the range of search, and S^{iter} is calculated as (4) to maintain the original increasing or decreasing direction. The main components of DSS2 are summarized in Algorithm 2.

Algorithm 2 The pseudo code of the DSS2

1: Input PS^{iter}, N, and the previous centroid $C^{\text{iter}-1}$;
2: Calculate the centroid C^{iter} of PS^{iter}, and D_1^{iter} and S^{iter}, according to (2) (3) (4);
3: **for** $i = 1, \cdots, r_2 \times N$ **do**
4: Randomly select a solution x from PS^{iter};
5: Generate a new point y according to (8);
6: Boundary check and repair according to (7) if necessary;
7: **end for**
8: Randomly replace the individuals in P^{iter} with generated individuals.

In line 1, N is the population size. Set $C^0 = (0, \cdots, 0)$, assuming that the starting location of centroid is at the origin.

In DSS2, the new individuals are generated according to the moving direction to search the most promis-

ing area of PS. This can be seen as a sort of "gradient" and is expected to provide useful information for the EA to converge to the changed Pareto front more rapidly.

2.4 Overall Framework of the Proposed Algorithm

The proposed directed search strategy (DSS) is incorporated into the most widely used Pareto-based MOEA, the elitist non-dominated sorting algorithm, NSGA-II (Deb et al (2002)). However, the crossover operator used in the original NSGA-II, simulated binary crossover (SBX) is replaced by the DE operator (Li and Zhang (2009); Iorio and Li (2005)). Thus, the main procedure of the proposed algorithm, denoted by NSGA-II/DE+DSS, is described in Algorithm 3:

Algorithm 3 The main procedure of NSGA-II/DE+DSS

1: iter $\leftarrow$ 0 $t \leftarrow$ 0 , Initialize a population P^{iter}, iter is generation, t is time step;
2: If an environmental change is detected, reinitialize the population P^{iter} using DSS1;
3: Apply tournament selection and crossover and mutation operator, and get population Q^{iter};
4: Select the population $P^{\text{iter}+1}$ from $P^{\text{iter}} \bigcup Q^{\text{iter}}$ using non-dominated rank and crowd distance;
5: Adjust $P^{\text{iter}+1}$ using DSS2;
6: If the termination is satisfied, then stop, otherwise iter $\leftarrow$ iter + 1, turn to Step 2.

The DE operator in line 4 of Algorithm 3 is described as follows:

$$y_i = x_i^{k_1} + F \times (x_i^{k_2} + x_i^{k_3}), \tag{9}$$

where F is a control parameter, $F = 0.5$ and k_1, k_2, k_3 are mutually exclusive integers randomly generated within the range $[1, N]$.

3 Test Instances and Performance Indicators

3.1 Test instances

Twelve dynamic multi-objective test instances as listed in Table 1 are adopted here to examine the performance of NSGA-II/DE+DSS in dynamic environments. The first ten test problems are taken from (Farina et al (2004); Goh and Tan (2009); Zhou et al (2014)). The last two problems are newly proposed to include two additional challenging environmental changes. Among these test problems, F1-F4 have linearly correlated decision variables, while F5-F12 have nonlinearly correlated decision variables. As to environmental changes,

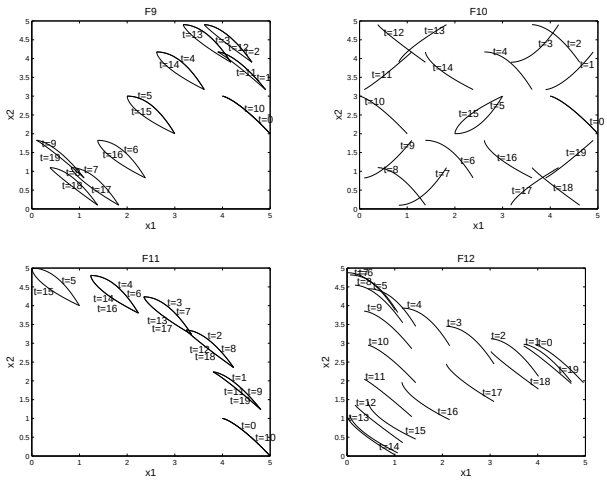


Fig. 2: Projections of the PSs on the lower dimensional space for F9-F12 with $t = 0, 1, \cdots, 19$. The parameters are $n_T = 10$ and $n = 20$.

F1-F8 are subject to continuous environmental changes, whereas F9-F12 have irregular and sharp environmental changes. So F9-F12 are the most difficult problems among the twelve test functions. As indicated in (Zhou et al (2014)), the Pareto fronts of F9 will have a large shift between two consecutive environments, which occurs occasionally. For F10, the shape of two consecutive PFs is substantially changed. By contrast, the PS of F11 suffers from rapid reverse movements near the boundary. F12's Pareto front will have both abrupt large shifts and big rotations, especially when n_T is small. Fig. 2 plots the projections of the PSs of F9-F12 on lower dimensional spaces. Details about the changes in PFs and PSs of other test problems can be found in (Zhou et al (2014)).

3.2 Performance Indicators

IGD (Zhang et al (2008)) is a widely used metric in static MOEAs, which can be used to measure both diversity and convergence of non-dominated optimums to true Pareto front. In this paper, we adopt a modified IGD metric to measure the performance of DMOEAs. Let $P_f^{t^*}$ be a set of uniformly distributed Pareto optimal points in PF^t and P_f^t be an approximation of PF^t. The original IGD metric is defined as (Zhang et al (2008))

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

where $d(v, P_f^t)$ is the minimum Euclidian distance between v and the point in P_f^t. $|P_f^{t^*}|$ is the cardinality of $P_f^{t^*}$.

Table 1: Test instances used in our experiments. F11, F12 are newly proposed test instances, where $x = (x_1, \cdots, x_n)$ is decision vector, $t = 0, 1, \cdots$ is time instances and n_T is an environmental parameter.

