Science in China Series F: Information Sciences

www.scichina.com
info.scichina.com
www.springerlink.com

Duple-EDA and sample density balancing

CAI YunPeng[1†], XU Hua[1], SUN XiaoMin[1], JIA PeiFa[1] & LIU ZeHua[2]

[1] State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;

[2] School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

In this paper, a new method is proposed to overcome the problem of local optima traps in a class of evolutionary algorithms, called estimation of distribution algorithms (EDAs), in real-valued function optimization. The Duple-EDA framework is proposed in which not only the current best solutions but also the search history are modeled, so that long-term feedback can be taken into account. Sample Density Balancing (SDB) is proposed under the framework to alleviate the drift phenomenon in EDA. A selection scheme based on Pareto ranking considering both the fitness and the historical sample density is adopted, which prevents the algorithm from repeatedly sampling in a small region and directs it to explore potentially optimal regions, thus helps it avoid being stuck into local optima. An MBOA (mixed Bayesian optimization algorithm) version of the framework is implemented and tested on several benchmark problems. Experimental results show that the proposed method outperforms a standard niching method in these benchmark problems.

evolutionary computation, estimation of distribution algorithm, Boltzmann selection

1 Introduction

In the past decade, estimation of distribution algorithms (EDA) has been developed as a new approach of evolutionary optimization which applies machine learning methods to speed up the evolution. By replacing the nature-inspired operators used in traditional genetic algorithm (GA) with explicit probabilistic formulations, the disorder in creating new population is reduced so that the optimization process is accelerated. On the other hand, by identifying the underlying interactions between variables, the algorithms become more powerful in solving deceptive problems. These advantages of EDAs have been proven in both combinatory and real-valued function optimization. Reviews on the development of EDA can be found in refs. [1–4].

The problem of local optimal traps is a tough problem that obsesses most optimization algorithms. Although EDAs strive to circumvent this problem by exploiting knowledge about the problem structure, it has been discovered[5] that they are still susceptible to the phenomenon of genetic drift, like traditional GA is. This will cause the algorithms to be trapped into local optima. Shapiro[5] analyzed the dynamic of EDA and pointed out that current EDA frameworks magnifies random fluctuations and cause the algorithm to search in a biased way, which leads to drifts. In this

Received October 26, 2007; accepted December 11, 2008
doi: 10.1007/s11432-009-0140-7
†Corresponding author (email: caiyp78@gmail.com)
Supported partially by the National Natural Science Foundation of China (Grant Nos. 60405011, 60575057, 60875073)

Citation: Cai Y P, Xu H, Sun X M, et al. Duple-EDA and sample density balancing. Sci China Ser F-Inf Sci, 2009, 52(9): 1640–1650, doi: 10.1007/s11432-009-0140-7

paper we illustrate that similar phenomenon exists in real-valued function optimization. The problem lies in the difficulty for the algorithm to transfer its exploring focus from one peak to another to correct the biases, and current approaches, such as niching techniques[6] and local structures[7−11], provide no mechanism to offset the fluctuations.

In this paper, we proposed that by taking feedbacks from the search history, the reinforcement of random fluctuations can be compensated and the above problem can be overcome. We extend the EDA framework to Duple-EDA, where both the current best solutions and the entire optimization history are modeled, so that long-term reward is considered in the search strategy and the searching biases caused by random fluctuation can be adjusted. After setting up the framework, we proposed sample density balancing (SDB) as a method of escaping local optima traps, specifically for real-valued function optimization. A two-objective Pareto ranking[12] selection scheme is designed taking into account not only the fitness, but also the density of visited points around the individual, so that repeated sampling in a small area is penalized and the algorithm is directed to exploit more potentially optimal regions. Compared to standard niching techniques, this strategy is more powerful in detecting and escaping traps in complex problems, thus speed up the search process, which is validated by our experiments on several benchmark problems.

In section 2, we provide necessary background knowledge on the problem we are discussing and explain the intuition behind our method. In section 3 we describe the duple-EDA framework and the SDB method. In section 4 we list previous work in related field and compare our approach with them. Some experimental results are given in section 5 before concluding in section 6.

2 EDA and the problem of local minimal traps

In this section we introduce the framework of EDA and address the possibility of creating an alternative architecture. Then we discuss the mechanism that leads EDA to local optima traps, so that the relevance of our method is made clear.

2.1 The EDA framework

Estimation of distribution algorithms mainly comprise the step of creating distribution probability model from current best solutions and the step of generating new solutions from the model. The detailed framework is shown on Table 1, which is coherent to most current EDA variants. The replacement step, which is not adopted by some early versions of EDA, is critical in this paper because existing niching methods usually take advantage of this step and restrict the competition between the individuals in one way or another.

