2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

Multi-Objective Differential Evolution with Self-Navigation

Ke Li, Sam Kwong, Ran Wang and Jingjing Cao
Department of Computer Science
City University of Hong Kong
Hong Kong SAR
keli.genius@gmail.com, cssamk@cityu.edu.hk

Imre J. Rudas
Institute of Intelligent Engineering Systems
Óbuda University
Budapest, Hungary
rudas@uni-obuda.hu

***Abstract*—Traditional differential evolution (DE) mutation operators explore the search space with no considering the information about the search directions, which results in a purely stochastic behavior. This paper presents a DE variant with self-navigation ability for multi-objective optimization (MODE/SN). It maintains a pool of well designed DE mutation operators with distinct search behaviors and applies them in an adaptive way according to the feedback information from the optimization process. Moreover, we deploy the neural network, which is trained by the extreme learning machine, for mapping an artificially generated solution in the objective space back into the decision space. Empirical results demonstrate that MODE/SN outperforms several state-of-the-art algorithms on a set of benchmark problems with variable linkages.**

***Index Terms*—Differential evolution, neural network, multi-objective evolutionary algorithm (MOEA)**

I. INTRODUCTION

Differential Evolution (DE), proposed by Storn and Price [1], is a popular and efficient population based, direct heuristic for solving global optimization problems in the continuous search space. The main benefits brought by DE are its simple structure, ease of use, fast convergence speed and robustness, which enable it to be widely applied to many real-world applications. For the generation of new solutions (trial vectors), each individual (target vector) is combined with others by means of different forms of weighted sums (mutation operators). A trial vector can replace the target vector only if it has a better fitness value. The aim of these iterations is basically to find a proper direction for the search process towards the optimum, by following the quality distribution of solutions in the current population.

One of the possible application domains of DE are the *Multi-objective Optimization Problems* (MOPs), which exist everywhere in real-world applications, such as engineering, financial, and scientific computing. Many different DE variants proposed to tackle MOPs can be found in the literature, such as PDE [2], GDE3 [3], etc. We refer readers to [4] for a recent comprehensive survey of DE, including its application to MOPs. However, most of the DE variants for MOPs are only simple grafts from global optimization cases, without the consideration of the properties of MOPs themselves. Specifically, it requires the algorithm to obtain an approximated solutions set with not only well convergence to the *Pareto Front* (PF), but also diversified and uniform distribution along this PF. By contrast, the primary goal of the global optimization is merely locating the single global optimum. Besides, the traditional DE mutation operators have not been equipped with any orientation information. Instead, they are usually designed with "single process", which either to explore the entire search space as much as possible, or to exploit some selected region exclusively. As a matter of fact, this kind of sightless search would be rather hazardous in some circumstances, especially in complicated optimization scenarios, such as problems with variable linkages. For example, when solutions have already approached the PF in the current time point, an explorative operator might generate offspring that are far away from the PF, which contributes to the typical degradation phenomenon. On the other hand, the exploitive operator can hardly help solutions jump out from the local optima.

Based on the above discussions, this paper presents a DE variant for MOPs with self-navigation ability (MODE/SN). It aims at equipping the orginal stochastic search behavior of DE with some intelligence, which can provide effective orientation information to solutions during the optimization process. Specifically, MODE/SN maintains an operators pool, which consists of four well designed DE mutation operators with the following distinct search behaviors:

- Explore the search space as much as possible;
- Exploit some selected elite solutions;
- Provide some guidance towards the convergence direction, when solutions are far away from the PF;
- Guide solutions that reside in the crowded regions move towards some relatively sparse areas.

In addition, an adaptive operator selection technique is employed as the coordinator to select an appropriate operator for offspring generation in each time point, according to the properties of the current fitness landscape. It is worth noting that, in order to provide elite solutions for orientation explicitly, we deploy a neural network, which is trained by the extreme learning machine [5], for learning and capturing the function that maps from the objective space to the decision space. In summary, solutions in MODE/SN can evolve with appropriate search behaviors during the entire optimization process.

The remainder of this paper is organized as follows. section II provides the detailed algorithmic description of the proposed method. After that, numerical experiments on

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

some benchmark problems are studied in section III. Finally, section IV concludes this paper and highlight some future directions.

II. Multi-Objective DE with Self-Navigation

In this section, we outline the framework of MODE/SN and discuss each step of the algorithm in detail.

