On Performance Metrics and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems

Xiaodong Li, Jürgen Branke, and Michael Kirley, *Member, IEEE*

***Abstract*— This paper describes two performance measures for measuring an EMO (Evolutionary Multiobjective Optimization) algorithm's ability to track a time-varying Pareto-front in a dynamic environment. These measures are evaluated using a dynamic multiobjective test function and a dynamic multiobjective PSO, $maximinPSOD$, which is capable of handling dynamic multiobjecytive optimization problems. $maximinPSOD$ is an extension from a previously proposed multiobjective PSO, $maximinPSO$. Our results suggest that these performance measures can be used to provide useful information about how well a dynamic EMO algorithm performs in tracking a time-varying Pareto-front. The results also show that $maximinPSOD$ can be made self-adaptive, tracking effectively the dynamically changing Pareto-front.**

I. Introduction

Multiobjective optimization represents an important class of optimization techniques which have a direct implication for solving many real-world problems. In recent years, using evolutionary algorithms to solve multiobjective optimization problems, commonly known as EMO (Evolutionary Multiobjective Optimization), has gained rapid popularity [1], [3]. Since Evolutionary Algorithms (EAs) make use of a population of candidate solutions, a diverse set of optimal solutions so called Pareto-optimal solutions can be found within a single run. EAs offer a distinct advantage over many traditional optimization methods where multiple solutions must be found in multiple separate runs.

Many EMO algorithms, most prominently NSGA II, SPEA, PAES [1], have shown to be very successful in solving multiobjective optimization problems. However, literature review shows that current EMO algorithms are largely focused on solving static multiobjective optimization problems. Very few studies have been devoted to solving dynamic multiobjective optimization problems where the approximate Pareto-front changes over the course of optimization [4]. As many real-world multiobjective optimization problems do show time-varying behaviour [5], it is important to measure how well an optimization algorithm adapts to a changing environment. Although there have been studies of performance measures on static MO problems [1], [15], [18], [19], research questions remain on performance measures for EMO algorithms in a dynamic environment. This paper attempts to answer some of these questions.

Xiaodong Li is with the School of Computer Science and IT, RMIT University, Melbourne, VIC 3001, Australia (phone: +61 3 99259585; fax: +61 3 96621617; email: xiaodong@cs.rmit.edu.au).

Jürgen Branke is with Institute AIFB, University of Karlsruhe, Germany.

Michael Kirley is with Department of Computer Science, Melbourne University, Australia.

Just like EAs, Particle Swarm Optimization (PSO) is also population-based. PSO is inspired by the social behaviours of animals and insects [6]. Since its introduction, it has been successfully applied to EMO problems [10], [11], [13], [14]. Compared with existing EAs, PSO shows relatively faster convergence behaviour and better adaptability to search landscape [6]. In a dynamic environment, such faster convergence behaviour and adaptability are desirable because the algorithm must be able to detect and respond to changes as quickly as possible. It is more important to keep tracking the changing Pareto-front than just locating the Pareto-front. A key question here is how to measure an EA's tracking ability to a time-varying Pareto-front in a dynamic environment.

The paper first describes related works in section 2, including some discussion on the important issues of detection and response, a brief description on PSO, and $maximinPSOD$ for handling optimization in a dynamic environment. Section 3 describes two performance measures for dynamic multiobjective algorithms, as well as collective mean errors based on these measures. Section 4 presents numerical results. Finally section 5 provides the concluding remarks.

II. Related Works

Although there are many studies on single objective optimization in dynamic environment [5], very few studies on multiobjective optimization in dynamic environment can be found in literature [2], [7], [8]. Recently, Jin and Sendhoff proposed to generate dynamic multiobjective optimization test functions using a dynamic weighted aggregation scheme [9]. They demonstrated that by varying weights, a dynamically changing Pareto-front and its corresponding values in the decision space can be produced. In another interesting work [4], Farina et al. proposed a set of test functions for dynamic multiobjective optimization, based on the well-known ZDT multiobjective test function sets. By adding a time variable to different components of the ZDT functions, various dynamic multiobjective optimization test functions can be generated. Farina et al. also proposed performance measures for an EMO in a dynamic environment. However, these measures are solely based on the convergence of an EMO algorithm in either the decision space or the objective space. More suitable performance measures that take into account both convergence and spread of the solutions are definitely needed.

For clarity, this paper uses $\mathcal{P}^*(t)$ to denote the Pareto-optimal front at time t, and $\mathcal{Q}(t)$ as the set of found non-dominated solutions at t. Minimization on all objectives is assumed throughout the paper, unless otherwise indicated.

A. Detection and Response

In the context of multiobjective optimization in a dynamic environment, when $\mathcal{P}^*(t)$ in the objective space (also Pareto-optimal solutions in the decision space) changes, an EMO algorithm is required to detect such changes so that appropriate response can be activated in order to track the newly updated $\mathcal{P}^*(t)$.

For single objective optimization in a dynamic environment, one common detection method is to randomly pick a point on the search landscape and re-evaluate its fitness. If this fitness value is different, then we assume a change has occurred in the environment. For multiobjective optimization in a dynamic environment, such a method becomes inadequate because it is the $\mathcal{P}^*(t)$ (made up of by a set of multiple optimal solutions) that must be tracked, instead of a single solution [1].