2.2 Drift and local optimal traps

In ref. [5] Shapiro illustrated that the finite population sampling in EDA will cause the phenomenon of drift, which leads the algorithm to poor performance. He pointed out that the EDA framework lacks two critical property for an effective search algorithm:

- The algorithm should be able to explore any unvisited point of the space, regardless of the current state.
- If the search space is flat, the algorithm should

Table 1 A generalized framework of EDA

Step 1 (Initialization)	Set t=0, randomly generate initial population $\boldsymbol{P}(0)$.
Step 2 (Selection)	Select a set of parent individuals from $\boldsymbol{P}(t)$ based on the fitness value.
Step 3 (Modeling)	Construct a statistical model based on the selected set.
Step 4 (Generation)	Generate a set of offspring $\boldsymbol{O}(t)$ according to the statistical distribution given by the constructed model.
Step 5 (Replacement)	Create a new population $\boldsymbol{P}(t+1)$ by replacing some individuals from $\boldsymbol{P}(t)$ with $\boldsymbol{O}(t)$, set $t = t+1$.
Step 6 (Iteration)	If the termination criteria are not met, go to Step 2.

search all points with equal likelihood.

The first property ensures that the global optimum will eventually be found. The second property provides that the algorithm will search the space in an unbiased way and will not be sensitive to random perturbations.

Unfortunately EDA possesses none of the above properties. The algorithm generates new solutions with similar statistical characteristics to the current population, via the distribution model. If the created model takes degenerative values for some parameter or gets biased in searching, the new population is more likely to inherit these defects than to fix them. For combinatory optimization, it is illustrated in ref. [5] that UMDA (univariate marginal distribution algorithm)[13], a version of EDA, might have a significant chance of not getting the optimal solution even after infinite steps in some benchmark problem.

For real-parameter optimization, the situation seems better because a Gaussian mutation is usually adopted in sampling, at any time all regions in the space will be visited with at least a minimal probability, so that the global optimum will always be hit given enough time. But the algorithm is still biased. The bias is introduced by two effects. Firstly, in order to retain the property of good solutions, mutation has to be biased towards the neighborhood of them, so that the variation of the Gaussian mutation will shrink when the population get close to the local optimal peak, and the chance of mutating to the other regions will decrease. Secondly, due to the selection strategy based on fitness, if the initial distribution of the population prefers the local optimum, new points will be produced around the peak and the exploring in other regions will be suppressed, even if a little number of individuals had been generated in these regions. With the progress of the evolution the algorithm is more and more biased to local search and the chance of discovering new peaks decreases. If the global optimum peak is not included in the favored peaks of the search in early stages, the algorithm will fail.

Niching methods[14] are standard solutions for evolutionary algorithms to preserve diversified population and deal with the problem of local optimal traps. However, these methods handle crowded individuals in the same generation but do not prevent iteratively reproducing similar samples. So, niching methods help to create multiple search directions, but they cannot suppress an exhausting search in a trapping direction and drive it to explore new regions, thus they are not helpful in correcting the bias once it has appeared. In section 4 we will provide a more detailed survey on niching methods.

Many EDA approaches also adopt local structures[7−11] in the distribution models to capture complex function landscapes and to perform multi-directional search. Like niching methods, this method strives to solve the problem by merely keeping as much diversified species as possible. As the problem scale increase, diversity-maintaining will be insufficient and the chance of missing the desired optima will increase.

This paper suggests a new approach to deal with the above problem by monitoring the optimization process and explicitly interfere the search direction to correct the biases. We set up the Dulpe-EDA framework to record the search history and propose sample density balancing as a selection scheme to change the search direction.

3 Duple EDA and sample density balancing

When EDA is trapped to local optima, the population will converge to one cluster (without niching) or several (with niching) clusters while no cluster covers the region of global optimum. Even in this time, with random mutation a small number of outliers will still be generated that are helpful to discover the local optimum. However, the fitness-based selection strategy will prefer the cluster than the random outliers. If we are able to detect that the search is biased to the clusters covering local optima, we can penalized the clusters and assign more probability to explore the outliers, thus compensate the bias.

The idea is that the sample density, which is the density of visited points in a given region, can act as a perfect measurement for over-sampling. If the

algorithm iteratively explore a region, the sample density around the point will increase quickly. After identifying this effect, the probabilistic model in EDA can be modified to suppress searching in this region temporally and encourage exploration in other promising regions.

To implement the above idea, three subproblems have to be coped with: 1) how to effectively express the sample density in a continuous space; 2) how to modify the model without disturbing the linkage learning and the evolutionary mechanism, thus keep the benefits of EDA; and 3) how to apply an effective compensation in large search space. The first problem is solved simply by replicating the idea of EDA and create an extra model of historical distribution, so that the sample density of any point in the space can be estimated. The latter two problems are solved by modifying the selection scheme, which will be described in this section.

3.1 Duple-EDA

In this section we extend EDA to Duple-EDA, under which we establish a feasible approach to record the search history and to fuse the objectives of both exploiting current best solutions and finding new potential solutions. The approach provides necessary foundation for solving the local optima trap problem, which we are interested in.

