Real-Coded Extended Compact Genetic Algorithm based on Mixtures of Models

Pier Luca Lanzi, Luigi Nichetti, Kumara Sastry
Davide Voltini, David E. Goldberg
IlliGAL Report No. 2008008
May, 2008

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

Real-Coded Extended Compact Genetic Algorithm based on Mixtures of Models

Pier Luca Lanzi*†, Luigi Nichetti†, Kumara Sastry*,
Davide Voltini†, David E. Goldberg*
*Illinois Genetic Algorithm Laboratory (IlliGAL)
University of Illinois at Urbana Champaign, Urbana, USA
†Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
pierluca.lanzi@polimi.it, kumara@kumarasastry.com, deg@uiuc.edu

May 13, 2008

Abstract

This paper presents a real-coded estimation distribution algorithm (EDA) inspired to the extended compact genetic algorithm (ECGA) and the real-coded Bayesian Optimization Algorithm (rBOA). Like ECGA, the proposed algorithm partitions the problem variables into a set of clusters that are manipulated as independent variables and estimates the population distribution using marginal product models (MPMs); like rBOA, it employs finite mixtures of models and it does not use any sort of discretization. Accordingly, the proposed real-coded EDA can be either viewed as the extension of the ECGA to real-valued domains by means of finite mixture models or as a simplification of the real-coded BOA to the marginal product models (MPMs). The results reported here show that the number of evaluations required by the proposed algorithm scales sub-quadratically with the problem size in additively separable problems.

1 Introduction

Estimation of distribution algorithms (EDAs) [16, 21, 26, 18] replace the traditional variation operators of genetic algorithms by building and sampling probabilistic models. EDAs have successfully solved boundedly-difficult single-level and hierarchal problems, oftentimes requiring only sub-quadratic number of function evaluations [21]. Despite their demonstrated scalability, most EDAs operate on binary variables, and their success has not been extensively carried over to other encodings such as permutation, program and real-codes. Moreover, existing real-coded EDAs are fairly complex, rendering them intractable for theoretical, analytical, or even semi-empirical analysis.

In the recent years, several real-coded extensions of the extended compact genetic algorithm (ECGA) [13] have been developed as a way to provide simple real-coded EDAs [7, 10, 17]. Such real-coded ECGAs usually combine a discretization algorithm, which is used to map real-values to discrete symbols, with a probabilistic model based on marginal product models (MPMs). In this paper, we follow a different approach and propose a native real-coded ECGA that directly works on the real variables and does not employ any sort of discretization. The proposed algorithm is inspired to the extended compact genetic algorithm (ECGA) [13] and to the real-coded Bayesian Optimization Algorithm [1, 2]. Similarly to what done in ECGA [13], the proposed algorithm works on probability distributions represented by marginal product models (MPMs) and, for this

purpose, it partitions the (real-valued) problem variables into clusters that are then manipulated as independent variables. However, instead of mapping the real values into discrete symbols and using the usual MDL to discriminate between candidate models, the proposed method models each cluster using a joint probability distribution and guides the partitioning process using an MDL metric for continuous distributions. Similarly to what done in rBOA [1, 2], the parameters of the probabilistic model are fitted using several mixtures of models, one for each cluster of variables identified during the model building step. Accordingly, the proposed real-coded ECGA can be either viewed as an extension of ECGA to real-valued domains by means of finite mixtures of models or as a simplification of the real-coded BOA to the case of a probabilistic model based on marginal product models (MPMs). We performed a scalability analysis an applied the proposed algorithm to problems taken from the literature, namely the sphere function and the real deceptive function [1, 2]. The results we present show that the number evaluations required to solve additively separable problems scales at most sub-quadratically with the problem size.

The paper is organized as follows. At first, we briefly overview related real-coded EDA designs. In Section 3, we give an outline of the extended compact genetic algorithm followed, in Section 4, by a brief description of the real-coded BOA. In Section 5, we describe the proposed real-coded ECGA and then, we discuss the experimental design (Section 6) which we followed for the population-sizing and scalability analysis we present in section 7.

