A Novel Memetic Algorithm for Constrained Optimization

Jianyong Sun and Jonathan M. Garibaldi, *Member, IEEE*

***Abstract*— In this paper, we present a memetic algorithm with novel local optimizer hybridization strategy for constrained optimization. The developed MA consists of multiple cycles. In each cycle, an estimation of distribution algorithm (EDA) with an adaptive univariate probability model is applied to search for promising search regions. A classical local optimizer, called DONLP2, is applied to improve the best solution found by the EDA to a high quality solution. New cycles are employed when the computational budget has not been reached. The new cycles are expected to learn from the search history to make the further search efficient and to enable escape from local optima. The developed algorithm is experimentally compared with ε-DE, which was the winner of the 2006 IEEE Congress on Evolutionary Computation (CEC'06) competition on constrained optimization. The results favour our algorithm against the best-known algorithm in terms of the number of fitness evaluations used to reach the global optimum.**

I. Introduction

The goal of this paper is to develop a memetic algorithm for the constrained optimization problem, also called nonlinear programming (NLP). The NLP process can be stated as follows:

$$\min f(\mathbf{x}), \mathbf{x} \in \mathcal{F} \in \mathbb{R}^n$$

where $\mathcal{F}$ is the set of *feasible* solutions that satisfies:

$$\begin{cases} g_i(\mathbf{x}) & \leq & 0, & i = 1, \cdots, q; \\ h_j(\mathbf{x}) & = & 0, & j = q+1, \cdots, m. \end{cases}$$

As is well known, one of the main concerns in developing evolutionary algorithms (EAs) for NLP is to select proper parent individuals, i.e. the individuals that carry useful information which can guide the search to feasible regions of the problem, for offspring reproduction. An effective selection method, or essentially individual ranking, should balance the feasibility and the fitness of the individuals. The development of effective ranking methods highly depends on the manner of constraint-handling, and so this has been an important research direction recently in constrained EAs (CEAs). For example, the stochastic ranking method [1] ranks the individuals taking account of their feasibilities and penalized fitness simultaneously; the ranking methods based on Pareto dominance relation [2] have been widely adopted in the CEAs based on a multi-objective perspective of constraint-handling [3], [4]. Readers are referred to [5], [6] and [7] for reviews, and [8] for an overview of recent advances.

An important issue in developing effective EAs for NLP is the offspring generation scheme. It is expected that the generated offspring can approach the feasible regions of the NLP in the early stages, while exploring the feasible regions for the global optimum later on. The abilities of a range of algorithms including evolutionary algorithms, such as genetic algorithms (GA) [9], evolution strategies (ES) [1], evolutionary programming (EP) [10], differential evolution (DE) [11], particle swarm optimization (PSO) [12], co-evolutionary algorithms (CA) [13], ant colony optimization (ACO) [14], artificial immune system [15], culture algorithm [16], [17], etc., have been widely explored for NLP. To our best knowledge, the application of estimation of distribution algorithm [18] to NLP is very limited. The only reference is [19], in which the performances of the UMDA [20] and PBIL [21] coupled with different constraint-handling methods, including penalty method and repair methods, were compared on two test problems.

Besides these research efforts, some researchers have made attempts to develop hybrid EAs for NLP problems. The development of hybrid EAs has proceeded in two main directions. On one hand, two meta-heuristics are combined to take advantages of their respective search strengths. For example, in [22], a combination of fuzzy logic and EP is proposed to handle constraints. In [23], EP is also hybridized with GENOCOP [24] for constrained optimization. In [25] and [26], GAs are combined with simulated annealing and PSO, respectively.

On the other hand, classical numerical optimization approaches for NLP have been hybridized in EAs. One of the main advantages of these classical approaches is that they are usually very efficient in locating feasible local optima, but the efficiency highly depends on the quality of the initial solutions. Starting from a 'bad' initial solution, the classical approaches could either only find an infeasible solution, or need a high computational cost to reach a feasible local optimum. Thus, effective strategies for the application of the classical approaches should be one of the main considerations in designing a hybrid EA.

In the literature, the hybridization strategies can be classified into three groups. Firstly, repair operators, which aim to make an infeasible individual feasible, are mostly applied in combinatorial optimization problems [27], [28]. In these repair-based algorithms, the repair operators are applied to all the newly generated infeasible individuals. In [29], the Nelder-Mead simplex search method is applied to repair *every* infeasible individual. From the view of computational cost, this indiscriminate strategy will result in high cost.

Jianyong Sun is with Centre for Plant Integrative Biology, School of Biosciences, The University of Nottingham, Sutton Bonington, LE12 5RD, United Kingdom (email: j.sun@cpib.ac.uk).

Jonathan M. Garibaldi is with School of Computer Science & Centre for Plant Integrative Biology, The University of Nottingham, Nottingham, NG8 1BB, United Kingdom (email: jmg@cs.nott.ac.uk).

JS and JMG are members of the Intelligent Modelling & Analysis (IMA) research group in School of Computer Science, The University of Nottingham.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

An obvious reason is that some individuals with low fitness cannot survive from the selection operation in the evolution procedure, and the repair efforts will be wasted. In response to this drawback, some more prudent strategies have been applied. For example, the Nelder-Mead simplex has been applied to only the best solution at each generation [30], [31]. The sequential quadratic programming (SQP) technique has been applied to the best individual in every ten generations [32]. In an alternative strategy, the classical optimization approach is only applied to the best individuals after the evolutionary search has finished. For examples, in [10] and [33], the best solution found after a whole EP search is optimized by a deterministic optimization neural network algorithm using a Lagrange multiplier method. In the hybrid simulated annealing (SA) method developed in [34], the best solution found by the SA is improved by the feasible SQP (FSQP) method after the SA has converged. The disadvantage of these prudent strategies is that the application of classical approaches to only *one* solution may be not sufficient to locate the global optimum.