The proposed Duple-EDA framework is depicted in Table 2, where the changes to the ordinal EDA framework is highlighted in bold. The framework involves two major modifications. The first one is that a historical distribution model $\boldsymbol{M}_h$ is created to record all visited points, and the second is that a hybrid-selection strategy is applied in the selection step.

In each iteration, the algorithm not only creates sampling model from elite individuals as normal EDA does, but also maintains a historical distribution model when every new sample is added. Any incremental model that had been adopted in EDA can be adopted for historical modeling but our experiments show that the univariate marginal distribution models are adequate. In our implementation we adopt the fixed-width histogram[15] and enhanced it with distance-weighted linear interpolation when adding a sample or acquiring the density of a point.

After acquiring the distribution of visited points, the algorithm is able to obtain the sample density and change the search direction to prevent oversampling. The modification should be careful not to violate the breed's strategy and linkage discovering in evolution so that the efficiency of EDA can be kept. The solution is that since learning the distribution of elite individuals relies on short-term feedbacks that exerts in a few generations, while sample density is a long-term feedback that will only change much after a number of evaluations, we can adopt a hybrid scheme to separate the purpose of exploiting current solution and exploring the space. In the normal phase, the algorithm recombines current best solutions and exploit the in-

Table 2 The framework of Dulpe-EDA

Step 1 (Initialization)	Set $t = 0$, randomly generate initial population $\boldsymbol{P}(0)$. **Create the historical distribution model $\boldsymbol{M}_h(0)$ from $\boldsymbol{P}(0)$.**
Step 2 (Hybrid-Selection)	**If** t mod $T_{\mathrm{SDB}} == 0$, **acquire the sample density of each point in $\boldsymbol{P}(t)$,** select a set of parent individuals $\boldsymbol{S}(t)$ from $\boldsymbol{P}(t)$ **with the SDB selection strategy. Otherwise** select a set of parent individuals $\boldsymbol{S}(t)$ from $\boldsymbol{P}(t)$ based on the fitness.
Step 3 (Modeling)	Construct the present distribution model $\boldsymbol{M}_p(t)$ based on the selected set $\boldsymbol{S}(t)$.
Step 4 (Generation)	Generate a set of offspring $\boldsymbol{O}(t)$ according to the statistical distribution given by the constructed model $\boldsymbol{M}_p(t)$. **Update $\boldsymbol{M}_h(t)$ from $\boldsymbol{O}(t)$.**
Step 5 (Replacement)	Create a new population $\boldsymbol{P}(t+1)$ by replacing some individuals from $\boldsymbol{P}(t)$ with $\boldsymbol{O}(t)$, set $t = t + 1$.
Step 6 (Iteration)	If termination criteria are not met, go to Step 2.

dicated best area, which is almost identical to normal EDA. Once after a given number of iterations (which is called the SDB-period in the rest of the paper, and denoted by T_{SDB}), the algorithm consults the historical model and applied a sophisticated scheme to search the space in a different manner. In this way the disturbing to the evolution mechanism of EDA is reduced.

With the above framework we are able to consult the search history and change the probabilistic model for generating new samples, thus change the search direction and avoid local optimal traps. The procedure of applying compensate to the model by consulting the accumulated sample density and balancing local and global search to avoid over-sampling is addressed as sample density balancing in this paper, and we will introduce it in the next subsection.

3.2 Sample density balancing

In order to change the search direction to offset the biases, it is straightforward to explicitly modify the probabilistic expression of specified regions in the model, but doing this is computationally expensive. In this paper we suggest an alternative approach, i.e., to modify the set of selected individuals fed to model building. Note that removing points in a penalized region is not enough to penalize the region since recombining other points may produce new samples that also fall into it. Fortunately, with the historical model this case has already been taken into consideration. Like normal EDA models, the historical model records the schemata of visited points. When the algorithm repeatedly explore a small region, the generated samples will have similar schemata and the estimated sample density of any points with common schemata will increase, although with different grade. If a penalty for high estimated sample density is added to the selection scheme, the individuals in the population that lead to exploration in the crowded region will no longer be selected for modeling after some generations, so that any possible recombination that lead to new crowded samples will be prohibited.

With the above framework, the algorithm is able to alter the probabilistic model and the search direction based on historical information. The last problem to deal with is to set up a compensation method to avoid drift. The task is two-fold. Besides penalizing the over-sampled region, some candidate regions must be picked out for exploring so that the algorithm will not degrade to random walk. In the next section we will propose a selection scheme to smoothly apply penalization to over-sampled regions and achieve a balanced search.