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

F9	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$, $f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$, $y_i=x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}}, H=1.25+0.75\sin(\pi\frac{t}{n_T})$ $a=2\cos((\frac{t}{n_T}-\lfloor\frac{t}{n_T}\rfloor)\pi)+2, b=2\sin(2(\frac{t}{n_T}-\lfloor\frac{t}{n_T}\rfloor)\pi)+2,$ $I_1=\{i\lvert 2\le i\le n, i \text{ is odd}\}, I_2=\{i\lvert 2\le i\le n, i \text{ is even}\}$ $PS(t): a\le x_1\le a+1, x_i=b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}}, for\ i=2,\cdots,n$ $PF(t): f_1=s^H, f_2=(1-s)^H, 0\le s\le 1$	F9 PF changes PS changes Two objectives
F10	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$, $f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$, $y_i=\begin{cases}x_i-b-\lvert x_1-a\rvert^{H+\frac{i}{n}} & \text{if t is odd}\\ x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}} & \text{otherwise}\end{cases}$ $H=1.25+0.75\sin(\pi\frac{t}{n_T}), a=2\cos(\pi\frac{t}{n_T})+2, b=2\sin(2\pi\frac{t}{n_T})+2$ $I_1=\{i\lvert 2\le i\le n, i \text{ is odd}\}, I_2=\{i\lvert 2\le i\le n, i \text{ is even}\}$ $PS(t): a\le x_1\le a+1, x_i=\begin{cases}b+\lvert x_1-a\rvert^{H+\frac{i}{n}} & \text{if t is odd}\\ b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}} & \text{otherwise}\end{cases}$ $for\ i=2,\cdots,n$ $PF(t): f_1=s^H, f_2=(1-s)^H, 0\le s\le 1$	F10 PF changes PS changes Two objectives
F11	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$, $f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$, $y_i=x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}}, H=1.25+0.75\sin(\pi\frac{t}{n_T})$ $a=4\lvert\cos(\pi\frac{t}{n_T})\rvert, b=4\lvert\sin(\pi\frac{t}{n_T})\rvert$ $I_1=\{i\lvert 2\le i\le n, i \text{ is odd}\}, I_2=\{i\lvert 2\le i\le n, i \text{ is even}\}$ $PS(t): a\le x_1\le a+1, x_i=b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}}, for\ i=2,\cdots,n$ $PF(t): f_1=s^H, f_2=(1-s)^H, 0\le s\le 1$	PF changes PS changes Two objectives
F12	$[0,5]^n$	$f_1(x,t)=\lvert x_1-a\rvert^H+\sum_{i\in I_1}y_i^2$, $f_2(x,t)=\lvert x_1-a-1\rvert^H+\sum_{i\in I_2}y_i^2$, $y_i=x_i-b-1+\lvert x_1-a\rvert^{H+\frac{i}{n}}, H=1.25+0.75\sin(\pi\frac{t}{n_T})$ $a=1.76\cos(\pi\frac{t}{n_T})+0.88\cos(2\pi\frac{t}{n_T})+1.32, b=1.5\sin(\pi\frac{t}{n_T})(1-\cos(\pi\frac{t}{n_T}))+1.05$ $I_1=\{i\lvert 2\le i\le n, i \text{ is odd}\}, I_2=\{i\lvert 2\le i\le n, i \text{ is even}\}$ $PS(t): a\le x_1\le a+1, x_i=b+1-\lvert x_1-a\rvert^{H+\frac{i}{n}}, for\ i=2,\cdots,n$ $PF(t): f_1=s^H, f_2=(1-s)^H, 0\le s\le 1$	PF changes PS changes Two objectives

The objective of DMOEAs is to track moving PF (PS) as closely as possible, not only to find a particular single PF (PS). To consider all PF (PS), the average IGD, which is denoted as MIGD (Zhou et al (2014))) over the time steps in the whole run can be defined as follows:

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

where T is a set of discrete time instances in a run and $|T|$ is the cardinality of T. The lower a MIGD value is, the better the tracking performance.

In our experiments, 2500 uniformly distributed points in the PFs of F4 and F8 and 500 uniformly distributed points in the PFs of the rest test problems are taken to form $P_f^{t^*}$ for computing IGD.

4 Experimental Results

4.1 Compared Algorithms and Parameter Settings

For fair comparative studies, two existing prediction strategies for handling dynamic MOEAs, namely, FPS (Hatzakis and Wallace (2006b)) and PPS (Zhou et al (2014)) are chosen to be incorporated into NSGA-II/DE. For simplicity, the three algorithms under comparison are denoted as DSS, FPS and PPS, respectively.

The parameter settings for the test problems and different algorithms are as follows. The severity of changes is set to be n_T=10. The dimensions of the test problems are $n=20$. The population size is set to be $N=100$ for all test problems. For change detection, at every generation, 5% randomly selected points from the population are reevaluated to detect environmental changes. The environments change for every $K=5500$ function evaluations. As a result, the environment changes every 50 generations for DSS and 52.5 generations for FPS and PPS. For simplicity, the change frequency for FPS and

Table 2: Mean and standard deviation of MIGD for FPS, PPS and DSS on F1-F12 over 20 runs