Apart from the application of classical optimization approaches, some work adopts crossover or mutation operators as alternatives to the classical approaches to exploit local areas of the search space. Such operators include the gradient-based mutation operator [31], [35], [11], the simplex crossover [36] and some others. These operators are usually applied to all the newly generated individuals. The problem with these operators is that they are not as efficient as the classical approaches.

In this paper, we present a new strategy which can reduce the computational cost in terms of number of fitness evaluations, and increase the possibility of finding the globally-optimal feasible solutions. In the rest of the paper, we first present the developed new strategy in the form of an algorithmic framework in Section II. An instantiation algorithm based on estimation of distribution algorithm (EDA) is described in Section III. The experimental results are shown in Section IV, including the component analysis and a comparison with the winning algorithm on the test problems from the 2006 IEEE Congress on Evolutionary Computation (CEC 2006) competition. Section V concludes the paper.

II. Novel Multi-Cycle Algorithmic Framework

As stated in Section I, in the present hybrid strategies, local optimizers either are tightly coupled within the evolutionary procedure, or completely isolated without contributing to the evolution procedure. The tightly-coupled strategy, called CS in this paper, will result in high computational costs due to the extravagant application of the local optimizers. This strategy not only inevitably wastes unnecessary improvements to the low-fitness individuals, but also causes unfairness, except for the consideration of computational cost, in the harvest of promising individuals for offspring generation. Note that if only a proportion of promising individuals employ local search at each generation, it is not fair to the other individuals when the selection operation is performed. The underlying rationale is that a solution with low fitness does not necessarily lead to a low-quality local optimum if feasible, or a high constraint violation if infeasible.

In the other strategy, called parsimonious strategy (PS), the local optimizers are applied only after EAs have converged. To obtain a good algorithm performance, the hope is that the best solution found after the EA running locates in the attraction basin of the global optimum, if it is not the global optimum. No scheme in this strategy is provided to escape from the local optima once the EAs fail to locate the global attraction basin. Moreover, considering the computational budget, the PS-based algorithm may be stopped early.

To overcome the drawbacks of the CS and PS, we propose a novel strategy that make the application of local optimizers adapt to the search procedure. The goal of the strategy is to decrease the number of local optimizer application but improve the solution quality. To realize this goal, one way is to provide local optimizers with 'good' initial solutions, i.e. solutions that are close to the feasible region or the attraction basin of the global optimum.

Our proposed strategy is different from the traditional EA design principle to balance the exploration and exploitation during the search procedure. It does not focus on the exploitation capability of the EAs (the exploitation task is handed to the local optimizers), but rather on strengthening the exploration aspect. This differs from both the CS and the PS approaches.

Firstly, in contrast to the CS approach, we do not apply local optimizers to the individuals during the evolution procedure. After the evolution procedure has converged, the best solution found so far can be considered as a relatively 'good' solution. Or, more accurately speaking, the best solution found so far is more likely to be located in the attraction basin of the global optimum than the other visited solutions.

Secondly, in contrast to the PS approach, we only terminate our search in the case that the computational budget has been exceeded: in which case, we restart the evolutionary search. To make the new search procedure efficient and effective, we can either diversify the new search from previously searched areas using techniques such as niching [37], fitness sharing [38], etc., or guide the new search intelligently using learned information from previous searches, or combine both strategies. The starting of a new search also aims to escape from local optima once previous searches have become trapped.

Based on the above described strategy, we develop a novel multi-cycle evolutionary scheme for optimization problems as shown in Alg. *MCE*, overpage.

In the Alg., Θ_1 and Θ_2 are the parameters of the EAs and the local optimizer, respectively, and c is the cycle index. In each cycle, **EvolutionaryAlgorithm(Θ_1,history)** takes the history information into account to help for an efficient search, and returns the best solution found x_c (line 3). In the first cycle, no history information is available (line 1). **LocalSearch**(x_c, Θ_2) improves x_c to local optimum x_c^*, called the cycle best solution (line 4). The global best solution x^* is updated after local search (line 5), and useful

Algorithm *MCE*(Θ_1, Θ_2)
Input: parameters Θ_1 and Θ_2.
Output: The best solution found x^*.
1. Initialization. Set $c = 0$, history= $\emptyset$;
2. **while** computational budget has not been exceeded,
3. $x_c =$ **EvolutionaryAlgorithm**(Θ_1, history);
4. $x_c^* :=$ **LocalSearch**(x_c,Θ_2);
5. $x^* := \min\{x_j^*, 1 \leq j \leq c\}$;
6. $S :=$ **LearningFromHistory**();
7. $c := c + 1$; history := history $\bigcup$ S.
8. **end while**;

information S should be learned from the current cycle (**LearningFromHistory**) (line 6). The cycle index and the history information are updated hereafter (line 7). A new cycle starts if the computational budget has not been exceeded. The global best solution x^* is returned on termination.