To prevent over-sampling and compensate the random bias from the knowledge of current population and sample density, we proposed SDB-selection as a selection scheme. The idea is to set up a criterion of evaluating the "potentially optimal" individuals rather than the current best individuals, so that a smooth trade-off between the fitness and the sample density is achieved. The concept of potential optimum is first proposed in a pattern search method called DIRECT (DIviding RECTangles)[16], and we first introduce the concept before extending it.

In the DIRECT algorithm, the space is partitioned into a set of hyper-rectangles containing a single visited point at the center of each rectangle. Assuming that a single Lipschitz constant exists for the entire space, the optimal boundary of the object function can be estimated by the sampled values, the constant and the size of the rectangles. For a given hyper-rectangle r, if there exists a Lipschitz constant K so that the optimal boundary will be achieved at the border of r with that constant, and the boundary is significantly better than the current best value, the hyper-rectangle is called a potential optimal rectangle.

For a maximization problem, the potential optimal criterion can be formulated as

$$\begin{aligned} &f(c(r)) + K \cdot l(r) \geqslant \max_{s \in \mathcal{S}} [f(c(s)) + K \cdot l(s)], \\ &f(c(r)) + K \cdot l(r) \geqslant f_{\max} + \varepsilon |f_{\max}|, \end{aligned} \tag{1}$$

where r is the referred hyper-rectangle, $c(r)$ is the center of it, and $l(r)$ is the length of its longest side[1)], $f(x)$ is the object function, $f_{\max}$ is the opti-

1) Here the side length is used to approximate the diagonal of the hyper-rectangle. Some modified version of DIRECT also take other sides into consideration.

mal value current achieved and ε is a small positive constant representing the expected improvement.

The DIRECT algorithm chooses all potential optimal rectangles to explore in every steps. The detail of the algorithm will be further discussed in section 4. The concept of potential optimum suggests a criterion to select promising regions to explore and perfectly balances local and global search. This can be adopted to EDA to prevent over-sampling.

In sample density balancing, we adopt the concept of potential optimal to select promising points. In DIRECT algorithm, sample points are regularly distributed and the size of the rectangles is the nature metric of sample density. To adopt the concept in EDA, where the sample points are irregularly distributed, the term $l(p)$ in eq. (1) must be redefined. In our case, for a certain point p, we define $l(p) = 1/N(p)$, where $N(p)$ is the sample density around p. With this modification the concept can be smoothly extended.

Following the concept of potential optimum a ranking scheme can be designed. The potential optimal individuals in the current population are marked as rank 0. After removing these individuals, the new potential optimal ones in the samples left are marked as rank 1, and so on. The selection scheme picks a given number of samples with the lowest ranks into the selected set, so that not only the best solutions but also the good solutions residing in less-explored regions are included. For points with the same rank, those with higher fitness will be selected first. The ranking is illustrated in Figure 1(b).

It should be noted that the selected individuals will not always survive to the next generation, for the replacement strategy will consider only the fitness.

When a point is selected for modeling, the sample density around it will increase due to the entering of new sampled points. The increase of sample density will in turn decrease the active length of the point by definition, so the ranking of the point is likely to remain the same or even become higher, which means that the chance of picking the same point for modeling next time is likely to be lower. However, if the new points around it can achieve better fitness, the chance of picking them as partial solutions and creating similar model is increased. In this way, sample density nicely plays a role of balancing the search between the current local optimum and other potential optimums, which is exactly what the term sample density balancing means.

With sample density balancing, the algorithm meets the second property suggested in section 2 that in a flat function landscape the algorithm should search the space with equal probability. Although SDB do not explicitly generate a formulation in the probabilistic model, it penalizes regions with higher sample density so that the distribution of visited points is driven towards the uniform distribution.

Because SDB has taken the role of diversity maintaining, we suggest that no niching method is needed in the replacement step of the EDA framework. Simple replacement strategy such as Tournament Replacement can be adopted.

4 Related works

The scope of this paper covers both recording

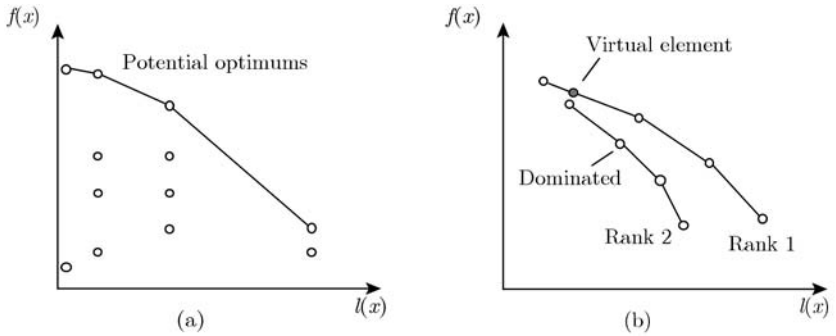

Figure 1 Illustration of potential optimum (a), potential optimum (b) ranking used by our method.

search history and diversity maintenance. Many results have been achieved in either scope but there is no works reported to fuse the two approaches, to the best of our knowledge. In this section we will list previous results on these two topics and compare our work to them.

