A First-Order Difference Model-Based Evolutionary Dynamic Multiobjective Optimization

Leilei Cao[1], Lihong Xu[1](✉), Erik D. Goodman[2], and Hui Li[3]

[1] Department of Control Science and Engineering, Tongji University, Shanghai 201804, China
mcaoleilei@sina.com, xulhk@163.com

[2] BEACON Center, Michigan State University, East Lansing, MI 48824, USA
goodman@egr.msu.edu

[3] School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China
lihui10@mail.xjtu.edu.cn

Abstract. This paper presents a novel algorithm to solve dynamic multiobjective optimization problems. In dynamic multiobjective optimization problems, multiple objective functions and/or constraints may change over time, which requires a multiobjective optimization algorithm to track the moving Pareto-optimal solutions and/or Pareto-optimal front. A first-order difference model is designed to predict the new locations of a certain number of Pareto-optimal solutions based on the previous locations when an environmental change is detected. In addition, a part of old Pareto-optimal solutions are retained to the new population. The prediction model is incorporated into a multiobjective evolutionary algorithm based on decomposition to solve the dynamic multiobjective optimization problems. In such a way, the changed POS or POF can be tracked more quickly. The proposed algorithm is tested on a number of typical benchmark problems with different dynamic characteristics and difficulties. Experimental results show that the proposed algorithm performs competitively when addressing dynamic multiobjective optimization problems in comparisons with the other state-of-the-art algorithms.

Keywords: Prediction model · Dynamic multiobjective optimization · Evolutionary algorithm · Decomposition

1 Introduction

In real-world multiobjective optimization problems, particularly in optimal control problems or problems requiring an on-line optimization [1], many of them are dynamic in nature, whose objective functions, constraints, or decision variables may change over time [2, 3]. This kind of problem is usually known as a dynamic multiobjective optimization problem (DMOP), or considered as multiobjective optimization in dynamic environments [4]. DMOP challenges evolutionary algorithms (EAs) since any environmental change may affect the objective functions or constraints, which resulting in that the Pareto-optimal solutions (POS) or Pareto-optimal front (POF) may change over time [2]. Therefore, the optimization goal for evolutionary algorithms (EAs) is to track the moving POS and/or POF and obtain a sequence of approximations over time [2, 4].

There are many different dynamic characteristics or uncertainties in optimization in nature. In this paper, we focus on the following class of DMOPs [5]:

$$\begin{aligned} &\mathit{minimize}\, F(x,t) = (f_1(x,t), f_2(x,t), \cdots, f_m(x,t))^T \\ &\mathit{subject\ to}\, x \in \prod\nolimits_{i=1}^{n} [a_i, b_i] \end{aligned} \tag{1}$$

where m is the number of objectives, $t = 0, 1, 2 \cdots$ represents discrete time instants. $\prod_{i=1}^{n} [a_i, b_i] \subset R^n$ defines the feasible region of the decision space, and $x = (x_1, x_2, \cdots, x_n)^T$ is the decision variable vector. $F(x,t)$ is the objective function vector that evaluates solution x at time t.

DMOP defined in (1) can be treated as a sequence of (stationary) multiobjective optimization problems [4], where the problem is considered stationary for some time period and an optimization algorithm be allowed to find optimal or near-optimal solutions within the time span in which the problem remains stationary [1].

This problem has been rapidly attracting the interest of the research community in recent years [2]. Many existing typical multiobjective optimization evolutionary algorithms (i.e., NSGA-II [1, 6], MOEA/D [7, 8], RE-MEDA [4]) have been modified to solve DMOPs. Adapting search behavior quickly to the environmental changes is a key issue in dealing with dynamic problems. In order to adapt the evolutionary algorithm to the new environment rapidly, diversity increasing [1], memory [8] and prediction [4, 7, 9, 10] are most used approaches in recent literature. Although a number of approaches have been proposed, the development of this research area is a relatively young field and more studies are greatly needed [2].

In this paper, we propose a simple first-order difference model incorporated into a multiobjective evolutionary algorithm based on decomposition (MOEA/D) to solve DMOPs. MOEA/D decomposes a multiobjective optimization problem into a number of single objective optimization subproblems through aggregation function and optimizes them simultaneously [11]. It has been paid widely attention since it was proposed. We use a simple yet effective model to predict the location of the new Pareto-optimal solutions based on the previous locations. The predicted locations of a number of solutions are mixed with some old solutions to form a new population once a change is detected. In such a way, the changed POS or POF can be found more quickly by the modified MOEA/D.