III. Algorithm Instantiation

In this section, we develop a simple algorithm instantiation according to the generic scheme proposed in Section II. The exemplar algorithm is based on an EDA [18]. An EDA constructs a probability model $p(x;t)$, at each generation t, to represent the statistical information extracted from the visited promising solutions, while offsprings are sampled from the probability model. The way to construct the mathematical model differentiates the EDA instantiations. In this paper, we only consider real variables.

A. The Adaptive Probability Model and Multiple Sampling Strategy

Existing EDAs for real variables can be classified with respect to the probability model $p(x;t)$. In those EDAs, the assumed probability models include a Gaussian distribution [18], a Gaussian mixture [39], and a histogram [40][41]. In this paper, we propose to construct an adaptive univariate model. This model is assumed to be fully factorized over the variables, i.e. $p(x;t) = \prod_{i=1}^{n} p(x_i;t)$. Moreover, we propose to adaptively shrink the search space in order to increase the exploration speed of the EDA.

Suppose that at generation t, the selected population consists of $\mathcal{P}^s(t) = \left\{x^1(t), x^2(t), \cdots, x^K(t)\right\}$ where K is the size of the selected population. For the i-th dimension, $p(x_i;t)$ is computed as described in Algorithm *ProbMod*.

One can imagine that along with the evolution procedure, the search space will be adaptively shrink. This can increase the evolution search speed, but will inevitably cause premature convergence problems. To moderate this problem, in the adaptive univariate model, we use a predefined small positive real number ϵ to expand the search space at each dimension. Moreover, the premature convergence problem can also be moderated by using a *multiple sampling strategy*.

To our best knowledge, in almost all EDAs, the sampled offspring size is usually less than, or equal to, the population size. However, it is well known that to accurately characterize $p(x;t)$, a large sampling size is needed. Therefore, statistically speaking, a small sample size will result in high sampling noise, which can result in the false guidance problem. That is, the search may be lead to possibly non-promising areas due to the sampling noise. Once the search has been trapped in non-promising areas, there is no way to escape. Moreover, the problem becomes worse when $p(x;t)$ is complex. The multiple sampling strategy, i.e. generating offspring which are several times more numerous than the population size, can reduce the sampling noise.

Algorithm *ProbMod*
Input: The selection population $\mathcal{P}^s(t)$
Output: The probability $p(x_i;t)$ for the i-th component x_i.
1. Find $\ell_i^{\min}(\ell_i^{\max}) = \min(\max)\{x_i^k(t), 1 \leq k \leq K\}$;
2. Assign a small probability to the intervals $[\ell_i^{\min} - \epsilon, \ell_i^{\min}]$ and $[\ell_i^{\max}, \ell_i^{\max} + \epsilon]$, and a big probability to $[\ell_i^{\min}, \ell_i^{\max}]$.

Algorithm *GMSampling*
Input: a template solution x^*, a real number $0 \leq \alpha \leq 1$ and a probability model $p(x;t)$.
Output: An offspring x.
1. Set $U = \{1, 2, \cdots, n\}$ and $N := \lceil \alpha n \rceil$; Randomly select a set of indices $V \subset U$ with $|V| = N$;
2. For an index $i \in V$, set $x_i := x_i^*$; For an index $j \in U \setminus V$, sample a value y from the probability model $p(x_j)$, set $x_j := y$;
3. Return x;

B. Learning from Previous Searches

An important factor of the proposed framework is to learn from previous searches in order to improve the new search efficiency. This section provides a simple learning method.

The most important message we obtained from previous searches is the location information of the cycle best x_c^*, or the global best x^*. This information should be incorporated in the new search. One possible way to take advantage of the location information is to combine it in the sampling procedure by using the guided mutation method [28]. Alg. *GMSampling* describes the guided mutation offspring sampling, where $\lceil A \rceil$ rounds the elements of A to the nearest integers greater than or equal to A. Basically speaking, an offspring is generated by coping a part of x^*, and filling the other part with samples from the present probability model. The underlying rationale is that the location information of x^* provides some guiding information to the promising areas since x^* is the best solution found so far. The parameter α is problem-dependent.

C. Selection and Replacement

The selection process has been widely studied in the constrained evolutionary optimization literature. Selection methods based on penalty methods will bias the search, while those based on multi-objective approaches will not. However, as stated in [42], the unbiased search does not necessarily improve search efficiency. Moreover, it is obvious

that solution feasibility is more important than optimality in NLP. Since the local optimizer used has a higher chance to improve a feasible solution than an infeasible solution to the global optimum, we prefer to use a selection method that favours feasible solutions. The selection method, called the over-penalized approach in [42], is adopted in this paper.

In the over-penalized approach, the feasible individuals are ranked higher than the infeasible individuals. The feasible solutions are sorted according to their objective function values f. The infeasible individuals are ranked according to the penalty function values ψ, which is defined as follows: $\psi(x) = \sum_j g_j^+(x)^\beta$ where $g_j^+(x) = \max\{0, g_j(x)\}$, and $\beta = 2$.

In the selection (replacement) operations, the over-penalized approach is applied to rank the individuals (offsprings). At each generation, the best individuals are used to construct the probability model, while the rest of the population is replaced by the sampled best offsprings.

D. The Local Optimizer

We adopt a classical optimization method developed for NLP, called DONLP2 [43] to improve the best solution found at each cycle. DONLP2 is based on sequential quadratic programming method (SQP), in which fully regularized mixed constrained subproblems are used to deal with non-regular constraints.