2 Related Work

We now provide a brief overview of the works that are relevant to this paper; for more references, we refer the reader to the two recent books [26, 18]. In [31], Pelikan et al. introduced a real-coded PMBGA working in the continuous domain; they used marginal histograms to model promising solutions. Both the marginal models used, fixed width and fixed height histograms, performed fairly well on test functions with no or weak interaction among the variables, but failed to recognize linkage among variables in functions with a medium level of linkage. An evolution of this approach is presented in [30] where the focus is on the linkage identification in real-coded GAs. In [30], the authors applied the SPX operator [15] as recombination operator and tried to identify linkage information by observing the distribution of the individuals in the population; in particular they examined the correlation coefficient matrix of parameter values of the individuals in the population.

Pelikan et al. [23, 24] combined the Bayesian Optimization Algorithm for recombination with evolutionary strategies (ES) for mutation. In [23, 24], a real population is first discretized, then BOA is applied to recombine individuals in the discrete population and the new population is mapped back to the real domain so that adaptive mutation from ES can be applied to obtain the next real population. Three discretization strategies were presented: *Fixed-Width Histograms*, *Fixed-Height Histograms* and *k-Means Clustering*. The approach was tested on three functions and the reported results showed good scalability. Ahn [1] developed a real-coded Bayesian Optimization Algorithm, rBOA, which constructs the Bayesian factorization graph using finite mixture models. All the relevant substructures are extracted from the graph and each substructure is fit and sampled independently.

Chen et al. [7] developed a real-valued version of the ECGA (rECGA) by combining a binary version of ECGA together with *Split-On-Demand* (SoD) discretization—an adaptive discretization technique that takes into account the distribution of the current population in creating discrete intervals. rECGA was tested on several problems and the accuracy of the computed solutions was shown. Fossati et al. [10] introduced a simple real-coded ECGA which combines basic discretization with a χ-ary ECGA [29, 9]. Minqiang et al. [17] also proposed another version of real-coded ECGA

in which continuous variables are initially discretized and then clustered together to build the probabilistic model.

3 Extended Compact Genetic Algorithm (ECGA)

The extended compact genetic algorithm (ECGA) [11, 13] is an estimation of distribution algorithm (EDA) that replaces traditional variation operators of genetic and evolutionary algorithms by building a probabilistic model of promising solutions and sampling the model to generate new candidate solutions. ECGA starts with a population of random individuals and repeats the following steps until a stopping criterion is satisfied. First, the fitness of individuals is computed, selection is applied, and a probabilistic model is learned from the population of selected individuals. The probability distribution used in ECGA is a class of probability models known as marginal product models (MPMs). MPMs partition genes into mutually independent groups and specifies marginal probabilities for each linkage group. To distinguish between better model instances from worse ones, ECGA uses a minimum description length (MDL) metric [27]. The key concept behind MDL models is that all things being equal, simpler models are better than more complex ones. The MDL metric used in ECGA is a sum of two components: *model complexity*, which quantifies the model representation size in terms of number of bits required to store all the marginal probabilities, and *compressed population complexity*, which quantifies the data compression in terms of the entropy of the marginal distribution over all partitions. In ECGA, both the structure and the parameters of the model are searched and optimized to best fit the data. While the probabilities are learned based on the variable instantiations in the population of selected individuals, a greedy-search heuristic is used to find an optimal or near-optimal probabilistic model. The search method starts by treating each decision variable as independent and then it continues by merging two partitions that yields greatest improvement in the model-metric score. The subset merges are continued until no more improvement in the metric value is possible. The probabilistic model is then sampled to create an offspring population which is combined with the original population to generate the next population.

4 Real-Coded BOA

The real-coded BOA has been been developed by Ahn et al. [4, 2] as an extension of Pelikan's BOA [26] to real-valued domains. rBOA constructs a Bayesian factorization graph using finite mixture models and it does not apply any sort of discretization. The real-coded BOA works as a typical EDA. It starts from an initial population of random individuals and then it applies the following steps until a stopping criterion is met. First, selection is applied to the current population and generates a new population containing most promising individuals. Then, a model building is applied to learn a probabilistic model of the population of selected individuals. Next, the model is sampled to generate a new population which is used together with the original population to generate the new population which will replace the original one. These four steps, selection, model building and sampling, and replacement are repeated until a termination criterion is satisfied.