Instance	Strategy	$t = 0$ mean(std)	$1 \leq t \leq 20$ mean(std)	$21 \leq t \leq 40$ mean(std)	$41 \leq t \leq 80$ mean(std)
F1	FPS	0.0335(0.0119)	0.0110(7.3987e-004)	0.0074(3.3263e-004)	**0.0068(2.2259e-004)**
	PPS	**0.0299(0.0094)**	0.0113(8.8305e-004)	0.0082(6.2491e-004)	0.0074(4.2087e-004)
	DSS	0.0307(0.0074)	**0.0077(2.2167e-004)**	**0.0070(1.2797e-004)**	0.0070(9.7054e-005)
F2	FPS	**0.6146(0.2118)**	0.0475(0.0249)	**0.0055(6.3386e-005)**	**0.0054(5.5716e-005)**
	PPS	0.6869(0.1604)	0.0517(0.0274)	0.0059(8.5134e-004)	0.0057(3.1957e-004)
	DSS	0.6895(0.1742)	**0.0088(0.0010)**	0.0070(1.1261e-004)	0.0069(8.8621e-005)
F3	FPS	0.0497(0.0153)	0.0148(0.0013)	0.0080(4.6996e-004)	**0.0071(2.9066e-004)**
	PPS	**0.0408(0.0097)**	0.0158(0.0013)	0.0090(6.7933e-004)	0.0093(7.1135e-004)
	DSS	0.0477(0.0192)	**0.0084(3.7780e-004)**	**0.0072(1.2601e-004)**	0.0073(1.1522e-004)
F4	FPS	**0.2846(0.0564)**	0.1223(0.0066)	0.0979(0.0030)	0.0960(0.0020)
	PPS	0.2873(0.0630)	0.1188(0.0062)	**0.0929(0.0021)**	**0.0886(0.0013)**
	DSS	0.3001(0.0649)	**0.1076(0.0030)**	0.1016(0.0022)	0.1032 (0.0012)
F5	FPS	0.6371(0.3793)	0.5924(0.1290)	0.2857(0.1090)	0.2154(0.1041)
	PPS	0.6878(0.4294)	0.5508(0.1925)	0.2662(0.1893)	0.1594(0.2207)
	DSS	**0.3957(0.1461)**	**0.0236(0.0016)**	**0.0221(0.0020)**	**0.0227(0.0014)**
F6	FPS	2.4267(0.9857)	0.3384(0.1220)	0.0956(0.0246)	0.0791(0.0136)
	PPS	3.0655(1.3283)	0.3729(0.1489)	0.0674(0.0156)	0.0547(0.0055)
	DSS	**0.8267(0.9751)**	**0.0262(0.0035)**	**0.0242(0.0023)**	**0.0249(0.0012)**
F7	FPS	2.0235(0.5762)	0.2523(0.0799)	0.0796(0.0333)	0.0585(0.0060)
	PPS	2.1944(1.1508)	0.2494(0.1320)	0.0664(0.0147)	0.0724(0.0422)
	DSS	**0.4602(0.5020)**	**0.0267(0.0050)**	**0.0200(0.0017)**	**0.0206(0.0010)**
F8	FPS	0.5475(0.1115)	0.1501(0.0056)	0.1189(0.0037)	**0.1239(0.0023)**
	PPS	0.4988(0.0744)	0.1439(0.0061)	**0.1167(0.0043)**	0.1273(0.0054)
	DSS	**0.4489(0.1059)**	**0.1338(0.0036)**	0.1240(0.0036)	0.1265(0.0019)
F9	FPS	0.5246(0.1826)	0.6453(0.2361)	0.3028(0.1131)	0.2597(0.0806)
	PPS	0.6849(0.3724)	0.6374(0.1218)	0.6904(0.3118)	0.6843(0.6271)
	DSS	**0.4205(0.1722)**	**0.0304(0.0056)**	**0.0316(0.0045)**	**0.0309(0.0040)**
F10	FPS	0.5916(0.2503)	0.6283(0.1173)	0.4246(0.1747)	0.3494(0.0701)
	PPS	0.6355(0.2780)	0.6237(0.2153)	1.4666(0.3209)	1.8133(0.3109)
	DSS	**0.4772(0.3732)**	**0.0391(0.0078)**	**0.0393(0.0112)**	**0.0372(0.0043)**
F11	FPS	5.0034(1.9732)	0.6744(0.2487)	0.3724(0.1178)	0.3256(0.0741)
	PPS	5.2397(2.1149)	0.3606(0.1347)	0.9110(0.1406)	1.4834(0.2291)
	DSS	**2.8563(2.5362)**	**0.0586(0.0246)**	**0.0421(0.0102)**	**0.0392(0.0058)**
F12	FPS	0.5732(0.2715)	0.2870(0.1079)	0.1269(0.0509)	0.0796(0.0226)
	PPS	0.6191(0.3356)	0.2460(0.1231)	0.1415(0.0975)	0.1691(0.1880)
	DSS	**0.4814(0.3454)**	**0.0333(0.0041)**	**0.0267(0.0023)**	**0.0281(0.0024)**

PPS is set to be 55 generations. The crossover and mutation probability are set to be 0.9 and 0.1, respectively. Twenty independent simulation runs are performed for each of the test problems. The parameter settings for different algorithms are listed as follows.