A. Preliminaries

In order to provide solutions effective orientation information, it is necessary to have a general recognition on the distribution of solutions timely. Motivated by this consideration, we would like to employ a grid structure to determine the location of a solution in this study. Moreover, the size and location of the grid is adaptively updated during the optimization progress. Here we only detail the grid setup in

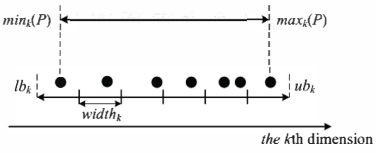


Fig. 1: The grid setup in the kth dimension

one dimensional case, without loss of generality. At first, the minimum and maximum values on the kth dimension are found and denoted as $min_k(P)$ and $max_k(P)$, respectively. Based on these two extreme values, the lower and upper boundaries of the grid on that dimension are determined as:

$$lb_k = min_k(P) - (max_k(P) - min_k(P))/(2 \times div) \quad (1)$$

$$ub_k = max_k(P) + (max_k(P) - min_k(P))/(2 \times div) \quad (2)$$

where div is a predefined parameter ($div = 6$ in Fig. 1) that determines the number of divisions on each dimension. In this case, the original m-dimensional objective space is thus divided into div^m grid cells with equal size. The width of a grid cell on the kth dimension (denoted as $width_k$) is calculated as:

$$width_k = (ub_k - lb_k)/div \quad (3)$$

As for a solution x_i, the coordinate of the grid cell that it resides in on the kth dimension is determined as:

$$C_k(x_i) = \lfloor (obj_k(x_i) - lb_k)/width_k \rfloor \quad (4)$$

where $obj_k(x_i)$ is the objective value of solution x_i on the kth dimension. Take the solution distribution in Fig. 1 as an example, the coordinates of grid cells for all solutions on the kth dimension are $0, 1, 2, 3, 4, 4$ and 5, respectively.

Definition 1 (Grid dominate relationship): Let solutions x_i and x_j reside in different grid cells, say grid cell A and grid cell B, respectively. A dominates B (denoted as $A \preceq_G B$) if and only if both the following two conditions are met:

1) The coordinate of A is no worse than that of B on all dimensions, i.e. $C_k(x_i) \leq C_k(x_j)$, where $k \in \{1, 2, \ldots, m\}$;
2) The coordinate of A is strictly better than that of B on at least one dimensions, i.e. $\exists l \in \{1, 2, \ldots, m\} : C_l(x_i) < C_l(x_j)$.

B. DE Mutation Operators with Orientation Information

As introduced in section I, four different DE mutation operators with distinct search behaviors are proposed in this following paragraphs.

1) Operator 1 (Exploration Oriented): In this paper, the classical "DE/rand/1" [1] is employed to fulfill the first task given previously, i.e. explore the entire search space as much as possible. This operator is featured by the slow convergence speed but strong exploration ability. For the current target vector x_i, it is formulated as:

$$u_{i,j} = x_{1,j} + F \times (x_{2,j} - x_{3,j}) \quad (5)$$

where $j \in \{1, 2, \cdots, n\}$ and n is the number of decision variables, and x_i, x_1, x_2 and x_3 are different from each other. The scaling factor $F > 0$ controls the impact of the vector differences on the mutant vector.

2) Operator 2 (Exploitation Oriented): This operator aims at guiding x_i exploit some selected elite solution x_{elite}. There are two important issues for designing this operator, one is the determination of x_{elite} and the other is the generation of the direction guiding x_i to move towards x_{elite}. As introduced in section II-A, the search space is artificially divided into several grid cells with equal size at each search step. It is obvious that the grid cell with many solutions resided in is usually not preferred, as it means a crowded distributed region. Thus, we apply the roulette wheel selection technique to select a less crowded grid cell at first. Then, within this selected grid cell, another round of roulette wheel selection is used to select x_{elite} (the black circle in Fig. 2(a)), based on the fitness values. On the other hand, in order to generate a direction guides x_i to move toward x_{elite}, we have to select another solution on the same side of x_i with respect to x_{elite}. Take Fig. 2(a) as an example, x_r is on the left side of x_{elite}, thus x_r is selected on this same side in between x_i and x_{elite} at random. In general, Operator 2 is formulated as follows:

