Evolutionary Multiobjective Optimization in Dynamic Environments: A Set of Novel Benchmark Functions

Subhodip Biswas, Swagatam Das, Ponnuthurai N. Suganthan and Carlos A. Coello Coello

R1. A New Re-initialization Strategy with the MOEA/D algorithm

Basic Re-initialization strategy

Re-initialization models are basically statistical time series models used to extrapolate the future positions of the population for a quick convergence towards the Pareto optimal Front. If we record solutions in the previous time windows, when a change is detected, the recorded solution set $Q_t, Q_{t-1}, ..., Q_1$ can provide information for predicting the new location of the Pareto set PS_{t+1} at time window *t*+1. We further assume that the location of PS_{t+1} is a function of the locations $Q_t, Q_{t-1}, ..., Q_1 : Q_{t+1} = \psi(Q_t, Q_{t-1}, ..., Q_1)$, where Q_{t+1} denotes the new location of the PS at time step *t*+1. The problem now boils down to: how to use the historical information $Q_t, Q_{t-1}, ..., Q_1$ to re-initialize the population for time window *t* + 1. In practice, the function $\psi(.)$ is not known and must be estimated by using a certain technique. In the following, we discuss how to generate initial solutions for time window *t* + 1.

One important approach for a re-initialization technique is to identify the memory or past states $Q_t, Q_{t-1}, ..., Q_1$ for any particular population member. It is very difficult to identify the relationship between the stored solutions in $Q_t, Q_{t-1}, ..., Q_1$ to build a time series. Zhou *et al.*[R1] adopted a heuristic approach to identify such time series. For a point $\vec{x}_t \in Q_{t,}$ its parent location in the previous time window can be defined as the nearest point in Q_{t-1}, i.e.,

$$\vec{x}_{t-1} = \arg\min_{\vec{y} \in Q_{t-1}} \left\| \vec{y} - \vec{x}_t \right\|_2 . \quad \text{(S1)}$$

The second part of the problem involves defining a relationship between past states to find the future state. It is practically a statistical extrapolation problem, the only issue being that due to computational limitations we cannot take a large number of past memories to predict the present which makes the problem even more difficult to handle. This problem is modeled in [R1] as: suppose that $\vec{x}_1, \vec{x}_2, ..., \vec{x}_t; \vec{x}_i \in Q_i$, $i = 1, 2..., t$ are a series of points (*n*-dimensional vectors) in the decision space and describing the movements of the PS. A generic model to predict the location of the initial individuals for the (*t*+1)-th time window can be formulated as follows: $\vec{x}_{t+1} = \psi(\vec{x}_t, \vec{x}_{t-1}, ..., \vec{x}_{t-K+1})$ where *K* represents the number of the previous time windows that $\vec{x}_{t+1}$ is dependent on in the prediction model. Any time series model can be used for modeling ψ. The following simple linear model was adopted in [R1]:

$$\vec{x}_{t+1} = \psi(\vec{x}_t, \vec{x}_{t-1}) = \vec{x}_t + (\vec{x}_t - \vec{x}_{t-1}). \quad \text{(S2)}$$

Controlled extrapolation and PF-based nearest distance approach

In this paper, we propose another heuristic to map a current population member to its past states. Unlike the method proposed in [R1] that considers the nearest solution in the past PS Q_{t-1} as the past parent of the solution concerned,

we consider here the mapping of the present state with the past, based on the nearest distance in the objective function space i.e., based on the relationships in the PF. Mathematically, it can be defined as:

$$\vec{x}_{t-1} = \arg\min_{\vec{y} \in Q_{t-1}} \left\| f(\vec{y}) - f(\vec{x}_t) \right\|_2 . \quad \text{(S3)}$$

In many problems, as will be shown later, this mapping relationship performs much better than the earlier case.

But the nature of variation in the PF or the PS equations is not always linear. So, the linear model may not be appropriate in cases in which complicated non-linear variations are involved. To overcome this, we suggest a feedback control mechanism to control the perturbation $(\vec{x}_t - \vec{x}_{t-1})$ in each dimension as the function changes with time. Also another important aspect of this scheme is to use very low order memory to perform the prediction. We can model this for the *i*-th decision variable as:

$$x_{t+1}^i = \psi(x_t^i, x_{t-1}^i) = x_t^i + m^i.(x_t^i - x_{t-1}^i), \forall i = 1, 2, ..., n \quad \text{(S4)}$$

where m^i is the scaling factor that assumes different values for different dimensions in a candidate solution $\vec{x}_t$.