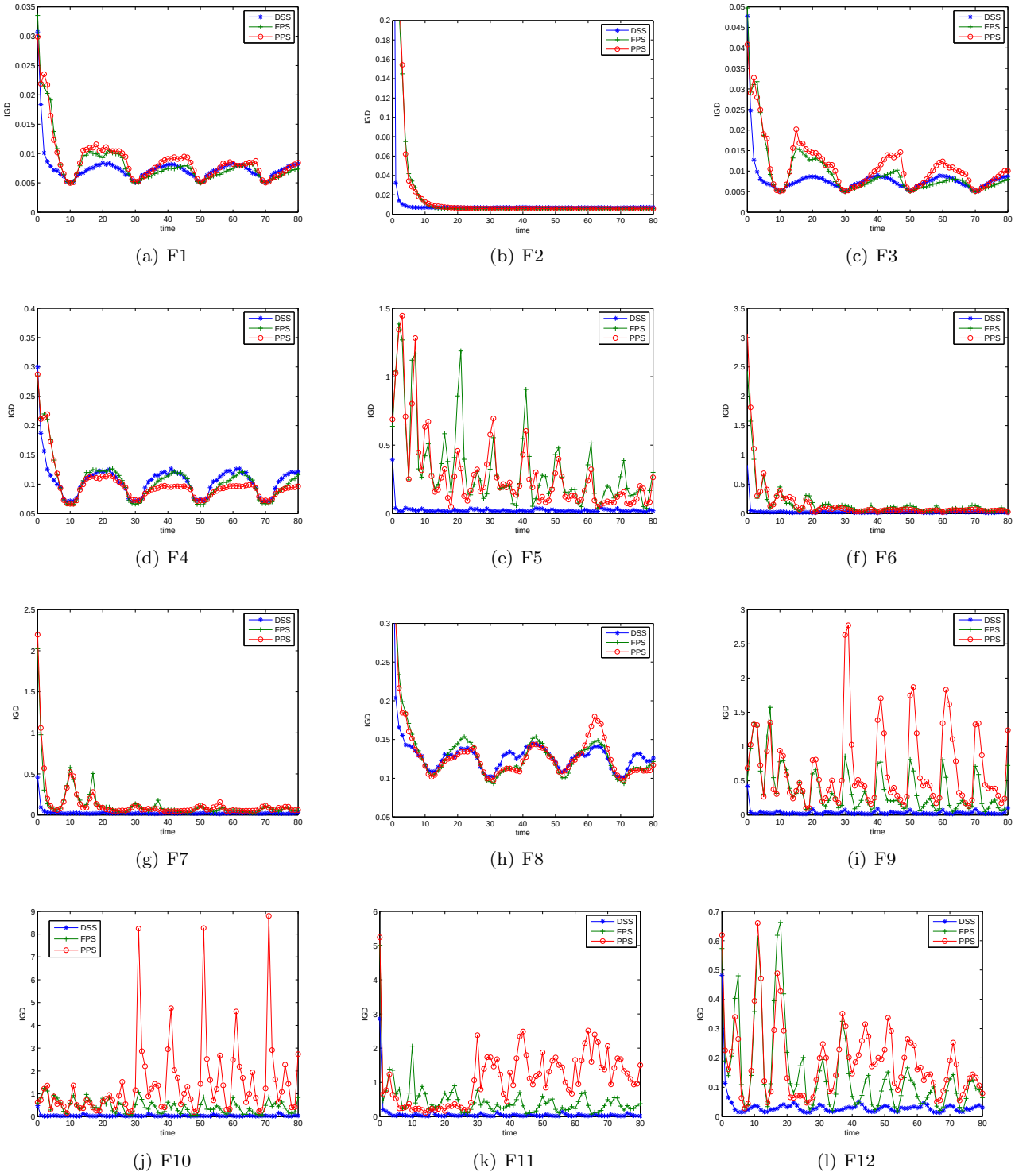
- Parameters in DSS: $r_1 = 0.5$ in DSS1, $r_2 = 0.05$ in DSS2
- Parameters in FPS (Hatzakis and Wallace (2006b)): The reinitialized population is composed of three parts, which are $3(m+1)$ predicted points, $30\%(N - 3(m+1))$ inherited points from the previous population, and $70\%(N - 3(m+1))$ randomly sampled points from the search space. The $AR(p)$ model order is $p = 3$ and the length of history mean point series is $M = 23$, The probability in the prediction model is 0.9.
- Parameters in PPS (Zhou et al (2014)): In the original PPS, the reinitialized population contains prediction points only. In this paper, since NSGA-II/DE is not as powerful as RM-MEDA (Zhang et al (2008)), in which the PPS was originally embedded (Zhou et al (2014)), we slightly modify the setup, i.e., when $t < 23$, the initialized population will combine 50% predicted points, 30% randomly sampled points and

20% inherited points from the previous population. When $t \geq 24$, PPS is exactly the same as in the original algorithm (Zhou et al (2014)). The $AR(p)$ model used here is the same as in (Zhou et al (2014)).

4.2 Experimental Results

Table 2 gives the mean and standard deviation values of the MIGD of each algorithm on each instance. These results are presented with $t = 0$, $1 \leq t \leq 20$, $21 \leq t \leq 40$, $41 \leq t \leq 80$. Fig. 3 shows the average IGD values versus the time on F1-F12. Fig. 4 plots the Pareto front of final populations obtained by FPS, PPS and DSS at different time steps. From the above results, we can make the following observations.

