Applied Mathematics and Computation 205 (2008) 6–18

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

An evolutionary algorithm for dynamic multi-objective optimization

Yuping Wang [a,*], Chuangyin Dang [b]

[a] Faculty of Computer Science and Technology, Xidian University, Xi'an, 710071 Shaanxi, China
[b] Department of Manufacturing Engineering, and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong SAR, China

ARTICLE INFO

Keywords:
Evolutionary algorithm
Dynamic multi-objective programming
Uniform design

ABSTRACT

In this paper, the dynamic multi-objective optimization problem (DMOP) is first approximated by a series of static multi-objective optimization problems (SMOPs) by dividing the time period into several equal subperiods. In each subperiod, the dynamic multi-objective optimization problem is seen as a static multi-objective optimization problem by taking the time parameter fixed. Then, to decrease the amount of computation and efficiently solve the static problems, each static multi-objective optimization problem is transformed into a two-objective optimization problem based on two re-defined objectives. Finally, a new crossover operator and mutation operator adapting to the environment changing are designed. Based on these techniques, a new evolutionary algorithm is proposed. The simulation results indicate that the proposed algorithm can effectively track the varying Pareto fronts with time.

1. Introduction

Many real world optimization problems exist in uncertain environments. Generally, uncertainties in these optimization problems can be classified into four categories [1]: (1) the fitness function is noisy; (2) the design variables and/or the environmental parameters may change after optimization, and the quality of the obtained optimal solution should be robust enough for any environmental changes or deviations from the optimal point; (3) the fitness function is often approximated, which means that the fitness function suffers from approximation errors; and (4) the optimum of the problem changes over time and, thus, the optimizer should be able to track the optimum continuously. In this paper, we will discuss the fourth problem. This kind of optimization problem is called dynamic optimization problem (DOP). DOPs include dynamic single-objective optimization problems (DSOPs) and dynamic multi-objective optimization problems (DMOPs). When DSOPs are considered, many studies are available in the literature (e.g.[2–5]). However, when DMOPs are concerned, only a few studies are available in the literature [6–11]. For DOPs, major modifications to the operators for the static optimization problems are required for a prompt reaction to time-dependent changing.

Generally speaking, there are three main approaches for the multi-objective optimization in the dynamic environments: diversity control approach, memory-based approach and multi-population approach.

Diversity control is a common topic in evolutionary algorithms (EAs) in general. To control the diversity in a dynamic environment, one can increase the diversity when a change happens, or maintain a high diversity all over the evolutionary run. The hyper-mutation method [12] and the variable local search technique [13] are the examples of the former, and the random immigrants [14] and the thermodynamical genetic algorithm [15] are the examples of the latter. Memory-based approaches employ an extra memory that implicitly or explicitly stores the useful information to guide future search. Implicit memory usually uses redundant representations to store information during the evolutionary process [16]. In the explicit

* Corresponding author.
E-mail address: ywang@xidian.edu.cn (Y. Wang).

0096-3003/$ - see front matter © 2008 Published by Elsevier Inc.
doi:10.1016/j.amc.2008.05.151

memory, the specific information get stored and retrieved when needed by the evolutionary mechanism [17]. Multi-population approach uses subpopulations to search the optima in different areas. The subpopulations are maintained and each becomes specialized on the specific relevant search space [2]. However, when the environment gradually and continuously changes with time, the existing algorithms are very difficult to react promptly to the environment change, i.e., it is impossible to obtain Pareto optimal solutions or high quality nondominated solutions all the time, because no algorithm can react fast enough.

To overcome this drawback and solve the problem in a practical way, one reasonable method is to obtain the Pareto optimal solutions or high quality nondominated solutions at some representative instant of time instead of all the time. In this paper, we design the new algorithm motivated by this concept. Firstly, the dynamic multi-objective optimization problem (DMOP) is approximated by a series of static multi-objective optimization problems (SMOPs) by dividing the time period into several equal subperiods. In each subperiod, the dynamic multi-objective optimization problem is seen as a static multi-objective optimization problem by taking the time parameter fixed. Secondly, to decrease the amount of computation and efficiently solve the static problems, each static multi-objective optimization problem is transformed into a two-objective optimization problem based on two re-defined objectives. Finally, a new crossover operator and mutation operator adapting to the environment changing are designed. Based on these, a new evolutionary algorithm is then proposed. The simulation results indicate that the proposed algorithm is effective.

2. Transformation of the problem

In this section, the problem is first transformed to optimizing simultaneously several static multi-objective optimization problems, then each transformed static multi-objective optimization problem is further transformed to a two-objective optimization problem, in which one objective is to optimize the distribution of the nondominated solutions (will be defined in Definition 2.1.1 later) found so far, and another is to optimize the quality of the nondominated solutions found so far. The details are as follows.

2.1. Transformation of the original problem into several static optimization problems

We consider the following dynamic multi-objective optimization problems:

$$\begin{cases} \min & f(x,t) = (f_1(x,t), f_2(x,t), \ldots, f_m(x,t)) \\ s.t & g_i(x,t) \leqslant 0, i = 1,2,\ldots,p \\ & x \in [L,U], t \in [t_0, t_s] \end{cases}, \tag{1}$$

where $t \in [t_0, t_x] \subset R$, $x \in R^n$ is called decision vector, $g_i(x,t)$ is called constraint condition depending on the time variable t.

$\Omega(t) = \{x | g_i(x,t) \leqslant 0,\ i = 1,2,\ldots,p\}$ is the decision vector space, and set $[L,U] = \{x = (x_1,x_2,\ldots,x_n) | l_i \leqslant x_i \leqslant u_i,\ i = 1,2,\ldots,n\}$ is the search space with respect to x, where $L = (l_1,l_2,\ldots,l_n)$, and $U = (u_1,u_2,\ldots,u_n)$.

Suppose that $f_j(x,t)$, $j = 1,2,\ldots,M$ and $g_j(x,t)$, $j = 1,2,\ldots,p$ are continuous with respect to t.

To simplify the problem, the time period $t \in [t_0,t_s]$ is equally divided into several subperiods $[t_{i-1},t_i]$, $i = 1,2,\ldots,s$. Because the related functions $f_j(x,t)$ and $g_i(x,t)$ are continuous with respect to t, they can be seen as unchanged with respect to time t in each subperiod when the length of this subperiod is sufficiently short. Thus, the original problem can be approximately seen as a static multi-objective optimization problem in each subperiod. As a result, the original problem can be approximately seen as several static multi-objective optimization problems as follows.

For the jth subperiod $[t_{j-1},t_j]$, define the following static multi-objective optimization problem

$$\begin{cases} \min & f(x,t_{j-1}) = (f_1(x,t_{j-1}), f_2(x,t_{j-1}), \ldots, f_M(x,t_{j-1})) \\ s.t & g_i(x,t_{j-1}) \leqslant 0, i = 1,2,\ldots,p \\ & x \in [L,U] \end{cases}, \tag{2}$$

for $j = 1,2,\ldots,s$, where t, is taken as a fixed value t_{j-1}. By using the aforementioned approximation, the original problem is transformed to optimizing simultaneously s multi-objective optimization problems in (2). Let $X(j)$ denote the feasible set of problem in (2).