The remainder of this paper is organized as follows. Section 2 presents a brief overview on evolutionary dynamic multiobjective optimization algorithms in recent literature. Section 3 presents our proposed approach. A comparative study and related discussions based on a set of benchmark functions are given in Sect. 4. Finally, a conclusion is given in Sect. 5.

2 Related Work

A dynamic multiobjective optimization evolutionary algorithm (DMOEA) should be composed of three major components, including change detection, change reaction, and multiobjective optimization [4]. The main steps of a typical DMOEA are described as follows.

```
Step 0. Set time step g = 0, and time window: T = 1. Initialize a population: $P^g$;
Step 1. If a change is detected:
        1.1. Update the population: reuse memory, tune parameters, or predict
             solutions;
        1.2. T = T + 1;
Step 2. Optimize the $T$-th MOP by using an MOEA for one generation;
Step 3. If the stop criterion is met, stop; else set g = g + 1 and go to Step 1.
```

To detect the environmental change in Step1, some solutions are reevaluated in the beginning of each generation, which is used by many algorithms [2, 7, 10]. It is easy to implement but it is based on the assumption that there is no noise in function evaluations [4].

In Step 1.1, the algorithm performs reaction on the population. There are three major approaches to update the population: memory maintenance, parameter tuning and model-based prediction.

Memory Maintenance: Memory based approach in DMOEA uses an addition set to store Pareto-optimal solutions from the previous environment in order to reuse them when necessary [6]. This approach is effective when periodic changes occurred. [6] utilizes an explicit memory to store a number of non-dominated solutions to reuse in later stages to reinitialize part of the population when an environment change occurs. Similarly, [8, 12] use the memory scheme to store the old Pareto-optimal solutions. [13] hybrids explicit memory and local search memory scheme to store the old Pareto-optimal solutions.

Parameter Tuning: This approach increases or maintains diversity of population through tuning parameters of DMOEA. All population or part of it can be reinitialized randomly whenever a change is detected, which is used by DNSGA-II-A [1]. Another widely used method is randomly reinitializing a fixed of changed percent of population in each generation regardless of the detected change [6]. Hyper mutation operator is a typical method to increase the diversity of population, in which the mutation probability can be increased rapidly once a change is detected [14].

Model-Based Prediction: In real-world DMOPs, many of them change as some regular rules, but not randomly. Thereafter, prediction based algorithms were proposed to solve DMOPs. They try to estimate the next locations of POS from the past sequence locations of POS when a change is detected [6]. A linear model with a Gaussian noise is most adopted to predict the new locations of POS [2, 5, 10]. An autoregressive model is used as a forecasting method by [4, 9, 15]. In addition, [7, 16] adopts a Kalman filter model to predict the new locations of POS, which was incorporated into MOEA/D to solve DMOP.

3 Prediction Model-Based MOEA/D Algorithm

In order to predict the locations of the POS, we have to make some assumptions: (1) a DMOP remains stationary within a fixed time span; (2) the POS or POF of consecutive MOPs changes as a regular rule. Based on two assumptions, we propose a first-order linear model to predict the new location of the centroid of the POS.

In DMOEA, a tradeoff between diversity and convergence is an important task. Diversity is maintained inherently in a static multi-objective optimization algorithm. Thus, we concentrate on fast convergence to the new POF when a change is detected in the environment by predicting the new locations of the POS [5]. We assume that the recorded solutions in the previous time windows when a change is detected, i.e. $P_T, \cdots, P_1$, can provide information for predicting the new locations of the POS at time window T + 1. The locations of POS can be considered as a function of the locations $P_T, \cdots, P_1$:

$$P_{T+1} = F(P_T, \cdots, P_1, t) \tag{2}$$

where P_{T+1} denotes the new location of the POS for time window T + 1. In practice, function $F(\cdot)$ is not known and must be estimated using a certain technique [5].

We assume that each POS $(P_T, \cdots, P_1)$ is composed of N solutions, then $x_{i,t}(i = 1, 2\cdots, N, t = 1, 2, \cdots, T)$ represents the i-th solution in the t-th POS $P_t, t = 1, 2, \cdots, T$. Accordingly, $x_{i,T}, x_{i,T-1}, \cdots, x_{i,1}$ are a series of solutions that describe the movement of the i-th solution in the POS over time, a prediction model can estimate its next location in the following time window $x_{i,T+1}$ by the former series solutions [5]. Theoretically, each solution should be estimated by a separate prediction model, therefore there will be N prediction models to describe the movement of N solutions in the POS exactly. However, it is too complex to use so many different prediction models. Actually in the real-world dynamic problems, the N solutions have similar moving characteristics, and the aim of prediction model is only to provide the initial solutions for the evolutionary algorithm, therefore a generic prediction model is enough for these N solutions. The generic model to predict the location of the initial i-th individual for the time window T + 1 can be formulated as follows [5]:

$$x_{i,T+1} = F\left(x_{i,T}, x_{i,T-1}, \cdots, x_{T-K+1}, t\right) \tag{3}$$

where K represents the number of the previous time windows that $x_{i,T+1}$ is dependent on in the prediction model.