1. From the Table 2, we see that DSS performs better than FPS and PPS at $t = 0$, except for F1, F3 and F4, while DSS performs comparably with FPS and PPS on F1, F3 and F4. The difference between DSS, FPS and PPS at $t = 0$ may be attributed to the fact that DSS introduces some promising individuals to guide the search by DSS2. This indicates that DSS2 does accelerate the convergence of the proposed algorithm. We will further investigate the influence of DSS2 in Section 4.4.
2. From Table 2 and Fig. 3, we can see that DSS has better performance than FPS and PPS on all test instances when $1 \leq t \leq 20$. Both FPS and PPS need a long history to sensibly predict the new points in a changed environment to improve the algorithm. However, when $1 \leq t \leq 20$, there is little history information available, which degrades the performance of FPS and PPS. On the contrary, DSS needs the moving direction of the centroid in the previous environment and no other history information is needed.
3. When $t \geq 21$, as shown in Table 2 and Fig. 3, the mean and deviation values of MIGD of DSS are better than that of FPS and PPS on most of test problems except for F1, F2, F4 and F8. Recall that F1-F4 are the easiest problems that have the linear correlation between the decision variables and are subject to smooth environmental changes. Therefore, DSS, FPS and PPS all perform well on these test problems. FPS and PPS perform better when $t \geq 21$ than when $1 \leq t \leq 20$ because the quality of history information stored by FPS and PPS improves.
   F5-F8 can be considered to be of medium difficulty because they have the nonlinear correlation between the decision variables and smooth changes in the environment. DSS outperforms FPS and PPS except on F8. FPS and PPS are slightly worse than DSS on F6 and F7 and outperform DSS on F8. The reason might be that DSS, FPS and PPS can all predict well the new location in the presence of smooth changes in the environment. However, as observed on F1-F4, the statistical results for FPS and PPS on F5-F8 when $t \geq 21$ are better when $1 \leq t \leq 20$. It indicates that history information can help improve the prediction accuracy of FPS and PPS. For DSS, it performs similarly well when $t < 21$. The results from PPS on F6 are similar to those presented in (Zhou et al (2014)).
   F9-F12 are the most difficult problems as they have nonlinear correlation between the decision variables and they experience sharp environmental changes. From Table 2 and Fig. 3, we can see that DSS consistently shows better performance than FPS and PPS on F9-F12. For PPS, the reinitialized population completely depends on the prediction. When there is a severe environmental change such as a large shift and rotation in the Pareto front, PPS often gets trapped in a local optimum if the whole reinitialized population based on a poor estimation is far away from the true new location of the PS and if the diversity of the reinitialized population is not large enough. Although the reinitialized population in FPS includes individuals randomly sampled from the whole space to address inaccurate prediction, FPS incurs more computational cost for convergence due to an overly large search space. So when $t \geq 21$, the history information cannot help improve the performance of PPS and FPS. By contrast, DSS aims to achieve good diversity resulting from the fact that half of the reinitialized population is based on a rough prediction of the new position of the Pareto set and the rest based on the local search along the directions orthogonal to the moving direction of the Pareto set. The convergence is further speed up by adding promising solutions generated along the moving direction of the non-dominated front in two consecutive generations, which is similar to the derivative information in single objective optimization.
4. From the overall MIGD results of DSS, FPS and PPS, we can see that DSS has robust performance except for $t = 0$, even when the environment changes irregularly. Meanwhile, from Table 2 and Fig. 4, we can see that the statistical results from the complex test problems are worse than those from the simple test problems.



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

(g) F7 (h) F8 (i) F9

(j) F10 (k) F11 (l) F12

Fig. 3: Average IGD values over 20 runs versus time for DSS, FPS and PPS on F1-F12

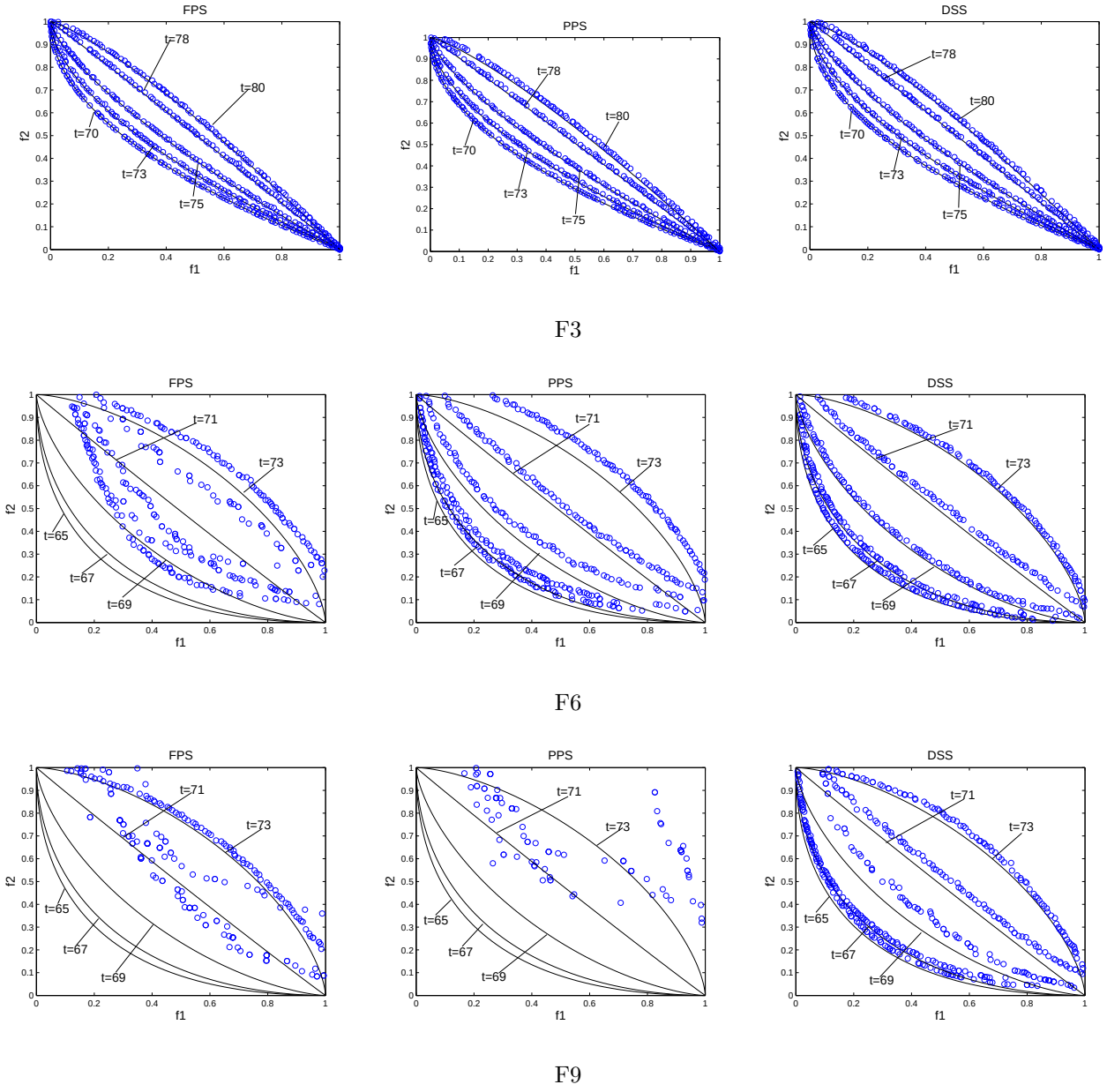


Fig. 4: Average IGD values over 20 runs versus time for DSS, FPS and PPS on F1-F12

4.3 Influence of Severity of Changes

In order to investigate the influence of the severity of changes, n_T, on the performance of DSS, we conduct additional tests on F9-F12 by setting $n_T = 2, 5, 10, 15$. The other parameters are the same as in Section 4.1.

