Sampling in Latent Space for a Mulitiobjective Estimation of Distribution Algorithm

Bing Dong, Aimin Zhou and Guixu Zhang
Shanghai Key Laboratory of Multidimensional Information Processing,
Department of Computer Science and Technology,
East China Normal University,500 Dongchuan Road, Shanghai, China
Email: bing@stu.ecnu.edu.cn, {amzhou,gxzhang}@cs.ecnu.edu.cn

***Abstract*—A regularity model-based multiobjective estimation of distribution algorithm (RM-MEDA) has been proposed for continuous multiobjective optimization problems. Generating promising solutions to approximate the population is significant to RM-MEDA. In the reproduction of RM-MEDA, it adopts a Latin square design strategy to sample points in the latent space that is extended to cover the whole Pareto set. However, the setting of the extension scale is problem-dependent to some extent. To circumvent this issue, we propose a differential evolution based sampling (DES) scheme for RM-MEDA. DES mutates the projections of the parent solutions in the latent space to generate promising candidate offspring solutions. The empirical experiment results have shown the significant advantages of the DES scheme comparing to the Latin square design.**

***Index Terms*—RM-MEDA, EDA, multiobjective optimization, latent space**

I. Introduction

Multiobjective optimization problems (MOPs) have arisen in many areas, gaining large attention from researchers with various backgrounds. This paper considers the following MOPs:

$$\begin{array}{ll} \min & F(x) = (f_1(x), \cdots, f_m(x)) \\ \text{s.t} & x \in \Omega \end{array} \tag{1}$$

where $x = (x_1, \cdots, x_n)^T \in R^n$ is a decision variable vector, and $\Omega = \Pi_{i=1}^{n}[a_i, b_i] \subset R^n$ denotes the feasible region of the search space, $f_i : R^n \to R, i = 1, \cdots, m$, is a continuous objective function, and $F(x)$ is an objective vector.

Let a, $b \in R^n$, a is said to dominate b if $a_i \leq b_i$ for all $i = 1, \cdots, n$ and $a \neq b$. A vector $x^* \in \Omega$ is called *Pareto optimal* if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. All the Pareto optimal vectors are called *Pareto set (PS)* and their objective vectors, $PF = \{F(x) \in R^m | x \in PS\}$, is called the *Pareto front (PF)*. The objectives in (1) usually conflict with each other, and consequently no single solution can optimize all the objectives at the same time. In many applications, the decision maker has to balance the relation among the solutions according to the approximation to the PS (PF).

Evolutionary algorithms (EAs) have shown their advantages addressing MOPs [1]–[3]. *Multiobjective evolutionary algorithms (MOEAs)* have advantages over other methods as they can produce a set of Pareto optimal solutions to approximate the PS in a single run. MOEAs have been accepted as the main approach, and a variety of MOEAs have been proposed [4], [5]. The widely used and investigated MOEAs can be categorized into three classes: the Pareto dominance-based [6]–[8], decomposition-based [9]–[11] and performance indicator-based methods [12]–[14].

Estimation of distribution algorithms (EDAs) are one of the most promising EAs [15]. And EDAs have been employed for MOPs with impressive performance [16]–[18]. Unlike conventional evolutionary algorithms, there is no mutation or crossover in EDAs. They use an explicit probabilistic distribution model, such as Bayesian network or multivariate normal distribution, to extract statistical information. Candidate offspring solutions will be sampled from the built model. Then they replace the parent solutions partly or fully in the population of previous generation. It is clear that the sampling strategy is significant to the performance of EDAs.

Under mild smoothness conditions, the PS of a continuous MOP is a piecewise continuous $(m-1)$-dimensional manifold (m is the number of objectives). This regularity property has been applied in the *regularity model based multiobjective estimation of distribution algorithm (RM-MEDA)* [16] for dealing with continuous MOPs. In RM-MEDA, the local principal component analysis is used to extract the solution distribution through a set of hyperplanes. In model sampling, the Latin square design method is firstly utilized to sample points in the extended hyperplanes and then some Gaussian noises are added to the sampled points to form the candidate offspring solutions.