Definition 2.1.1. Let $x^* \in X(j)$ if there does not exist $x \in X(j)$ such that $f_i(x,t_{j-1}) \leqslant f_i(x^*,t_{j-1})$ for $\forall i$ and $f_k(x,t_{j-1}) < f_k(x^*,t_{j-1})$ for some k, x^* is then called the Pareto optimal solution or nondominated solution in $X(j)$. The set of all Pareto optimal solutions in $X(j)$, denoted as $E(f,X(j))$, is called Pareto optimal solution set in $X(j)$, and the set $\{f(x) | x \in E(f,X(j))\}$ is called the Pareto front of $f(x)$ in $X(j)$.

2.2. Transformation of each static problem into a two-objective optimization problem

For notation convenience, denote $\chi = (x,t_{j-1})$, $f_i(\chi) = f_i(x,t_{j-1})$, for $i = 1,2,\ldots,M$ for fixed t_{j-1}, $\Omega = X(j)$ the feasible solution set, and $F(\chi) = (f_1(\chi), f_2(\chi), \ldots, f_M(\chi))$ in problem (2). Also denote $D = \{F(\chi) | \chi \in \Omega\}$.

Note that the main objective for solving a multi-objective optimization problem is to get a set of solutions with well distribution and high quality. To realize this objective and make the problem easy, our idea is to introduce a new problem with only two-objective functions. One objective function is responsible for optimizing distribution of the solutions, known as U-measure function, and another is for optimizing the quality of the solutions, known as rank-based function.

To introduce the U-measure function, we need to introduce the definition of neighbors of a point in an objective space.

Definition 2.2.1. For each point $F(\chi) = (f_1(\chi), f_2(\chi), \ldots, f_M(\chi)) \in D$ in objective space, let $f_k^{\min} = \min\{f_k(\chi) | \chi \in \Omega\}$ and $f_k^{\max} = \max\{f_k(\chi) | \chi \in \Omega\}$ for $k = 1, 2, \ldots, M$. The kth axis neighbors of $F(\chi)$ are defined by

$$Q_{2k-1} = \begin{cases} F(\chi), & \text{if} \quad f_k(\chi) = f_k^{\min}, \\ V \in D_k^L, & \text{otherwise}, \end{cases}$$

$$Q_{2k} = \begin{cases} F(\chi), & \text{if} \quad f_k(\chi) = f_k^{\max}, \\ V \in D_k^R, & \text{otherwise}, \end{cases}$$

$k = 1, 2, \ldots, s$, where $(V = V_1, V_2, \ldots, V_M)$,

$$D_k^L = \{V \in D | d(V, F(\chi)) = \min\{d(V, F(\chi)) : V_k < f_k(\chi)\},$$
$$D_k^R = \{V \in D | d(V, F(\chi)) = \min\{d(V, F(\chi)) : V_k > f_k(\chi)\},$$

and $d(V, F(\chi))$ is the distance between V and $F(\chi)$.

2.2.1. The objective based on U-measure

U-measure is to measure the uniformity and the extent of the distribution of a set of Pareto solutions found by an algorithm. This metric was proposed in [19]. There are two cases for this metric.

Case 1: The problem with two objectives $F = (f_1, f_2)$.

Algorithm 2.1. (U-measure for the case with two objectives $F = (f_1, f_2)$)

1. Let $f_k^{\min} = \min_{\chi \in \Omega}, f_k^{\max} = \max_{\chi \in \Omega} f_k(\chi)$, $k = 1, 2$, where Ω is the decision vector space.
2. Suppose that $T = \{F_1, F_2, \ldots, F_N\}$ is the set of Pareto solutions in the objective space obtained by Algorithm A. Add two reference points $F_0 = (f_1^{\min}, f_2^{\max})$ and $F_{N+1} = (f_1^{\max}, f_2^{\min})$ to the set T to get a set $P = \{F_0, F_1, \ldots, F_{N+1}\}$.
3. Sorting the points in P according to the ascending order of the first components of these points. Without loss of generality, suppose that the sequence is also $F_0, F_1, \ldots, F_{N+1}$.
4. Calculate the distances $d_{i,i+1}$ of every pair of adjacent points F_i and F_{i+1} for $i = 0, 1, \ldots, N$. Calculate $d = \frac{1}{N-1}\sum_{i=0}^{N} d_{i,j+1}$, and let $d_{01} = d_{01} + d$, $d_{N,N+1} = d_{N,N+1} + d$. Calculate the mean distance and standard deviation of all distances d_{i+1} for $i = 0, 1, \ldots, N$ as follows:

$$d_{\text{mean}} = \frac{1}{N+1}\sum_{i=0}^{N} d_{i,j+1},$$

$$U_m(T) = d_{\text{std}} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(d_{i,i+1} - d_{\text{mean}})^2},$$

$U_m(T)$ can be used as a metric for the uniformity and spread of the Pareto front T. The smaller the $U_m(T)$ is, the more uniformity and wider spread of the points in the set T will be.

Case 2: The problem with M objectives $F = (f_1, f_2, \ldots, f_M)$ and $M > 2$.

Let $T = \{(f_1^i, f_2^i, \ldots, f_M^i)\}_{i=1,2,\ldots,m}$ be a set of Pareto solutions in the objective space found by Algorithm A. For each $r \in \{1, 2, \ldots, M\}$, classify T into some subsets $T_1^r, T_2^r, \ldots, T_k^r$ according to the rth objective values of the points in T such that

- all points in each T_i^r have the same rth objective values ($i = 1, 2, \ldots, k$) and,
- for any two points $Z = (z_l, z_2, \ldots, z_M) \in T_i^r$ and $Y = (y_l, y_2, \ldots, y_M) \in T_s^r$, if $i < s$ then $z_r < y_r$. For each point $g \in T$, its rth axis neighbors are Q_{2r-1} and Q_{2r} (defined by Definition 2.2.1) for $r = 1, 2, \ldots, M$. Their distances to g are denoted as d_{2r-1} and d_{2r}, respectively.

Algorithm 2.2. (U-measure for the case with M objectives)

1. Let $\bar{d}_r = \frac{1}{2|T-\{T_l^r, T_k^r\}|}\sum_{g \in T-\{T_l^r, T_k^r\}}(d_{2r-1} + d_{2r}), r \in \{1, 2, \ldots, M\}$. For any $g \in T_i^r \subset T$, $r \in \{1, 2, \ldots, M\}$, let $d_{2r-1} = d_{2r-1} + \bar{d}$ if $i = 1$ and let $d_{2r} = d_{2r} + \bar{d}$ if $i = k$, where $|T - \{T_1^r, T_k^r\}|$ represents the number of the points in $T - \{T_1^r, T_k^r\}$.