$$u_{i,j} = x_{i,j} + F \times (x_{elite,j} - x_{r,j}) \quad (6)$$

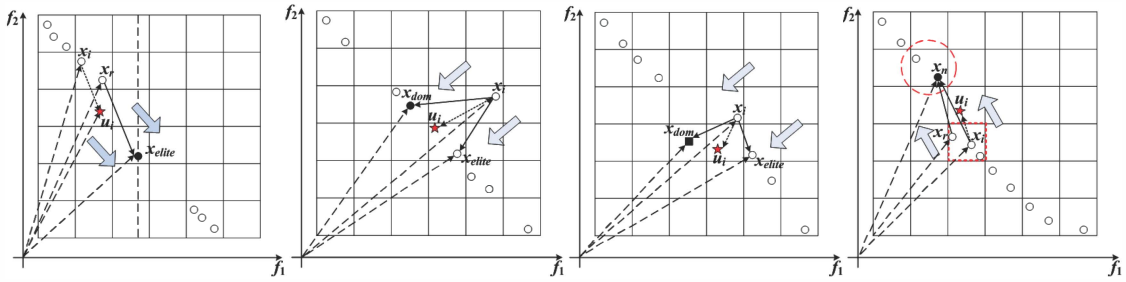
where $j \in \{1, 2, \cdots, n\}$ and n is the number of decision variables.

3) Operator 3 (Convergence Guided): As the name suggested, the behavior of this operator is to guide x_i to move towards the direction of PF, so that the pressure for the improvement of convergence can be constantly provided in this case. The formula of Operator 3 is given as follows:

$$u_{i,j} = x_{i,j} + F \times (x_{dom,j} - x_{i,j}) + F \times (x_{elite} - x_{i,j}) \quad (7)$$

where $j \in \{1, 2, \cdots, n\}$ and n is the number of decision variables.

In practise, however, it is never known prior the exact position of PF before the optimization. Nevertheless, the directional information, which points to the direction of convergence, can be obtained by the dominated solution points to its dominator. This direction is relatively easy to obtain if x_i itself is a dominated solution in the population, otherwise, it is far from trivial to know which direction can improve the convergence.



(a) Operator 2 (b) Operator 3 (dominated) (c) Operator 3 (non-dominated) (d) Operator 4

Fig. 2: Intuitive illustration on different DE mutation operators

To this end, we scan the entire population to find out whether there are solutions dominating x_i at first. On the one hand, the solution which is farthest to x_i and dominate it is chosen as x_{dom} (the black circle in Fig. 2(b)), when x_i is a dominated solution. Afterwards, the grid cell (denoted as *B*) which is nearest to that x_i resided in (denoted as *A*) and with the least crowd is chosen, and a solution x_{elite} is randomly selected from *B*. Then the two vectorial differences in equation (7) and their linear combination generates the directional vector pointing to the convergence direction. At last, this vector is further combined with x_i thus generate the trial vector u_i. On the other hand, it is difficult to generate the directional vector pointing to the convergence direction if x_i itself is a non-dominated solution. In this case, first of all, we locate the non-dominated grid cell which is nearest to the grid cell (denoted as *A*) that x_i resided in and with the least crowd. Then a solution x_{elite} is chosen from this selected grid cell in a random manner. Afterwards, the solution x_{dom} that dominates x_i is generated artificially in the objective space. Generally speaking, infinite number of solutions can be generated to dominate x_i. In this paper, the solution that resides in the center of the grid cell (denoted as *B*) which dominates *A* is generated (the black square in Fig. 2(c)). In particular, the coordinates of *B* is determined as follows:

$$C_k(x_{dom}) = C_k(x_i) - 1 \tag{8}$$

Then the coordinates of x_{dom} in the objective space can be calculated as follows:

$$obj_k(x_{dom}) = lb_k + C_k(x_{dom}) \times width_k + \frac{width_k}{2} \tag{9}$$

Afterwards, x_{dom} is mapped back from the objective space to the decision space, according to the global model built by a trained neural network. This technique would be detailed in section II-C.