Based on the above considerations, we propose an alternative sampling strategy, named as *differential evolution based sampling (DES)*, for RM-MEDA. The basic idea is to regard the extracted hyperplanes as a latent space and use genetic operators to produce new trail solutions in the latent space. The proposed DES works as follows: it firstly maps the current solutions into the latent space, then it applies the *differential evolution* (DE) [19], [20] to mutate the solutions in the latent space, and finally it projects the newly generated points from the latent space to the decision space. The newly generated points will be added with Gaussian noise to create new trial offspring solutions. A major character of DES is that much part of the reproduction operation is done in the latent space instead of the decision space with utilizing the eigenvector of the population. It should be noted that similar idea has been proposed for single objective optimization [21], [22].

The rest of the paper is organized as follows. Section II introduces the related work. Section III presents the details of the DES scheme and gives the framework of DES assisted RM-MEDA (DES-RM-MEDA). Section IV empirically studies the proposed strategy. Finally, Section V concludes the paper and outlines the future work.

II. Related Work

A. RM-MEDA

Algorithm 1: RM-MEDA Framework

```
1 Initialize a population $Pop(0)$, and set $t = 0$.
2 while not terminate do
3     Modeling: Build the probabilistic model $\delta$ in order to model the distribution of the solutions in $Pop(t)$.
4     Reproduction: Generate a new solution set $Q$ from the probabilistic model.
5     Selection: Select $N$ solutions from $Q \bigcup Pop(t)$ to construct a new population $Pop(t+1)$.
6     $t = t + 1$
7 end
8 Return the solutions in $Pop(t)$.
```

RM-MEDA [16] works as in Algorithm 1, and we would like to make the following comments on RM-MEDA.

- t is the generation counter.
- $Pop(t) = \{x^1, x^2, \cdots x^N\}$ is the population at generation t, which contains N solutions.
- $\delta = \zeta + \varepsilon$ is the probabilistic model built in RM-MEDA. It is assumed that the solutions scatter around an $(m-1)$-dimensional piecewise continuous manifolds, i.e., the PS. Therefore, each solution can be regarded as a random vector $\delta \in R^n$ where ζ is uniformly scattered around the piecewise continuous $(m-1)$-dimensional manifolds, and ε is an n-dimensional noise vector subjects to Gaussian distribution.
- Q is a new solution set sampled from the probabilistic model. A set of $(m-1)$-dimensional hyperplanes are extracted to approximate the PS manifold. The new trial solutions are generated by adding points sampled in the hyperplanes by Latin square design and noises sampled from a Gaussian distribution.

Fig. 1 illustrates the model building and sampling in RM-MEDA in the case that the number of the clusters (K) is 3. Three line segments, N^1, N^2, N^3, are used to approximate the PS. To cover the whole PS, three extended line segments, M^1, M^2 and M^3, are used in the model sampling. The extension scale is set to be 0.25 for all test instances in [16]. However, the extension scale is problem dependent. A large extension may be redundant to cover the PS, and a small extension may be unable to cover the PS. Hence, the selection of the extension scale is hard in practice. In Section IV, we also conduct a systematic experiment to analyze the performance of RM-MEDA with different extension scale settings. This is another reason that motivates us to improve the sampling strategy in RM-MEDA.

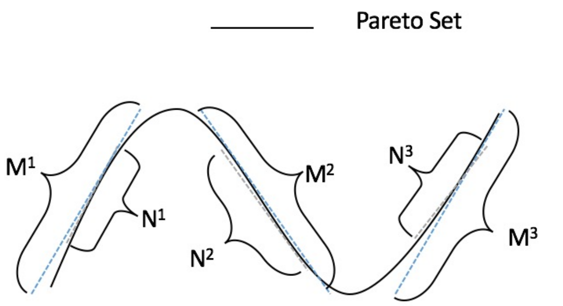


Fig. 1. An illustration of model building and sampling in RM-MEDA.

More details about RM-MEDA are available in [16].

III. DES Assisted RM-MEDA

A. DES

Inspired by the idea of DE [19], [20], the mutation is adopted in the sampling strategy to generate promising solutions.

A modified mutation strategy is employed in this paper. A mutated solution X is generated as

$$X = X_{r_1} + rand \cdot (X_{r_2} - X_{r_3}) + F \cdot (X_{r_2} - X_{r_3}) \qquad (2)$$

where X_{r_1}, X_{r_2}, X_{r_3} are sampled randomly from the population, and r_1, r_2 and r_3 are mutually exclusive integers selected randomly from 1 to NP (NP is the size of the population). F is the scaling factor working in the mutation scheme, and $rand$ is a random number from $[0.0, 1.0]$ subjects to a uniform distribution.