Existing convergence measures can be employed to calculate how well the algorithm converges to the time-varying $\mathcal{P}^*(t)$. For example, the GD measure (Generational Distance) proposed by Van Veldhuizen [3], can be adopted to find the average distance of the solutions of $\mathcal{Q}(t)$ from $\mathcal{P}^*(t)$:

$$GD(t) = \frac{\sum_{i=1}^{|\mathcal{Q}(t)|} d_i}{|\mathcal{Q}(t)|}, \tag{1}$$

where d_i is the Euclidean distance between the i-th member of $\mathcal{Q}(t)$ and its closest sampling point from $\mathcal{P}^*(t)$:

$$d_i = min_{k=1}^{|\mathcal{P}^*(t)|} \sqrt{\sum_{j=1}^{M} (f_j^{(i)} - f_j^{*(k)})^2}, \tag{2}$$

where $|\mathcal{P}^*(t)|$ specifies the number of sampling points on $\mathcal{P}^*(t)$. $f_j^{*(k)}$ is the j-th objective function value of the k-th sampling point on $\mathcal{P}^*(t)$. M is the number of objectives. Equation (1) can be used by an EMO algorithm to detect if a $\mathcal{P}^*(t)$ has changed. If $GD(t)$ is greater than a threshold, it indicates that there might be a change occurred, consequently this may trigger a response method such as randomizing partially or fully an EA population in order to maintain a sufficient level of diversity to allow the algorithm to converge towards the new $\mathcal{P}^*(t)$ in the changed environment. In [4], Farina et al. proposed to use some random sampling points and calculate the averaged fitness difference of these points between two consecutive iterations. If the difference is greater than some threshold value, then a change is considered to have occurred.

In this paper, we will show in the following sections that a multiobjective PSO can be made self-adaptive to the changing $\mathcal{P}^*(t)$, hence no explicit detection methods are required.

[1]It is possible that $\mathcal{P}^*(t)$ does not change, but its corresponding points in decision space change [4]. However, this paper focuses on dealing with changing $\mathcal{P}^*(t)$.

B. $maximinPSOD$

$maximinPSO$ makes use of a maximin fitness function to evaluate the fitness of an individual in a swarm population in terms of non-dominance and as well as diversity [14]. $maximinPSO$ has shown to have a rapid convergence with good solution distribution [14]. This motivated us to develop $maximinPSOD$ ($maximinPSO$ for Dynamic environment) which seems to be suitable for tracking a time-varying $\mathcal{P}^*(t)$ in a dynamic environment. In the following paragraphs, we will first describe PSO, then the maximin fitness function, followed by the basics of $maximinPSOD$. For an introduction to the basic concepts on multiobjective optimization, readers can refer to [1].

PSO is an optimization technique inspired by studies of the social behaviour of insects and animals [6]. The social behaviour is modelled in a PSO to guide a population of particles (or potential solutions) moving towards the most promising region of the search space. In PSO, each particle represents a candidate solution, $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})$. D is the dimension of the search space. The i-th particle of the swarm population knows: a) its personal best position $\mathbf{p_i} = (p_{i1}, p_{i2}, \ldots, p_{iD})$, i.e., the best position this particle has visited so far that yields the highest fitness value; and b) the global best position, $\mathbf{p_g} = (p_{g1}, p_{g2}, \ldots, p_{gD})$, i.e., the position of the best particle that gives the best fitness value in the entire population; and c) its current velocity, $\mathbf{v_i} = (v_{i1}, v_{i2}, \ldots, v_{iD})$, which represents its position change. The following 2 equations use the above information to calculate the new updated velocity for each particle in the next iteration step, and to update each particle's position:

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{3}$$
$$x_{id} = x_{id} + v_{id}\ , \tag{4}$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$; N is the population size; w is the inertia weight, which is often used as a parameter to control exploration/exploitation; c_1 and c_2 are two coefficients (positive constants); r_1 and r_2 are two random numbers within the range [0, 1]. There is also a V_{MAX}, which sets the upper and lower bound for velocity values. For more information on PSO, readers can refer to [6].

The maximin fitness for a decision vector $\mathbf{u}$ can be calculated through the following steps. First the $\mathbf{min}$ function is called to obtain the minimal value from set $\{f_i(\mathbf{u}) - f_i(\mathbf{v}) \mid \forall i \in \{1, \ldots, m\}$:

$$\mathbf{min}_{i=1,\ldots,M}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\} \tag{5}$$

where $\mathbf{v}$ denotes a decision vector $\mathbf{u}$ is compared with. Basically equation (5) is applied over all possible pairs of $\mathbf{u}$ and another decision vector (other than $\mathbf{u}$) in the population, to obtain the minimal values for all the pairs. The maximin fitness for $\mathbf{u}$ can be obtained by applying the $\mathbf{max}$ function over this set of minimal values:

$$f = \mathbf{max}_{j=1,\ldots,N;\mathbf{u}\neq\mathbf{v}}\{\mathbf{min}_{i=1,\ldots,M}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\} | \forall \mathbf{v} \in \mathcal{N}; \mathbf{v} \neq \mathbf{u}\}, \tag{6}$$

where $\mathcal{N}$ denotes the set representing the population. Two loops of comparisons take place here, with the $\mathbf{min}$ first applied to all pairs of $\mathbf{u}$ and every other vector in the population over the objectives from 1 to M, and then the $\mathbf{max}$ is applied the set of minimal values obtained.

It is now obvious that for any solution (i.e., a decision vector) to be a non-dominated solution with respect to the current population, its maximin fitness value must be less than zero. Any solution with a maximin fitness equal to zero is a weakly-dominated solution. Any solution with a maximin fitness value greater than zero is a dominated solution. One unique property about maximin fitness function is that it can be used to reward diversity and penalize clustering of non-dominated solutions, therefore no additional diversity maintaining mechanism is required.

Our goal is to propel particles in the population towards $\mathcal{Q}(t)$ as well as the less crowded areas along $\mathcal{Q}(t)$. For each particle, we can choose randomly its p_{gd} for each dimension $d = 1, \ldots, D$ of the particle, from a pool of non-dominated particles with the smallest maximin fitness values (they should be negative too), which represent particles in the least-crowded areas of $\mathcal{Q}(t)$ [14]. By selecting $\mathbf{p_g}$ this way, we hope to obtain a $\mathcal{Q}(t)$ that not only converges nicely towards $\mathcal{P}^*(t)$, but also well-distributed along $\mathcal{P}^*(t)$.