4) Operator 4 (Diversity Guided): This operator tries to guide x_i move towards the region that is relatively less crowded. In this case, the most important thing is the detection of gaps between x_i and its neighbors. The grid cell (denoted as *A* and marked as the dashed red square in Fig. 2(d)) that x_i resided in is located at first. Afterwards, the Euclidean distances between *A* and the other non-empty grid cells in the objective space are calculated based on the coordinates of those grid cells. Then those non-empty grid cells are sorted in ascending order based on these calculated Euclidean distances. The nearest grid cells in the left, right, up and down sides, with respect to *A*, are located separately, and the one that is farthest to *A* is chosen as the neighbor grid cell (denoted as *B* and marked as the dashed red circle in Fig. 2(d)). If all four nearest grid cells are with equal distance to *A*, we choose the least crowded one as *B*. Next, solutions x_r and x_n are selected from *A* and *B*, respectively, in a random manner. x_r is just x_i whenever only one solution contained in *A*. The vectorial differences between x_n and x_i, x_r generate two vectors whose linear combination gives the direction towards the relatively less crowded vicinity of x_i. The formula of this operator is give as follows:

$$u_{i,j} = x_{i,j} + F \times (x_{n,j} - x_{i,j}) + F \times (x_{n,j} - x_{r,j}) \tag{10}$$

where $j \in \{1, 2, \cdots, n\}$ and n is the number of decision variables.

C. Mapping from Objective Space to Decision Space

As discussed in section II-B, the implementation of Operator 3 might need to generate a non-dominated solution in the objective space artificially. However, it is far from trivial to know the decision variables correspond to the generated solution explicitly. Inspired by [6], we deploy a neural network to capture the function which maps a certain objective solution back to its corresponding decision variables. This is achieved by training a neural network with the objective function values as inputs and their corresponding decision variables as outputs. Neural network is a useful modeling technique to model and capture the pattern of data, so that to produce some predictions for the parameter values of unknown systems. The neural network needs to be trained to achieve desirable prediction accuracies. In our case, the training data is the whole set of objective function values obtained within a single run of a MOEA. Moreover, as the same in [6], the training of the neural network takes place offline before the execution of our algorithm. In this case, a so called global model can be well established by this trained neural network, which has a "good knowledge" about the global landscape of the objective and decision space of a certain optimization problem. During the execution of our algorithm, the trained neural network is then

used to provide a more or less accurate prediction, whenever an artificially generated solution is required to be mapped from the objective space back into the decision space. If a invalid decision variable value is produced by this mapping, it is reflected back to its nearest value in its domain of definition.

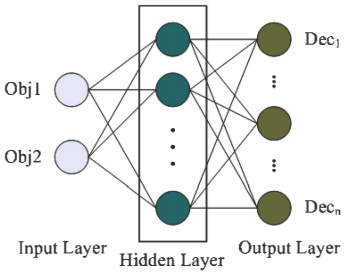


Fig. 3: The grid setup in the kth dimension

As shown in Fig. 3, a single hidden layer feed-forward network is deployed in this paper as the structure of our neural network. The extreme learning machine is employed as the learning algorithm for training this neural network, in view of its reported good results [5]. The number of input neurons is the number of objective functions, while the number of output neurons is the number of decision variables. The training samples for this neural network are data of objective function values and their corresponding decision variables obtained from all the executions of NSGA-II [7] within 500 generations.

D. Adaptive Operator Selection

As discussed earlier, the appropriate operator for a solution is determined by the fitness landscape that is currently explored, which might be different from time to time during the optimization process. The four DE mutation operators proposed in section II-B have different behaviors on exploring the search space, which, in turn, provide solutions with specific orientation information. By adaptively selecting the appropriate opeartor, according to the current fitness landscapes and the population distribution, we can provide sufficient guidelines to the solutions to move towards the desired regions, thus, can enable the algorithm to perform constantly well in different circumstances. In order to meet the aforementioned objectives, we propose an *Adaptive Operator Selection* (AOS) paradigm to implement an autonomous control of which operator should be applied at each time point of the optimization process, based on their recent performances. The AOS paradigm here generally consists of two components: the credit assignment scheme which defines how an operator should be rewarded based on its recent performance; the operator selection mechanism that decides which operator should be applied next.

1) Credit Assignment: Here we improve the credit assignment module proposed in [8], and the impact of the application of operator a is evaluated as follows:

$$\eta_a = f_{best} \times \frac{|pf - cf|}{cf} \tag{11}$$

where f_{best} is the best fitness value in the current population; pf and cf are the fitness values of the target vector and its offspring, respectively. In case no improvement has been achieved (i.e. $pf - cf \leq 0$), η_a is set to zero. It is worth noting that evaluating the fitness value of a solution is pretty difficult in solving MOPs, as some solutions might be incomparable from each other. Here we employ the efficient fitness assignment scheme proposed in [9], in view of its reported good performance.