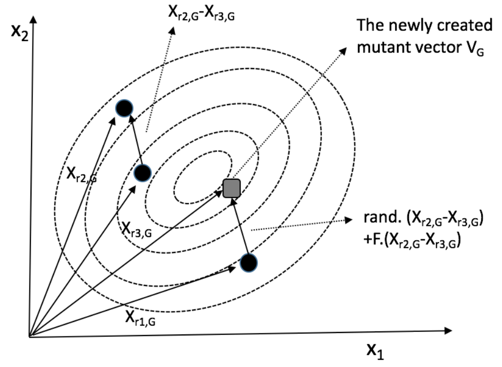

x₂
X_{r2,G}-X_{r3,G}
The newly created mutant vector V_G
X_{r2,G}
X_{r3,G}
rand. (X_{r2,G}-X_{r3,G}) +F.(X_{r2,G}-X_{r3,G})
X_{r1,G}
x₁

Fig. 2. Illustrating the mutation scheme in a 2-D space.

Fig. 2 illustrates the mutation in a 2D-space [23]. It should be noticed that the vector $rand \cdot (X_{r_2} - X_{r_3}) + F \cdot (X_{r_2} - X_{r_3})$ is a random vector that is significant for the diversity of the population. The DE mutation operation can be regarded as sampling a random vector around X_{r_1}. If a set of points are generated around each solution of a population by the DE mutation operator, the set of points can be regarded as a sampled around the current population. It should be noted that distribution of the generated solutions may not have the same distribution as the current population, and this might be worth for investigation.

In RM-MEDA modeling process, the population is partitioned into K clusters by the local principal component analysis. Our new sampling strategy works for each cluster. Basically, DES maps the solutions in the decision space into the latent space, operates mutation in the latent space, and maps the new points back to the decision space, the new generated points are added with the Gaussian noise to produce offspring solutions.

The algorithm framework of DES is presented in Algorithm 2.

Algorithm 2: DES

```
1 Compute the covariance matrix $C$ of the given cluster,
  and decompose it as
      $$C = EDE^T$$
  where $E$ is the eigenvector matrix of $C$, and $D$ is a
  diagonal matrix composed of eigenvalues.
2 For each solution $x$ in the cluster, project it to a latent
  space as
      $$y = x \cdot R.$$
  where $R$ is a matrix that contains the first $(m-1)$
  components of the eigenvector matrix $E$.
3 Generate $M$ points in the latent space as
      $$y' = y_{r_1} + rand \cdot (y_{r_2} - y_{r_3}) + F \cdot (y_{r_2} - y_{r_3})$$
  where $r_1$, $r_2$ and $r_3$ are integers randomly selected from
  $\{1, 2, 3, \cdots, M\}$, $M$ is the size of the cluster.
4 Map $y'$ back to the decision space.
      $$x' = y' \cdot R^T.$$
5 Return the generated solution.
      $$x'' = x' + \varepsilon'$$
  where $\varepsilon'$ is the Gaussian noise subjects to the distribution
  $\mathcal{N}(0, \sigma_\tau I)$ ($\tau \in \{1, 2, \cdots, K\}$ is a randomly generated
  integer).
```

The DES scheme is incorporated in the RM-MEDA framework to generate offspring solutions, and DES-RM-MEDA is presented in Algorithm 3.

Algorithm 3: DES-RM-MEDA

```
1 Initialize a population $Pop(0)$, and set $t = 0$.
2 while not terminate do
3     Modeling: Build the probabilistic model $\delta$ in order to
      model the distribution of the solutions in $Pop(t)$.
4     Reproduction: Partition the population into different
      clusters $C_i$ according to the probabilistic model. For
      each cluster, use DES to generate a set of candidate
      solutions $Q_i$. Set $Q = \cup_i Q_i$.
5     Selection: Select $N$ solutions from $Q \bigcup Pop(t)$ to
      construct a new population $Pop(t+1)$.
6     $t = t + 1$
7 end
8 Return the solutions in $Pop(t)$.
```

IV. Experimental Study

A. Experimental Settings

The 10 test instances, $F1-F10$, introduced in [16] are used as the benchmark problems.

The *inverted generational distance (IGD)* [24] is applied to evaluate the performance of the algorithms. Let P^* be a set of uniformly distributed points in the objective space along the PF. P is assumed to be an approximation of the PF, and the IGD from P to P^* is defined as follows:

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

where $d(v, P)$ is the minimum Euclidean distance between v and any point in P. If P^* is large enough to represent the PF very well, $IGD(P^*, P)$ is able to measure the convergence and diversity of P. The lower the value of $IGD(P^*, P)$ is, the closer P to the PF.