The most important algorithmic parameters of the DONLP2, i.e. Θ_2 in Alg. *MCE*, include τ_0 which gives a bound describing how much the unscaled penalty-term (the L_1-norm of the constraint violation) may deviate from zero and δ_0 which is a binding constraint. In our experimental simulations, we set $\tau_0 = 1.0$ and $\delta_0 = 0.2$ as suggested in [43]. Moreover, we do not calculate the analytical form of the gradients and Hessian of the Lagrangian during our implementation, but using numerical differentiation to compute these values.

IV. Experimental Results

In the developed algorithm, called MCE, the parameters Θ_1 of the EDA includes the population size M, the selection size K, the sampling size L which is a multiple of the population size, i.e. $L = k * M$, and the guided mutation parameter α. In our implementation, the EDA is considered to have converged if in T consecutive generations, no better solution is found.

In this section, firstly we analyze the exploration capability of the developed EDA and its sensitivity to the parameters. Then we compare the MCE with the algorithms in the CEC 2006 competition. The test problems used in the CEC 2006 competition are used as our test problems. Readers are referred to [44] for detailed problem definitions. The same experimental configurations are applied. That is, the positive number to relax the equality constraints is set as $\varepsilon = 0.0001$, the number of runs is 25 and the maximum number of fitness evaluations (NFEs) is 500,000. The best known solutions x^* are given. In each run, the NFEs needed for finding a solution satisfying $f(x) - f(x^*) < 0.0001$ are to be recorded. Moreover, the feasible rate (number of feasible runs / total runs) and success performance (the mean of the NFEs for successful runs $\times$ number of total runs / number of successful runs) will be used as the comparison criteria between different algorithms.

A. Component Analysis

The two aspects which affect the performance of the MCE the most are the exploration capability of the EDA, and the guiding information learned from previous searches. The component analysis aims to investigate the respective contributions of these two aspects.

1) Components Study: The most important components of the algorithm are the probability model construction and the selection operation. The test problem g_{02} is used to study the effects of these components to the performance of the MCE in terms of the criteria described above.

To test the effects of the probability model, we compared the MCE with the proposed adaptive model (MCEA), and the MCE with the histogram model (MCEH) developed in [41], and the MCE with the Gaussian model (MCEG) developed in [18][45]. Those probability models are all univariate. In the histogram model, the bound of each variable is divided into 10 subintervals (as suggested in [41]), and the histogram of the selected individuals is normalized as the probability distribution over these subintervals. The Gaussian model [18][45] assumes that the selected individuals at each variable follows a Gaussian distribution. The mean and standard deviation are the sufficient statistics. It can be seen that the adaptive model shrinks the search space along with the evolution procedure. However, it also prevents premature convergence by assigning the same probability on the shrink interval, while the other two models do not shrink the search space. The comparison among these different probability models can be used to gain insight into their exploration capabilities.

The parameter settings of these algorithms are $M = 100$, $k = 1$ and $\alpha = 0.3$. Both the MCEA and the MCEH reach the global optimum in all the 25 runs, but the success rate of the MCEG is zero. We recorded the number of cycles used by these algorithm. Fig. 1 shows the boxplots of the number of fitness evaluations consumed by the MCEA, MCEH and MCEG, while the means of the number of cycles for the MCEA, MCEH and MCEG are 14.48, 32.24 and 83.36, respectively. From this figure, we see that the algorithm with the adaptive model needs fewer NFEs to reach the global optimum. Considering that the three algorithms have the same learning and intelligent sampling components, we may conclude that the adaptive univariate model has better exploration capability than the others.

We now study the effects of the over-penalized selection and the stochastic ranking selection on the algorithm performance. The compared algorithm, MCES, is the same as the MCEA, except the stochastic ranking selection is applied (where the stochastic ranking parameter $\rho = 0.45$ as suggested in [1]). The same algorithmic parameters as above are applied. The success rate of the MCES is only 0.24, i.e.

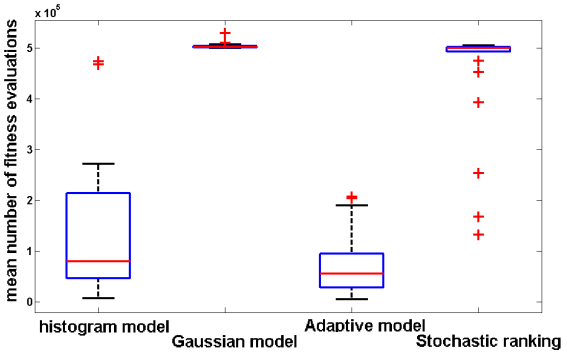


Fig. 1. Comparison of different statistical model and selection methods for $p(x;t)$ in terms of the NFEs used to reach the global optimum.

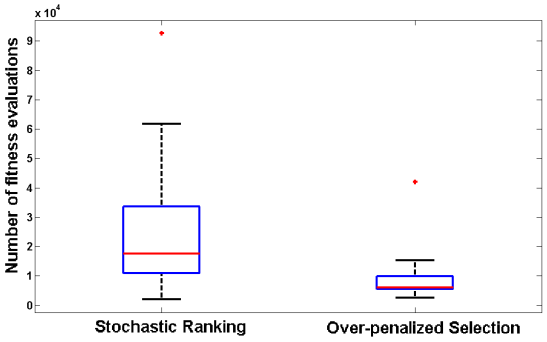


Fig. 2. The boxplot of the number of fitness evaluations consumed by using stochastic ranking and over-penalized approach.

only in six runs, the MCES reaches the global optimum. The success performance of the MCEA and the MCES are 73224.5 and 11,413,763, respectively. The last column in Fig. 1 shows the boxplot of the NFEs in the 25 runs of the MCES. From these experiments, it is clear that stochastic ranking does not necessarily improve the search efficiency. Moreover, in comparison with the results in the histogram model column, one can see that the algorithm performance with stochastic ranking is even worse than the MECH. This shows that the exploration capability of the developed EDA does not benefit from the application of stochastic ranking.