In general the nature of the function that causes the dynamicity may vary with time i.e., it may be increasing or decreasing with time or even have an increasing or decreasing gradient (i.e., their second order derivative may be positive or negative). A linear prediction model adds a perturbation, which is simply the difference between the present and immediate past state. In several occasions it may overestimate the change or may underestimate it. To circumvent the problem, we use another past memory of the solution under consideration i.e. $\vec{x}_{t-2}$. The two different perturbations $d_1^i = (x_t^i - x_{t-1}^i), d_2^i = (x_{t-1}^i - x_{t-2}^i), \forall i = 1, 2, ..., n$ give a measure of whether the function is decreasing or increasing or if it is having a decreasing or an increasing slope. This knowledge about the functional changes is very important to predict the next state. For example, if d_1^i is positive, it means that the function is increasing, i.e., $x_{t+1}^i > x_t^i$, but this does not indicate the nature of the change (i.e., whether it has an increasing or a decreasing gradient). But if $\left|d_1^i\right| < \left|d_2^i\right|$, then this indicates that the function has a negative gradient. When the function is increasing or decreasing, the linear model with a perturbation term $(\vec{x}_t - \vec{x}_{t-1})$ adds positive or negative values respectively, but it cannot estimate the change in the gradient. Here, the role of the controlling factor comes into play. The controlling factor m^i increases or decreases the perturbation according to the increasing or decreasing nature of the gradient, respectively. The value of m^i, which is 1 for the linear model, can be varied within bounds both deterministically and stochastically. m^i can be determined in the following way:

Let, $k^i = \left|d_1^i\right| - \left|d_2^i\right|$. Then, for the deterministic model we select:

$$m^i = 1 + \tanh(k^i), \quad \text{(S5a)}$$

and for the stochastic model, we consider:

$$m^i = 1 + k^i \left|N(0,1)\right|, \quad \text{(S5b)}$$

where *N* stands for a normal distribution. We invoke both models with equal probabilities in the following way:

$$\text{If } rand_i(0,1) < 0.5, m^i = 1 + \tanh(k^i); \text{ else } m^i = 1 + k^i \left| N(0,1) \right|, \quad \text{(S5c)}$$

where $rand_i(0,1)$ is a uniformly distributed random number in (0,1) . To check the strength of the prediction models we incorporate these models into MOEA/D [R2].

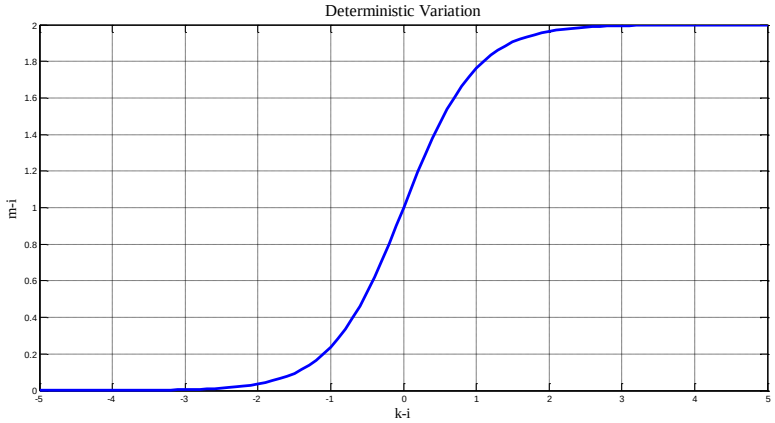


Fig. S1: Deterministic Variation of m^i with k^i : $m^i = 1 + \tanh(k^i)$

In this context, a question may arise regarding the reasons behind choosing such particular functions as in (S5a) and (S5b). We should note that if $k^i = \left|d_1^i\right| - \left|d_2^i\right|$ is zero, the nature of variation is a linear one and the value of m^i should be 1 in that case. For positive values of k^i , m^i should be greater than 1. The more positive k^i is greater should be the value of m^i correspondingly. However, the rate of change of m^i with respect to k^i should be decreasing for higher values of k^i , otherwise the perturbation would make change too drastic to follow the actual variation. Similarly, for negative values of k^i , m^i should be less than 1. The more negative k^i is, lesser should be the value of m^i correspondingly. Nevertheless, the rate of change of m^i with respect to k^i should be decreasing for higher values of k^i .