The idea of recording the search history to avoid over-sampling has been adopted by various optimization algorithms. Tabu search[17] is a popular procedure in combinatory optimization, which memorizes recently-visited points and prohibits exploring their neighborhood. Many variations of the procedure have been developed to adopt more complex tabu rules. But in real-valued optimization, it is difficult to apply tabu rules to the continuous state space, thus this approach is not applicable.

The DIRECT (DIviding RECTangles) algorithm, on the other hand, records the entire history and assigns a dynamic priority to each region. The algorithm divides the search space into a set of hyper-rectangles regions $\mathcal{S}$. For each hyper-rectangle, the object function is evaluated on the center point and treated as the value of the hyper-rectangle. At the beginning of the search, $\mathcal{S}$ contains a single hyper-rectangle covering the entire space. In each iteration, all hyper-rectangles in $\mathcal{S}$ that are identified to be potentially optimal are divided into three parts along the longest side of them. Here $\mathcal{S}$ contains the search history and the potential optimum criterion provides dynamic priority for accessing all regions. The concept of potential optimum had been introduced in section 3.2.

The DIRECT algorithm has been proven to be excellent in balancing local and global search in many industrial applications. However, the algorithm bears a severe defect that it lacks effective modeling of the historical information. In every step the algorithm has to consult every sampled point to determine the potential optima, which causes it to slow down quickly as the search proceeds. Saving the result of the last step rarely helps because changing a few points changes the entire solution structure. In the case of high dimension, things become even worse that the algorithm degrades to greedy search for almost all rectangles are with the same side-length. Furthermore, since its sampling scheme is rigid, i.e., picking the center point of each divided rectangle, it is inefficient for many problems.

In our work, the estimated sample density, rather than the exact sample record, is utilized as a way to keep the information about the history to avoid over-sampling. Compared to the method used by DIRECT, the computational effort is reduced from exponential complexity to quadratic one. The probabilistic sampling style in EDA also overcomes the problem of rigid-sampling.

Niching methods has long been introduced to evolutionary algorithms in the purpose of diversity maintenance. Surveys on existing niching methods can be found in refs. [13, 15]. They take into account some attributes of the individuals other than their fitness, and give bias to the selection strategy (in traditional GA) or the replacement strategy (in EDA), so that individuals having different attribute values can coexist. In ref. [15] by proposing restrict tournament replacement (RTR) niching methods are introduced to EDA under the framework of hBOA (hierarchical Bayesian optimization algorithm).

Although niching methods are successful in preventing the excessive reproduction of a single solution, they cannot prevent fruitless exploring in a fixed region (i.e., over-sampling) because they produce the same selection result statistically given the same population. For example, fitness sharing penalizes over-crowded individuals, but the summed probability of choosing their schemata is still high. Similar new samples will still have a high chance of being generated consistently although the amount of surviving individuals of the same kind is under control.

Moreover, niching methods preserve diversified solutions, but they do not ensure that these solutions are selected to generate new ones, so they do not help to correct the bias in the searching process once it has appeared. This has been illustrated

in the third case in the above section. In another word, niching methods do not increase the probability of transforming from one peak to another in real-valued optimization, thus they are limited in preventing drift.

5 Experimental results

In this section we created an implementation of Duple-EDA and conducted a set of experiments to test the improvements given by SDB. We implement our code based on the MBOA (mixed Bayesian optimization algorithm) code provided by Ocenasek[8], and we called the variation the Duple-MBOA. Duple-MBOA is achieved by removing the niching method, i.e., RTR, from the original MBOA code and incorporating the Duple-EDA framework.

In the following experiments, altogether 5 benchmark functions are used. For each function we execute tests on different dimensions to see how both algorithms perform with the problem scale increases.

The Duple-MBOA is configured with a 10-division fixed-width histogram in each dimension, and the SDB-period is set to 4 generations, unless explicitly specified.

The first experiment is to show that the proposed method enhanced the ability of MBOA for escaping local optimal traps. The two-peak function and the continuous two-deceptive function proposed in ref. [15] are used for test.

We perform the test on these two functions from 10 dimension to 80 dimension. The terminal condition suggested in ref. [15] is used, namely, the algorithm stops if it reaches solutions whose Euclid distance is less than 0.01 to the global optimum, the optimal parameter set is determined by the fastest one that find the global optimum in all 30 successive runs.

The results on the above two functions is depicted in Figures 2 and 3. It can be concluded from the results that Duple-MBOA clearly outperforms the original MBOA in these functions, especially when the problem size increases. For the 2-peak function, the population size for Duple-MBOA ranges from $N = 300$ for $n = 10$ to $N = 1000$ for $n = 80$, and the required number of evaluations is approximately $O(n^{1.13})$. For the 2-deceptive function, the population size for Duple-MBOA ranges from $N = 1000$ for $n = 10$ to $N = 4500$ for $n = 80$, and the required number of evaluations is approximately $O(n^{1.39})$. This result is also better than previous ones on the same experiments reported in refs. [8, 15].