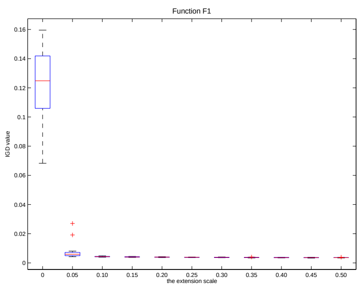
Both RM-MEDA and DES-RM-MEDA are implemented in Matlab and executed in the same computer. In this paper, the experimental setting is as follows:

- *Initialization of the population*: The initial population in algorithms is randomly generated.
- *The number of new trial solutions generated*: The number of new solutions generated at each generation is set to be 100 for instances ($F3$, $F7$, $F9$), and 200 for other instances.
- *The number of decision variables*: The number of decision variables is set to 30 in all algorithms.
- *The number of clusters*: The number of clusters is set to be 5.
- *The scaling factor F*: The scaling factor F is set to be 0.4.
- *The number of runs*: We run each algorithm 30 times independently for every instance.
- *The number of generation*: The number of generation is set to 100 for instances ($F1$, $F2$, $F5$, $F6$), 200 for instances $F4$ and $F8$, and 1000 for instances $F3$, $F7$ and $F9$.

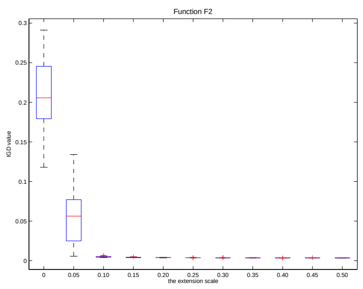
B. Extension Scale in RM-MEDA

Fig. 3 shows the boxplots of the IGD values by RM-MEDA with different extension scale settings from 0 to 0.5 with an interval of 0.05.

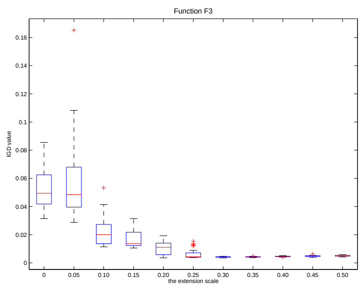
It clears shows that the RM-MEDA does not work well without extension. Furthermore, RM-MEDA with a large extension scale usually works better than RM-MEDA with a small extension scale on all the test instances expect on F10.