3.1 A First-Order Difference Model

Any time series model can be used for modeling F in (3). The major problem in making a prediction is that it is very difficult to identify the relationship between the stored solutions in $P_T, \cdots, P_1$ to build a time series [5]. However, in MOEA/D, this problem can be ignored, since all solutions are arranged by weight vectors [17, 18]. Although each solution's location at time window T + 1 can be predicted through its historical locations, how can we guarantee the accuracy of these historical locations? In addition, each solution that obtained during the given evaluations may deviate its trajectory over time windows in different ways. To simplify the prediction model and enhance the predicting accuracy, the centroid of solutions is used to describe the movement of solutions over time [4]. We assume that each solution has the same movement as the centroid, including the moving direction and the moving step size. However this condition may be too strict, since the solutions may have similar movements to the centroid but not completely the same in most situations in nature [2]. We still can predict the new location of the centroid to represent the other solutions, since the prediction model is only used to provide the initial individuals to the evolutionary algorithm.

Let C_T be the centroid of the POS and P_T be the obtained approximation POS set at time window T, then C_T can be computed by:

$$C_T = \frac{1}{|P_T|} \sum\nolimits_{x \in P_T} x \tag{4}$$

where $|P_T|$ is the cardinality of P_T, x is a solution in P_T.

Then the movement vector of C_T at the end of time window T is $\overrightarrow{C_T - C_{T-1}}$, which is considered as the inertia of C_T's movement. Consequently, the location of the centroid in the solution space for the next time window T + 1 is predicted as follows:

$$C_{T+1} = C_T + \overrightarrow{C_T - C_{T-1}} \tag{5}$$

This model can be treated as a first-order difference model that is also called linear model by many researchers in their algorithms [5, 10]. In most algorithms, a Gaussian noise is added to the linear model, which is used to compensate possible errors in the prediction [10] and intends to increase the probability of the reinitialized population to cover the POS in the new environment [2]. However, the noise is not needed when predicting since a prediction model is only to provide the initial individuals to the evolutionary algorithm as mentioned before. We don't need exactly accurate locations of the POS in the new environment, but the moving directions and a rough moving step-size are enough to provide valuable information to the evolutionary algorithm. The successful prediction locations are close to the next POS, which assists the algorithm in discovering the new POS quickly. Instead, a noise may perturb the original prediction locations and lead to wrong directions. Therefore, a Gaussian noise is not adopted by the first-order difference model.

3.2 Response to Change

Once an environmental change is detected, the population in the evolutionary algorithm must be updated to respond to the change. Instead of completely predicting all solutions, a part of old solutions are retained to the new population. In real-world DMOPs, the POS of consecutive MOPs are similar to each other in most cases [4]. Therefore, these solutions found just before the change can also be retained to the initial population in the new environment. The old solutions may perform better than the reinitialized population as done in most current DMOEAs.

The new population is composed of two kinds of solutions: the old solutions and the prediction solutions. In MOEA/D, all solutions found just before the change have been arranged by the uniformly distributed weight vectors automatically. Since each subproblem is optimized by using information only from its neighboring subproblems [17], the old solutions and the prediction solutions must be separately distributed, which intends to place the prediction solutions into the population uniformly. As mentioned before, the predicting solutions are assumed to have the same or similar movement with the centroid, therefore each predicting solution can refer to formula (5) to obtain a predicted location in the next time window. The algorithm of updating population works as follows:

```
1): Input $P_T$, $N$ and the historic centroid points $C_{T-1}$, calculate $C_T$;
2): for $i$=1:$N$ do
3):     if mod (i,3)==0 do
4):             $x_i^{T+1} = x_i^T + \overrightarrow{C_T - C_{T-1}}$;
5):             Boundary check $x_i^{T+1}$;
6):     else $x_i^{T+1} = x_i^T$;
7):     end if
8): end for
9): Output $P_{T+1}$.
```

3.3 Summary of the Proposed Algorithm

A first-order difference model is incorporated into MOEA/D (simplify as MOEA/D-FD) in this paper. There are many variants of MOEA/D in recent literature; MOEA/D with a differential evolution (DE) operator and a polynomial mutation operator is adopted in this paper [17]. The main steps of the proposed algorithm are described as follows. (The detail steps in MOEA/D-DE can refer to [17]).