2) Sensitivities of the MCEA to the Algorithmic Parameters: In this section, we study the effect of the guiding information. The effect can be investigated by looking at the performance of the MCEA with different α values and population size M. The population size M seriously affects the exploration capability of an EA, and α controls the contributions of the location information of the best solution found so far for the further search.

The most important parameters of the MCEA is the population size M, the multiplier k and the guided mutation parameter α. The rest of the parameters, i.e. the selection size K and the EDA stop parameter T are set as $K = M/2$ and $T = 5$. From our experiments, we found the settings for K and T can be fixed for all the problems, i.e. they are problem-independent. The population size and the guided mutation parameter are problem-dependent. The population size M can seriously affect the exploration capability of an EA, k should be studied to verify our claim about the sampling noise, and α controls the usefulness of the location information of the best solution found so far to further search.

In our experiment, the study was carried out by varying $\alpha \in \{0.0, 0.1, \cdots, 0.9\}$ values and $M \in \{20, 40, 60, 80, 100\}$. It can be imagined that the population size will affect the setting of the optimal α. A large population size will increase the NFEs used in a cycle, but in the meanwhile will improve the EA's search ability. A large α value can increase the search speed of the EDA at later cycles, and hence a large population size is required to compensate to avoid a quick loss of diversity which will deteriorate the exploration ability. Therefore, optimal settings of the population size and α will trade-off the computational cost and the search quality.

In this section, we also use g_{02} as an example to study the interactions between the population size M and α. Fig. 3 shows the results obtained when studying the relationship between M and α. The MCEA with all M and α settings, except the combinations $\alpha = 0.0$ and $M = 20, 40, 60, 80$, reach the global optimum in all the 25 runs. Fig. 3(a) shows the success rate and the mean number of cycles used in the search. From the figure, it can be seen that with a small population size, the MCEA cannot reach the global optimum. On the other hand, a large population size ($M = 100$) can guarantee success in the case of $\alpha = 0.0$. Note that, in this case, there is no learning from the previous searches, which suggests that the exploration ability of the adaptive univariate model is seriously affected by the population size. Moreover, note that all the runs when $\alpha > 0.0$ successfully locate the global optimum, indicating that the proposed learning method is effective.

Fig. 3(b) shows the number of fitness evaluations used to reach the global optimum in case $\alpha \neq 0$, while (c) shows the average number of cycles. From Fig.3(c), one can see that, along with the increase of M, the MCEA requires fewer cycles. This again indicates that the exploration capability increases along with the increase in M. However, it can be seen that the minimal number of cycles used to reach the global optimum does not occur when the maximum population size (100) is applied. This implies that there is a trade-off between the optimal population size and α. From Fig. 3(a), one can see that (roughly speaking) the smaller the population size, the less computational cost. Recalling that a larger population size indicates a better exploration capability, this observation indicates that the learning method can compensate for the quick loss of diversity due to a small population size. Fig. 3(d) shows the change of the number of fitness evaluations and the number of cycles along with the increase of the guided mutation α. It can be seen that $\alpha = 0.2$ achieves the best performance, and the number of cycles is highly correlated with the number of fitness evaluations.

Fig. 4 shows the results obtained in the study of the population size and the sampling size. In the study, α is

fixed as 0.2. Fig. 4(a) shows the results obtained from the combination study of the population size and the sampling size, while (b) shows the number of cycles. From Fig. 4(b), it can be seen that the number of cycles tends to decrease along with the increase of the population size and the increase of the sampling size. In Fig. 4(a), one can see that a larger sampling size does not result in a reduced computational cost. There should be a balance between the sampling size and the population size. More specifically, we can see that in case $k = 1$, the consumed computational cost is fewer than that in the case $k = 0.5$ when the population size is less than 80. This observation justifies that more samples can reduce the sampling noise, as claimed in Section III-A.

In summary, we may conclude from the above experiments that (i) the adaptive univariate model can improve the exploration ability of the proposed EDA; (ii) the learning strategy can compensate for the loss of diversity caused by employing small population size in the search; and (iii) the multiple sampling strategy works in cases of small population size.

B. Comparison with the Best-Known EA

In this section, we present the comparison of our algorithm with the algorithms in the CEC 2006 competition. The 24 test problems except g_{20} and g_{22} are used as the test bed (as discovered in the CEC 2006 competition, no algorithms were able to find the global optimum of the two test functions g_{20} and g_{22}). The winning algorithm of the competition was $\varepsilon-$DE. It has 8 parameters. In contrast, our algorithm has 5 main parameters. In our experiments, we set these parameters as $M = 2 \times n$, where n is the problem dimension, $K = n$, $T = 5$, and $\alpha = 0.2$.