2. Let

$$d_{\text{mean}} = \frac{1}{2MN}\sum_{g\in T}\sum_{r=1}^{M}(d_{2r-1} - d_{2r}),$$

$$U_m(T) = d_{\text{std}} = \sqrt{\frac{1}{2MN-1}\sum_{g\in T}\sum_{r=1}^{2M}(d_r - d_{\text{mean}})^2},$$

3. $U_m(T)$ can be used as a measure of the uniformity and spread for the points in T. The smaller the $U_m(T)$ is, the more uniformity and wider spread of the points in T will be. The metrics in Algorithms 2.1 and 2.2 are called U-measure.

U-measure can be defined as one objective, and gradually minimizing $U_m(T)$ will get better and better distributed Pareto optimal solutions.

2.2.2. The objective based on rank

For each subperiod $[t_{j-1},t_j]$, the time variable t is seen as unchanged and taken as a fixed value $t = t_j-1$. Suppose that the kth generation population $P(k)$ is made of individuals $x_1(k),x_2(k),\ldots,x_N(k)$. Let $n_j(x)$ denote the number of individuals in $P(k)$ dominating x, then $r_j(x) = 1 + n_j(x)$ is called the rank of individual x. The smaller the value of $r_j(x)$ is, the better the quality of the solution x will be. $r_j(x)$ is called the rank of individual x, and it can be as another objective.

2.2.3. Models for two-objective optimization problems

Based on the objectives defined above, for each subperiod $[t_{j-1},t_j]$, the following two-objective optimization problem can be introduced as

$$\begin{cases} \min & f(x) = (U_m(T), r_j(x)) \\ s.t & g_i(x,t_{j-1}) \leqslant 0, \quad i = 1,2,\ldots,p \\ & x \in [L,U] \end{cases} \tag{3}$$

for $j = 1,2,\ldots,s$.

3. The proposed evolutionary algorithm

3.1. Crossover operator using the uniform design

In this subsection, we first briefly introduce a uniform design method, by which a crossover operator was designed in [20] and has been proved efficient by experiments. However, at the beginning stages of the evolution in each subtime period, if two parents are too close to each other, the offspring generated by crossover in [20] may be crowded in a small region and the diversity of the population may not be good. To improve the shortcoming, we avoid the execution of the crossover of two parents which are close at the beginning stage of the evolution, and introduce an improved crossover operator using the uniform design.

The main objective of a uniform design is to sample a small set of points from a given closed and bounded set $G \subset R^M$ such that the sampled points are uniformly scattered on G. In the following, we consider only a specific case of G and describe the main features of a uniform design. For more details, we refer to [21].

Let $l = (l_1,l_2,\ldots,l_I)$ and $u = (u_l,u_2,\ldots,u_I)$ be two points of R^I, where $l_i \leqslant u_i, i = 1,2,\ldots,I$. Denote

$$[l,u] = \{\theta_1,\theta_2,\ldots,\theta_I)|)l_i \leqslant \theta_i \leqslant u_i = 1,2,\ldots,I\} \tag{4}$$

Finding a set of exactly uniformly scattered points on $[l,u]$ is, in general, very difficult, but there are some efficient methods to look for a set of well approximately uniformly scattered points on $[l,u]$. One of the simple and efficient methods is the good-lattice-point method (GLP) [21], which generates a set of q uniformly scattered points on

$$[0,1]^I = \{(x_1,x_2,\ldots,x_I)|0 \leqslant x_i \leqslant 1, i = 1,2,\ldots,I\},$$

denoted by $C(I,q)$, and a set of q uniformly scattered points on $[l,u]$, denoted by $O(I,q)$, respectively, as follows:

- Given q and I, determine a number σ (Table 1 lists the values of σ for different values of q and I).
- Generate a $q \times I$ integer matrix called uniform array denoted by

$$G(I,q) = [G_{ij}]_{q\times I}, \tag{5}$$

where $G_{ij} = (i\sigma^{j-1} \bmod q) + 1$, $i = 1,2,\ldots,q$, and $j = 1,2,\ldots,I$.
- Each row of matrix $G(I,q)$ defines a point $C_i = (c_{i1},c_{i2},\ldots,c_{iI})$ of $C(I,q)$ and a point $O^i = (o_1^i, o_2^i, \ldots, o_I^i$ of $O(I,q)$ by

$$c_{ij} = \frac{2G_{ij}-1}{2q}, \quad i = 1,2,\ldots,q, \quad j = 1,2,\ldots,I$$

Table 1
The corresponding values of parameter σ for different values of q and I

q	l	σ
5	2–4	2
7	2–6	3
11	2–10	7
13	2	5
	3	4
	4–12	6
17	2–16	10
19	2–3	8
	4–18	14
23	2, 13–14, 20–22	7
	8–12	15
	3–7, 15–19	17
29	2	12
	3	9
	4–7	16
	8–12, 16–24	8
	13–15	14
	25–28	18
31	2, 5–12, 20–30	12
	3–4, 13–19	22

and

$$o_j^i = l_j + c_{ij}(u_j - l_j), \quad i = 1,2,\ldots,q, \quad j = 1,2,\ldots,I,$$

respectively. $C(I,q)$ and $O(I,q)$ are given by

$$C(I,q) = \{C_i | i = 1,2,\ldots,q\}, \tag{6}$$

$$O(I,q) = \{O^i | i = 1,2,\ldots,q\}. \tag{7}$$

Now we use the uniform design method to construct a crossover operator. The offspring generated by this operator is uniformly scattered in a region, and thus can effectively exploit the search space. The main idea is as follows. For two parents $Y = (y_1, y_2, \ldots, y_n)$ and $X = (x_1, x_2, \ldots, x_n)$, we define two vectors,

$$l = (l_1, l_2, \ldots, l_n) \text{ and } u = (u_1, u_2, \ldots, u_n), \tag{8}$$

where $l_i = \min\{x_i, y_i\}$ and $u_t = \max\{x_i, y_i\}$ for $i = 1,2,\ldots,n$. These two vectors define a hyper-rectangle,

$$[l,u] = \{(z_1, z_2, \ldots, z_n) | l_i \leqslant z_i \leqslant u_i, i = 1,2,\ldots,n\}. \tag{9}$$

Choose a proper integer q_1. The uniform design method is then used to generate $q = q_1$ points uniformly distributed on $[l, u]$, and these q_1 points can be regarded as the offspring of two parents X and Y.

Note that the parameters I and q in the uniform design should satisfy $I \leqslant q - 1$ [21]. Thus, given I and $q = q_1$, the set of q_1 offspring, denoted by

$$O(n,q_1) = \{O^k | k = 1,2,\ldots,q_1\} = \{(o_1^k, o_2^k, \ldots, o_n^k) | k = 1,2,\ldots,q_1\} \tag{10}$$

can be generated according to two cases: $I \leqslant q_1 - 1$ and $I > q_1 - 1$. The detail is given by the following Algorithm 3.1.