All the impacts achieved by the application of operator a during each generation g are stored in a specific memory R_a. At the end of each generation g, a unique credit (or reward) value is assigned to each operator. It is calculated as the combination of the average of all impacts operator a has achieved and its success rate:

$$r_a(g) = \sum_{i=1}^{|R_a|} \frac{R_a(i)}{|R_a|} + \frac{success}{total} \tag{12}$$

where $success$ is the number of times that the offspring generated by the application of operator a at generation g can survived to the next generation; while $total$ is the number of times that operator a has already been applied. In our original proposal in [8], only the average term is considered as the credit value. However, due to the incomparable characteristic of MOP itself, it is difficult to precisely define the advantage achieved by the application of an operator merely based on the fitness value. Thus, here we add another item, i.e. success rate of an operator, to aggregate more information to the credit assignment module.

2) Operator Selection: The operator selection mechanism used here is the *Probability Matching* (PM) [10]. Formally, let the operators pool be denoted by $S = \{s_1, \ldots, s_K\}$ where $K > 1$. The probability vector $P(g) = \{p_1(g), \ldots, p_K(g)\} (\forall t : p_{min} \leq p_i(g) \leq 1; \sum_{i=1}^{K} p_i(g) = 1)$ represents the selection probability of each operator at generation g. At the end of generation g, the empirical quality estimate $q_a(g)$ of operator a is updated as [11]:

$$q_a(g+1) = q_a(g) + \alpha \times [r_a(g) - q_a(g)] \tag{13}$$

where $\alpha \in (0, 1]$ is the adaptation rate; the selection probability is updated as:

$$p_a(g+1) = p_{min} + (1 - K \times p_{min}) \times \frac{q_a(g+1)}{\sum_{i=1}^{K} q_i(g+1)}. \tag{14}$$

where $p_{min} \in (0, 1)$ is the minimal selection probability value of each operator. It ensures that all the operators have a nonzero selection probability, which can avoid losing a currently bad operator that might become useful in the future.

E. Replacement Mechanism

In the original DE algorithm, the offspring replaces its parent in the next generation only if it has a better fitness value. In the case of MOPs, a different replacement mechanism is needed due to the already mentioned properties of them. To this end, a two-step comparison method is proposed in this work, it works as follows.

The first step of the replacement mechanism, which is based on the Pareto dominance relation, uses the non-dominated sorting method proposed in the NSGA-II [7]. Briefly, at each round, the non-dominated solutions of the hybrid population, which are merged by solutions from the parent and offspring populations, are chosen to survive to the next generation, and then they are removed from the hybrid population afterwards. This is done iteratively up to the completion of the population for the next generation (i.e. required number of solutions have been chosen after the non-dominated sorting procedure), or until there are no less than this required number of solutions with assigned rank values in the population.

In case there are still solutions need to be filtered for the next generation, the next step considers the *nTND* values proposed in [9] as the measurement. At each iteration, the solution that has the lowest *nTND* value (i.e. the one that locates in the least crowded region) is preserved, until the exact number of solutions for the completion of the new population is achieved.

III. Empirical Study

In this section, we compare the performance of our proposed MODE/SN with three state-of-the-art MOEAs, namely, GDE3 [3], NSGA-II [7] and MOEA/D [12].

A. Experimental Settings

In our experiments, decision variables are encoded in real numbers. The population size is constantly set to 100 for all benchmark problems. The maximum number of generations is set to 300 for two-objective problems and 500 for three-objective problems. 50 independent runs are conducted to collect the statistical results, and Wilcoxon's rank sum test at a 0.05 significance level is adopted to compare the significance of the difference between the solutions sets obtained by two competing algorithms. The number of divisions on each dimension is set as $div = 10$, and the hyper-parameters for the AOS paradigm, i.e. adaptation rate $\alpha = 0.8$ and $p_{min} = 0.05$. For the sake of a fair empirical comparison, the parameters of the three state-of-the-art MOEAs are set as in their respective original papers.

Various features of MOPs might cause difficulties to MOEAs, such as non-convexity, multi-modality, discontinuity, and non-uniformality. As Deb et al discussed in [13], most of the widely used benchmark problems do not consider the variable linkages, which can introduce significant difficulties to MOEAs. In this paper, we investigate the performance of the algorithms on the benchmark problems proposed in [14] ($F1$ to $F10$), which are with explicit linear and non-linear variable linkages. Details about these ten benchmark problems can be found in [14].