To learn the probabilistic model $M(\zeta, \theta)$, with structure ζ and parameters θ, for the current population rBOA first applies k-means [14] to partition the population into a set of k_s clusters. Then, it applies an incremental greedy search, guided by the Bayesian Information Criterion, to find for the best Bayesian factorization graph that can model all the k_s clusters. Then, model fitting is performed and for this purpose another clustering algorithm, the randomized leader algorithm (RLA) [14], is applied to extract the k_f substructures that are then separately fitted using different

Algorithm 1 Pseudo-code of the real-coded ECGA.

```
1:  procedure RECGA
2:      var t;                                          ▷ Time step
3:      var P(·);                                       ▷ Population at time t
4:      var ζ;                                          ▷ Structure of the probabilistic model
5:      var Θ;                                          ▷ Parameters of the probabilistic model
6:      t ← 0;
7:      RandomInit(P(t));
8:      repeat
9:          EvaluateFitness(P(t));
10:         P_sel ← Selection(P(t));
11:         ζ ← ComputeModelStructure(P_s(t));
12:         Θ ← ComputeModelParameters(P_sel,ζ);
13:         P_sam ← Sample(M(ζ, Θ_ζ));                  ▷ Sample the model
14:         P_new ← Replacement(P(t),P_sam);
15:         P(t + 1) ← P_new;
16:         t ← t + 1;
17:     until StopCriterionNotMet
18: end procedure
```

mixtures of models. All the k_f mixtures are then combined together in an overall probabilistic model which is finally sampled to generate a new population. The sampled population and the original population are then combined to produce the next population.

5 Real-Coded ECGA

Our real-coded ECGA is largely inspired to the real-valued Bayesian Optimization Algorithm (rBOA) [4, 2] and to the χ-ary Extended Compact Genetic Algorithm (χ-ECGA) [9]. Like rBOA, our real-coded ECGA use clustering and joint normal distributions to build and sample probabilistic models defined over the real domain, thus is does not employ any sort of discretization as other versions of real-coded ECGA [8, 10, 6]. Like χ-ECGA [9], it uses marginal product models (MPMs) to partition variables into mutually independent groups and specifies marginal probabilities for each linkage group.

The algorithm is structured as the typical EDA for real-valued domains, see Algorithm 1. At first, the population is randomly initialized (line 7) using a uniform distribution. Then the following steps are repeated until a stop criterion is verified (line 17). The fitness of individuals in the current population $P(t)$ is computed (line 9) and selection is applied to $P(t)$ which generates P_{sel}. Several selection strategies can be used, in this works we tested two of them, tournament selection, which has been used both in BOA [22] and χECGA [9], and truncation selection used in rBOA [4, 2]. Next, the structure ζ of the probabilistic model that best represent the selected population P_{sel} is computed (line 11) and from the structure ζ and the population P_{sel} the parameter vector Θ is computed. Then the probabilistic model $M(\zeta, \Theta)$, with structure ζ and parameters Θ, is sampled to generate the sampled population P_{sam} (line 13). At the end, restricted tournament replacement (RTR) [20, 12] is applied to the original population $P(t)$ and to the sampled population P_{sam} to generate the new population P_{new} (line 14) which will replace the original one (line 15).

5.1 The Probabilistic Model

Our approach is characterized by a probabilistic model $M(\zeta, \Theta_\zeta)$ described by the structure ζ and the parameter set Θ_ζ. The former describes the relations among the problem variables as in the typical ECGA, that is,

$$\zeta = \{S_1, \ldots, S_{|\zeta|}\}$$

where S_j is a subset of $|S_j|$ genes, and $|\zeta|$ is the number of subsets. The parameter set Θ_ζ contains the parameters of the $|\zeta|$ joint normal distributions which model the distribution for each variable subset S_j. Since the subsets are independent, $M(\zeta, \Theta)$ is computed as,

$$M(\zeta, \Theta) = \prod_{j=1}^{|\zeta|} f(S_j, \vartheta_j)$$

where $f(S_j, \vartheta_j)$ is the joint normal distribution for the variables in the subset S_j whose parameters ϑ_j are defined by an array of means (one for each variable) and a covariance matrix; overall, ϑ_j contains $\frac{1}{2}|S_j|^2 + \frac{3}{2}|S_j|$ parameters; the model parameter set Θ_ζ denotes the set of all the parameter vectors ϑ_j.

5.2 Model Selection

The goal of this step is to find the model structure ζ that better fits the current population. The search procedure is based on the same greedy-search heuristic used in ECGA [11, 13]. Initially, the decision variables are considered independent and, in the next steps, the search continues by merging two partitions that yields greatest improvement in the model-metric score. The subset merges are continued until no more improvement in the metric value is possible.