Algorithm 3.1. (Crossover operator)

1. Given two positive constants C_1 and $d_{\min}$. For two chosen parents X and Y, if $T_{\text{initial}} \leqslant C_1$ and $\|X - Y\| \leqslant d_{\min}$ where T_{initial} is the length of time, randomly select another parent replacing Y, go to step 1; otherwise, go to step 2.
2. Define two vectors l and u by the formula (8) and a hyper-rectangle $[l,u]$ by the formula (9). Choose a proper prime number q_1.
3. If $n \leqslant q_1 - l$, take $I = n$ in the formulas (4)–(7). Then the kth offspring $O_k = (o_1^k, o_2^k, \ldots, o_n^k)$ can be generated by the formulas (4)–(7) with $I = n$ for $k = 1,2,\ldots,q$.
4. If $n > q_1 - 1$, take $I = q_1 - 1$ and randomly divide l and u into I blocks of subvectors, respectively, in the following way:

$$\begin{cases} l = (A_l^1, A^2, \ldots, A^{q_1-1}) \\ u = (B^1, B^2, \ldots, B^{q_1-1}) \end{cases}, \tag{11}$$

where A^j and B^j are subvectors of l and u in the same dimension. Then the kth offspring

$$O^k = (O_1^k, O_2^k, \ldots, O_{q_1-1}^k) \tag{12}$$

can be generated by

$$O_j^k = A^j + \frac{2G_{kj} - 1}{2q_1}(B^j - A^j), j = 1, 2, \ldots, q_1 - 1 \tag{13}$$

for $k = 1,2,\ldots,q_1$, where $G(q_1-1, q_1) = [G_{kj}]_{q_1 \times (q1-1)}$ is defined by (7) with $I = q_l-l$.

3.2. Selection scheme for the next generation population

In order to search for Pareto optimal solutions in different regions for problems in different time subperiods in the next generation evolution, we can select the next generation population by considering all problems in (3) as a whole, and select almost the same number of solutions for each problem in (3). The detail is as follows.

3.2.1. Selection scheme

- Select the nondominated solutions of each problem in (3) from the current population and offspring generated by crossover and mutation. The set of the nondominated solutions for the problem in the jth time subperiod in (3) is denoted by $S_n(j)$, $j = 1,2,\ldots,s$.
- Compute the mean number N_{pop}/s of the solutions for these problems in (3), where N_{pop} is the population size.
- For the problems in (3) with more than N_{pop}/s nondominated solutions, randomly delete the more than N_{pop}/s nondominated solutions for each of these problems, so that the number of nondominated solutions for each of these problems is $[N_{\text{pop}}/s]$.
- For the problems in (3) with fewer than N_{pop}/s nondominated solutions, for example, for the problem in the jth time subperiod, $S_n(j)$ contain fewer than N_{pop}/s nondominated solutions, randomly select $N_{\text{pop}}/s-|S_n(j)|$ solutions from the current population and offspring generated by crossover and mutation to put into $S_n(j)$, such that the total number of the selected solutions in steps 3 and 4 is N_{pop}/s.

3.3. The proposed algorithm

Algorithm 3.2

1. Choose the proper parameter q_1, population size N_{pop}, the expected number N_1 of nondominated solutions in each time subperiod, crossover probability p_c and mutation probability p_m. Generate an initial population randomly.
2. For each time subperiod $[t_{j-1},t_j]$ determine the set of the nondominated solutions in objective space for problem (3), denoted by $F_p(j)$ for $j = 1,2,\ldots,s$.
3. Select individuals for crossover according to the current crossover probability p_c. Randomly match every two selected parents and execute crossover for each pair of matched parents by Algorithm 3.1.
4. Select individuals from the offspring of crossover according to the mutation probability p_m. For each selected offspring, say $X = (x_1,x_2,\ldots,x_n)$, randomly change it into another individual $X = X + \Delta X$, where each component of X is generated by a random number generator obeying Gaussian distribution with mean 0 and deviation $\sigma = 1$ and all components are independent.
5. Update the sets of nondominated solutions $F_p(1),\ldots,F_p(s)$ for all time subperiods, respectively, so that the number of solutions in each set is no more than N_1 and the distribution of these solutions is as uniform as possible.
6. Select the next generation population by using the selection scheme in the previous subsection.
7. If stopping criterion is not met, go to step 3; otherwise, stop.

4. Simulation results

4.1. Test problems

In this subsection, we use three test problems for dynamic multi-objective optimization problems which are similar to the problems FDA1 to FDA3 in ([6]), respectively. The only difference is that we divide the time period into $s = 10$ time subperiods and take the fixed value for t in each subperiod instead of using $t = \frac{1}{n_t}[\frac{\tau}{\tau_T}]$.

Test problem F_1:

where $X = (x_1,x_2,\ldots,x_n)$

$$f_1(x_1) = x_1$$

is a function of the first decision variable only,

$$g = 1 + \sum_{i=2}^{n} (x_i - G(t))^2$$

is a function of the remaining (n−1) variables,

$$h = 1 - \sqrt{\frac{f_1}{g}}$$

is a function of two variables,

$$G(t) = \sin(0.5\pi t),$$

$n = 20$, $x_1 = [0,1]$ and $x_1 \in [-1,1]$, $i \geqslant 2$.

Since $G(t)$ changes with time, the Pareto optimal solutions in a variable space change with time as well. However, the resulting Pareto optimal fronts do not change. In order to track the dynamic Pareto optimal fronts, x_i has to converge to the new $G(t)$ value every time there is a change for each $2 \leqslant i \leqslant n$.

Test problem F_2:

It is same as test problem F_1 except for the formats of functions g and h. g and h are defined as follows:

$$g(x_2, \ldots, x_n) = 1 + \sum_{i=2}^{n} x_i^2,$$

$$h(y_2, \ldots, y_n, f_1, g) = 1 - \left(\frac{f_1}{g}\right)^{\left(H(t) + \sum_{i=2}^{n} (y_i - H(t))^2\right)^{-1}},$$

$$H(t) = 0.75 + 0.7\sin(0.5\pi t),$$

where $n = 16$, $X = (x_l, x_2, \ldots, x_n, y_2, \ldots, y_n)$, $x_1 \in [0,1]$ and $x_i, y_i \in [-1,1]$, $i \geqslant 2$.

For this test problem, the Pareto optimal fronts swing from a convex shape to a nonconvex shape due to the change in the $H(t)$ function, while the corresponding Pareto optimal solutions in the variable space remain unchanged.