Step 0. Initialize and evaluate an initial population P^0, then initialize the ideal point z;
Step 1. Detect the environmental change;
  Step 1.1. If the environment changes, go to Step 1.2, else go to Step 2.
  Step 1.2. Predict next location of the centroid, and update the current population;
  Step 1.3. Reevaluate the new population, and update the ideal point z;
Step 2. Update the population using DE and the polynomial mutation operator;
Step 3. If the stopping criterion is met, stop; else go to Step 1.

In Step 1, to detect the environmental change, 10% randomly selected population members are re-evaluated for change detection, which is widely used by many algorithms [2, 4, 7]. Note that the first-order difference model needs former two locations of the centroid to predict the next location, therefore Step 1.2 works beginning from the third time window. In the second time window, Step 1.2 is omitted, and all solutions of the POS are retained to be a new population.

4 Experimental Study

4.1 Benchmark Problems and Performance Metrics

The proposed algorithm is tested on six benchmark problems, including FDA1–FDA3 [19] and JY1–JY3 [20]. The FDA test suite is commonly used to evaluate the performance of DMOEAs. JY test suite is a recently proposed benchmark framework, which is able to tune a number of challenging characteristics, including mixed Pareto-optimal front, nonmonotonic and time-varying variable linkages, mixed types of changes [20]. The time instance t involved in these problems is defined as $t = (1/n_t) * \lfloor \tau/\tau_t \rfloor$ [19] (where n_t, τ_t, and τ represent the severity of change, the frequency of change, and the iteration counter, respectively). The definition of these problems can be found in [19, 20], respectively. Note that, FDA2 has been modified, and the modified version of FDA2 can be found in [1], but the time instance in this modified FDA2 is still defined as mentioned before. The dimensions of FDA1 and FDA2 are set to be 11 and 13, respectively. In FDA3, the dimension is set to be $|X_I| = 2$ and $|X_{II}| = 8$. The domains of decision variables in FDA test suites are set as [19]. The dimensions of JY test suites are set to be 10, and the domains are set as [20].

There are a number of metrics to be used for performance assessment of DMOEAs, i.e., Inverted Generational Distance (IGD), Schott's Spacing Metric, Maximum Spread and Hypervolume Difference. In this paper, a modified version of the IGD (MIGD) metric is used to assess the algorithms, which is suggested in [4, 7].

Let Q^{t*} be a set of uniformly distributed points in the true POF^t, and Q^t be an approximation of POF^t. The IGD metric is defined as:

$$IGD(Q^{t*}, Q^t) = \frac{\sum_{v \in Q^{t*}} d(v, Q^t)}{|Q^{t*}|} \tag{6}$$

where $d(v, Q^t) = \min_{u \in Q^t} \|F(v) - F(u)\|$ is the distance between v and Q^t, and $|Q^{t*}|$ is the cardinality of Q^{t*}. The IGD metric can measure both diversity and convergence. To have a low IGD value, Q^t should be very close to the true POF^t and cannot miss any part of the whole POF^t. In the experiments, 500 points are uniformly sampled from the true POF^t of biobjective problems for computing the IGD metrics.

The MIGD metric is defined as the average of the IGD values in some time windows over a single run:

$$MIGD = \frac{1}{|T|} \sum_{t \in T} IGD(Q^{t*}, Q^t) \tag{7}$$

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

4.2 Compared Algorithms and Parameter Settings

Three popular DMOEAs are used for comparison in our empirical studies, including MOEA/D [17], DNSGA-II-A [1] and DDS [10]. To adapt the dynamic change of problems, the original MOEA/D is modified. A part of population members are randomly reinitialized when the environmental changes, which is similar to the idea of DNSGA-II-A. DDS combined a linear prediction model as well as a directed local search with NSGA-II. These three compared algorithms also use DE and the polynomial mutation operator to generate new individuals. The problem and algorithms parameter settings are as follows.