Table 3 shows the mean and standard deviation values of MIGD over 20 runs for DSS with different settings of n_T. From the statistical results, we can see that the performance of DSS becomes better as n_T increases. It means that DSS performs better when the severity of change becomes smaller. This is fairly expected, indicating that DSS is able to focus more on convergence when the change is milder.

4.4 Influence of DSS2 and the parameter of r_2

We are also interested in examining the influence of r_2 on the search performance of DSS. When $r_2 = 0$, it means that the proposed algorithm does not apply DSS2 mechanism. Table 4 shows the mean and standard deviation of MIGD metric for DSS with different settings for $r_2 = 0, 0.05, 0.1, 0.2, 0.5$ on F9-F12 over 20 runs. We can observe that the proposed algorithm

Table 3: Mean and standard deviation of MIGD metric for DSS with different severity of changes n_T, on F9-F12 over 20 runs

	F9		F10		F11		F12	
	$t=0$	$1 \le t \le 80$	$t=0$	$1 \le t \le 80$	$t=0$	$1 \le t \le 80$	$t=0$	$1 \le t \le 80$
$n_T = 15$	0.3957 (0.1461)	0.0241 (0.0016)	0.4360 (0.1336)	0.0324 (0.0037)	3.5122 (2.1679)	0.0294 (0.0049)	0.4695 (0.1954)	0.0249 (0.0016)
$n_T = 10$	0.4205 (0.1722)	0.0309 (0.0028)	0.4772 (0.3732)	0.0382 (0.0046)	2.8563 (2.5362)	0.0448 (0.0072)	0.4814 (0.3454)	0.0290 (0.0017)
$n_T = 5$	0.4216 (0.2127)	0.0255 (0.0028)	0.3957 (0.1461)	0.0337 (0.0053)	2.3070 (1.5099)	0.0902 (0.0123)	0.4422 (0.1749)	0.0417 (0.0033)
$n_T = 2$	0.4299 (0.2050)	0.1004 (0.0077)	0.4607 (0.2026)	0.1110 (0.0172)	2.9113 (1.8599)	0.0642 (0.0304)	0.3961 (0.1227)	0.1134 (0.0169)

Table 4: Mean and standard deviation of MIGD metric for DSS with different parameter of r_2 on F9-F12 over 20 runs

	F9		F10		F11		F12	
	$t=0$	$1 \le t \le 80$	$t=0$	$1 \le t \le 80$	$t=0$	$1 \le t \le 80$	$t=0$	$1 \le t \le 80$
$r_2 = 0$	0.7664 (0.3388)	0.0529 (0.0044)	0.5707 (0.1605)	0.0582 (0.0075)	5.5522 (2.6407)	0.0602 (0.0091)	0.8451 (0.4213)	0.0301 (0.0016)
$r_2 = 0.05$	0.4205 (0.1722)	0.0309 (0.0028)	0.4772 (0.3732)	0.0382 (0.0046)	2.8563 (2.5362)	0.0448 (0.0072)	0.4814 (0.3454)	0.0290 (0.0017)
$r_2 = 0.1$	0.3477 (0.1023)	0.0278 (0.0022)	0.3563 (0.1195)	0.0357 (0.0038)	1.4961 (1.2303)	0.0378 (0.0053)	0.3558 (0.1014)	0.0293 (0.0012)
$r_2 = 0.2$	0.2902 (0.1185)	0.0278 (0.0016)	0.2902 (0.1185)	0.0349 (0.0028)	0.4369 (0.3292)	0.0378 (0.0024)	0.2675 (0.0723)	0.0343 (0.0019)
$r_2 = 0.5$	0.3037 (0.0911)	0.0368 (0.0020)	0.3037 (0.0911)	0.0407 (0.0045)	0.1140 (0.0592)	0.0591 (0.0035)	0.2281 (0.0683)	0.0561 (0.0020)

performs the worst when $r_2 = 0$, which indicates that DSS2 can help enhance the exploitation of DSS. For other values of r_2, the obtained results are similar when $r_2 = 0.05, 0.1, 0.2$. DSS performs increasingly reliably with the increase of r_2. However, the performance deteriorates again when $r_2 = 0.5$, indicating that introducing too many individuals by DSS2 may reduce the diversity thus degrade the performance. In this paper we choose $r_2 = 0.05$, also taking computational time into consideration.

4.5 The influence of the local search in DSS1

Local search in DSS plays an important role in finding good and diverse individuals especially when prediction is inaccurate. Here we test the influence of the local search along the direction orthogonal to the moving direction of the Pareto front in comparison with a random local search. For a random local search, a new solution is generated as follows:

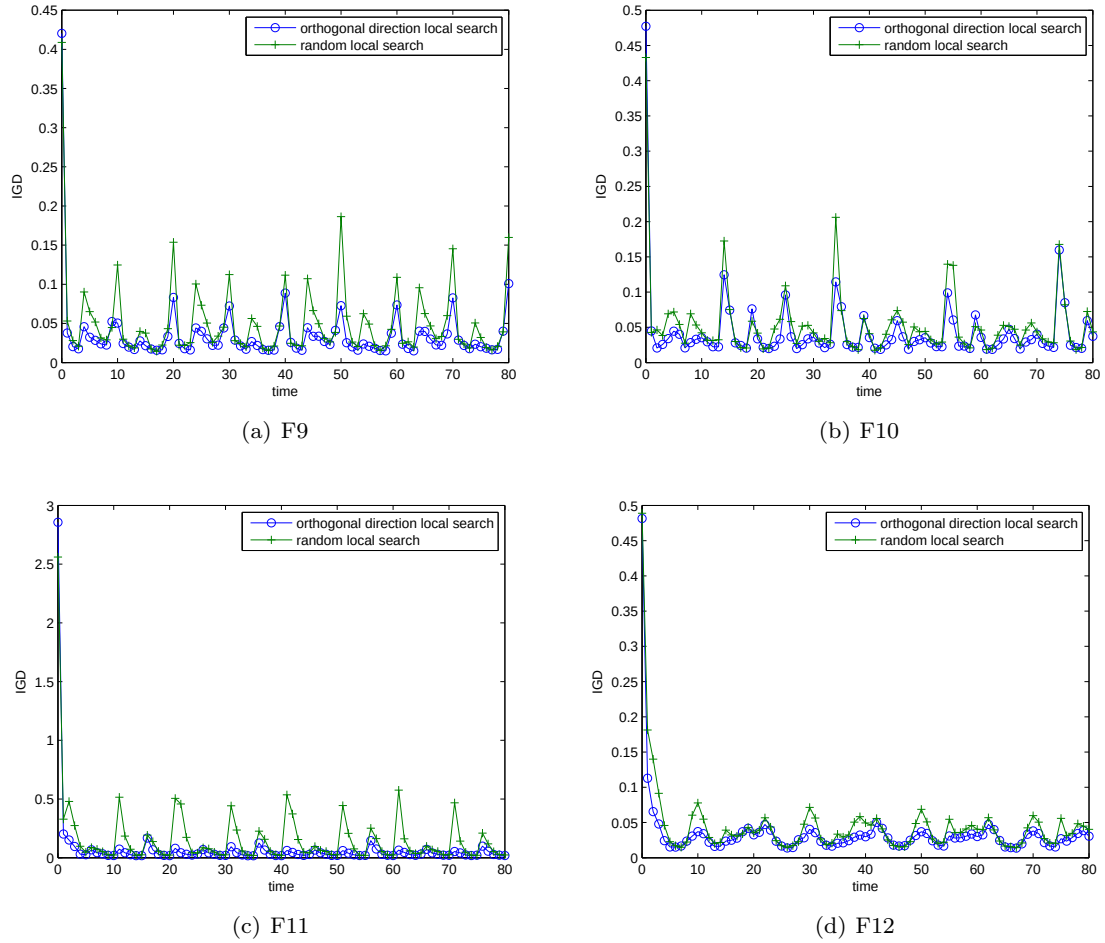
$$y_i = x_i + N(0,1), \tag{10}$$

where $x = (x_1, \cdots, x_n) \in PS^t$. Then we use formula (10) replace formula (6) when a random local search is performed. All other parameters are the same as in Section 4.1.

Fig. 5 plots the average IGD values over 20 runs over time when a local search along the orthogonal directions of the moving Pareto set and a random local search is carried out for optimization of F9-F12, respectively. It demonstrates that the algorithm with the orthogonal local search performs better than that with a random local search. This clearly shows that the local search along the orthogonal direction contributes to the diversity better than the random local search.

5 Conclusion

For MOEAs tracking a moving Pareto front, it is extremely important to achieve a good balance between maintaining diversity and accelerating convergence. In this paper, we propose a DSS to improve the performance of MOEAs in dynamic environments. DSS in-



(a) F9 (b) F10

(c) F11 (d) F12

Fig. 5: Average IGD values over 20 runs versus time for DSS with different local search strategy on F9-F12

cludes two mechanisms aiming to encourage exploration and enhance convergence. The proposed DSS is embedded into a variant of NSGA-II with SBX being replaced by a DE operation. DSS has been compared with FPS and PPS, two state-of-the-art prediction based strategies on twelve test problems having smooth or non-smooth environmental changes. Our experimental results show that DSS performs better than FPS and PPS on most of the test problems considered in this work.

Although DSS has showed very promising performance in dealing with DMOPs, several ideas remain to be verified. For example, due to the good performance of DSS in the early stage of the search, DSS can be combined with FPS or PPS to improve the performance of FPS or PPS. In addition, DSS only uses the information in the previous environments and it may be of interest to explore additional history information. Embedding DSS in other MOEAs such as MOEA/D (Zhang and Li (2007)) is of potential interest. Finally, as indicated in (Jin et al (2013)), how DSS can contribute to finding trade-off solutions that are robust over time is another future work.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (No. 61105065 and No.11326188), the Fundamental Research Funds for the Central Universities (No. K5051270009), and the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (Grant No. 61428302).

References

Abello M, Bui L, Michalewicz Z (2011a) An adaptive approach for solving dynamic scheduling with time-varying number of tasks: Part I. In: Proc. IEEE CEC, IEEE, pp 1703–1710

Abello M, LTBui, Michalewicz Z (2011b) An adaptive approach for solving dynamic scheduling with time-varying number of tasks: Part II. In: Proc. IEEE CEC, IEEE, pp 1711–1718

Aragon V, Esquivel S, Coello CC (2005) Evolutionary multiobjective optimization in non-stationary environments. J Comput Sci Technol 5(3):133–143

Azevedo C, Araujo A (2011) Generalized immigration schemes for dynamic evolutionary multiobjective optimization. In: Proc. IEEE CEC, IEEE, pp 2033–2040

Camara M, Ortega J, de Toro F (2009) A single front genetic algorithm for parallel multiobjective optimization in dynamic environments. Neurocomputing 72(16-18):3570–3579

Camara M, Ortega J, de Toro F (2010) Generalized immigration schemes for dynamic evolutionary multiobjective optimization. In: Proc. Advances Multi-Objective Nature Inspired Computation, Springer, pp 63–86