With sufficiently large numbers of particles, it has been shown that $maximinPSO$ is able to produce a set of well-distributed non-dominated solutions, regardless the front being convex or concave [14]. In this paper, $maximinPSO$ is extended to $maximinPSOD$ for handling dynamic environments. $maximinPSOD$ consists of following steps:

1) An initial population of N particles is generated at random, and stored in $PSOList$; For each particle on $PSOList$, personal best $\mathbf{p_i}$ is set to $\mathbf{x_i}$ by default; $\mathbf{v_i}$ is set to be within the variable ranges, with a probability of 0.5 being positive or negative; V_{MAX} is set to the bounds of the variable ranges.
2) Calculate the maximin fitness for all particles in the initial population; get the non-dominated solutions based on the maximin fitness values; sort the non-dominated solutions according to their maximin fitness values (in ascending order), and store them in $nonDomPSOList$.
3) Iterate through the whole swarm population $PSOList$. For the i-th particle (from 1 to $PSOList$'s current size), do the following:
    a. Choose randomly a p_{gd} for each dimension $d = 1, \ldots, D$ of the i-th particle, from the top few particles (e.g., top 10%) of the $nonDomPSOList$, as the new $\mathbf{p_g}$ for the i-th particle.
    b. Produce an offspring for the i-th particle based only on its $\mathbf{p_g}$ (This is done by resetting $\mathbf{p_i}$ to $\mathbf{x_i}$).
    c. Store both the i-th particle and its offspring in a list $nextPopList$. Note that $nextPopList$ should be twice the size of $PSOList$, since it now contains both the parent and its offspring.
4) Calculate maximin fitness for all particles on $nextPopList$; get non-dominated solutions; copy the non-dominated solutions from $nextPSOList$ to the new $PSOList$ for the next iteration; if there are less than N particles, then randomly add weakly-dominated or dominated solutions from $nextPopList$, until $PSOList$ is filled up; Note that the size of $PSOList$ is fixed at N.
5) Calculate performance measures on $rGD(t)$ or $HVR(t)$ (section 3).
6) Go back to Step 3), until a termination criterion is met.

There are only 2 modifications to $maximinPSO$ in the above - at step 3b) produce a particle's offspring based only on its $\mathbf{p_g}$, leaving $\mathbf{p_i}$ out; step 5) the calculation of performance measures. Readers can refer to [14] for a detailed description of $maximinPSO$.

C. Self-adaptiveness

In the above $maximinPSOD$, no explicit detection methods are employed. This is because $maximinPSO$ can be easily modified to self-adapt to a changing environment. This is achieved by resetting each particle's personal best $\mathbf{p_i}$ to its current position $\mathbf{x_i}$ at each iteration, to ensure it is always up-to-date with the possible changes occurred in the environment. As indicated in step 3b) of $maximinPSOD$, this essentially eliminates $\mathbf{p_i}$ from the PSO velocity update equation, which means that only the global best $\mathbf{p_g}$ has influence in a particle's next movement. In the case of a change occurring, each particle will not be influenced by its "memory" (ie., $\mathbf{p_i}$, the best position visited so far), since this "memory" becomes meaningless once the environment has changed.

The way $\mathbf{p_g}$ being chosen also helps to make $maximinPSOD$ self-adaptive to changes. Since $\mathbf{p_g}$ for each particle is randomly chosen from the top few particles (e.g., top 10%) of the $nonDomPSOList$, which are in the least-crowded areas of $\mathcal{Q}(t)$, attracting particles towards these positions not only helps to better maintain solution diversity, but also adjust particles' positions with respect to the changing $\mathcal{P}^*(t)$.

III. Performance Measures

The optimization goal for an EMO in a dynamic environment is to follow as closely as possible the dynamically changing Pareto-front $\mathcal{P}^*(t)$. The EMO algorithm is required to continuously make a good progress on both the convergence (towards $\mathcal{P}^*(t)$) and the distribution of the solution set $\mathcal{Q}(t)$ (along $\mathcal{P}^*(t)$).

Farina et al. proposed a convergence measure which is based on the time-varying $GD(t)$ values and the time-dependent *nadir* and *utopia* points [4]. However, this method does not take into account of the diversity of the non-dominated solutions in $\mathcal{Q}(t)$. This section describes and analyzes the properties of two performance measures (or indicators). Both are able to measure an EMO's performance in terms of convergence and diversity, therefore they should

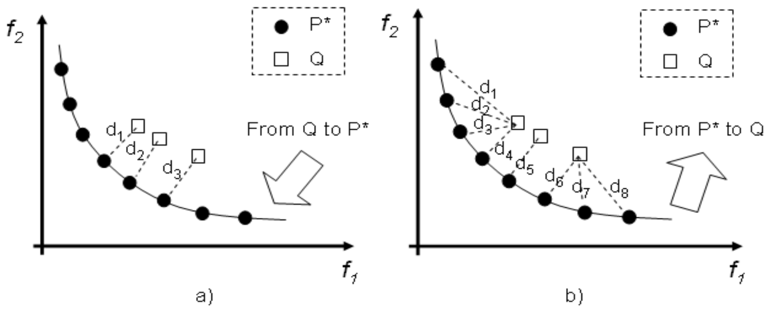

Fig. 1. Calculating a) $GD(t)$, and b) $rGD(t)$, which gives a larger average distance value than $GD(t)$ for the same $\mathcal{Q}(t)$ and $\mathcal{P}^*(t)$. Note that black dots represent sampling points on $\mathcal{P}^*(t)$.

improve the accuracy over the convergence-based only measures.