Test problem F_3:

It is same as test problem F_1 except for the formats of functions f_1 and g. f_1 and g are defined as follows:

$$f_1(x_1, \ldots, x_5) = \sum_{i=1}^{5} x_i^{Q(t)},$$

$$g(y_1, \ldots, y_{25}) = 1 + G(t) + \sum_{i=1}^{25} (y_1 - G(t))^2,$$

$$G(t) = |\sin(0.5\pi t)|,$$

$$Q(t) = 10^{2\sin(0.5\pi t)},$$

where $x_1, \ldots, x_5 \in [0,1]$ and $y_i \in [-1,1]$ $i = 1, 2, \ldots, 25$.

For this test problem, both the Pareto optimal fronts and the Pareto optimal solutions in the variable space change with time, and the density of the solutions on the Pareto optimal fronts varies with time. The task of an algorithm for this problem would be to find a widely distributed set of solutions each time if there is a change in t.

4.2. Results and comparisons

In the simulation, the time period considered is taken from $t_0 = 0$ to $t_s = 10$ min, and it is equally divided into $s = 10$ subperiods. The length of each subperiod is 1 min. We execute the proposed algorithm 30 times for each test problem, and record the approximated Pareto fronts found by the proposed algorithm in 10 time subperiods for each problem in each run, respectively. To make the results intuitively, we represent these results graphically for each test problem in a typical run. Figs. 1–3 depict the approximated Pareto fronts found by the proposed algorithm in 10 time periods for these test problems in a typical run, respectively.

Because the method to handle the time in the proposed algorithm is different from that in the existing algorithms, it is difficult to make a fair comparison with the existing algorithms directly. To make a fair comparison, we can adopt the same method of handling time for both the compared algorithm and the proposed algorithm. We choose one of the best and well-known algorithms, real-coded NSGA-II [22], for comparison. In the simulation, we execute the real-coded NSGA-II for these problems in a similar way (i.e., the time period considered is taken from 0 to 10 min and is equally divided into 10 subperiods). In each subperiod, the time is taken as the fixed value and compare the quality of the solutions found by the proposed algorithm and by real-coded NSGA-II by using coverage metric (C-metric) [18] and uniformity measure (U-measure) [19].

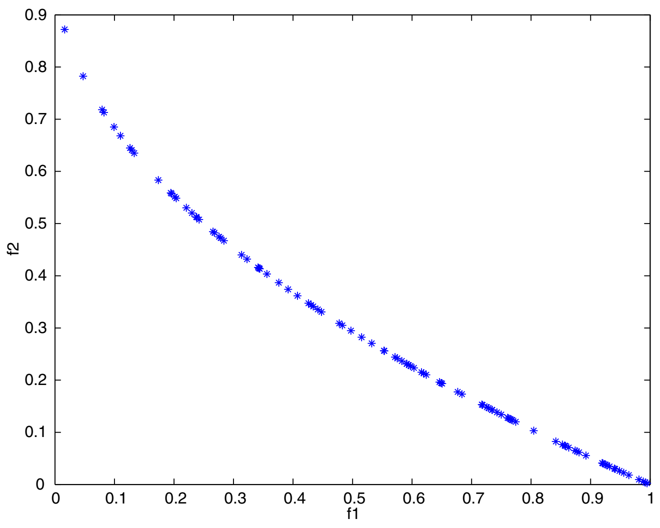

Fig. 1. The Pareto fronts found by the proposed algorithm for FDA1 in a typical run.

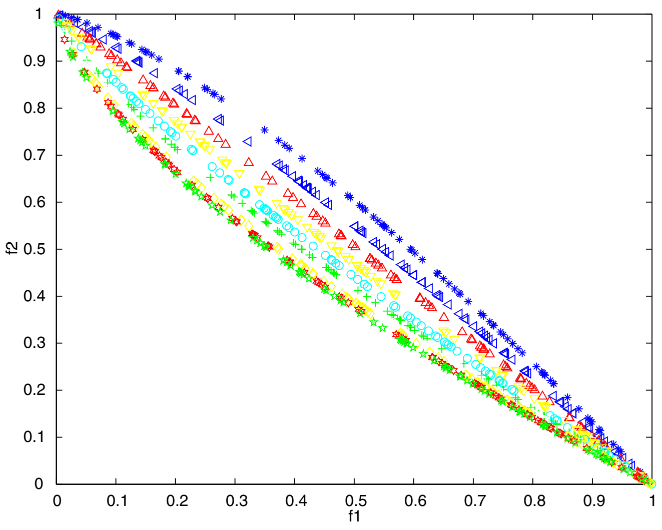

Fig. 2. The Pareto fronts found by the proposed algorithm for FDA2 in a typical run.

4.2.1. Parameter values

In the simulation, we take the following parameter values:

For the proposed algorithm:

- Then number of offspring generated by each pair of parents: $q_1 = 5$;
- The population size: $N_{\text{pop}} = 100$;
- The expected number of nondominated solutions in each time subperiod: $N_1 = 30$;
- The crossover probability: $p_c = 0.2$;
- The mutation probability: $p_m = 0.02$.

For the real-coded NSGA-II

- Population size: $N = 100$.
- Crossover probability: 0.9.
- Mutation probability: $1/n$, where n is the problem dimensions.
- Distribution indexes: $c = 20$ and $m = 20$ for SBX crossover and polynomial mutation.

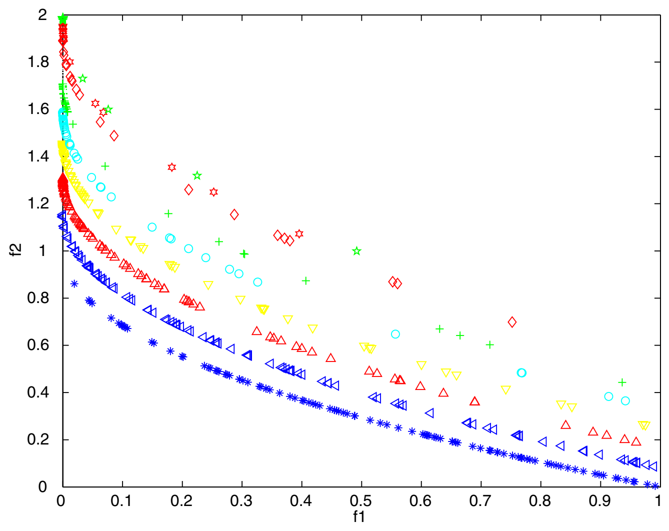

Fig. 3. The Pareto fronts found by the proposed algorithm for FDA3 in a typical run.