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans on Evol Comput 6(2):182–197

Deb K, Rao U, Karthik S (2007) Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In: Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO, Springer, pp 803–817

Farina M, Deb K, Amato P (2004) Dynamic multiobjective optimization problems: Test cases, approximations, and applications. IEEE Trans Evol Comput 8(5):425–442

Goh CK, Tan K (2009) A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans Evol Comput 13(1):103–127

Greeff M, Engelbrecht A (2008) Solving dynamic multiobjective problems with vector evaluated particle swarm optimisation. In: Proc. IEEE CEC, IEEE, pp 2922–2929

Hatzakis I, Wallace D (2006a) Dynamic multiobjective optimization with evolutionary algorithms: A forward-looking approach. In: Proc. GECCO, ACM, pp 1201–1208

Hatzakis I, Wallace D (2006b) Topology of anticipatory populations for evolutionary dynamic multiobjective optimization. In: Proc. 11th AIAA/ISSMO Multidisciplinary Anal. Optimization Conf., AIAA, pp 1944–1950

Helbig M, Engelbrecht A (2012) Analyses of guide update approaches for vector evaluated particle swarm optimisation on dynamic multiobjective optimisation problems. In: Proc. IEEE CEC, IEEE, pp 2621–2628

Iorio A, Li X (2005) Solving rotated multi-objective optimization problems using differential evolution. In: Advances in Artificial Intelligence. LNAI:3339, Springer, pp 861–872

Jin Y, Branke J (2005) Evolutionary optimization in uncertain environments: A survey. IEEE Trans Evol Comput 9(3):303–317

Jin Y, Sendhoff B (2004) Constructing dynamic test problems using the multi-objective optimization concept. In: Applications of Evolutionary Computing. LNCS 3005, Springer, pp 525–536

Jin Y, Tang K, Yu X, Sendhoff B, Yao X (2013) A framework for finding robust optimal solutions over time. Memetic Computing 5(1):3–18

Li H, Zhang Q (2009) Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Transactions on evolutionary computation 13(2):284–302

Liu C, Wang Y (2006) New evolutionary algorithm for dynamic multiobjective optimization problems. In: Advances in Natural Computation,LNCS:4221, Springer, pp 889–892

Liu C, Wang Y (2009) Multiobjective evolutionary algorithm for dynamic nonlinear constrained optimization problems. J Syst Eng Electron 20(1):204–210

Liu R, Zhang W, Jiao L, Liu F, Ma J (2011) A sphere-dominance based preference immune-inspired algorithm for dynamic multiobjective optimization. In: Proc. GECCO, ACM, pp 423–430

Ma Y, Liu R, Shang R (2011) A hybrid dynamic multiobjective immune optimization algorithm using prediction strategy and improved differential evolution crossover operator. In: Neural Information Processing. LNCS:7063, Springer, pp 435–444

Manriquez A, Pulido G, Torres J (2010) Handling dynamic multiobjective problems with particle swarm optimization. In: Proceedings of the International Conference on Agents and Artificial Intelligence, ICAART, pp 337–342

Martins F, Carrano E, Wanner E, Takahashi R, Mateus G (2009) A dynamic multiobjective hybrid approach for designing wireless sensor networks. In: Proc. IEEE CEC, IEEE, pp 1145–1152

Nguyen T, Yang S, Branke J (2012) Evolutionary dynamic optimization: A survey of the state of the art. Swarm Evol Comput 6:1–24

Tang J, Alam S, Lokan C, Abbass H (2012) A multiobjective evolutionary method for dynamic airspace re-sectorization using sectors clipping and similarities. In: Proc. IEEE CEC, IEEE, pp 3565–3572

Vinek E, Beran P, Schikuta E (2011) A dynamic multiobjective optimization framework for selecting distributed deployments in a heterogeneous environment. Procedia Comput Sci 4:166–175

Wang Y, Li B (2009) Investigation of memory-based multiobjective optimization evolutionary algorithm in dynamic environment. In: Proc. IEEE CEC, IEEE,

pp 630–637

Wei J, Wang Y (2012) Hyper rectangle search based particle swarm algorithm for dynamic constrained multiobjective optimization problems. In: Proc. IEEE CEC, IEEE, pp 259–266

Wei J, Zhang M (2011) Simplex model based evolutionary algorithm for dynamic multiobjective optimization. In: Advances in Artificial Intelligence.LNCS:7106, Springer, pp 372–381

Wu PY, Campbel D, Merz T (2011) Multiobjective four-dimensional vehicle motion planning in large dynamic environments. IEEE Trans Syst, Man, Cybern B, Cybern 41(3):621–634

Yang M, Kang L, Guan J (2008) Multialgorithm coevolution strategy for dynamic multiobjective TSP. In: Proc. IEEE CEC, IEEE, pp 466–471

Zhang Q, Li H (2007) MOEA/D: A multiobjective evolutionary algorithm based on decomposition,. IEEE Trans Evol Comput, 11(6):712–731

Zhang Q, Zhou A, Jin Y (2008) RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. IEEE Trans Evol Comput 12(1):41–63

Zhang Z (2008) Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control. Appl Soft Comput 8(2):959–971

Zheng B (2007) A new dynamic multiobjective optimization evolutionary algorithm. In: Third International Conference on Natural Computation.ICNC, IEEE, pp 565–570

Zhou A, Jin Y, Zhang Q, Sendhoff B, Tsang E (2007) Prediction based population re-initialization for evolutionary dynamic multiobjective optimization. In: Evolutionary Multi-Criterion Optimization.LNCS:4403, Springer, pp 832–846

Zhou A, Jin Y, Zhang Q (2014) A population prediction strategy for evolutionary dynamic multiobjective optimization,. IEEE Transactions on Cybernetics, 44(1):40–53