A. rGD(t)

The first measure, $rGD(t)$ (reversed $GD(t)$), is derived from $GD(t)$ (see equation (1)) [3]. $GD(t)$ can be modified to measure an EMO's performance in terms of both convergence and spread of the solution set $\mathcal{Q}(t)$ with respect to a time-varying $\mathcal{P}^*(t)$, in the following manner:

$$rGD(t) = \frac{\sum_{i=1}^{|\mathcal{P}^*(t)|} d_i}{|\mathcal{P}^*(t)|}, \tag{7}$$

where $d_i = min_{k=1}^{|\mathcal{Q}(t)|} \sqrt{\sum_{j=1}^{M} (f_j^{*(i)} - f_j^{(k)})^2}$. Note that $f_j^{(k)}$ is the j-th objective function value of the k-th member of $\mathcal{Q}(t)$. Equation (7) calculates d_i in a 'reversed' direction as compared to the original $GD(t)$ measurement. Instead of iterating over $Q(t)$, equation (7) iterates over $\mathcal{P}^*(t)$, that is, for each sampling point in $\mathcal{P}^*(t)$, the closest solution in $\mathcal{Q}(t)$ is found. And d_i is the Euclidean distance between these two points. By using $rGD(t)$, an EMO algorithm can get useful information on both the convergence and spread of solutions in $\mathcal{Q}(t)$. An ideal $rGD(t)$ would be 0.0, which means $\mathcal{Q}(t)$ has the best convergence and spread of solutions. Fig. 1 shows an example of how $rGD(t)$ is calculated differently from $GD(t)$. The $\mathcal{Q}(t)$ in this example represents an especially poor spread of solutions, since $rGD(t)$ gives a relatively larger value.

$rGD(t)$ assumes $\mathcal{P}^*(t)$ is known a priori, and its value depends on the distribution of sampling points on $\mathcal{P}^*(t)$ chosen for the calculation. A similar measure, $D1_R$, as proposed in [16] also measures the average distance from sampling points on $\mathcal{P}^*(t)$ to the nearest point on $\mathcal{Q}(t)$. $D1_R$ is defined as follows [18]:

$$D1_R(A, \Lambda) = \frac{1}{|R|} \sum_{\mathbf{r} \in R} \min_{\mathbf{z} \in A} \{d(\mathbf{r}, \mathbf{z})\} \tag{8}$$

where A and R are equivalent to $\mathcal{Q}(t)$ and $\mathcal{P}^*(t)$ respectively in this paper, $d(\mathbf{r}, \mathbf{z}) = \max_j\{\lambda_j(r_j - z_j)\}$ and $\Lambda = [\lambda_1, \ldots, \lambda_J]$, $\lambda_j = 1/R_j, j = 1, \ldots, J$ with R_j being the range of objective j in set R.

Note that $d(\mathbf{r}, \mathbf{z})$ is computed using a $\mathbf{max}$ function. However, $rGD(t)$ relies on calculation of the Euclidean distance between any two closest points from $\mathcal{P}^*(t)$ to $\mathcal{Q}(t)$. $D1_R$ was shown to give a result which does not tell the correct *outperformance* relation for the following bi-objective maximization example used in [18]. For example, if the reference set is $R = \{\mathbf{r}^1 = [9, 4], \mathbf{r}^2 = [4, 9]\}$, and two nondominated solution sets are: $\mathcal{A} = \{\mathbf{u}^1 = [8, 2], \mathbf{u}^2 = [2, 8]\}$, and $\mathcal{B} = \{\mathbf{u}^3 = [8.5, 2], \mathbf{u}^4 = [2, 8.5]\}$. Since $\mathbf{u}^3$ dominates $\mathbf{u}^1$ and $\mathbf{u}^4$ dominates $\mathbf{u}^2$, $\mathcal{B}$ should outperform $\mathcal{A}$. However, $D1_R$ would consider $\mathcal{A}$ and $\mathcal{B}$ are equally good, since $D1_R(\mathcal{A}) = D1_R(\mathcal{B}) = 0.2$. In contrast, $rGD(t)$ value for $\mathcal{A}$ is 4.47 and for $\mathcal{B}$ is 4.12. This result tells the correct outperformance relation, i.e., $\mathcal{B}$ outperforming $\mathcal{A}$. Hansen and Jaszkiewicz's work in [18] has provided details on formal definitions of outperformance relations, and how existing performance measures are compliant with these relations.

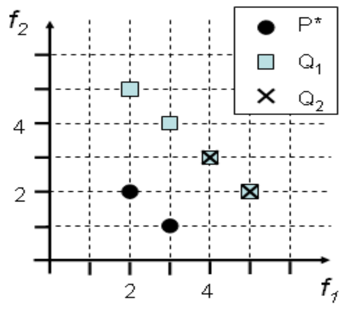

Fig. 2. In this example $rGD(t)$ considers both Q_1 and Q_2 are equally good when 2 sampling points are used in representing $\mathcal{P}^*(t)$. However, if an additional sampling point (1, 3) is added, $rGD(t)$ for Q_2 will increase while $rGD(t)$ for Q_1 remains unchanged.

Like some existing performance measures, $rGD(t)$ also has its drawbacks. It strongly depends on the distribution of sampling points on $\mathcal{P}^*(t)$. For example, in Fig. 2, sets Q_1 and Q_2 would be considered as equally good according to their $rGD(t)$ values, where in fact Q_1 is *better* than Q_2. However, if an additional sampling point (1, 3) is added to the existing two sampling points in the figure, $rGD(t)$ for Q_2 will increase while the $rGD(t)$ for Q_1 remains unchanged. In such a case, Q_1 will be considered better than Q_2, which indicates the correct outperformance relation. This example shows that if we have a sufficient number of sampling points covering the $\mathcal{P}^*(t)$, then it is reasonable

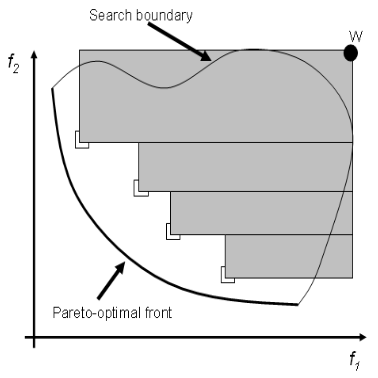

Fig. 3. HV as given by the sum of shaded areas.

to expect $rGD(t)$ will inform the correct outperformance relation. In our experiments that follow (see section 4), we ensure that the coverage of sampling points on $\mathcal{P}^*(t)$ is always sufficient.