5.2.1 Model Scoring

The evaluation of a model structure is based on the same MDL principle used in ECGA, which takes into account the accuracy of the model and its complexity. The score of a model represented by $f(\zeta, \Theta_\zeta)$ is computed as,

$$Accuracy(f(\zeta, \Theta_\zeta)) + Complexity(f(\zeta, \Theta_\zeta))$$

As done in rBOA [1, 2], we evaluate the accuracy of $f(\zeta, \Theta_\zeta)$ using a mixture of k_s models that are obtained by clustering the selected population into k_s clusters and applying the same structure ζ to model each cluster. This step is performed, as in rBOA [1, 2], by applying the k-means algorithm with the Euclidean distance as the similarity measure among individuals. The model $f(\zeta, \Theta_\zeta)$ is thus computed as,

$$f(\zeta, \Theta_\zeta) = \sum_{i=1}^{k_s} \alpha_i \prod_{j=1}^{|\zeta|} f(S_j, \vartheta_j^i)$$

where, k is the number of models used in the mixture, α_i is the relative weight of cluster k_i with respect to the entire population, $\alpha_i = |k_i|/|P_{sel}|$, and ϑ_j^i represents the parameters of the joint normal distribution which describes the variables in S_j for the individuals in cluster k_i.

Model complexity takes into account the number of parameters $|\Theta_\zeta|$ computed as the sum of the parameters of each MPM model for each one of the k clusters. However, since all the MPMs have the same structure ζ, each component has the same number of parameters. More precisely,

Algorithm 2 Estimation of the Model Parameters.

procedure ComputeModelParameters(P_{sel},ζ)
  var $\Theta = \{\}$; ▷ The set of model parameters is initially empty.
  for $S_j \in \zeta$ **do** ▷ For each variable subset in the structure ζ
    ▷ Apply clustering on P_{sel} using the variables in S_j.
    $\langle k_j^1, \ldots, k_j^{C_j} \rangle \leftarrow$ Clustering(P_{sel},S_j);
    for (i=1; i$\leq C_j$; i++) **do**
      $\mu_j^i \leftarrow$ ComputeAverages(P_{sel},S_j,k_j^i);
      $\Sigma_j^i \leftarrow$ ComputeCovarianceMatrix(P_{sel},S_j,k_j^i);
      $\Theta \leftarrow \Theta \cup \mu_j^i \cup \Sigma_j^i$;
    end for
  end for
  return Θ;
end procedure

each MPM is a product of ζ marginal distributions on independent subsets. Thus, each distribution $f(S_j, \vartheta_j^i)$ linked to subset S_j and the cluster k_i, being a joint normal distribution on the variables in S_j, needs $\frac{1}{2}|S_j|^2 + \frac{3}{2}|S_j|$ parameters. The overall model complexity is therefore computed as,

$$Complexity(f(\zeta, \Theta_\zeta)) = \lambda \times \ln(|PopSel|) \times K \times \sum_{j=1}^{|\zeta|} \left(\frac{1}{2}|S_j|^2 + \frac{3}{2}|S_j| \right) \tag{1}$$

where the parameter λ determines how much the model complexity influences its evaluation, like in rBOA [1].

5.3 Model Fitting

Given the model structure ζ, the model parameters Θ are computed for each subset S_j separately. To improve the overall model accuracy, each marginal distribution for S_j is represented as a mixture of distributions that are obtained through a clustering step. As in the initial clustering, the similarity measure among individuals is measured using the Euclidean distance. However, instead of using a fixed number k_s of clusters, as in rBOA [1], we apply an adaptive clustering algorithm, namely the BEND random leader algorithm (RLA) [5, 14] with the same threshold used in [1]. Given, the subset S_j, to compute the model parameters clustering is applied to extract C_j clusters. For each cluster, the parameters of the joint normal distribution $f(S_j, \vartheta_j^i)$ are computed. All the distributions are then mixed to build the overall model as follows,

$$M(\zeta, \Theta_\zeta) = \prod_{j=1}^{|\zeta|} \sum_{j=1}^{C_j} \beta_{i,j} f(S_j, \vartheta_j^i) \tag{2}$$

where C_j is the number of clusters extracted for the data identified by the variable subset S_j; $\beta_{i,j}$ is the weight of the i-th cluster of the j-th subset with respect to the whole population and it is computed as the ratio between the cluster size $|k_j^i|$ and the population size. The procedure is reported as Algorithm 2.