In (6a), the deterministic functional relationship between m^i and k^i satisfies such criterion. The range of $\tanh(k^i)$ is from -1 to 1, hence the range of m^i is from 0 to 2. If $k^i = \left|d_1^i\right| - \left|d_2^i\right|$ is zero, the nature of variation is a linear one and the value of m^i becomes 1. If k^i is positive, m^i takes some value between 1 and 2. On the other hand, for negative values of k^i , m^i lies in between 0 to 1. In (6b), the stochastic model generates a value of m^i based on k^i . The model, though not pre-determinable, satisfies the requirements mentioned earlier. For $k^i = 0$, $m^i = 1$ as expected for linear extrapolation. It can be seen that the probability of m^i having a value more than 1 or less than 1 depends on k^i and thus controls the extent of perturbation required to re-initialization. Another point to be mentioned here is that since the objective functions we have used are normalized, i.e. in the range 0 to 1, we do not

need to normalize k^i . If this is not case, we have to normalize k^i by some means (for example multiplying suitable constant to keep value of m^i between 0 to 2) to ensure the perturbation caused by the re-initialization is independent of the range of the objectives.

The proposed DMOP solver

In this paper we propose Controlled Extrapolation and PF based nearest distance approach. This re-initialization strategy has been incorporated with the MOEA/D-DE framework to form a complete algorithm.

A pseudo-code representation of the algorithm is provided below:

Input : i) MOP(1) ii) a stopping criterion iii) *N* the number of the subproblems considered in MOEA/DFD iv) a uniform spread of *N* weight vectors: $\lambda^1,\ldots,\lambda^N$ v) *T* : the number of the weight vectors in the neighborhood of each weight vector;

Output : { $\vec{x}^1,\ldots,\vec{x}^N$} and { $F(\vec{x}^1),\ldots,F(\vec{x}^N)$ }

Step 1 Initialization

Step 1.1: Compute the Euclidean distances between any two weight vectors and then find the *T* closest weight vectors to each weight vector to find *B(i)*

Step 1.2: Generate an initial population $\vec{x}^1,\ldots,\vec{x}^N$ in the search space.

Step 1.3: Initialize $\vec{z}$ by setting $\vec{z}$ =min { $f_i(\vec{x}^1), f_i(\vec{x}^2),\ldots, f_i(\vec{x}^N)$}.

Step 1.4: Set *gen*=0 for all $i=\{1,\ldots,N\}$.

Step 1.5: Set Q_t, Q_{t-1}, Q_{t-2} as the initial positions of the individuals.

Step 2: If mod(*gen, T*)=0

Call *Reinitialize;* \\ *Reinitialize* is the function module for prediction based re-initialization\\

Set *P*= Q_{t+1} ;

end

Step 3: For each $i=\{1,\ldots,N\}$ do:

Step 3.1: Selection of mating/update range

Step 3.2: Reproduction: Reproduce the offspring $\vec{u}_i$ corresponding to parent $\vec{x}_i$ by DE/rand/1/bin scheme. For *j*-th component of the *i*-th vector:

$u_{i,j} = x_{r_1^i,j} + F.(x_{r_2^i,j} - x_{r_3^i,j})$, with probability *Cr*

$= x_{i,j}$, with probability 1 – *Cr*,

F being the standard scale factor for DE.

Step 3.3: Repair: If an element of $\vec{y}$ is out of the boundary reset its value.

Step 3.4: Update of z: For each *j*=1,…,*m*, if $z_j > f_j(\vec{y})$, then set $z_j = f_j(\vec{y})$.

Step 3.5: Calculation of fuzzy dominance level:

For each $j \in P$,

For each $i \in \{1,\ldots,m\}$

if $f_i(\vec{y}) \le f_i(\vec{x}^j)$, $\mu_i^j = 1$;

else $\mu_i^j = e^{-A(f_i(\vec{y}) - f_i(\vec{x}^j))}$; end.