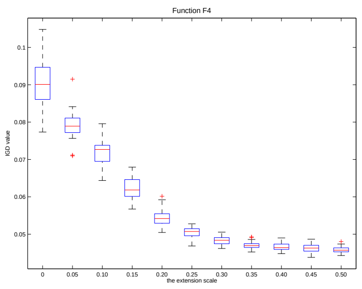
(a) F1



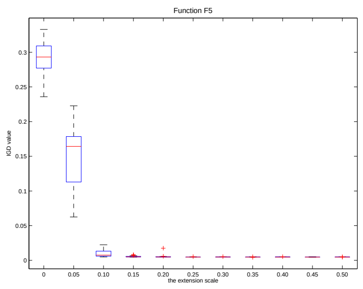
(b) F2



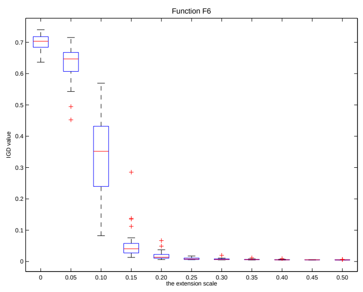
(c) F3



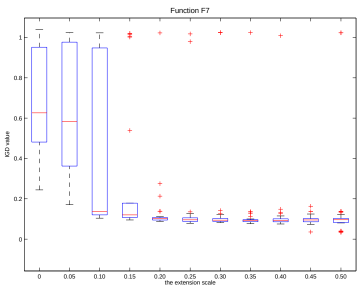
(d) F4



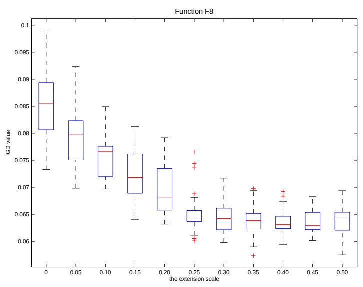
(e) F5



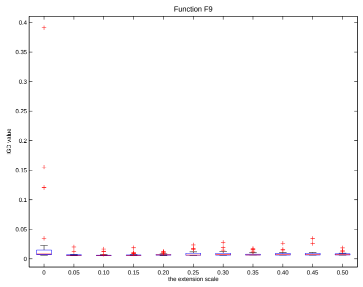
(f) F6



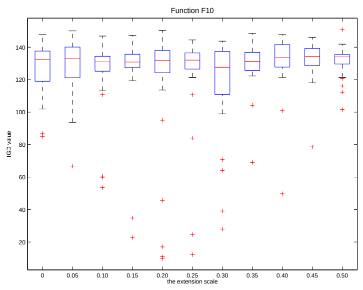
(g) F7



(h) F8



(i) F9



(j) F10

Fig. 3. Boxplots of the IGD values versus extension scale on the results obtained by RM-MEDA over 30 runs.

However, large extension scale may also lead to unstable as shown in Fig. 3(g) and (h).

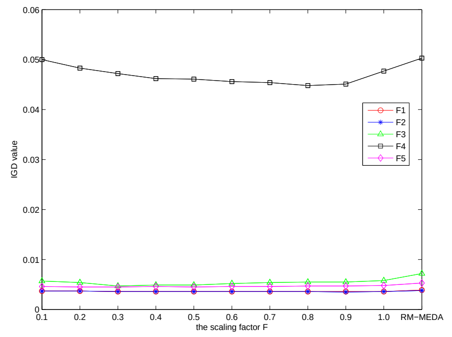
The best extension scale setting for all the instances is not the same. It reflects the extension scale setting is relatively problem-dependent to some extent.

In conclusion, it might be difficult to decide the appropriate extension scale setting in practice if better performance is desired.

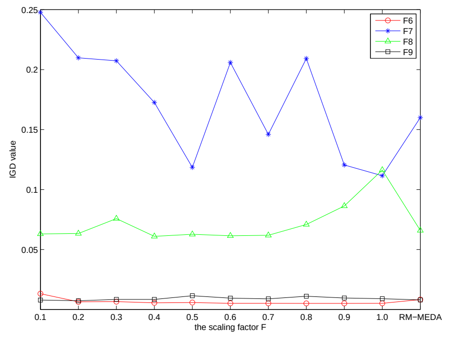
C. *The sensitivity of the scaling factor F*

To verify the sensitivity of the scaling factor F of DES-RM-MEDA, we run the algorithm with the settings of the scaling factor F from 0.1 to 1.0 with an interval of 0.1 and compare them with RM-MEDA.

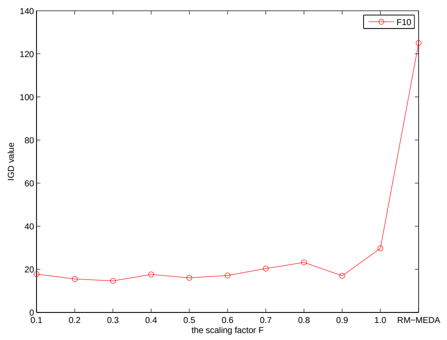
Fig. 4 shows the experimental results. It demonstrates the insensitivity of the scaling factor F. On most instances except $F7$, the performance curve of DES-RM-MEDA with different settings of the scaling factor F is smooth. And it is noteworthy that the comparison between the performance of DES-RM-MEDA and RM-MEDA is distinct.



(a)



(b)



(c)

Fig. 4. Influence of the scaling factor F in DES-RM-MEDA on F1-F10.

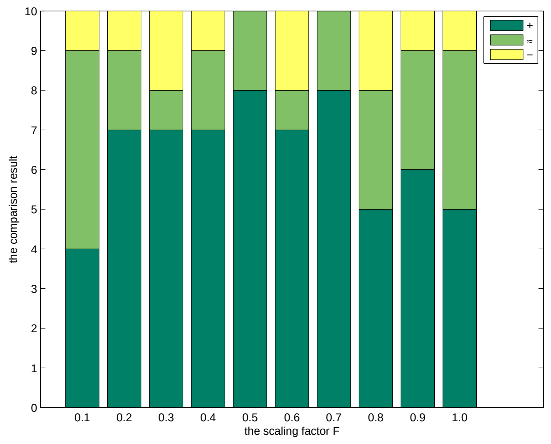


Fig. 5. ”+”, ”≈”, and ”−” denotes that the performance of DES-RM-MEDA with the corresponding setting of the scaling factor F is better than, similar to, and worse than that of RM-MEDA.