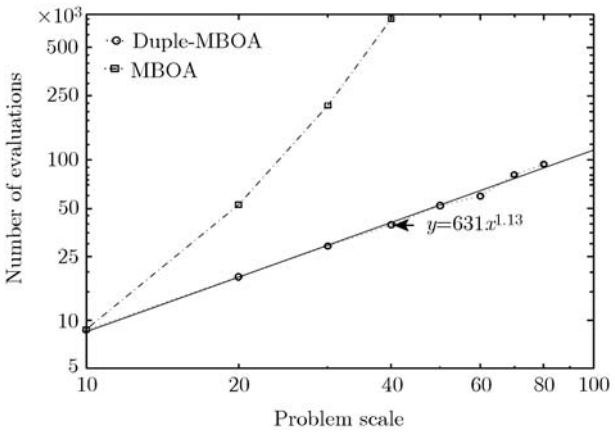

Figure 2 Result for the 2-peak function.

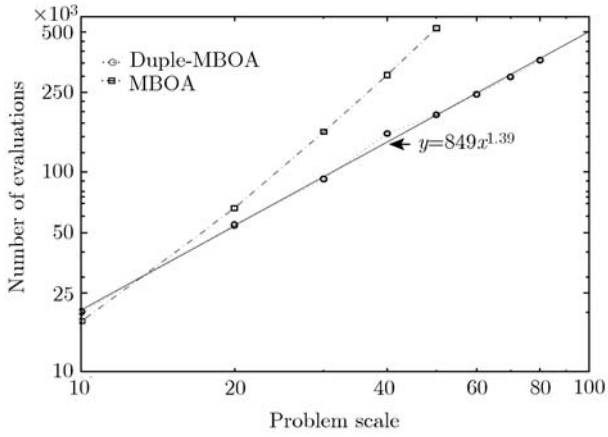

Figure 3 Result for the 2-deceptive function.

The improvements can be attributed to the distinctive exploring style of SDB exhibited in the experiment. The original MBOA maintains multiple optimal regions and converges to them. If the global optimum is not found before the convergence occurs, the algorithm will be trapped in some local optimum. With the problem size increases, it is very hard to keep as much candidate regions as is required, unless the population size is increased fiercely, so the algorithm slows down. On the contrary, Duple-MBOA focuses on the current best

region and at the same time adaptively explores other regions with the variation of sample density. In this way Duple-MBOA could lead the search out of local optimums to explore other potential optimums and help the algorithm find more links. This explains why Duple-MBOA is more likely to find the global optimum quickly.

The average time consumption for a single run in the above experiments are also given in Table 3. The time data is acquired on a PC with Pentium IV 1.6 GHz processor and 256 MB RAM under Win2k/cygwin environment. We can see that Duple-EDA does not introduce notable extra computational effort to the algorithm compared to normal niching methods.

The second experiment is to compare the performance in normal benchmark problems. The benchmark problems used in ref. [1] are chosen, namely, the SumCan function, the Schwefel function and the Griewangk function. These functions are all multi-modal functions featuring with strong, moderate and weak dependencies among variables, respectively.

We perform 30 runs for each set of parameters for both algorithms, and the one with the least average error to the global optimum after 500000 evaluations is chosen as the result. It is notable that for the first two functions, the parameter set that achieves the best average performance also achieves the best single-run result during the experiments.

The experiment results on these functions are depicted in Table 4, Figures 4 and 5. For the SumCan function and the Schwefel function, both algorithms perform excellently in low dimensional case, the original MBOA even outperforms Duple-MBOA. However, with the problem size increases, it is more and more difficult to learn the entire linkage structure, even with increased population. For the original MBOA, if the algorithm does not learn the precise links in the early stage of the searching, it is unlikely to correct the model later due to the reinforcement nature of the algorithm, so that search will not make more progresses. On the other hand, for Duple-MBOA there is more chances to continue the search with incomplete linkage information and correct the faults in modeling with sample density balancing in the later stages. This explains why Duple-MBOA can preserve its quality with increased problem size and outperform the ordinal approach. In the case of the Schwefel function, it can be seen that such phenomenon affects not only the average performance of the algorithm, but also the best result. For many application problems people concern more about the best result rather than the average one, so that this experiment provides more evidence to the relevance of our approach.