4.2.2. Performance measures

We use the following two performance measures in the comparisons:

- *Coverage metric (C-metric)* [18]: This metric compares the percentage of the solutions found by one algorithm in each time period covered by solutions found by another algorithm. For each time period $t = t_1, t_2, \ldots, t_{10}$, it is defined as follows: let A_i and B_i be two nondominated solution sets found by two compared algorithms in time period t_i for $i = 1, 2, \ldots, 10$. The C-metric maps ordered pair (A_i, B_i) to the interval $[0, 1]$

$$C(A_i, B_i) = \frac{|\{b \in B_i | \exists a \in A_i, a \text{ cov ers } b\}|}{|B|}$$

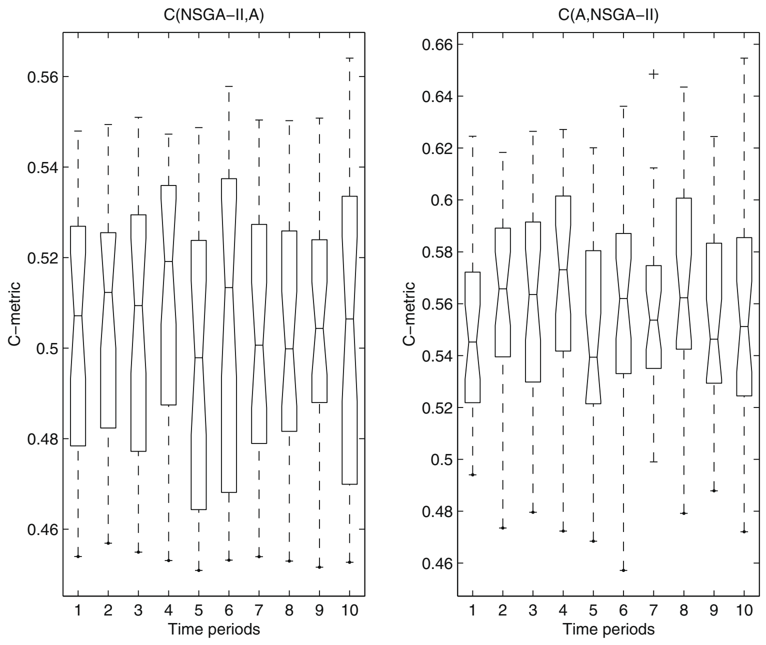

Fig. 4. The box plot of C-metric in the 10 time periods for FDA1, where the numbers 1, 2, ..., 10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding C-metric values in 30 runs.

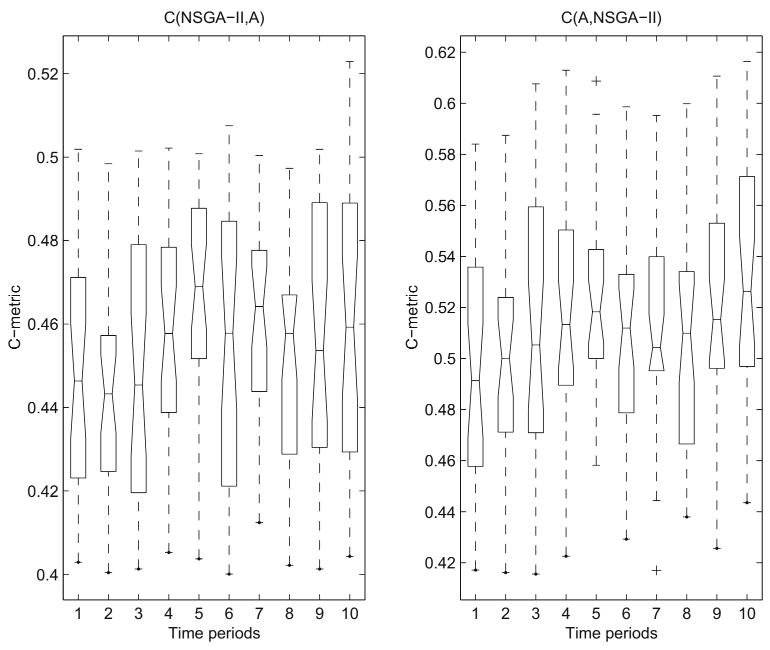

Fig. 5. The box plot of C-metric in the 10 time periods for FDA2, where the numbers l,2,…,10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding C-metric values in 30 runs.

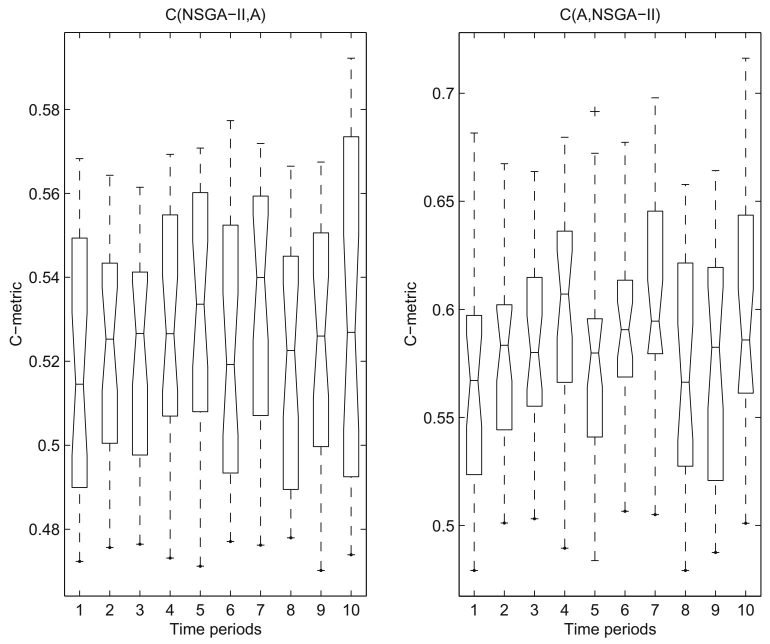

Fig. 6. The box plot of C-metric in the 10 time periods for FDA3, where the numbers l,2,…,10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding C-metric values in 30 runs.

where $|B|$ is the number of the elements in set B, and a cov *ers* b means that a dominates b or a and b are equally good. Thus, C-metric in 10 time periods can be defined by vector

$$\{C(A_1,B_1),C(A_2,B_2),\ldots,C(A_{10},B_{10})\}$$

$C(A_i,B_i) = 1$ means all solutions in B_i are weakly dominated by A_i. Note that $C(A_i,B_i)$ is not necessarily equal to l-$C(B_i,A_i)$. If $C(A_i,B_i) > C(B_i,A_i)$), then the solution set A_i is better than the solution set B_i with respect to C-metric.