Two comprehensive performance indicators, *Inverted Generational Distance* (IGD) [15] and *Hypervolume* (HV) [16], are employed to quantitatively evaluate the performance of each algorithm at the end of each run. Both of them are comprehensive indicators, which can evaluate the convergence and diversity simultaneously. The reference points are set as (2.0, 2.0) for two-objective problems and (2.0, 2.0, 2.0) for three-objective problems, respectively, when calculating the HV. Generally speaking, the lower the IGD value, the better the performance is; oppositely, for HV, the higher the better.

B. Experimental Results

The comparative results, including the mean and variance, for each of them are presented in Table I and Table II. In the last column of each table, the † indicating that MODE/SN is significantly better than all its competitors in the corresponding benchmark problem, and ‡ representing that the best competitor significantly outperforms MODE/SN. Moreover, the best results for each indicator on each problem function are highlighted in bold face with grey background.

From these comparative results, it clearly shows that MODE/SN is the best choice when compared to its competitors: it achieves the best results in all performance metrics, and performing significantly better in 18 out of 20 comparisons. All these ten benchmark problems are modified from the classical ZDT [17] and DTLZ [18] benchmark problems, on which the state-of-the-art GDE3 and NSGA-II reported good performances [3] [7]. However, the introduction of variable linkages presented significant difficulties to both of them for optimization. MOEA/D was the winning algorithm in CEC 2009 MOEA contest. It decomposes a MOP into a number of single objective subproblems. Then a population based method is used to solve these subproblems simultaneously. Though it shows significant better performance to GDE3 and NSGA-II, our proposed MODE/SN still outperform it in all performance comparisons. All those three competing MOEAs are with traditional stochastic evolutionary operators, in which the optimization process does not consider the orientation information. We conclude that the success of MODE/SN should benefit from the new designed DE mutation operators which consider the effective orientation information from the optimization process. At the same time, the AOS paradigm adaptively select the most appropriate operator to apply also well balance the trade-off between the exploration and exploitation.

IV. Conclusion

In this paper, we propose a new DE variant for multi-objective optimization, i.e. MODE/SN, which considers utilizing the effective orientation information drawn from the optimization process. It maintains an operators pool which consists of four different DE mutation operators with distinct search behaviors. We also deploy a neural network, which is trained by the extreme learning machine, for mapping an artificially generated solution from the objective space to its corresponding decision variable values. An adaptive operator selection paradigm is proposed to control the application of different operators in an online manner. Numerical experiments demonstrate that the proposed MODE/SN significantly outperforms three state-of-the-art MOEAs, namely GDE3 [3], NSGA-II [7] and MOEA/D [12], in all performance comparisons.

TABLE I: Performance Comparisons on IGD

	MOEA/D	NSGA-II	GDE3	MODE/SN	S
F1	6.938E-3(4.92E-4)	1.093E-1(3.89E-2)	1.441E-1(4.19E-2)	**4.235E-3(5.12E-4)**	†
F2	1.533E-2(5.11E-3)	1.539E-1(4.35E-2)	3.732E-1(6.79E-2)	**4.231E-3(4.67E-4)**	†
F3	4.887E-1(7.71E-2)	8.978E-1(1.02E-1)	7.912E-1(9.85E-2)	**7.650E-2(6.19E-3)**	†
F4	1.429E-1(8.70E-2)	2.824E-1(1.37E-1)	9.461E-2(1.94E-1)	**5.438E-2(6.58E-3)**	†
F5	6.979E-3(5.01E-4)	3.006E-1(6.59E-2)	3.842E-1(6.28E-2)	**4.540E-3(4.18E-4)**	†
F6	1.054E-2(6.10E-3)	2.449E-1(5.11E-2)	2.877E-1(5.50E-2)	**7.921E-3(8.02E-4)**	†
F7	5.521E-1(7.17E-2)	8.613E-1(1.77E-1)	5.433E-1(1.03E-1)	**8.388E-2(8.98E-3)**	†
F8	3.843E-1(1.00E-1)	3.052E-1(1.35E-1)	4.924E-1(1.90E-1)	**7.637E-2(9.00E-3)**	†
F9	5.268E-1(2.03E-1)	5.271E-1(2.22E-1)	5.267E-1(2.53E-1)	**6.250E-3(1.08E-3)**	†
F10	7.812E-1(4.98E-1)	8.092E-1(5.30E-1)	8.053E-1(5.19E-1)	**9.358E-2(1.17E-1)**	†