(1) The problem parameters: To study the impact of change frequency and change severity, different parameters are adopted. The severity of changes is $n_t = 10, 5$. The frequency of changes is $\tau_t = 5, 10, 20$. Thereafter, each problem will have 6 different cases of changes, and the total number of changeable cases is 36.
(2) Common parameters in all algorithms: the population size is set as 100. CR = 0.5, F = 0.5 in the DE operator. $\eta = 20$, $p_m = 1/n$ in the polynomial mutation operator (n is the dimension of variables).
(3) Other parameters in MOEA/D-FD and MOEA/D: T = 20, $\delta = 0.8$, $n_r = |P|$ ($|P|$ is the cardinality of P). The meaning of P can refer to [17]. Note that the value of n_r in this paper is different from [17], which is a key parameter in solving DMOPs. Besides, the Techebycheff method is used to be as a decomposition approach in MOEA/D-FD and MOEA/D.
(4) In MOEA/D and DNSGA-II-A, 20% of population members are randomly reinitialized within the domain when the environment changes. The other parameters of DSS can refer to [10].
(5) Algorithm cost: The number of total generations in each run is fixed to be $40 * \tau_t$, which ensures that 40 changes will happen in each experiment. Each algorithm run 30 times for each test instance independently.

4.3 Comparative Study

Results on FDA: Table 1 shows the obtained average MIGD values and standard deviations over 30 runs by four algorithms on the FDA test instances, where the best values are highlighted in bold. The nonparametric statistical test results called Wilcoxon's rank sum test are in the brackets followed by the standard deviations in Table 1. The statistical test is conducted at the 5% significance level, and the results are marked as "+," "−," or "$\sim$" when MOEA/D-FD is statistically significantly better than, worse than, or statistically equivalent to the corresponding algorithm, respectively. It is obvious that MOEA/D-FD performed best on the majority of the FDA instances, implying that it has the best tracking ability of changing POS and/or POF in most cases. MOEA/D performed better than MOEA/D-FD on FDA3 with three cases. The POS and POF of FDA3 changed over time, in which environmental changes shifted the POS and affected the density of points on the POF. The reinitialized approach for MOEA/D was better than prediction when the change frequency was relatively slow ($\tau_t = 20$) or the change severity was relatively smooth ($n_t = 10, \tau_t = 10$) on this problem, which implied that the diversity of population was more needed on these cases. However, when the change frequency was fast, the prediction approach could enhance the searching efficiency, in which the moving direction and the moving step-size may be predicted correctly. The prediction approach worked well, viewing

Table 1. Mean values and standard deviations of MIGD metric obtained by four algorithms on FDA1–FDA3

Problem	(n_t, τ_t)	MOEA/D-FD	MOEA/D	DNSGA-II-A	DSS
FDA1	(5,5)	**0.0261 ± 0.0016**	0.1080 ± 0.0044(+)	0.2472 ± 0.0002(+)	0.0942 ± 0.0093(+)
	(5,10)	**0.0106 ± 0.0005**	0.0211 ± 0.0022(+)	0.0966 ± 0.0027(+)	0.0484 ± 0.0010(+)
	(5,20)	**0.0063 ± 0.0001**	0.0077 ± 0.0002(+)	0.0286 ± 0.0014(+)	0.0195 ± 0.0022(+)
	(10,5)	**0.0187 ± 0.0055**	0.0374 ± 0.0017(+)	0.1458 ± 0.0016(+)	0.0817 ± 0.0027(+)
	(10,10)	**0.0080 ± 0.0005**	0.0134 ± 0.0006(+)	0.0551 ± 0.0001(+)	0.0283 ± 0.0013(+)
	(10,20)	**0.0055 ± 0.0001**	0.0072 ± 0.0001(+)	0.0214 ± 0.0024(+)	0.0127 ± 0.0011(+)
FDA2	(5,5)	**0.0204 ± 0.0013**	0.0283 ± 0.0022(+)	0.0525 ± 0.0021(+)	0.1880 ± 0.0087(+)
	(5,10)	**0.0083 ± 0.0001**	0.0114 ± 0.0003(+)	0.0244 ± 0.0002(+)	0.0434 ± 0.0027(+)
	(5,20)	**0.0058 ± 0.0001**	0.0070 ± 0.0002(+)	0.0125 ± 0.0003(+)	0.0148 ± 0.0004(+)
	(10,5)	**0.0114 ± 0.0009**	0.0137 ± 0.0001(+)	0.0317 ± 0.0008(+)	0.0716 ± 0.0072(+)
	(10,10)	**0.0072 ± 0.0001**	0.0083 ± 0.0001(+)	0.0168 ± 0.0005(+)	0.0203 ± 0.0001(+)
	(10,20)	**0.0053 ± 0.0001**	0.0059 ± 0.0001(+)	0.0103 ± 0.0001(+)	0.0123 ± 0.0009(+)
FDA3	(5,5)	**0.0973 ± 0.0102**	0.1035 ± 0.0128(+)	0.1588 ± 0.0117(+)	0.1202 ± 0.0090(+)
	(5,10)	**0.0559 ± 0.0011**	0.0716 ± 0.0084(+)	0.1083 ± 0.0045(+)	0.0928 ± 0.0002(+)
	(5,20)	0.0514 ± 0.0019	**0.0486 ± 0.0022**(−)	0.0867 ± 0.0023(+)	0.0788 ± 0.0060(+)
	(10,5)	**0.0741 ± 0.0072**	0.0816 ± 0.0097(+)	0.1374 ± 0.0052(+)	0.1064 ± 0.0082(+)
	(10,10)	0.0535 ± 0.0013	**0.0428 ± 0.0016**(−)	0.1009 ± 0.0036(+)	0.0776 ± 0.0033(+)
	(10,20)	0.0495 ± 0.0002	**0.0366 ± 0.0055**(−)	0.0779 ± 0.0010(+)	0.0695 ± 0.0047(+)