DES-RM-MEDA with the settings of the scaling factor $F = 0.2, 0.5, 0.7$ outperforms RM-MEDA on 9 instances. And the DES-RM-MEDA with setting the scaling factor $F = 0.1, 0.4, 0.6, 0.9$ works better than RM-MEDA on 8 instances. With other settings of the scaling factor F, the IGD value of DES-RM-MEDA is lower than that of RM-MEDA on 7 instances.

For instances $F1$, $F2$, $F3$, $F4$ and $F5$, DES-RM-MEDA with any setting of the scaling factor F outperforms RM-MEDA. Except setting the scaling factor F to be 0.1, DES-RM-MEDA performs better than RM-MEDA on instance $F6$. And on $F7$, $F8$ and $F9$, DES-RM-MEDA dose not work as well as RM-MEDA. On the instance $F10$, the difference between RM-MEDA and DES-RM-MEDA with any setting of the scaling factor F is distinct. Hence, it is comprehensive evidence to prove the advantages of DES-RM-MEDA and the insensitivity of the scaling factor F.

To evaluate the performance of DES-RM-MEDA with different scaling factor F settings in a deeper level, Wilcoxon's rank sum test at a 0.05 significance level is performed between RM-MEDA and DES-RM-MEDA with different settings of the scaling factor F. The statistical result is presented in Figure 5 as a stacked bar, and it demonstrates the advantages of DES-RM-MEDA from another aspect. With most settings of the scaling factor F, DES-RM-MEDA is better than RM-MEDA on more than 4 instances. RM-MEDA only outperforms DES-RM-MEDA on one or two instances. For DES-RM-MEDA with the setting of the scaling factor F (0.5 and 0.7), RM-MEDA is not better than DES-RM-MEDA on all instances. On the whole, DES-RM-MEDA with different settings of the scaling factor F performs impressively better in comparison with RM-MEDA.

Thus, the performance of DES-RM-MEDA with different settings of the scaling factor F is remarkable according to the above systematic comparison. In conclusion, the setting of the scaling factor F of DES-RM-MEDA is considerably insensitive, thus we can select the setting of the scaling factor F in the interval [0.2,0.9].

D. Comparison Study

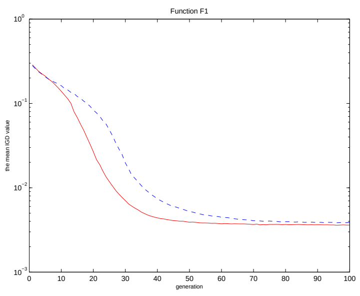
This section compares RM-MEDA and DES-RM-MEDA on the test instances. Fig. 6 shows run time performance curves obtained by the two algorithms. Table I presents the statistical comparison on the final obtained results.

From Table I, we can see that DES-RM-MEDA outperforms RM-MEDA on 8 instances, and the performance of DES-RM-MEDA is similar to that of RM-MEDA on 2 instances. It is noteworthy that DES-RM-MEDA has a remarkable performance on hard instances, $F3$ and $F10$.

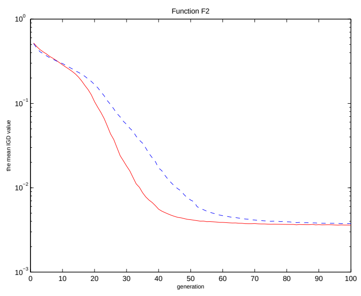
The run time performance curves in Fig. 6 also indicate that DES-RM-MEDA converges faster than RM-MEDA on most of the test instances. This suggests the proposed DES strategy can produce better solutions than the original Latin square design.

We further analyze the results as follows.