TABLE II: Performance Comparisons on HV

	MOEA/D	NSGA-II	GDE3	MODE/SN	S
F1	3.6553(3.56E-4)	3.3368(4.32E-4)	3.2553(4.19E-4)	**3.6604(2.41E-4)**	†
F2	3.3177(5.87E-4)	2.5511(4.35E-3)	2.1247(4.02E-3)	**3.3269(4.89E-4)**	†
F3	1.5894(6.77E-3)	1.0931(7.01E-3)	1.2936(6.82E-3)	**3.0215(4.38E-4)**	†
F4	5.5555(3.27E-3)	5.0497(4.11E-3)	5.5284(3.96E-3)	**7.4045(2.55E-3)**	†
F5	3.6552(3.71E-4)	3.0248(4.90E-4)	2.8847(1.99E-3)	**3.6595(3.01E-4)**	
F6	3.3133(7.82E-4)	2.3293(5.15E-3)	2.2545(5.72E-3)	**3.3183(5.09E-4)**	
F7	1.5140(2.66E-2)	1.0378(3.31E-2)	1.8003(3.18E-2)	**2.4832(3.81E-3)**	†
F8	6.0001(2.97E-2)	5.8060(2.16E-2)	6.6013(2.34E-2)	**7.0346(1.80E-2)**	†
F9	2.3484(2.51E-2)	2.3470(2.45E-2)	2.3496(2.39E-2)	**3.6446(5.52E-3)**	†
F10	2.1135(1.09E-1)	2.0731(2.11E-1)	2.0824(2.32E-1)	**2.2253(1.35E-1)**	†

In future, we would like to further analyze each module of MODE/SN, including the four well designed DE mutation operators and AOS paradigm, to better understand the underlying mechanism of it. Moreover, the mapping from the objective space to the decision space can introduce some prediction errors, especially on some difficult problems. We consider building some local models to better capture the fitness landscapes online during the optimization process.

REFERENCES

[1] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[2] H. Abbass, R. Sarkar, and C. Newton, "A pareto differential evolution approach to vector optimisation problems," in *CEC '01: Proc. of 2001 IEEE Congress on Evolutionary Computation*. IEEE press, 2001, pp. 971–978.

[3] S. Kukkonen and J. Lampinen, "GDE3: the third evolution step of generalized differential evolution," in *CEC'05: Proc. of 2005 IEEE Congress on Evolutionary Computation*, vol. 1, Sep. 2005, pp. 443–450.

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

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1C3, pp. 489–501, 2006.

[6] S. F. Adra, I. Griffin, and P. J. Fleming, "An informed convergence accelerator for evolutionary multiobjective optimiser," in *GECCO'07: Proc. of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 734–740.

[7] 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, pp. 182–197, Apr. 2002.

[8] K. Li, A. Fialho, and S. Kwong, "Multi-objective differential evolution with adaptive control of parameters and operators," in *LION'11: Porc. of 2011 Learning and Intelligent OptimizatioN*, C. Coello, Ed. Springer Berlin / Heidelberg, 2011, pp. 473–487.

[9] K. Li, S. Kwong, J. Cao, M. Li, J. Zheng, and R. Shen, "Achieving balance between proximity and diversity in multi-objective evolutionary algorithm," *Information Sciences*, vol. 182, no. 1, pp. 220–242, 2012.

[10] D. E. Goldberg, "Probability matching, the magnitude of reinforcement, and classifier system bidding," *Machine Learning*, vol. 5, pp. 407–425, 1990.

[11] D. Thierens, "An adaptive pursuit strategy for allocating operator probabilities," in *GECCO'05: Proc. of the 7th annual conference on Genetic and evolutionary computation*, H.-G. Beyer et al., Ed. ACM, 2005, pp. 1539–1546.

[12] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *CEC '09: Proc. 2009 IEEE Congress on Evolutionary Computation*. IEEE press, May 2009, pp. 203–208.

[13] K. Deb, A. Sinha, and S. Kukkonen, "Multi-objective test problems, linkages,and evolutionary methodologies," in *GECCO '06: Proc. of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006, pp. 1141–1148.

[14] 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, Feb. 2008.

[15] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 174–188, 2003.

[16] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 257–271, 1999.

[17] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, pp. 173–195, June 2000.

[18] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing, L. Jain, X. Wu, A. Abraham, L. Jain, and R. Goldberg, Eds. Springer Berlin Heidelberg, 2005, pp. 105–145.