The experimental results are shown in Table I. Since both algorithms can reach the global optimum of the test problems with total reliability, the success rate is omitted. The minimal, mean and maximum number of fitness evaluations in the 25 runs for the MCEA are shown in the 'min', 'mean' and 'max' columns, respectively. The column '$\varepsilon-$DE' shows the average number of fitness evaluations obtained in [35], and the last column shows the least NFEs used by the algorithms which appeared in the CEC 2006 competition. These algorithms are SaDE [46], MDE [7], MPDE [47], GDE [48], PCX [49], DMS [50] and jDE-2 [51].

From the table, it can seen that for g_{17} the average NFEs used by the MCEA is more than that of the MDE, but fewer than that of the ε-DE. Except g_{17}, the MCEA can reach the global optima of the test functions by using much fewer NFEs than all the other compared algorithms. Thus, we may conclude that the MCEA proposed here outperforms all the evolutionary algorithms that appeared in the CEC 2006 competition.

TABLE I

COMPARISON RESULTS BETWEEN THE MCEA, THE WINNER OF THE CEC2006 COMPETITION $\varepsilon-$DE, AND THE BEST RESULTS OBTAINED BY THE ALGORITHMS APPEARED IN THE CEC 2006.

function	MCEA			$\varepsilon-$DE	Best (Alg.)
	min	mean	max		
g_{01}	4,166	7,859	20,237	59,309	25,115 (SaDE)
g_{02}	8,033	45,555	111,784	149,827	96,222 (MDE)
g_{03}	2,207	3,673	9,551	89,407	24,861 (MPDE)
g_{04}	1,021	1,201	1,891	26,216	15,281 (GDE)
g_{05}	722	834	1,160	97,430	21,306 (MDE)
g_{06}	346	489	850	7,381	5,202 (MDE)
g_{07}	2,685	3,588	4,921	74,304	26,578 (DMS)
g_{08}	649	1,068	2,035	1,139	918 (MDE)
g_{09}	1,483	1,632	1,852	23,121	16,152 (MDE)
g_{10}	3,928	17,319	82,377	105,234	25,520 (DMS)
g_{11}	346	362	392	16,420	3000 (MDE)
g_{12}	324	348	434	4,124	1,308 (MDE)
g_{13}	1,118	6,595	48,287	31,096	21,723 (MDE)
g_{14}	2,230	8,012	34,465	113,439	25,220 (DMS)
g_{15}	375	477	826	83,655	10,458 (MDE)
g_{16}	1,096	7,092	24,627	19,122	8,730 (MDE)
g_{17}	20,845	86,721	274,511	98,860	26,364 (MDE)
g_{18}	4,724	11,095	25,979	59,153	28,261 (DMS)
g_{19}	5,223	13,355	19,629	356,350	21,830 (DMS)
g_{21}	2,654	18,488	56,676	135,142	38,217 (PCX)
g_{23}	2,832	5,141	11,970	200,763	129,550 (SaDE)
g_{24}	346	425	1,357	2,952	1,794 (jDE-2)

V. CONCLUSION

In this paper, we have presented a constrained evolutionary algorithm based on an estimation of distribution algorithm and a novel hybridization strategy with a classical local optimizer DONLP2. The novel strategy clearly distinguishes the roles of the EA and the local optimizer in the search procedure, and attempts to improve the search efficiency of the advanced evolutionary stages through learning from previous cycles. In the experiments, we studied the components of the developed EDA to investigate its exploration capability, and the proposed learning strategy to show its advantages. The developed algorithm was then compared with the evolutionary algorithms in the CEC 2006 competition. The comparison results show that the proposed algorithm outperforms all these CEC2006 algorithms including the winner of the competition, ε-DE. Future work will focus on the development of new learning strategies to reduce the degree of randomness of the advance evolutionary search stages, and new EAs that have better exploration capabilities.

ACKNOWLEDGEMENT

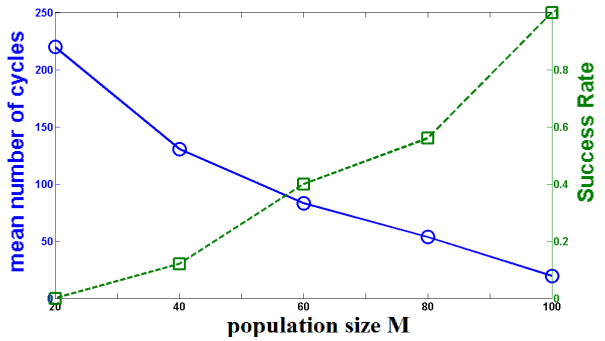
JS was funded to carry out this work by the grant BB/D019613/1 the Centre for Plant Integrative Biology, which is a Centre for Integrative Systems Biology funded by the UK BBSRC and EPSRC.

REFERENCES

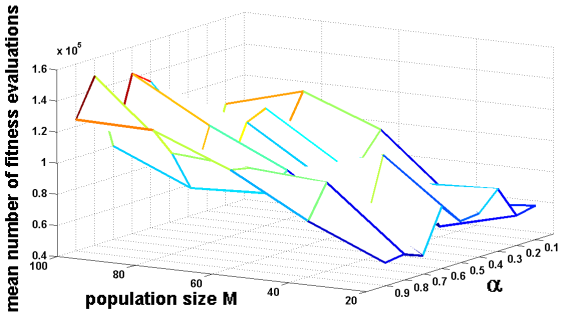
[1] T. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2002.

[2] C. Fonseca and P. Fleming, "Multiobjective optimization and multiple constrained handling with evolutionary algorithms– part i: A unified formulation," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.