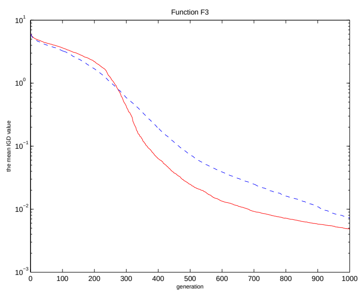
1) Instances with two objectives: For instances with two objectives, DES-RM-MEDA has an impressive performance. On easy instances, such as $F1$ and $F2$, and hard instances, such as $F3$ and $F10$, DES-RM-MEDA has a much better performance according to both the convergence speed and the quality of the final obtained approximations. $F9$ is a relatively simple instance, and the two algorithms performs similarly. On instance $F7$,



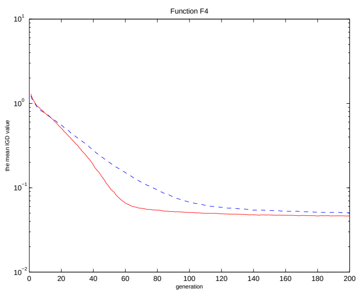
(a) F1



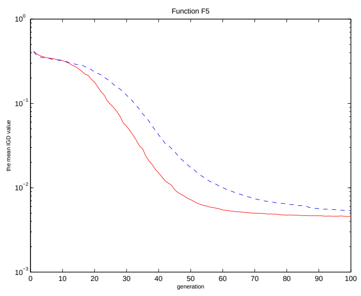
(b) F2



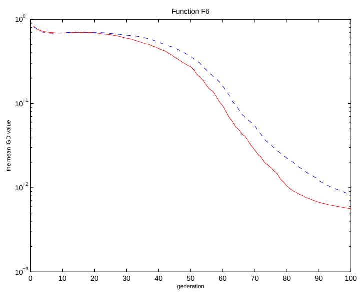
(c) F3



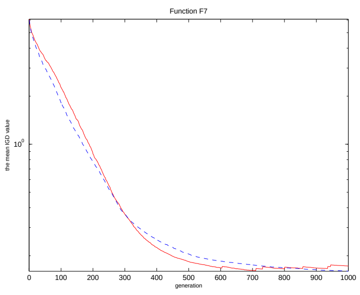
(d) F4



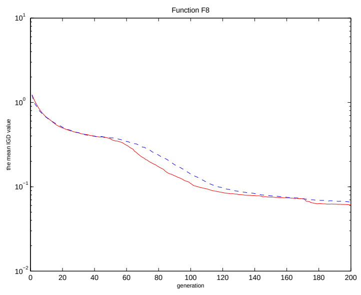
(e) F5



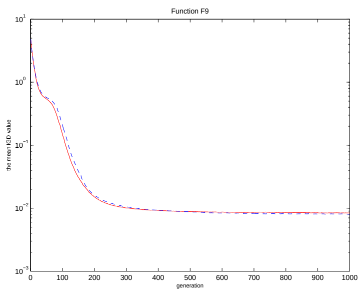
(f) F6



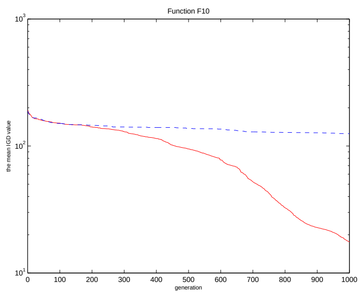
(g) F7



(h) F8



(i) F9



(j) F10

Fig. 6. The mean IGD values versus generations obtained by two algorithms over 30 runs. The solid curves are with DES-RM-MEDA, and the dash curves are with RM-MEDA.

DES-RM-MEDA has a better performance in the middle stages, while in the latter stages, DES-RM-MEDA does not perform steadily.

2) Instances with three objectives: $F4$ and $F8$ have three objectives. On these two instances, DES-RM-MEDA converges faster. But in the later stages, the two algorithms tend to perform similarly.

From the statistical results, we can conclude that DES-RM-MEDA outperforms RM-MEDA on most of the instances. Since the only difference between the two algorithms is on the sampling strategy, these results suggest that the newly proposed DES strategy can help to generate better solutions.

V. Conclusions and Future Remarks

This paper proposed a *differential evolution based sampling (DES)* scheme to generate points in the latent space. The basic idea is to project the parent solutions into the latent space, use a DE mutation operator to generate new points in the latent space based on the projected points, and finally map the points back to the decision space added with the Gaussian noise

TABLE I

STATISTICAL RESULTS OF THE IGD VALUES OF THE FINAL POPULATIONS OBTAINED BY RM-MEDA AND DES-RM-MEDA ON THE 10 TEST INSTANCES OVER 30 RUNS.