from the results of MOEA/D-FD and DDS. DDS performed better than DNSGA-II-A on FDA1 and FDA3, but worse on FDA2. When the change severity was fixed, the performance of all algorithms were better with slower change frequency (more evaluations were allowed). When the change frequency was fixed, all algorithms performed better if the change severity was smoother.

Figure 1(a)–(c) show the tracking of the IGD values with the environmental change obtained by four algorithms for FDA1-FDA3 with $n_t = 10, \tau_t = 10$, which is obtained from the average value over 30 runs. Observing from Fig. 1(a) and (b), the tracking ability of MOEA/D-FD was most stable, especially after the first two time windows (the prediction model works beginning from the third time window). The reinitialized MOEA/D also had better tracking ability than DNSGA-II-A and DDS. These two algorithms that based on NSGA-II could not steadily track the moving POS and/or POF, especially on FDA1. FDA3 challenged all algorithms, in which the IGD curves of four algorithms fluctuated a lot in Fig. 1(c). It is difficult to track the moving POS and POF on stable manners for these algorithms.

Results on JY: Table 2 shows the obtained average MIGD values and the standard deviations over 30 runs by four algorithms on the JY test instances, where the best values are highlighted in bold. The nonparametric statistical test results are also in the brackets. Unlike FDA, the JY test instances have nonlinear linkages between decision variables, and JY test suites introduce some new dynamic features that are not included in FDA. Obviously, DNSGA-II-A and DDS were challenged a lot on the JY test suites, comparing with their performance on FDA test suites. However, MOEA/D-FD and the reinitialized MOEA/D were not affected a lot by these new dynamic features and