Another measure which is almost identical to $rGD(t)$ was recently suggested in [12]. However, more importantly in the context of dynamic multiobjective optimization, this section illustrates clearly that $rGD(t)$ is a more informative measure than $GD(t)$.

B. HVR(t)

The second measure proposed is based on Hypervolume (HV) also suggested by Van Veldhuizen [3]. In HV the area covered by all the solutions in $\mathcal{Q}$ is calculated with the help of using a reference point $\mathcal{W}$, which is a vector identified as having the worst objective function values (see Fig. 3). HV is calculated as follows [1]:

$$HV = volume(\cup_{i=1}^{|\mathcal{Q}|} v_i), \tag{9}$$

where for each solution $i \in \mathcal{Q}$, a hypercube v_i is constructed with reference to $\mathcal{W}$. The HV value is the union of all hypercubes. HV does not assume the Pareto-optimal front is known a priori. However, HV depends strongly on the choice of $\mathcal{W}$. Two comparing non-dominated sets can have a different ordering when a different choice of $\mathcal{W}$ is used [15]. It has a complexity $O(n^{k+1})$ (k is the number of objectives), so it becomes very expensive when computing over a large number of objectives.

For static EMO problems, it is obvious that the larger this HV value, the better the performance by an EMO algorithm is. However, for dynamic EMO problems, since $\mathcal{P}^*(t)$ is time-varying, it becomes meaningless to compare different $HV(t)$ values at different iterations. One way to overcome this problem is to compute $HVR(t)$, the ratio of the $HV(t)$ of $\mathcal{Q}(t)$ and $\mathcal{P}^*(t)$. $HVR(t)$ (for a static environment) was first suggested by Deb to deal with the problem of bias ([1]:p.333). It can be also used as a performance measure for EMO in a dynamic environment:

$$HVR(t) = \frac{HV(\mathcal{Q}(t))}{HV(\mathcal{P}^*(t))}. \tag{10}$$

The reference point $\mathcal{W}(t)$ can be determined by identifying the worst value in each objective dimension, of all the non-dominated solutions at iteration t. $\mathcal{W}(t)$ is also referred to as the time-dependent *nadir* point [4].

The $HVR(t)$ should give a more accurate indication of an EMO algorithm's tracking ability because it tells how well $HV(\mathcal{Q}(t))$ covers $HV(\mathcal{P}^*(t))$ regardless of their absolute values. The maximum value of $HVR(t)$ should be 1.0 when $\mathcal{Q}(t)$ is equal to $\mathcal{P}^*(t)$. Since $\mathcal{P}^*(t)$ is used, $HVR(t)$ does assume $\mathcal{P}^*(t)$ is known a priori, and it also depends on the distribution of sampling points on $\mathcal{P}^*(t)$.

C. Collective Mean Error

To measure how well an EMO algorithm performs over a run, we can calculate the *Collective Mean Error* (CME) for $rGD(t)$ by averaging the $rGD(t)$ values over the entire run:

$$CME_{rGD} = \frac{1}{T}\sum_{t=1}^{T} rGD(t), \tag{11}$$

where T is the total number of iterations of a run. CME provides a way of measuring the aggregated performance of an EMO algorithm over a run, after exposing to a representative range of fitness landscape dynamics. Similarly CME_{HVR} can be also defined:

$$CME_{HVR} = \frac{1}{T}\sum_{t=1}^{T} HVR(t). \tag{12}$$

Equations (11) and (12) measure the overall tracking performance of an EMO algorithm in the given time of a run. CME_{rGD} with a value of 0.0 indicates the best possible optimal tracking performance. For CME_{HVR}, this optimal value is 1.0. To assess an EMO algorithm's robustness, it is generally assumed that the algorithm has been exposed to all dynamic environments considered during a run.

IV. Numerical Results

Jin et al. in [9] proposed to define a dynamic two-objective optimization problem from reformulation of a three-objective optimization test function: $minimize(f_1, f_2, f_3)$, where

$$\begin{aligned} f_1 &= x_1^2 + (x_2-1)^2, \\ f_2 &= x_1^2 + (x_2+1)^2 + 1, \\ f_3 &= (x_1-1)^2 + x_2^2 + 2, \\ subject\ to \quad &: -2 \le x_1, x_2 \le 2. \end{aligned}$$

These 3 functions are used to minimize the following two functions:

$$\begin{aligned} F_1 = w f_1 + (1-w) f_2, \\ F_2 = w f_1 + (1-w) f_3, \end{aligned}$$

where w can be substituted by t ($0 \le t \le 1$), which can be varied by $t = \frac{1}{n_T}\lfloor \frac{\tau}{\tau_T} \rfloor$. Note that τ is iteration number, and

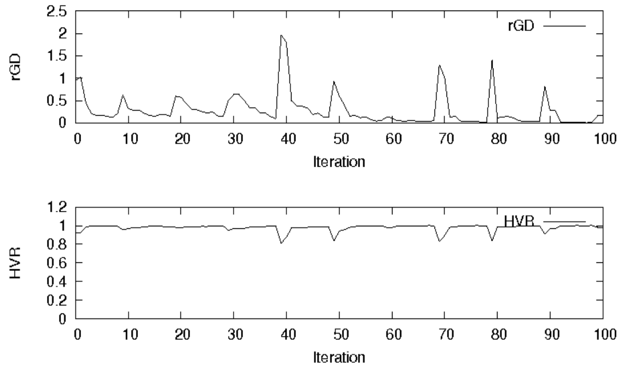

Fig. 4. $rGD(t)$ and $HVR(t)$ values over 100 iterations of a run; The $maximinPSOD$ population is randomized (ie., using a "restart" strategy) when $\mathcal{P}^*(t)$ changes at every τ_T = 10 iterations.