5.4 Model Sampling

The model $M(\zeta, \Theta_\zeta)$ is finally sampled to generate a new population. This step is perfomed as in the typical ECGA and χ-ECGA, but in our case separate joint normal distributions are sampled.

The generation of a new individual from the model $M(\zeta, \Theta_\zeta)$ works as follows. At first, for every subset S_j, one of the model components $f(S_j, \vartheta_j^i)$ is selected with a probability proportional to their weight $\beta_{i,j}$ (Equaton 2). Then, as in ECGA, each selected component is sampled and the new individual is generated by merging the different components corresponding to each one of the subsets that describe the model structure ζ.

6 Design of Experiments

We performed a scalability analysis of our simple real-coded ECGA using a problem that does not require linkage learning and a deceptive problem that needs linkage learning, both taken from the literature [1, 2, 25]. The former is the sphere function function (Figure 1),

$$f_s(\vec{x}) = \sum_{i=0}^{n-1} x_i^2,$$

where n is the number of variables and $x_i \in [0, 1]$, the latter is the *real deceptive function* f_{rdp} [1, 2],

$$f_{rdp}(\mathbf{x}) = \sum_{i=0}^{m-1} f_{trap}\left(x_{2i}, x_{2i+1}\right)$$

where $x_i \in [0, 1]$, m is the number of subproblems ($m = n/2$), and $f_{trap}(\cdot, \cdot)$ is the two-dimensional trap (Figure 2),

$$f_{trap}\left(x_j, x_{j+1}\right) = \begin{cases} 1 & \text{if } x_j, x_{i+1} \geq 0.8 \\ 0.8 - \sqrt{\frac{x_j^2 + x_{j+1}^2}{2}} & \text{otherwise} \end{cases} \tag{3}$$

6.1 Design of Experiments

The analysis was performed by applying the typical bisection procedure used in [28, 21] to determine the smallest population size and the smallest number of evaluations which guarantee the convergence to the optimum. The procedure is illustrated in Algorithm 3. Given a problem with n variables, the minimum population size *PopMin*, and the maximum population size *PopMax*, the following bisection procedure is run *NoBis* times. Initially, the lower and upper bounds for the population size, *PopLow* and *PopUp*, are set to *PopMin* and *PopMax* respectively; then the population size *PopSize* is randomly initialized between *PopLow* and *PopUp*. The real-coded ECGA is applied *NoRuns* times with a population size of *PopSize* individuals and the percentage of of individuals which converged in each run is measured. If more than the 99% of the population consists of optimal individuals the run is successful and the variable *NoSuccess* is incremented. When the *NoRuns* runs are completed, the number of converged runs, *noSuccess*, is checked: if at most one run did not converged, it is assumed that the real-coded ECGA can solve the problem with *PopSize* individuals and therefore the upper bound for the population size is decreased (line 20); otherwise, the lower bound for the population size is increased (line 22); then, the new population size is computed and the real-coded ECGA is run again *NoRuns* times. This process stops when the interval between the lower and upper bound for the population size is small enough (line 11). At the end of all the *NoBis* bisection runs, we determine the smallest population size for which the optimum is reached as,

$$BestPop = \frac{\sum_{i=1}^{NoBis} PopSize\,[i]}{NoBis}.$$

Note that, this value can be reached only if it is contained in the interval [*PopMin*, *PopMax*]. In the experiments performed for this work, we run ten bisections, *NoBis* was 10, and for each value of *PopSize* we run the real-coded ECGA 30 times.