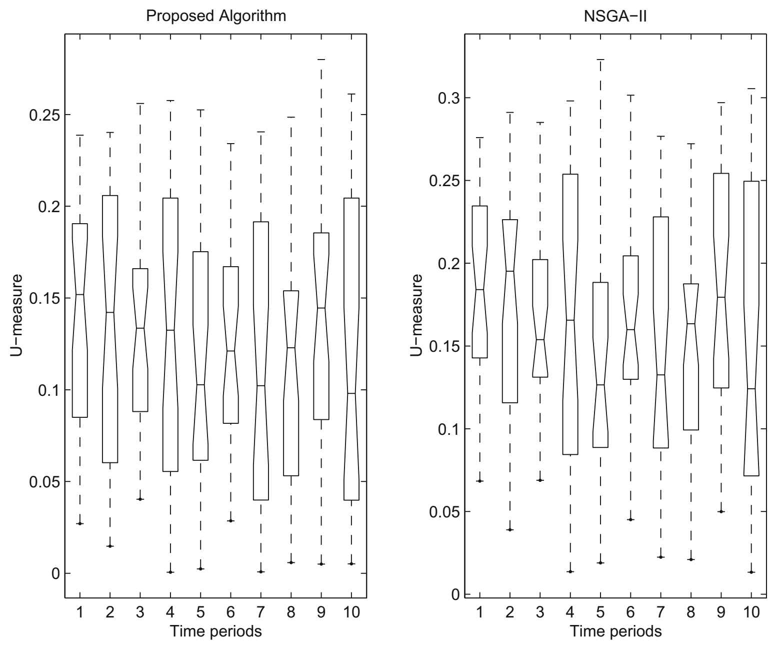

Fig. 7. The box plots of U-measure in the 10 time periods for FDA1, where the numbers l,2,. . .,10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding U-measure values in 30 runs.

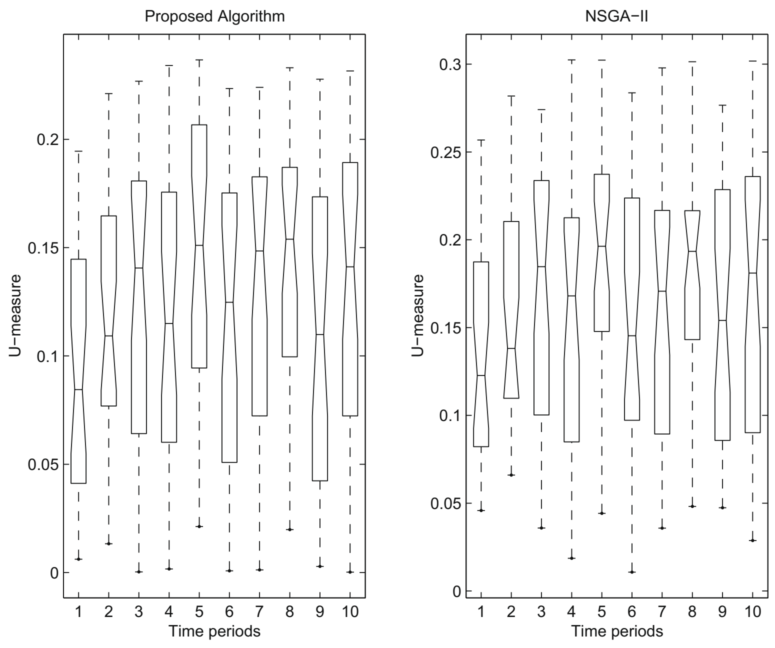

Fig. 8. The box plot of U-measure in the 10 time periods for FDA2, where the numbers l,2,. . .,10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding U-measure values in 30 runs.

- *Uniformity measure (U-measure)* [19]: This metric was proposed in [19] and used to measure the uniformity and the extent of the resulting Pareto front. For problems with two objectives $F(f_1,f_2)$, it can be used to the solutions found in each time period. The details are as follows:
  1. For each time period $t = t_1, t_2, \ldots, t_{10}$, compute $U_m(t_1), U_m(t_2), \ldots, U_m(t_{10})$ by Algorithm 2.1.
  2. U-metric in 10 time periods can be defined by $\{U_m(t_1), U_m(t_2), \ldots, U_m(t_{10})\}$.

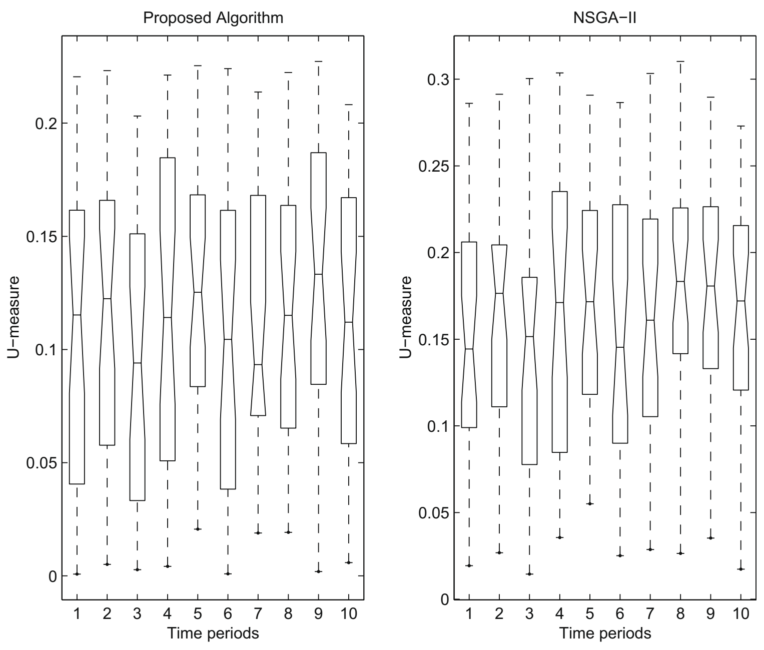

Fig. 9. The box plot of U-measure in the 10 time periods for FDA3, where the numbers 1, 2, . . ., 10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding U-measure values in 30 runs.