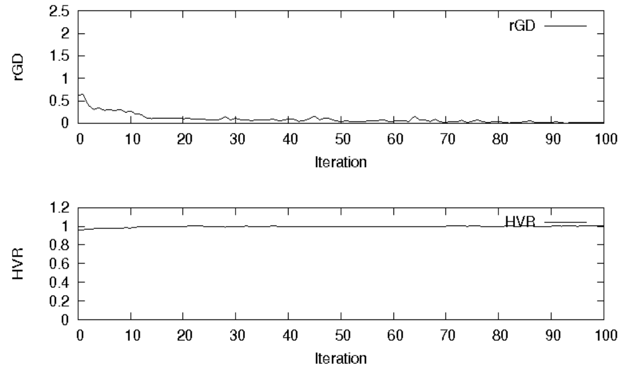

Fig. 5. $rGD(t)$ and $HVR(t)$ values over 100 iterations of a run; $maximinPSOD$ self-adapts to the changed $\mathcal{P}^*(t)$ at every τ_T =10 iterations.

τ_T is the number of iterations for which t remains fixed. n_T is the number of distinct steps in t. For experiments in this paper, we set $\tau_T \in \{5, 10, 20\}$, and $n_T = 10$.

Experiments were carried out on the above test function with the following three aims: a) to study the efficacy of $rGD(t)$ and $HVR(t)$ for measuring the performance of $maximinPSOD$ in a dynamic environment; b) to find out how well $maximinPSOD$ self-adapt to the changing $\mathcal{P}^*(t)$; c) to examine the effectiveness of CME_{rGD} and CME_{HVR}.

A population size of 5 or 10 was used for the experiments. The $maximinPSOD$ was run for 100 iterations. For the PSO parameters, c_1 and c_2 were set to 2.0. w was gradually decreased from 1.0 to 0.4. V_{MAX} was set to the bounds of decision variables. This is a set of common parameter values used in a simple PSO model [6]. To calculate the $\mathcal{P}^*(t)$, 1000 sampling points were taken for the search space at iteration t, and then $\mathcal{P}^*(t)$ was defined by identifying the non-dominated solutions from the 1000 sampling points. This might not be ideal, as for some test function it is possible to derive directly the $\mathcal{P}^*(t)$ rather than an approximation.

A. rGD(t) and HVR(t)

In this experiment, with Jin's test function, we set $\tau_T = 10$, which means $\mathcal{P}^*(t)$ changes every 10 iterations, so there are 10 changes over 100 iterations of a run. Firstly, we deliberately re-randomized the population whenever $\mathcal{P}^*(t)$ changes. By doing this, the performance measures are expected to indicate performance deterioration of an EMO algorithm using a "restart" strategy, at the iteration when the change occurs. Fig. 4 shows clearly this is exactly the case. In contrast, without any "restart" strategy, $maximinPSOD$ with a population size of 10 was able to self-adapt to the changes and minimize the $rGD(t)$ and maximize $HVR(t)$ effectively throughout the run, as shown in Fig. 5. This can be attributed to the fact that in $maximinPSOD$ particles do not use "memory" and are always up-to-date with possible changes in the environment. No explicit detection methods were used.

B. Self-adaptiveness

Fig. 6 shows a run of $maximinPSOD$ on the test function. To better illustrate the effect of $maximinPSOD$'s self-adaptiveness to the changing $\mathcal{P}^*(t)$, a population size of 30 was used. τ_T was still set to 10, so the $\mathcal{P}^*(t)$ changes at every 10 iterations. At step 9, 29 and 59, it can be seen that $maximinPSOD$ converged reasonably well to $\mathcal{P}^*(t)$ just before the end of an interval of $\tau_T = 10$ iterations. At step 10, 30, 60, 90 and 100 (ie., iteration immediately after an interval), it managed to immediately self-adapt to the change with at least some particles continuing to track the changed $\mathcal{P}^*(t)$. Since $\mathcal{Q}(t)$ is able to relocate $\mathcal{P}^*(t)$ almost immediately after each change interval, the performance as measured by $rGD(t)$ should give a relatively small value close to 0.0, whereas $HVR(t)$ should give a value close to 1.0, similar to Fig. 5.

C. Collective Mean Errors

In this experiment, $maximinPSOD$'s collective mean errors for $rGD(t)$ and $HVR(t)$ were measured. Three test cases were used, $\tau_T \in \{5, 10, 20\}$ (which means $\mathcal{P}^*(t)$ changes at every 5, 10 and 20 iterations respectively), so it was exposed to 20, 10 and 5 different $\mathcal{P}^*(t)$ respectively. Among the three test cases, $\tau_T = 20$ is the easiest problem, since $\mathcal{P}^*(t)$ changes only 5 times, and each time it remains unchanged for 20 iterations. A population size of 10 and 5 were used over the three test cases to see the effect of different population sizes.

Table I shows the self-adaptive $maximinPSOD$'s CME_{rGD} and CME_{HVR} at the end of 100 iterations of a run, averaged over 30 runs. Generally speaking, $maximinPSOD$ was able to track the $\mathcal{P}^*(t)$ well over the run, especially when a population size of 10 was used. All CMEs when using a population size of 10 are better than those of population size of 5. This can be explained by the fact that to have a good solution distribution along $\mathcal{P}^*(t)$, it is important to have a sufficient number of particles, otherwise the performance would deteriorate. Another observation is that CME_{HVR} values for the $popsize = 10$ case are all slightly greater than 1.0. On one hand it shows that $maximinPSOD$ was able to track well $\mathcal{P}^*(t)$; on the other hand it shows an inaccurate approximation of the true $\mathcal{P}^*(t)$