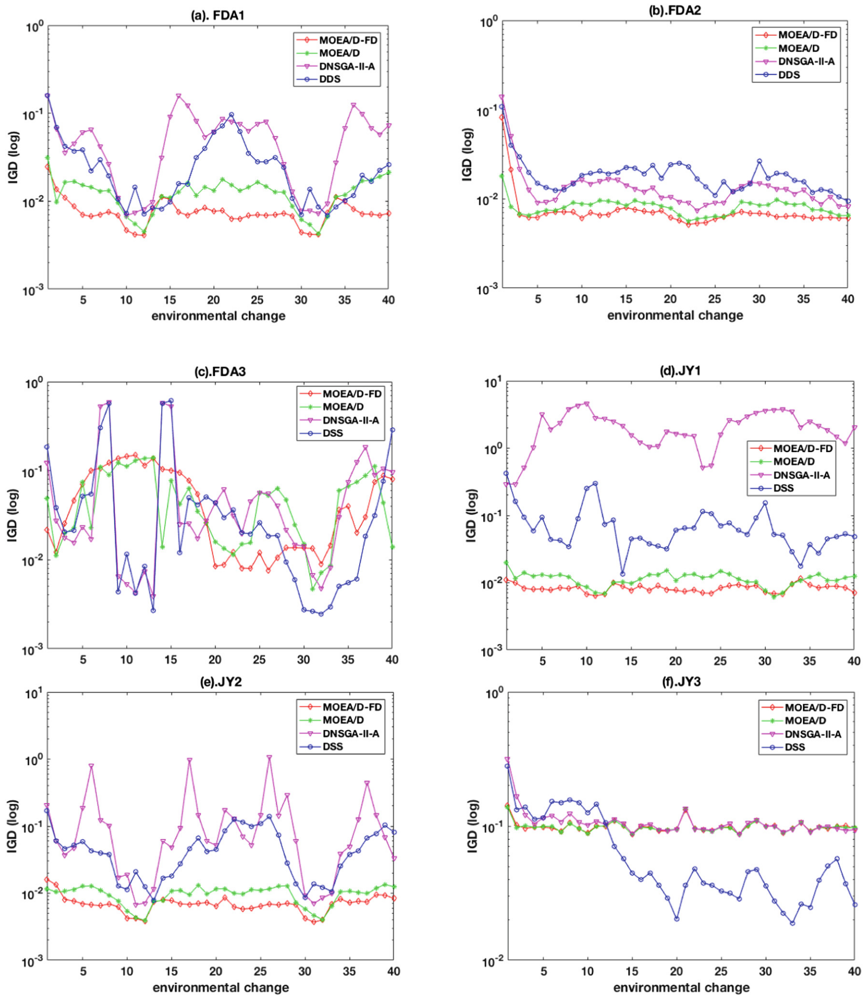

Fig. 1. Tracking the IGD values with the environmental change obtained by four algorithms for all test problems with $n_t = 10, \tau_t = 10$.

nonlinear linkages between decision variables. Similarly, MOEA/D-FD performed best on most test instances, while it was only slightly worse than the reinitialized MOEA/D on JY3 with $n_t = 5, \tau_t = 5$ and $n_t = 10, \tau_t = 5$. The fast change frequency $(\tau_t = 5)$ and the serve change severity $(n_t = 5)$ leaded to hard tracking the changing POS and/or POF for all algorithms on the JY test instances.

Figure 1(d)–(f) show the tracking of the IGD values with the environmental change obtained by four algorithms for JY1–JY3 with $n_t = 10, \tau_t = 10$, which is obtained from the average value over 30 runs. MOEA/D-FD and the reinitialized MOEA/D performed well and stably on JY1–JY3, in which their tracking abilities were not affected by the environmental changes too much. However, DNSGA-II-A and DDS

Table 2. Mean values and standard deviations of MIGD metric obtained by four algorithms on JY1–JY3

Problem	(n_t, τ_t)	MOEA/D-FD	MOEA/D	DNSGA-II-A	DSS
JY1	(5,5)	**0.0178 ± 0.0007**	0.0346 ± 0.0003(+)	2.4965 ± 0.1513(+)	0.1507 ± 0.0050(+)
	(5,10)	**0.0099 ± 0.0001**	0.0127 ± 0.0001(+)	2.3222 ± 0.0551(+)	0.0975 ± 0.0002(+)
	(5,20)	**0.0076 ± 0.0001**	0.0080 ± 0.0001(+)	2.1691 ± 0.0641(+)	0.0740 ± 0.0035(+)
	(10,5)	**0.0118 ± 0.0002**	0.0225 ± 0.0009(+)	2.3450 ± 0.1215(+)	0.1228 ± 0.0015(+)
	(10,10)	**0.0085 ± 0.0001**	0.0109 ± 0.0001(+)	2.1358 ± 0.1850(+)	0.0940 ± 0.0026(+)
	(10,20)	**0.0072 ± 0.0001**	0.0079 ± 0.0001(+)	1.8928 ± 0.1526(+)	0.0734 ± 0.0066(+)
JY2	(5,5)	**0.0154 ± 0.0002**	0.0032 ± 0.0001(+)	0.3917 ± 0.0169(+)	0.1075 ± 0.0028(+)
	(5,10)	**0.0086 ± 0.0003**	0.0117 ± 0.0004(+)	0.1773 ± 0.0201(+)	0.0693 ± 0.0009(+)
	(5,20)	**0.0060 ± 0.0001**	0.0065 ± 0.0001(+)	0.1589 ± 0.0058(+)	0.0383 ± 0.0040(+)
	(10,5)	**0.0115 ± 0.0011**	0.0208 ± 0.0007(+)	0.2199 ± 0.0053(+)	0.0852 ± 0.0014(+)
	(10,10)	**0.0072 ± 0.0001**	0.0098 ± 0.0001(+)	0.1297 ± 0.0052(+)	0.0548 ± 0.0012(+)
	(10,20)	**0.0056 ± 0.0001**	0.0062 ± 0.0001(+)	0.1149 ± 0.0118(+)	0.0283 ± 0.0001(+)
JY3	(5,5)	0.1054 ± 0.0010	**0.1027 ± 0.0001**(−)	0.1398 ± 0.0051(+)	0.1592 ± 0.0161(+)
	(5,10)	**0.0997 ± 0.0006**	0.1008 ± 0.0011(+)	0.1111 ± 0.0009(+)	0.1188 ± 0.0076(+)
	(5,20)	**0.0985 ± 0.0001**	0.0986 ± 0.0001(+)	0.1033 ± 0.0012(+)	0.1097 ± 0.0026(+)
	(10,5)	0.1037 ± 0.0015	**0.1021 ± 0.0009**(−)	0.1371 ± 0.0035(+)	0.1522 ± 0.0230(+)
	(10,10)	**0.0982 ± 0.0008**	0.0986 ± 0.0003(+)	0.1077 ± 0.0019(+)	0.1359 ± 0.0083(+)
	(10,20)	**0.0936 ± 0.0043**	0.0977 ± 0.0001(+)	0.1034 ± 0.0010(+)	0.0811 ± 0.0036(+)

fluctuated a lot with environmental changes on JY1–JY2. DSS performed unstably on JY3 in spite of good tracking on later time windows. DNSGA-II-A had the similar tracking ability to MOEA/D-FD and MOEA/D on JY3.