Algorithm 3 Pseudo-code for the bisection procedure.

```
1: procedure BISECTION(numBisection)
2:     var NoBis;                                          ▷ number of bisections
3:     var NoRuns;                                         ▷ number of test runs for each interval
4:     var PopSize;                                        ▷ current population size
5:     var PopMin,PopMax;                                  ▷ min and max population size
6:     var PopLow, PopUp;                                  ▷ lower and upper bounds for binary search
7:     for NoBis times do
8:         PopLow := PopMin;
9:         PopUp := PopMax;
10:        PopSize := random(PopLow,PopUp);                ▷ Random population
                                                           ▷ size between PopLow and PopUp
                                                           ▷ Bisection continues while interval is large enough
11:        while ( (PopMax - PopMin) > 0.1×PopMin) do
12:            NoSuccess = 0;                              ▷ number of successful runs
13:            for NoRuns times do
14:                var pci;                                ▷ percetage of optimal individuals in the population
15:                pci = rECGA(PopSize);                   ▷ run RECGA
16:                if pci≥0.99 then                        ▷ 99% of individuals are optimal, success! NoSuccess :=
                       NoSuccess + 1;
17:                end if
18:            end for
19:            if NoSuccess≥NoRuns-1 then                  ▷ at most one failed
20:                PopUp := PopSize                        ▷ population bound is decreased
21:            else
22:                PopLow := PopSize                       ▷ population bound is increased
23:            end if
24:            PopSize := (PopUp + PopLow)/2;
25:        end while
26:        print PopSize, the convergence time, and the number of evaluations
27:    end for
28: end procedure
```

7 Experimental Results

We performed two sets of experiments to analyze how our rECGA scales up in a problem which does not require any linkage and in a problem which requires linkage, but also to study how the type of clustering algorithm applied during model selection and model fitting influences the scalability.

7.1 Experiments with the Sphere Function

At first, we applied our real-coded ECGA to the sphere function which does not require any linkage learning and therefore it can be solved by a simple greedy search. The optimum is in the axes origin and, for bisection purposes, we considered that an individual of size n is converged when all the n variables are comprised between -10^{-5} and 10^{-5}. We applied our rECGA with the same settings used for the real-coded BOA by Ahn in [1]: during the model selection the k-means is applied with one cluster ($k_s = 1$) so that the population is modeled using only one joint normal distribution; during the model fitting, the RLA algorithm is applied with an upper bound of 100 clusters, i.e., $k_f = 100$. Figure 3 reports (a) the number of evaluations and (b) the population size as functions of the problem size for our rECGA when the probabilistic model is enabled (solid dots) and when the probabilistic model is disabled (empty dots), i.e., when all the variables are considered independent. When we fit the data in Figure 3a with a polynomial we find that when the probabilistic model is used the number of fitness evaluations (Figure 3a) grows sub-quadratically as $O(n^{1.75})$ while, when all the variables are considered independent (similar to what done by UMDA [19]), the number of function evaluations grows almost linearly, more precisely as $O(n^{1.1})$. The population size (Figure 3b) required grows sub-quadratically, as $O(n^{1.4})$, when the probabilistic model is enabled (solid dots), while when all the variables are modeled independently the population size grows with the square root of n (empty dots).

Model Selection. As in [1], the first step of model selection in our rECGA involves the clustering of the population aimed at identifying k subproblems that are then modeled using k joint distributions based on the same underlying structure. In the experiments reported in [1], the number of clusters is usually one therefore only one distribution is used to model the entire population. However, a higher number of clusters (i.e., a higher number of distributions) should improve the model selection, accordingly, we performed another set of experiments aimed at studying how the clustering algorithm used in the model selection step influences the scalability of our algorithm. For this purpose, we applied our rECGA to the sphere function when the model selection is performed using k-means with 1, 5, and 10 clusters (i.e., $k_s \in \{1, 5, 10\}$) or using the adaptive clustering (RLA) which in rBOA is employed for model fitting [1]. Figure 4a compares the number of evaluations as a function of the problem size for the different versions of our rECGA. When more distributions are used to model the initial population, when k_s increases, the number of evaluations decreases, from $O(n^{1.75})$ to $O(n^{0.5})$.The problem is very simple and the best improvement is obtained with just five clusters ($k_s = 5$). Population size follows a similar behavior (Figure 4b).