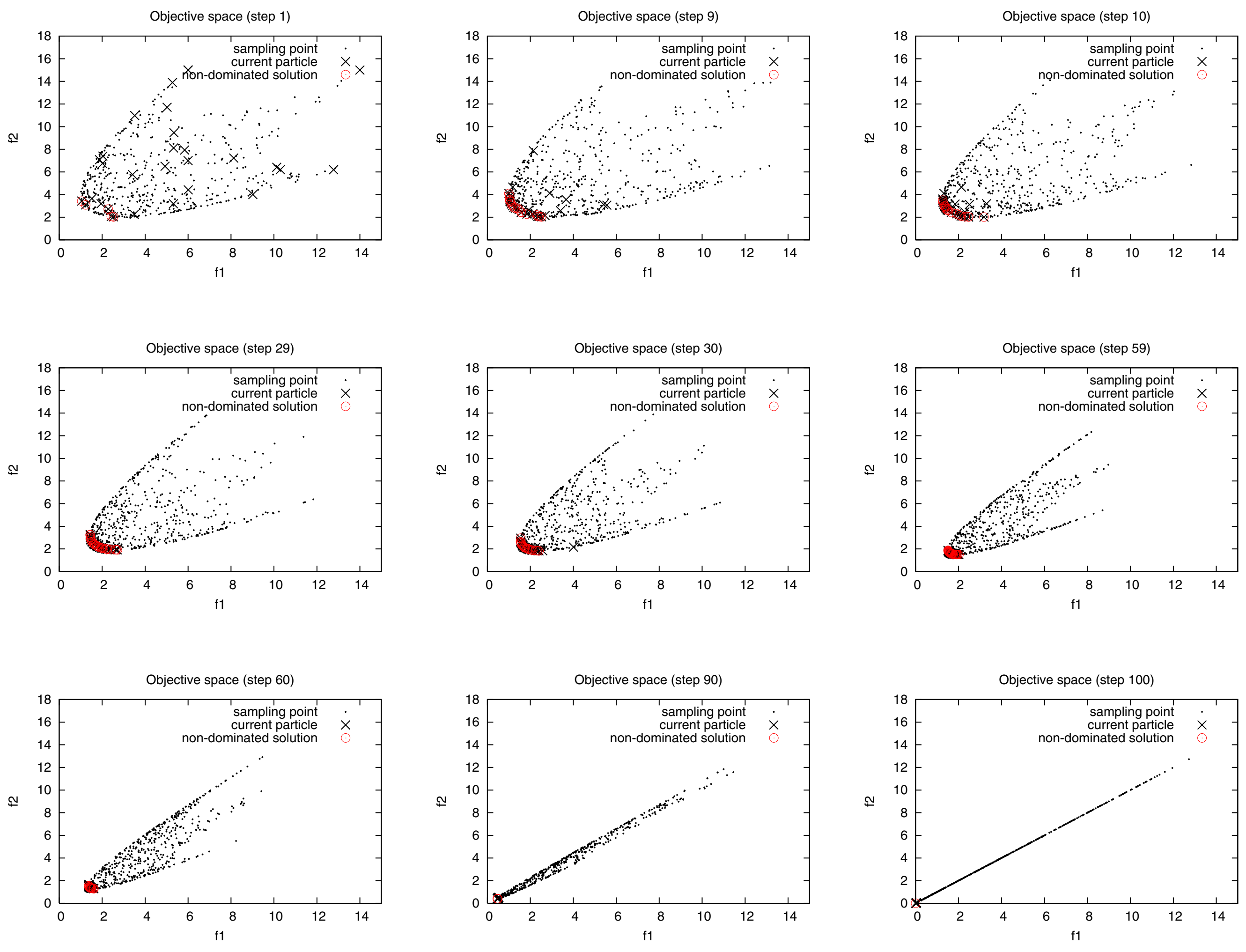

Fig. 6. Snapshots of a run of $maximinPSOD$ on the test function ($popsize = 30$; $\tau_T = 10$).

when using 1000 sampling points over the entire search space. It is possible that the coverage of sampling points on $\mathcal{P}^*(t)$ became less than that of $\mathcal{Q}(t)$ at some iterations. Since $HVR(t)$ depends on the distribution of sampling points, a larger (than sampling) coverage in $\mathcal{Q}(t)$ would result in a $HVR(t)$ value greater than 1.0. More accurate approximation or a direct derivation of the true $\mathcal{P}^*(t)$ should eliminate this problem.

The original $maximinPSO$ (without resetting $\mathbf{p_i}$ to its $\mathbf{x_i}$) was also run using the same setups. The results of CME_{rGD} and CME_{HVR} shown in Table II are much worse than those corresponding results in Table I, except for the case $\tau_T = 20$. Since $\tau_T = 20$ is the easiest case, where $\mathcal{P}^*(t)$ remains unchanged for every 20 iterations. $maximinPSO$ would have enough time to recover and relocate the changed $\mathcal{P}^*(t)$. $maximinPSO$ performed much poorly for cases $\tau_T = 5$ and $\tau_T = 10$. These results demonstrate that the self-adaptive $maxininPSOD$ is definitely more effective than the original $maximinPSO$ in tracking $\mathcal{P}^*(t)$.

V. Conclusions

This paper has described in details two performance measures, $rGD(t)$ and $HVR(t)$, and their collective mean errors (CME) for measuring an EMO's performance in a dynamic environment. The performance measures were evaluated using a simple dynamic EMO test function and $maximinPSOD$, a dynamic multiobjective PSO extended from a previously proposed static multiobjective PSO algorithm. Our results suggest that both $rGD(t)$ and $HVR(t)$ are useful performance indicators for measuring EMO algorithms in their ability to track a time-varying Pareto-optimal front $\mathcal{P}^*(t)$, although there are drawbacks such as the assumption of a known $\mathcal{P}^*(t)$. Our results also show that by resetting $\mathbf{p_i}$ to its $\mathbf{x_i}$ at each iteration, $maximinPSOD$ can be made self-adaptive to the changing $\mathcal{P}^*(t)$. One advantage of this PSO model is the avoidance of explicit detection methods, which are often required by an optimization algorithm in dynamic environments.