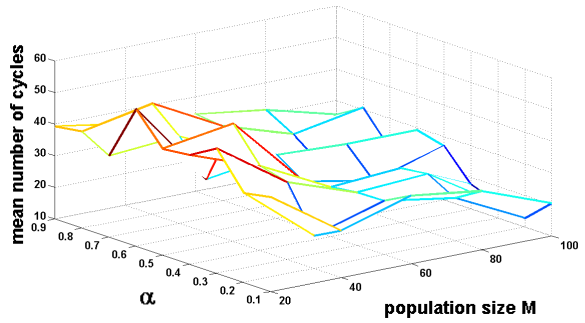
[3] E. Mezura-Montes and C. Coello, "A survey of constraint-handling techniques based on evolutionary multiobjective optimization," in *PPSN workshop on Multiobjective Problem Solving from Nature*, 2006.



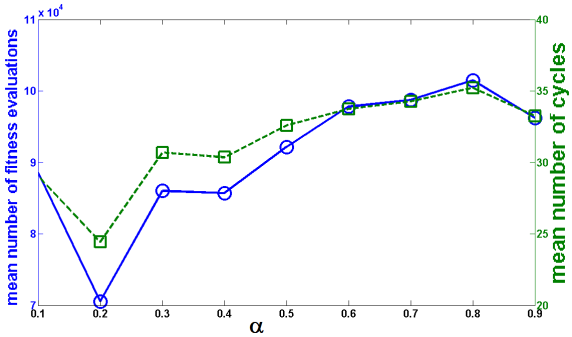
(a) The success rate and number of cycles against the population size when $\alpha = 0.0$.



(b) The interactions of M and α in terms of NFEs.

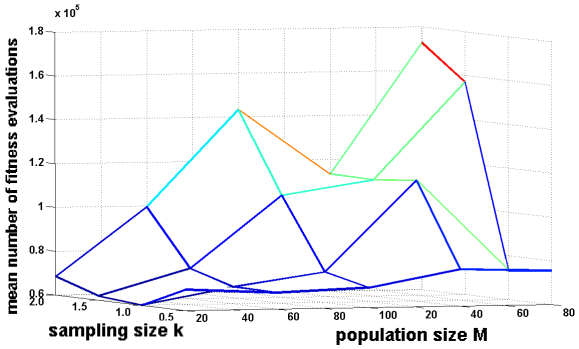


(c) The interactions of M and α in terms of NCs.

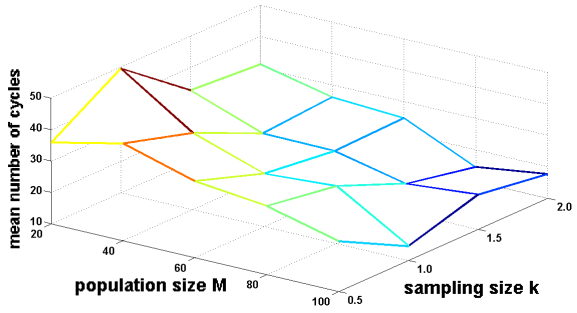


(d) The mean NFEs and cycles against α.

Fig. 3. The study of the interactions of the algorithmic parameters M and α to the performance of the MCEA. (a) is the success rate and the number of cycles used when $\alpha = 0.0$; (b) shows the results in terms of the NFEs; (c) is the results in terms of the number of cycles; (d) shows the mean NFEs and cycles w.r.t. different α.



(a) The interactions of M and k in terms of NFEs.



(b) The interactions of M and k in terms of NCs.

Fig. 4. The study of the interactions of the algorithmic parameters M and k to the performance of the MCEA. (a) shows the results in terms of the NFEs; (b) is the results in terms of the number of cycles.

[4] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

[5] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[6] C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[7] E. Mezura-Montes and C. Coello, *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing. Springer Berlin Heidelberg, 2005, ch. Use of Multiobjective Optimization Concepts to Handle Constraints in Genetic Algorithms, pp. 229–254.

[8] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence Series. Springer, 2009, vol. 198.

[9] T. Takahama and S. Sakai, "Constrained optimization by α constrained genetic algorithm (αga)," *Systems and Computers in Japan*, vol. 35, no. 5, pp. 11–22, 2004.

[10] J.-H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 129–140, 1997.

[11] T. Takahama and S. Sakai, *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer

Berlin / Heidelberg, 2009, vol. 198, ch. Solving Difficult Constrained Optimization Problems by the ε Constrained Differential Evolution with Gradient-Based Mutation, pp. 51–72.

[12] A. El-Gallad, M. El-Hawary, and A. Sallam, "Swarming of intelligent particles for solving the nonlinear constrained optimization problem," *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 9, no. 3, pp. 155–163, 2001.

[13] R. Hinterding and Z. Michalewicz, "Your brains and my beauty: Parent matching for constrained optimization," in *Proceedings of the 5th International Conference on Evolutionary Computation*, 1998, pp. 810–815.

[14] G. Bilchev and I. Parmee, "Constrained and multi-modal optimization with an ant colony search model," in *Proceedings of the 2nd International Conference on Adaptive Computing in Engineering Design and Control*, I. Parmee and M. Denham, Eds., University of Plymouth, Plymouth, UK, March 1996.

[15] P. Hajela and J. Yoo, "Constrained genetic search via schema adaptation: An immune network solution," *Structural Optimization*, vol. 12, pp. 11–15, 1996.