The smaller the $U_m(t)$ is, the more uniformity and the wider spread of the solutions found in the tth time period will be. We record the data of C-metric and U-measure of the proposed algorithm and NSGA-II for each test problem in 10 time periods in 30 independent runs, and compare these results by the box plots as shown in Figs. 4–9. For test problem FDA1, the Pareto solutions in the variable space changed with the time t, but the corresponding Pareto fronts do not change with time. Fig. 1 shows the Pareto fronts found by the proposed algorithm for FDA1 in a typical run. It can be seen from Fig. 1 that the Pareto fronts found by the proposed algorithm also do not change with time, which is consistent to the theoretical results. For test problem FDA2, the Pareto fronts swing from a convex shape to a nonconvex shape due to the change in $H(t)$ function, while the corresponding Pareto solutions in variable space remain unchanged. Fig. 2 shows that the Pareto fronts found by the proposed algorithm for FDA2 really change from a convex front to a nonconvex front in the 10 time periods in a typical run. For test problem FDA3, the Pareto fronts change with time due to the change in $F(t)$ and $G(t)$ functions, and the density of the Pareto fronts varies with time. Fig. 3 indicates that both the Pareto fronts found by the proposed algorithm for FDA3 and the density of them vary with time in a typical run. Figs. 4–6 depict the box plots of the C-metric for FDA1 to FDA3 in 10 time periods in 30 runs. Each figure represents the box plots of the C-metric $C(A, \text{NSGA-II})$ and $C(\text{NSGA-II}, A)$ for one test problem in the 10 time periods in 30 runs, where the numbers 1, 2, . . ., 10 in the horizontal axis represent the 10 time periods, the vertical axis represents the box plot of the corresponding C-metric values in 30 runs, and A in C-metric represents the proposed algorithm. It can be seen from Fig. 4 that $C(A, \text{NSGA-II}) > C(\text{NSGA-II}, A)$ in every time period. This means the solution set found by the proposed algorithm is better than the solution set found by NSGA-II in every time period with respect to C-metric. From Figs. 5 and 6, we can see that the similar conclusion is true. These results indicate that the solution sets found by the proposed algorithm in each time period have a higher quality than those found by NSGA-II. Figs. 7–9 depict the box plots of the U-measure for FD A1 to FD A3 in 30 runs. Each figure represents the box plots of the U-measure for the proposed algorithm and NSGA-II on one test problem in the 10 time periods in 30 runs, where the numbers 1, 2, . . ., 10 in the horizontal axis represent the 10 time periods, and the vertical axis represents the box plot of the corresponding U-measure values in 30 runs. It can be seen from Fig. 7 that the values of U-measure in 30 runs for the proposed algorithm in each time period for FDA1 are smaller than those for NSGA-II. For example, the maximum median value of U-measure in the 10 time periods for the proposed algorithm is close to 0.15, while that value for NSGA-II is close to 0.2. This illustrates the solutions found by the proposed algorithm in the objective space in each time period are distributed more uniformly and wider spread than those found by NSGA-II. From Figs. 8 and 9, we also can see that the values of U-measure for the proposed algorithm in 30 runs in each time period for FDA2 and FDA3 are smaller than those for NSGA-II, respectively. Thus, the proposed algorithm can also find more uniformly distributed and wider- spread nondominated solutions in the objective space in all time periods for these test problems. All these results indicate that the proposed algorithm can effectively track the Pareto fronts with time.

5. Conclusion

In this paper, a new evolutionary algorithm for dynamic multi-objective optimization problem (DMOP) is proposed. Firstly, the dynamic multi-objective optimization problem (DMOP) is approximated by a series of static multi-objective optimization problems (SMOPs) by dividing the whole time period into several equal subperiods. In each subperiod, the dynamic multi-objective optimization problem is then regarded as a static multi-objective optimization problem by taking the time parameter fixed. Secondly, the each static multi-objective optimization problem is further transformed into a two-objective optimization problem based on two re-defined objectives. Finally, a new crossover operator and mutation operator adapting to the environment changing are designed. The simulation results indicate that the proposed algorithm can effectively track the varying Pareto fronts with time and has a better performance than the compared algorithm.

Acknowledgements

The authors are grateful to the anonymous reviewers for their valuable comments and suggestions that greatly enhanced the quality of this paper. This work was jointly supported by the National Natural Science Foundation of China (No. 60374063) and (SRG: 7002006) of the City University of Hong Kong.

References

[1] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments – a survey, IEEE Trans. Evolutionary Comput. 9 (2005) 303–317.
[2] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer Academic Publishers, Massachuaetts, USA, 2002.
[3] H.G. Beyer, Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice, Comput. Methods Appl. Mech. Eng. 186 (2000) 239–267.
[4] M. Bhattacharya, G. Lu, A dynamic approximate fitness-based hybrid EA for optimization problems, in: Process of Congress of Evolutionary Computation, IEEE Press, Canberra, 2003, pp. 1879–1886.
[5] J. Branke, C. Schmidt, Fast convergence by means of fitness estimation, Soft Comput. 9 (2005) 13–20.
[6] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: test cases, approximations, and applications, IEEE Trans. Evolutionary Comput. 8 (2004) 425–442.
[7] K. Deb, H. Gupta, Searching for robust Pareto-optimal solutions in multi-objective optimization, in: Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 3410, Spriner, Berlin, 2005, pp. 150–164.
[8] M. Li, S. Azarm, V. Aute, A multi-objective genetic algorithm for robust design optimization, in: Process of Genetic and Evolutionary Computation Conference, ACM Press, Washington, DC, USA, 2005, pp. 771–778.
[9] Z. Bingul, A. Sekmen, S. Zein-Sabatto, Adaptive genetic algorithms applied to dynamic multiobjective programs, in: Process of Artificial Neural Networks Engineering Conference, Springer, Berlin, 2000, pp. 273–278.
[10] K. Deb, U.B.N. Rao, S. Karthik, Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling, in: S. Obayashi et al. (Eds.), Proceedings of EMO 2007, LNCS, vol. 4403, Springer-Verlag, 2007, pp. 803–817.
[11] Yaochu Jin, Bernhard Sendhoff, Constructing dynamic optimization test problems using the multiobjective optimization concept, in: Process of Evolution Workshops 2004, Lecture Notes in Computer Science, vol. 3005, Spriner, Berlin, 2004, pp. 525–536.
[12] H.G. Cobb, An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, Technical Report AIC-90-001, Naval Research Laboratory, 1990.
[13] F. Vavak, K. Jukes, T.C. Fogarty, Learning the local search range for genetic optimization in nonstationary environments, in: Process of 1997 IEEE Internatioal Conference Evolutionary Computation, IEEE Publisher, USA, 1997, pp. 355–360.
[14] A. Ghosh, T. Tstutsui, H. Tanaka, Function optimization in nonstationary environment using steady state genetic algorithms with aging individuals, in: IEEE International Conference Evolutionary Computation, IEEE Publisher, USA, 1998, pp. 666–671.
[15] N. Mori, H. Kitaand, Y. Nishikawa, Adaptation to changing environments by means of the thermodynamical genetic algorithms, in: H.M. Voigt (Ed.), Parallel Problem Solving From Nature, LNCS, vol. 1411, Springer, Berlin, 1996, pp. 513–522.
[16] D. Dasgupta, Incorporating redundancy and gene activation mechanisms in genetic search, in: L. Chambers (Ed.), Practical Handbook of Genetic Algorithms, Taylor and Francis Group/CRC Press, USA, 1995.
[17] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Congress on Evolutionary Computation CEC99, IEEE Publisher, USA, 1999, pp. 1875–1882.
[18] E. Zitzler, K. Deb, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evolutionary Comput. 3 (1999) 257–271.
[19] Y.W. Leung, Y.P. Wang, U-measure: a quality measure for multiobjective programming, IEEE Trans. Systems, Man, Cybernetics – Part A: Syst. Humans 33 (2003) 337–343.
[20] Y.P. Wang, An effective uniform genetic algorithm for hard optimization problems, in: Jian Qin Mao, Xi Ren Cao (Eds.), Process of 3rd World Congress on Intelligent Control and Automation, Sponsored by IEEE Robotics and Automation Society, Press of University of Science and Technology of China, 2000, pp. 656–660. June.
[21] K.T. Fang, Y. Wang, Number-Theoretic Method in Statistics, Chapman and Hall, London, 1994.
[22] K. Deb, A. Pratap, S. Agrawal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evolutionary Comput. 6 (2002) 182–197.