$\hat{\mu}^j = \cap \mu_i^j = \prod_{i=1}^{m} \mu_i^j$;end.

$\tau = 0.1 * gen / (\max\_gen)$;end.

Step 3.6: Update of solutions: For each $j \in P$;If $\hat{\mu}^j > \tau$ then , = and $FV^j = F(\ )$; $\vec{y}$

else, if $g^{te}(\vec{y}^j \mid \lambda^j, \bar{z}^*) \leq g^{te}(\vec{x}^j \mid \lambda^j, \bar{z}^*)$, then $\vec{x}^j = \vec{y}$, $FV^j = F(\vec{y})$;

Step 3.7: Update Q_{t-1}, Q_{t-2}

Step 4: If the stopping criteria is satisfied then stop and output.

Step 5: *gen=gen*+1. Go to **Step 2.**

Reinitialize will differ depending on which re-initialization strategy we are using.
We are using the proposed re-initialization strategy based on controlled extrapolation.

Function ***Reinitialize*/CE**

Input: Q_t, Q_{t-1}, Q_{t-2} **:** Current population(*gen=t*) and last population(*gen=t-1*)

Output: Q_{t+1} : Re-initialized population based on prediction model

For each $i = \{1, 2, ..., N\}$ do:

Step 1: find the nearest solution in PF of last generation corresponding to each solution:

$$\vec{x}_{t-1} = \arg\min_{\vec{y} \in Q_{t-1}} \left\| f(\vec{y}) - f(\vec{x}_t) \right\|_2$$

Step 2.1: Calculate $d_1^i = (x_t^i - x_{t-1}^i), d_2^i = (x_{t-1}^i - x_{t-2}^i), \forall i = 1:n$

Step 2.2: Calculate $k^i = \left|d_1^i\right| - \left|d_2^i\right|$

Step 2.3: If $rand_i(0,1) < 0.5$,

Then $m^i = 1 + \tanh(k^i)$

Else $m^i = \dfrac{k^i}{\left|k^i\right|} N(1, \left|k^i\right|)$

Step 2.4: Predict the next position of that solution:

$$x_{t+1}^i = \psi(x_t^i, x_{t-1}^i) = x_t^i + m^i.(x_t^i - x_{t-1}^i), \forall i = 1, 2, ..., n$$

End

Return Q_{t+1}

Experimental Results

Experimental setup

Window size T and step size n_s are important problem parameters which control the time window for the algorithm to converge as well as the severity of the change, respectively. For all the test functions, we take T = 5; n_s=5. The lower the values of these two parameters, the more difficult the problems are as far as their dynamic behavior is concerned. Due to space limitations, detailed results obtained from the variation of window and step sizes are not reported in this paper. However, it is quite obvious that with a higher window size and a larger step size the performance of all the algorithms would improve. Since the UDF series of functions are more challenging, a larger population size is required to achieve satisfactory performance. For UDF1-UDF9, except for UDF7, the population size is kept at 300; for UDF7 (which has three objective functions), the population size is kept at 500. The time variation of the Pareto set is governed by the function:

Evolutionary Multiobjective Optimization in Dynamic Environments: A Set of Novel Benchmark Functions

Subhodip Biswas, Swagatam Das, Ponnuthurai N. Suganthan and Carlos A. Coello Coello

$$G(t) = \sin(0.5\pi t), \text{ where } t = \frac{1}{n_s}\left\lfloor \frac{\tau}{T} \right\rfloor \text{ (} \tau \text{ is the current iteration number)}$$

with a period of 4 in terms of the time variable t i.e. $4 \times T_s \times n_s = 4 \times 5 \times 5 = 100$ iterations. Here, 300 iterations have been taken for all functions. It implies that 3 complete cycles of $G(t)$ have been considered. However, in case of PF variation, $|G(t)|$ is the time dependent factor with a period equal to half of the period of $G(t)$. So, 300 iterations mean 6 cycles as far as the variation of Pareto front is concerned. We consider five Dynamic MO algorithms from the literature and run them over our test-beds to compare their performances and to assess the nature and difficulty of these benchmark functions.