[16] R. Becerra and C. Coello, "Cultured differential evolution for constrained optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33-36, pp. 4303–4322, 2005.

[17] F. Gao, G. Cui, and H. Liu, *ICONIP 2006, Part III*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2006, vol. 4234, ch. Integration of Genetic Algorithm and Cultural Algorithms for Constrained Optimization, pp. 817–825.

[18] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[19] J. Grahl and F. Rothlauf, "PolyEDA: Combining estimation of distribution algorithms and linear inequality constraints," in *GECCO*, 2004, pp. 1174–1185.

[20] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV*, 1996, pp. 178–187.

[21] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuos search spaces," in *Parallel Problem Solving from Nature - PPSN V*. Springer-Verlag, 1998, pp. 418–427, berlin.

[22] T. V. Le, "A fuzzy evolutionary approach to constrained optimization problems," in *Proceedings of the second IEEE Conference on Evolutionary Computation*. Perth: IEEE, November 1995, pp. 274–278.

[23] C.-J. Chung and R. Reynolds, "A testbed for solving optimization problems using culture algorithms," in *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*. Cambridge, Massachusetts: MIT Press, 1996.

[24] Z. Michalewicz and C. Janikow, "Handling constraints in genetic algorithms," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo, California: Morgan Kaufmann Publishers, 1991, pp. 151–157.

[25] B. Wah and Y. Chen, "Hybrid constrained simulated annealing and genetic algorithms fornonlinear constrained optimization," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 2, 2001, pp. 925–932.

[26] T. Takahama, S. Sakai, and N. Iwane, *AI 2005*, ser. LNAI. Springer-Verlag Berlin Heidelberg, 2005, vol. 3809, ch. Constrained Optimization by the ε Constrained Hybrid Algorithm of Particle Swarm Optimization and Genetic Algorithm, pp. 389–400.

[27] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[28] Q. Zhang, J. Sun, and E. Tsang, "Evolutionary algorithm with the guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 192–200, 2005.

[29] S. Belur, "CORE: Constrained optimization by random evolution," in *Late Breaking Papers at the Genetic Programming 1997 Conference*, J. Koza, Ed. Stanford University, California: Stanford Bookstore, July 1997, pp. 280–286.

[30] E. Zahara and C.-H. Hu, "Solving constrained optimization problems with hybrid particle swarm optimization," *Engineering Optimization*, vol. 40, no. 11, pp. 1031–1049, 2008.

[31] E. Zahara and Y.-T. Kao, "Hybrid nelder-mead simplex search and partical swarm optimization for constrained engineering design problems," *Expert Systems with Applications*, vol. 36, pp. 3880–3886, 2009.

[32] K. Deb, S. Lele, and R. Datta, *ISICA 2007*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2007, vol. 4683, ch. A Hybrid Evolutionary Multi-objective and SQP Based Procedure for Constrained Optimization, pp. 36–45.

[33] H. Myung and J.-H. Kim, *Proceedings of the Sixth Annual Conference on Evolutionary Programming*, ser. Lecture Notes on Computer Science. Springer-Verlag, 1997, vol. 1231, ch. Evolian: Evolutionary Optimization based on Lagrangian with constraint scaling, pp. 177–188.

[34] C. Pedamallu and L. Ozdamar, "Investigating a hybrid simulated annealing and local search algorithm for constrained optimization," *European Journal of Operational Research*, vol. 185, pp. 1230–1245, 2006.

[35] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006, pp. 308–315.

[36] Y. Wang, Z. Cai, Y. Zhou, and Z. Fun, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," *Struct Multidisc Optim*, vol. 37, pp. 295–413, 2009.

[37] K. D. Jong, "An analysis of behavior of a class of genetic adaptive systems," Ph.D. dissertation, The University of Michigan, Ann Arbor, Michigan., 1975.

[38] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on In Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[39] P. A. N. Bosman and D. Thierens, "Expanding from discrete to continuous estimation of distribution algorithms: The IDEA," in *Parallel Problem Solving from Nature – PPSN VI. Lecture Notes in Computer Science 1917*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., 2000, pp. 767–776.

[40] S. Tsutsui, M. Pelikan, and D. Goldberg, "Evolutionary algorithm using marginal histogram models in continuous domain," in *Proc. of the 2001 Genetic and Evolutionary Computation Conference Workshop*, San Francisco, CA, 2001, pp. 230–233.

[41] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Hybrid estimation of distribution algorithm for global optimisation," *Engineering Computations,*, vol. 21, no. 1, pp. 91–107, 2003.

[42] T. Runarsson and X. Yao, "Search biases in constrained evolutionary optimization," *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, vol. 35, no. 2, pp. 233–243, 2005.

[43] P. Spellucci, "An SQP method for general nonlinear programs using only equality constrained subproblems," *Math. Prog.*, vol. 82, pp. 413–448, 1998.

[44] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," Tech. Rep., 2006.

[45] J. Sun, Q. Zhang, and E. Tsang, "DE/EDA: A new evolutionary algorithm for global optimisation," *Information Sciences*, vol. 169, no. 3-4, pp. 249–262, 2005.

[46] V. Huang, A. Qin, and P. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006.

[47] M. F. Tasgetiren and P. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problems," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006.

[48] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006.

[49] K. Deb, A. Sinha, and S. Aravind, "A population-based, parent centric procedure for constrained real-parameter optimization," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006.

[50] J. Liang and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006.

[51] J. Brest, V. Zumer, and M. Maućec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006.