Future works on evaluating both $rGD(t)$ and $HVR(t)$ on more complex and challenging EMO problems will be

TABLE I

$maximinPSOD$'S CME_{rGD} AND CME_{HVR} AVERAGED OVER 30 RUNS (MEAN AND STANDARD DEVIATION).

Popsize	τ_T	5	10	20
10	CME_{rGD}	1.05E-01 (2.34E-02)	1.03E-01 (1.91E-02)	1.66E-01 (2.96E-02)
	CME_{HVR}	1.01E+00 (2.29E-03)	1.01E+00 (2.32E-03)	1.01E+00 (2.90E-03)
5	CME_{rGD}	1.34E+00 (3.72E+00)	2.30E-01 (9.64E-02)	4.64E-01 (6.63E-01)
	CME_{HVR}	9.44E-01 (1.76E-01)	1.00E+00 (1.02E-02)	9.71E-01 (8.00E-02)

TABLE II

$maximinPSO$'S CME_{rGD} AND CME_{HVR} AVERAGED OVER 30 RUNS (MEAN AND STANDARD DEVIATION).

Popsize	τ_T	5	10	20
10	CME_{rGD}	4.05E+01 (1.52E+01)	8.53E+00 (9.50E+00)	2.13E-01 (2.91E-02)
	CME_{HVR}	9.48E-01 (1.78E-02)	9.51E-01 (1.91E-02)	9.32E-01 (2.25E-02)
5	CME_{rGD}	3.09E+01 (2.98E+01)	8.33E+00 (9.26E+00)	3.72E-01 (7.97E-02)
	CME_{HVR}	8.85E-01 (1.40E-01)	9.63E-01 (2.22E-02)	9.53E-01 (1.28E-02)

carried out. More sophisticated multiobjective PSOs incorporating the archive and ϵ approaches will be investigated, as they may lead to further improvement in efficiency in tracking the dynamically changing $\mathcal{P}^*(t)$. More works on how to measure an EMO algorithm's performance in decision space (rather than tracking $\mathcal{P}^*(t)$ in the objective space) will be also explored.

REFERENCES

[1] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Chichester, UK (2001).

[2] Deb, K., Bhaskara Rao, U. and Karthik, S.: Dynamic Multi-Objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-Thermal Power Scheduling Bi-Objective Optimization Problems. KanGAL Report No. 2006008 (2006).

[3] Van Veldhuizen, D.: Multiobjective Evolutionary Algorithms: Classifications, Analysis, and New Innovations, Ph.D. Thesis, Dayton, OH: Air Force Institute of Technology. Technical Report No. AFIT/DS/ENG/99-01 (1999).

[4] Farina, M., Deb, K. and Amato, P.: Dynamic multiobjective optimization problems: test cases, approximation and applications. In *IEEE Trans. on Evolutionary Computation* 8(5), (2004) 425-442.

[5] Branke, J.: Evolutionary Optimization in Dynamic Environments. Boston: Kluwer Adademic Publishers (2002).

[6] Kennedy, J., Eberhart, R.C.:Swarm intelligence, San Francisco: Morgan Kaufmann Publishers (2001).

[7] Yamasaki, K.: Dynamic Pareto optimum GA against the changing environments. In Proc. Genetic Evolutionary Computation Conf. Workshop, San Francisco, CA (2001) 47-50.

[8] Bingul, Z., Sekmen, A. and Zein-Sabatto, S.: An alife-inspired evolutionary algorithm applied to dynamic multi-objective problems. In Proc. Artificial Neural Networks Engineering Conf., C.H. Dagli, et al. (eds.), New York, (2000) 273-278.

[9] Jin, Y. and Sendhoff, B.: Constructing dynamic test problems using the multi-objective optimization concept. In: Applications of Evolutionary Computing. LNCS 3005, Raidl, G.R. et al. (eds.) Springer (2004) 525-536.

[10] Fieldsend, E. and Singh, S.: A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence, Proc. of the 2002 U.K. Workshop on Computational Intelligence, Birmingham, UK(2002) 37-44.

[11] Coello, C.A.C. and Lechuga, M.S.: MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization, in Proc. of Congress on Evolutionary Computation (CEC'2002), Vol. 2, IEEE Press (2002) 1051-1056.

[12] Coello, C.A.C. and Cortes, N.C.: Solving Multiobjective Optimization Problems using an Artificial Immune System, *Genetic Programming and Evolvable Machines*, Vol.6, No.2, pp.163–190, June 2005.

[13] Li, X.: A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization, in Erick Cant-Paz et al. (editors), Proc. of Genetic and Evolutionary Computation (GECCO'03), Part I, Springer, LNCS 2723, (2003) 37-48.

[14] Li, X.: Better Spread and Convergence: Particle Swarm Multiobjective Optimization using the Maximin Fitness Function. In K. Deb, et al., (eds.) Proc. of Genetic and Evolutionary Computation Conference 2004 (GECCO'04), LNCS 3102. Seattle, USA (2004) 117-128.

[15] Knowles, J.D. and Corne, D.W.: On metrics for comparing non-dominated sets. In Proc. of the 2002 Congress on Evolutionary Computation Conference (CEC'02). IEEE Press (2002) 711-716.

[16] Czyzak, P. and Jaszkiewicz A.: Pareto simulated annealing - a metaheuristic technique for multipleobjective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7, (1998) 34-47.

[17] Balling, R.: The Maximin Fitness Function - Multiobjective City and Regional Planning. In Proc. of EMO 2003, (2003) 1-15.

[18] Hanson, M.P. and Jaszkiewicz, A.: Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Demark, March 1998.

[19] Zitzler, E., Thiele, L. Laumanns, M., Fonseca, C.M. and Fonseca, V.G.d.: Performance assessment of multiobjective optimizers: an analysis and review. In *IEEE Trans. on Evolutionary Computation* 7(2), (2003) 117-132.