A variant of MOEA/D [R2] was developed using the re-initialization proposed in [R1] and is included in the comparative study: MOEA/D-BR (MOEA/D + PS-based nearest distance +Basic Re-initialization Strategy). Four other well-known dynamic MOEAs are also considered and their performances are investigated over all the benchmark functions. These algorithms include DEMO (direction based method) [R3], DMEA/PRI [R1], DNSGA-II [R4] (based on NSGAII by Deb *et.al.* [R5]) and DQMOO/AR [R6]. DMEA/PRI and MOEAD/BR adopt the same re-initialization strategy, and their only difference is in the basic algorithms employed. As MOEA/D is a very popular MO algorithm, we have incorporated the above mentioned re-initialization scheme to this algorithm. To make a proper comparison of our benchmark functions with that of FDA given by Farina *et.al.* [R3], we also ran the above mentioned algorithms on this test-bed.

Results

For multi-objective problems, one of the most commonly used performance assessment measures is IGD. For dynamic multi-objective optimization, we generally use an average of the IGD values over all the iterations performed [R7]. In Tables 1 and 2, we present the average values of the mean IGD values and their standard deviations over 25 independent runs for the eight algorithmic variants considered.

Table 2: Mean IGD metric values, standard deviations and individual ranks (within parentheses) for benchmarks FDA1-FDA3 and UDF1 – UDF4. The best results are marked in **boldface**.

Function / Strategy	FDA1	FDA2	FDA3	UDF1	UDF2	UDF3	UDF4
DEMO [9]	0.1134± 0.0098 (4)	0.0255± 0.0032 (4)	0.1643± 0.0877 (3)	0.2240± 0.0022 (6)	0.0527± 0.0028 (4)	0.5843± 0.0983 (6)	0.4533± 0.0905 (6)
DMEA/PRI [19]	0.1228± 0.0035 (5)	0.0205± 0.0018 (3)	0.1786± 0.0564 (5)	0.1558± 0.0136 (3)	0.0455± 0.0017 (3)	0.5856± 0.0701 (5)	0.3092± 0.0236 (4)
DNSGA-II [29]	0.1253± 0.0243 (6)	0.0298± 0.0061 (5)	0.1853± 0.8432 (6)	0.2153± 0.0558 (5)	0.0584± 0.0025 (6)	0.5251± 0.0141 (3)	0.3456± 0.0609 (3)

DQMOO/AR [31]	0.1037± 0.0042 (3)	0.0299± 0.0015 (6)	0.1769± 0.0221 (4)	0.1989± 0.0018 (4)	0.0575± 8.359e-4 (5)	0.4775± 0.0387 (2)	**0.1985± 0.0903 (1)**
MOEA/D-BR	0.0932± 0.0082 (2)	**0.0143± 8.544e-4 (1)**	0.1578± 0.0483 (2)	0.1399± 0.0268 (2)	**0.0351± 9.640e-4 (1)**	0.6923± 0.1217 (6)	0.4056± 0.0730 (5)
MOEA/D-CER	**0.0583± 0.0086 (1)**	0.0177± 0.0015 (2)	**0.1405± 0.0516 (1)**	**0.1322± 0.0029 (1)**	0.0358± 0.0017 (2)	**0.4308± 0.0275 (1)**	0.2269± 0.0614 (2)

Tables 1 and 2 provide the mean IGD values for benchmarks UDF1 – UDF9 and FDA1-FDA3. Along with the IGD values we also indicate the rank of each algorithm for that problem in brackets. The last column presents the average rank of all the algorithms over the entire test-bed. A non-parametric statistical test called Wilcoxon's rank sum test for independent samples [R8, R9] was conducted at the 5% significance level in order to judge whether the results obtained with the best performing algorithm differed from the final results of the rest of the competitors in a statistically significant way. *P*-values obtained through the rank sum test between the best algorithm and each of the remaining ones over all the benchmark functions are presented in Table 3. In these tables, NA stands for *Not Applicable* and occurs for the best performing algorithm itself in each case. If the *P*-values are less than 0.05 (5% significance level), there is strong evidence against the null hypothesis, indicating that the better final objective function values achieved by the best algorithm in each case is statistically significant and has not occurred by chance [R9]. The relative performance can also be visualized from the IGD vs. iteration graphs presented in Figures S2(a-f). Due to space limitations, the graphs of all nine functions have not been plotted, and instead, only the important ones are shown. These graphs also show the good performance of the proposed scheme. However, in most cases it is hard to say which algorithm performs best from these graphs. The mean IGD value is (as in Tables 1 & 2) the most evident indicator.