5 Conclusion

In this paper, a novel algorithm based on MOEA/D was proposed to solve dynamic multiobjective optimization problems. A simple yet effective first-order difference model was incorporated into MOEA/D algorithm, in which the model was used to predict the new location of the centroid of the POS based on its historical locations. The performance of the proposed algorithm was validated using six benchmark problems. The results show that MOEA/D-FD is competitive with other state-of-the-art dynamic algorithms in terms of tracking the POS and/or POF. The proposed prediction strategy for solving dynamic multiobjective optimization problems illustrates that a proper prediction approach is indeed effective to enhance the tracking ability of MOEAs in dynamic environments.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China under Grant 61573258, in part by the National High-Technology Research and Development Program (863 Program) of China under Grant 2013AA103006-2, and in part by U. S. National Science Foundation's BEACON Center for the Study of Evolution in Action, funded under Cooperative Agreement No. DBI-0939454.

References

1. Deb, K., Rao, N.U.B., Karthik, S.: Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 803–817. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70928-2_60
2. Jiang, S., Yang, S.: A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization. IEEE Trans. Evol. Comput. **21**, 65–82 (2017)
3. Azzouz, R., Bechikh, S., Ben Said, L.: Multi-objective optimization with dynamic constraints and objectives : new challenges for evolutionary algorithms (2015)
4. Zhou, A., Jin, Y., Zhang, Q.: A population prediction strategy for evolutionary dynamic multiobjective optimization. IEEE Trans. Cybern. **44**, 40–53 (2014)
5. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 832–846. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70928-2_62
6. Shaaban, S., Haluk, T.R.: A memory based NSGA2 algorithm for DMOP. In: LNCS, pp. 296–310 (2016)
7. Muruganantham, A., Tan, K.C., Vadakkepat, P.: Evolutionary dynamic multiobjective optimization via Kalman filter prediction. IEEE Trans. Cybern. **46**, 2862–2873 (2016)
8. Chen, X., Zhang, D., Zeng, X.: A stable matching-based selection and memory enhanced MOEA/D for evolutionary dynamic multiobjective optimization. In: Proceedings of the International Conference on Tools with Artificial Intelligence, ICTAI 2016, pp. 478–485 (2016)
9. David, H.I.W.: Dynamic multi-objective optimization evolutionary algorithms: a forward-looking approach. In: GECCO 2006, pp. 456–459 (2006)
10. Wu, Y., Jin, Y., Liu, X.: A directed search strategy for evolutionary dynamic multiobjective optimization. Soft. Comput. **19**, 3221–3235 (2015)
11. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**, 712–731 (2007)
12. Liu, M., Zeng, W.-H.: Memory enhanced dynamic multi-objective evolutionary algorithm based on decomposition. J. Softw. **24**, 1571–1588 (2014)
13. Wang, Y., Li, B.: Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment. In: CEC 2009, pp. 630–637 (2009)
14. Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol. Comput. **6**, 1–24 (2012)
15. Hatzakis, I., Wallace, D.: Topology of anticipatory populations for evolutionary dynamic multi-objective optimization. In: 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, pp. 1–10 (2006)
16. Muruganantham, A., Zhao, Y., Gee, S.B., Qiu, X., Tan, K.C.: Dynamic multiobjective optimization using evolutionary algorithm with Kalman filter. Procedia Comput. Sci. **24**, 66–75 (2013)
17. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comput. **13**, 284–302 (2009)
18. Li, H., Zhang, Q., Deng, J.: Biased multiobjective optimization and decomposition algorithm. IEEE Trans. Cybern. **47**, 52–66 (2017)
19. Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problem: test cases, approximation, and applications. IEEE Trans. Evol. Comput. **8**, 425–442 (2004)
20. Jiang, S., Yang, S.: Evolutionary dynamic multiobjective optimization: benchmarks and algorithm comparisons. IEEE Trans. Cybern. **47**, 198–211 (2017)