instance	RM-MEDA				DES-RM-MEDA			
	mean	std.	best	worst	mean	std.	best	worst
$F1$	$3.90e-03$	$1.39e-04$	$3.70e-03$	$4.20e-03$	**3.60e-03**	$9.16e-05$	**3.43e-03**	**3.77e-03**
$F2$	$3.80e-03$	$1.43e-04$	$3.50e-03$	$4.10e-03$	**3.60e-03**	$1.01e-04$	**3.35e-03**	**3.82e-03**
$F3$	$7.20e-03$	$3.90e-03$	$3.60e-03$	$1.55e-02$	**4.90e-03**	$8.09e-4$	$3.90e-03$	**7.31e-03**
$F4$	$5.03e-02$	$1.30e-03$	$4.82e-02$	$5.35e-02$	**4.62e-03**	$9.32-04$	**4.44e-02**	**4.85e-02**
$F5$	$5.30e-03$	$3.00e-03$	$4.40e-03$	$2.12e-02$	**4.60e-03**	$1.49e-04$	**4.33e-03**	**4.96e-03**
$F6$	$8.30e-03$	$2.10e-03$	$5.70e-03$	$1.50e-02$	**5.60e-03**	$9.04e-04$	**4.52e-03**	**8.30e-03**
$F7$	$1.60e-01$	$2.35e-01$	$7.96e-02$	$1.03e+0$	$1.73e-01$	$2.28e-01$	**3.20e-02**	**1.02e+0**
$F8$	$6.59e-02$	$3.50e-03$	$6.05e-02$	$7.69e-02$	**6.10e-02**	$2.03e-03$	**5.67e-02**	**6.39e-02**
$F9$	$8.00e-03$	$2.80e-03$	$5.80e-03$	$1.48e-02$	$8.40e-03$	$3.20e-03$	**5.51e-03**	$2.12e-02$
$F10$	$1.25e+02$	$2.35e+01$	$2.27e+01$	$1.44e+02$	**1.76e+0**	$1.29e+01$	**4.73e+0**	**7.15e+01**

[1] The bolder ones mean better.

to generate offspring solutions. The DES scheme is applied in RM-MEDA. A preliminary study has indicated that the DES scheme outperforms the Latin square design method on sampling new solutions. The reason might be that DES can generate more diverse points than the Latin square design.

The preliminary study in this paper has shown the possibility to do reproduction in the latent space, which has much lower dimension than the decision space. There still needs much work to investigate the latent space sampling strategy, and possible directions include: (a) trying other reproduction operators in the latent space, and (b) applying the DES scheme to other MOEAs.

REFERENCES

[1] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2002, vol. 242.

[2] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective evolutionary algorithms and applications*. Springer Science & Business Media, 2006.

[3] J. Knowles and D. Corne, "Memetic algorithms for multiobjective optimization: issues, methods and prospects," in *Recent advances in memetic algorithms*. Springer, 2005, pp. 313–352.

[4] K. Deb, "Multi-objective evolutionary algorithms," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 995–1015.

[5] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

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

[7] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 19, no. 2, pp. 201–213, 2015.

[8] R. Filomeno Coelho, "Probabilistic dominance in multiobjective reliability-based optimization: Theory and implementation," *Evolutionary Computation, IEEE Transactions on*, vol. 19, no. 2, pp. 214–224, 2015.

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

[10] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 284–302, 2009.

[11] S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 3, pp. 442–446, 2012.

[12] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary computation*, vol. 19, no. 1, pp. 45–76, 2011.

[13] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications," *Theoretical Computer Science*, vol. 425, pp. 75–103, 2012.

[14] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669.

[15] P. Larranaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media, 2002, vol. 2.

[16] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

[17] T. Okabe, Y. Jin, B. Sendoff, and M. Olhofer, "Voronoi-based estimation of distribution algorithm for multi-objective optimization," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2. IEEE, 2004, pp. 1594–1601.

[18] M. Costa and E. Minisci, "MOPED: a multi-objective parzen-based estimation of distribution algorithm for continuous problems," in *Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 282–294.

[19] R. Storn and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995, vol. 3.

[20] ——, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[21] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *Evolutionary Computation, IEEE Transactions on*, vol. 19, no. 1, pp. 31–49, 2015.

[22] Y. Wang, H.-X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Applied Soft Computing*, vol. 18, pp. 232–247, 2014.

[23] S. Das and P. N. Suganthan, “Differential evolution: a survey of the state-of-the-art,” *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, 2011.

[24] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, “Performance assessment of multiobjective optimizers: an analysis and review,” *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.