Table 3: Mean IGD metric values, standard deviations and individual ranks (within parentheses) for benchmarks UDF5 – UDF9. The best results are marked in **boldface**.

Function / Strategy	UDF5	UDF6	UDF7	UDF8	UDF9	Final Rank
DEMO [9]	0.0456± 0.0079 (6)	1.7439± 0.0229 (6)	0.5742± 0.0473 (5)	0.5328± 0.1251 (5)	0.1628± 0.0281 (5)	5
DMEA/PRI [19]	0.0433± 0.0024 (5)	1.4639± 0.0308 (4)	0.4836± 0.0948 (4)	0.4910± 0.0728 (4)	0.1496± 0.0826 (4)	4
DNSGA-II [29]	0.0379± 0.0098 (4)	1.5648± 0.0965 (5)	0.6846± 0.0404 (6)	0.5625± 0.0825 (6)	0.1972± 0.0638 (6)	6
DQMOO/AR [31]	0.0276± 7.78e-4 (2)	1.2318± 0.0530 (2)	0.3936± 0.0372 (3)	0.4414± 0.1283 (3)	0.1418± 0.0273 (3)	3
MOEA/D-BR	0.0327± 0.0028 (3)	1.2587± 0.0226 (3)	**0.2317± 0.0073 (1)**	0.4032± 0.1793 (2)	0.1376± 0.0317 (2)	2
MOEA/D-CER	**0.0235± 8.35e-4 (1)**	**1.0080± 0.0406 (1)**	0.2483± 0.0087 (2)	**0.3843± 0.0826 (1)**	**0.1149± 0.0376 (1)**	**1**

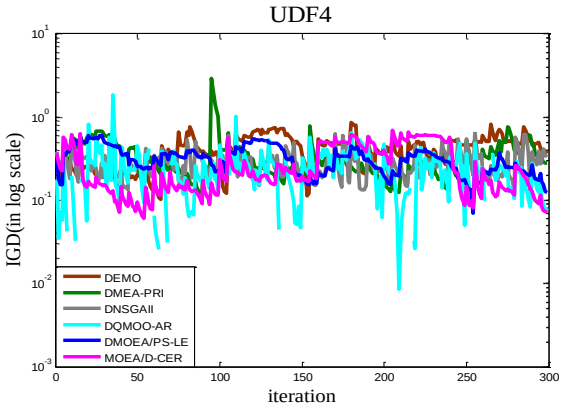
From the results presented in Tables 1 and 2 and the statistical testing done in Table 3, it can be inferred that our proposed algorithm MOEA/D-CER outperforms all the other algorithms with respect to which it was compared. Also, from a total of twelve benchmark functions, our proposed approach outperforms the others in 8 cases and ranks second in four other cases. In case of FDA2 and UDF2, where one has to deal with a polynomial Pareto set, the performance of MOEA/D-CER degrades, but only behind MOEA/D-BR. Also, for the 3-dimensional problems UDF7 and UDF4, MOEA/D-CER ranks second. Thus, we can say that the linear extrapolation strategy for the Pareto set works better than the Controlled Extrapolation strategy on the Pareto front in some limited cases mainly when dealing with polynomial functions. Test problems such as UDF 3, UF4, UDF 6, UDF 7, UDF 8 have higher associated values of the average IGD measure than the other benchmark functions. This correctly points out the difficulty associated with these functions. A closer look reveals that, in general, a trigonometric Pareto set variation is much more difficult than a polynomial one. Also, the curvature variation of the Pareto front makes the problem more challenging than the shifting. Other than this, in general, a discrete front and a 3-dimensional front is always a tough situation to deal with. The non-cyclic changes or rather the non-deterministic changes associated with UDF 8 and UDF 9 make the problem a little bit more challenging. The average IGD values that we obtained while simulating these algorithms over our benchmark test-bed is generally much higher than that obtained in the FDA test-bed. Also, in the FDA test-bed, the average IGD values of each algorithm differ in a smaller proportion with the others compared to our UDF test-bed. That is, the UDF test-bed gives a more diversified result compared to the FDA test-bed, which allows us to clearly distinguish the performance of each algorithm through proper statistical testing. Also, from the above discussion, we can rightly conclude that the UDF test-bed is not only more difficult and challenging than the FDA test-bed, but that it is also more diversified. Finally, from a theoretical point of view, the UDF test-bed consists of a collection of a larger number of changes than the FDA test-bed.