Table 3 Time consumption for MBOA and Duple-MBOA (in second)

2-peak function									
Prob. size		10	20	30	40	50	60	70	80
Duple-MBOA	time	1.0	9.1	30.8	77.9	160.5	279.6	369.5	569.2
	evals.	8700	18725	29080	39440	51920	59560	80800	93950
MBOA	time	1.0	25.4	247.0	1658.3	N/A	N/A	N/A	N/A
	evals.	8786	52560	218400	756100	N/A	N/A	N/A	N/A
2-deceptive function									
Prob. size		10	20	30	40	50	60	70	80
Duple-MBOA	time	2.8	30.2	133.0	498.7	1002.3	2201.9	3354.8	5550.6
	evals.	20186	54495	92085	155600	193280	244080	298266	361950
MBOA	time	2.2	36.0	248.2	1028.6	2984.9	N/A	N/A	N/A
	evals.	18135	66020	158960	305493	521933	N/A	N/A	N/A

Table 4 Performance of MBOA and Duple-MBOA on normal benchmark problems (error after 5e5 evals)

SumCan function							
Prob. Size		10	12	14	16	18	20
MBOA	aver.	2.48e-10	3.30e+3	4.55e+3	5.25e+3	1.27e+4	2.19e+4
	best	1.46e-11	1.46e-11	4.37e-11	8.92e-7	1.34e-3	0.99
Duple-MBOA	aver.	1.42e-9	9.31e-9	8.47e-9	0.0131	0.700	164
	best	1.46e-11	1.46e-11	1.46e-11	1.96e-7	0.0156	24.1
Schwefel function							
Prob. Size		150	180	210	240	270	300
MBOA	aver.	5.31e-16	1.75e-8	0.250	23.4	393	1440
	best	4.86e-18	5.76e-10	6.45e-5	0.0722	11.5	401
Duple-MBOA	aver.	6.07e-14	1.21e-10	2.81e-7	2.67e-5	1.41e-3	0.0202
	best	6.63e-20	9.03e-15	2.72e-11	2.76e-7	1.88e-5	2.19e-3
Griewangk function							
Prob. Size		5	10	15	-	-	-
MBOA	aver.	7.63e-3	5.09e-3	3.61e-3	-	-	-
	best	<1e-40	<1e-40	<1e-40	-	-	-
Duple-MBOA	aver.	<1e-40	<1e-40	<1e-40	-	-	-
	best	<1e-40	<1e-40	<1e-40	-	-	-

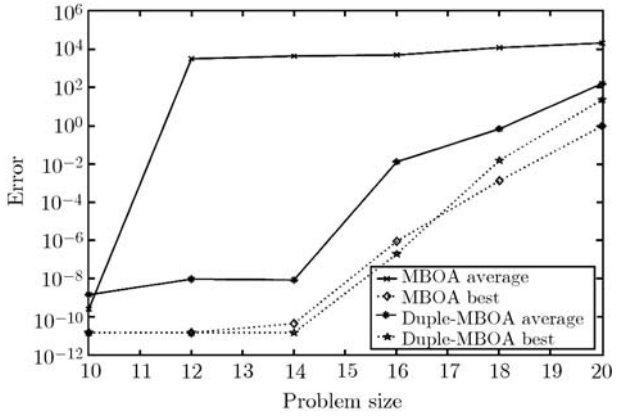

Figure 4 Result for the SumCan function.

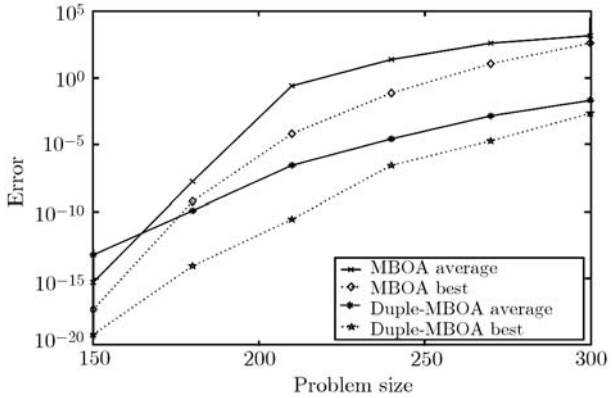

Figure 5 Result for the Schwefel function.

For the Griewangk function, it seems that the original MBOA with Restrict Tournament Replacement often fails to distinct the global optimum peak from the local ones, and we did not find a parameter set that can achieve the global optimum within the given number of evaluations in all 30 runs. On the other hand, the Duple-MBOA seems to be successful in migrating from the local optimum peaks to the global one.

Since Duple-MBOA introduced a new parameter, i.e, the SDB-period, we perform the third experiment to show that the quality of the algorithm will not depend much on it, so that it will not involve heavy works on parameter-tuning. The result

on the 20-dimensional SumCan function with varied SDB-period is depicted in Figure 6. 10 runs are performed for each value of the parameter. It can be seem that even for problems with complex dependency, there is a wide range for the parameter that it will not affect the quality of searching.

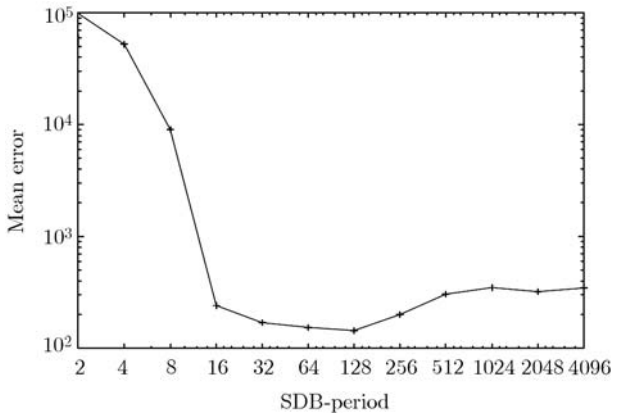

Figure 6 Performance of Duple-MBOA to hybrid frequency.

6 Conclusions

In this paper, SDB is proposed as an enhancement to EDA by preventing repeated sampling in local optimum regions. The Duple-EDA framework is proposed as an extension of EDA to effectively record historical information such as sample density so that long-term feedbacks can be used to aid the search. A hybrid-selection strategy is proposed to utilize both long-term and short-term information and to preserve the quality of search and linkage learning in EDA. A two-objective Pareto ranking is introduced as a criterion for selecting individuals and penalizing individuals with higher sample density, so that an adaptive search on potentially optimal regions can be performed. Experiment results have shown that the proposed method outperforms a standard niching method in the ability of escaping traps, without sacrificing the speed of convergence. Furthermore, Duple-EDA does not incur extra computation burden, neither does it require extensive parameter-tuning.

The Duple-EDA introduces a new dimension in the EDA framework and could lead to various implementations. The framework could potentially be applied to many research topics that are currently active in the field of optimization. For example, whether the framework is helpful to multi-object optimization or dynamic problem optimization, such questions are left open.

1 Larrañaga P, Lozano J. Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Norwell: Kluwer, 2002
2 Pelikan M, Goldberg D, Lobo F. A survey to optimization by building and using probabilistic models. Comput Optim Appl, 2002, 21(1): 5–20
3 Pelikan M, Sastry K, Cantú-Paz E. Scalable Optimization via Probabilistic Modeling: from Algorithms to Applications. Berlin: Springer, 2006
4 Lozano J, Larrañaga P, Inza I, et al. Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Berlin: Springer, 2006
5 Shapiro J. Drift and scaling in estimation of distribution algorithms. Evol Comp, 2005, 13(1): 99–124
6 Pelikan M, Goldberg D. Escaping hierarchical traps with competent genetic algorithms. In: Beyer H, Cantú-Paz E, Goldberg D, et al., eds. Proceedings of the Genetic and Evolutionary Computation (GECCO 2001). San Francisco: Morgan Kaufmann, 2001. 511–518
7 Pelikan M, Goldberg D. A hierarchy machine: Learning to optimize from nature and humans. Complexity, 2003, 8(5): 36–45
8 Oceanasek J, Schwarz J. Estimation of distribution algorithm for mixed continuous-discrete optimization problems. In: Kacprzyk J, ed. 2nd Euro-International Symposium on Computational Intelligence. Kosice: IOS Press, 2002. 227–232
9 Bosman P, Thierens D. Expanding from discrete to continuous estimation of distribution algorithms: the IDEA. In: Schoenauer M, Deb K, Rudolph G, et al., eds. Parallel Problem Solving from Nature-PPSN VI. Paris: Springer, 2000. 767–776
10 Peña J, Lozano J, Larrañaga P. Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of bayesian networks. Evol Comp, 2005, 13(1): 43–66
11 Ahn C, Ramakrishna R, Goldberg D. Real-coded bayersian optimization algorithm: bringing the strength of boa into the continuous world. In: Deb K, Poli R, Banzhaf W, et al., eds. Proceedings of the GECCO-2004 Conference on Genetic and Evolutionary Computation. Berlin: Springer, 2004. 840–851
12 Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning. Reading: Addision-Wesley, 1989
13 Mühlenbein H. The equation for response to selection and its use for prediction. Evol Comp, 1997, 5(3): 303–346
14 Mahfoud S. Niching methods for genetic algorithms. Ph.D. thesis. Urbana: University of Illinois at Urbana-Champaign, 1995
15 Pelikan M, Goldberg D, Tsutsui S. Combining the strengths of Bayesian optimization algorithm and adaptive evolution strategies. In: Cantú-Paz E, Mathias K, Roy R, et al., eds. Proceedings of the Genetic and Evolutionary Computation (GECCO 2002). San Francisco: Morgan Kaufmann, 2002. 512–519
16 Jones D, Perttuned C, Stuckman B. Lipschitzian optimization without the Lipschitz constant. J Optim Theo Appl, 1993, 79(1): 157–181
17 Bland J, Dawson G. Tabu search. Desn Optim, 1991, 23(3): 195–201