Table 4: *P*-values calculated for Wilcoxon's rank sum test

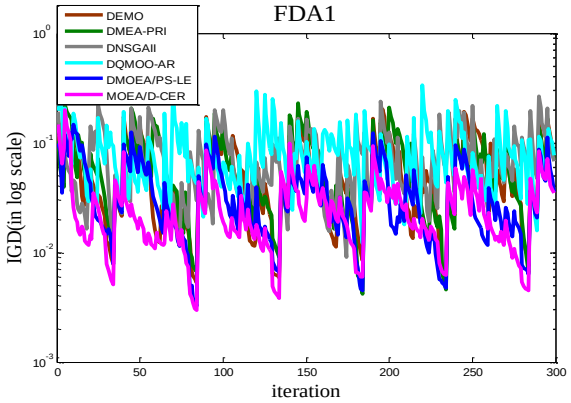
Algo / Func.	DEMO [9]	DMEA/PRI [19]	DNSGA-II [29]	DQMOO/AR [31]	MOEA/D-BR	MOEAD/CER
FDA1	0.0008	0.0038	0.0003	0.0026	0.1173	NA
FDA2	0.0093	0.0482	0.0036	0.0163	NA	0.2163
FDA3	0.0018	0.0128	0.0063	0.0035	0.1234	NA
UDF1	0.0026	0.0027	0.0273	0.0017	0.2192	NA
UDF2	0.0281	0.0027	0.0002	0.0021	NA	0.1963
UDF3	0.0008	0.0284	0.0003	0.1173	0.0620	NA
UDF4	0.0129	0.0182	0.0018	NA	0.0096	0.1973
UDF5	0.0283	0.0092	0.0047	0.1218	0.1273	NA
UDF6	0.0046	0.0138	0.0003	0.0918	0.0936	NA
UDF7	0.0036	0.0928	0.0059	0.0073	NA	0.1836
UDF8	0.0381	0.0483	0.0031	0.0389	0.1730	NA

UDF9	0.0384	0.0083	0.0027	0.0217	0.1129	NA

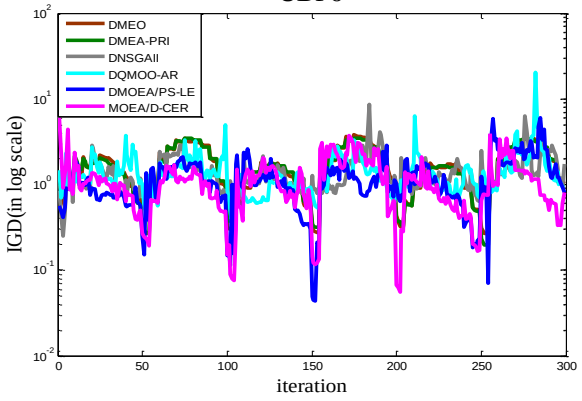
Now, when considering the above results without including our proposed algorithm DQMOO/AR, then MOEA/D-BR has a better performance than the three other algorithms. A closer look at the results reveals that, although MOEA/D-BR performs best in 5 cases and DQMOO/AR performs best in 4 cases, the overall ranking chooses DQMOO/AR as the best algorithm. But the difference between their performances is very narrow. Thus, it can be rightly inferred that both algorithms perform equally good on the test-bed and no single algorithm can outperform the others in most cases. Thus, the benchmark test-bed is quite diverse in nature, since, as expected, different algorithms excel in different types of functions. While DQMOO/AR performs best mainly in those functions with discrete Pareto fronts (UDF3 and UDF6) and in problems with angular shift + curvature variation (UDF4 and UDF5), MOEA/D-BR performs best with mainly three-dimensional Pareto fronts (UDF 7) and in random problems (UDF8 and UDF9).



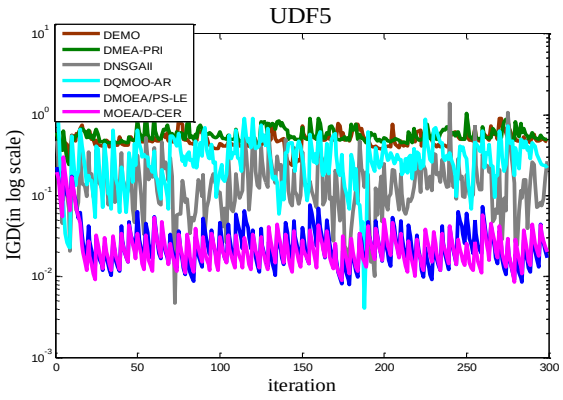
a) FDA1



b) UDF4



c) UDF5

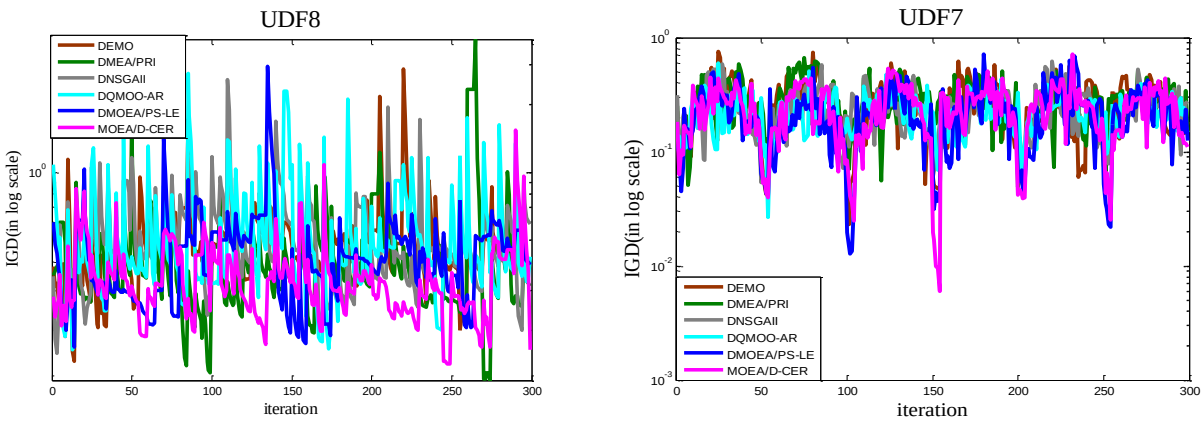


d) UDF6

MOEA/CER

Evolutionary Multiobjective Optimization in Dynamic Environments: A Set of Novel Benchmark Functions

Subhodip Biswas, Swagatam Das, Ponnuthurai N. Suganthan and Carlos A. Coello Coello



e) UDF7 f) UDF8

Fig. S2 (a-f). IGD vs. iteration plots for different algorithms over the tested benchmarks

MOEA/CER

Evolutionary Multiobjective Optimization in Dynamic Environments: A Set of Novel Benchmark Functions

Subhodip Biswas, Swagatam Das, Ponnuthurai N. Suganthan and Carlos A. Coello Coello

References

[R1] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization", *4th Int. Conference on Evolutionary Multi-Criteria Optimization*, LNCS, Springer, 2007.

[R2] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Trans. on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[R3] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications", IEEE *Transactions on Evolutionary Computation*, 8(5):425–442, Oct. 2004.

[R4] K. Deb, U. Bhaskara Rao, and S. Karthik, "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*, Sept. 2006.

[R5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 182-197, 2002.

[R6] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach", In Proc. of *Genetic and Evolutionary Computation Conference* (GECCO 2006), pp. 1201–1208, Seattle, Washington, USA, July 2006.

[R7] X. Li, J. Branke and M. Kirley, "On performance metrics and particle swarm methods for dynamic multiobjective optimization problems", in *Proceeding of Congress of 2007 Evolutionary Computation* (CEC'07), IEEE Service Center, Piscataway, NJ 08855-1331, p.1635 - 1643.

[R8] F. Wilcoxon, "Individual comparisons by ranking methods", *Biometrics*, 1, pp. 80-83, 1945.

[R9] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 3-18, Mar. 2011.