Model Fitting. The number of clusters used during model fitting influences the accuracy of the approximation used to sample the target distribution. A higher number of clusters means that the distribution is approximated using more joint distributions and therefore it should be more accurate. However, if the population is small or the number of distributions is excessive the obtained approximation might be misleading. In rBOA [1], the model fitting step applies the adaptive RLA clustering algorithm to find the number of joint distributions which can provide an accurate approximation. In the last set of experiments with the sphere function, we investigated how the choice of the clustering algorithm used for model fitting influences the scalability of our real-coded ECGA. Figure 5a compares the number of evaluations for different versions of our rECGA in which model selection is based on a single cluster (k_s=1) and the model fitting is performed using a k-means with 1, 5, and 10 clusters ($k_f \in \{1, 5, 10\}$) or the same adaptive clustering (RLA) used in rBOA [1]. The sphere function is simple since all the variables can converge to same value independently, therefore we should expect no major advantage for using more clusters for model fitting. The results in Figure 5 confirm this in that the best scalability is obtained when only one

cluster is used both for model selection and model fitting. As we should expect in such a simple problem, when the number of clusters used for model fitting increases, the number of evaluations increases. Noticeably, an adaptive clustering like RLA seems to provide a good trade-off between the different k-mean settings, although it tends to grow with an order that is slightly higher than the one provided by simple k-mean.

Selection and Fitting using Adaptive Clustering. Finally, we applied a version of our rECGA in which adaptive clustering, namely the RLA algorithm used in rBOA [1] only for model fitting, is applied both during model selection and model fitting. Figure 6 compares three versions of our rECGA: one without the probabilistic model (empty dots), one with the same clustering strategy used in rBOA [1], that is, model selection is based on one cluster, model fitting is based on the adaptive RLA clustering, while the last one is fully adaptive in that RLA is used both during model selection and model fitting. As can be noted, the fully adaptive version requires slightly more evaluations and slightly larger populations than the version with no probabilistic modeling. However, the two versions scale similarly in that the number of evaluations grows as $O(n^{1.1})$ in both cases.

7.2 Experiments with the Real Deceptive Function

We repeated the same set of experiments using the real deceptive function (RDP) [1, 2] which requires linkage. Figure 7 compares (i) our real-coded ECGA with the same clustering strategies used in the real-coded BOA in [1], and (ii) a version of the same algorithm without probabilistic model building. As it should be expected, without linkage learning only problems of limited size can be solved while, when the probabilistic model is used, the number of function evaluations and the population size both grow sub-quadratically, respectively as $O(n^{1.7})$ (Figure 7a) and $O(n^{1.4})$ (Figure 7b). In Figure 8, we report the percentage of converged building block for an RDP with 50 variables. When we compare our results on the RDP function with the ones available for rBOA in [1], we note that our real-coded ECGA requires slightly larger populations but it appears to require fewer generations to converge, while overall, the two methods scale up similarly [3]. Figure 9 compares three versions of our rECGA in which the model selection is based on a k-mean with 1, 5, and 10 clusters ($k_s \in \{1, 5, 10\}$). In this case a higher number of clusters during model selection does not improve the overall performance both in terms of number evaluations nor in terms of population size. The building blocks are rather small in that they involve only two variables, therefore the influence of the initial clustering appears to be limited. Figure 10 compares three versions of our rECGA in which the model fitting is based on a k-mean with 1, 20, and 40 clusters ($k_f \in \{1, 20, 40\}$) with the version using the clustering strategy of rBOA. One cluster for model fitting is insufficient to reach good scalability and around 20 clusters are required to reach a performance similar to the one obtained with the usual clustering strategy. As in the previous case, the further increase of the number of clusters involve has no effect on the scalability. Finally, Figure 11 compares the version of our rECGA without the probabilistic model, with the probabilistic model based on the clustering strategy used in rBOA [1], and with fully adaptive clustering. Again, the choice of the clustering for model selection and model fitting does not show a relevant effect which basically explains the choice of Ahn in [1] where the same clustering strategy is always employed.

7.3 Discussion

The scalability analysis we performed shows that our real-coded ECGA scales up well on the two simple problems we considered. The analysis also shows that the choice of the clustering algorithm

used for model selection and model fitting may have some effect but it is most likely to be problem dependent and needs further investigations. Interestingly, a fully adaptive strategy, which applies adaptive clustering both for model selection and fitting, seems to provide a good trade-off and it actually scales up as the best configuration in both problems. Accordingly, it would also be interesting, and it will be a matter of future work, to investigate a fully adaptive version of Ahn's real-coded BOA [1].

8 Summary

In this paper, we have presented a real-coded estimation distribution algorithm inspired to ECGA and to the real-coded Bayesian Optimization Algorithm. Our algorithm, like ECGA [13], partitions the real-valued variables into clusters, that are manipulated as independent variables, and works on probability distributions represented by marginal product models (MPMs). However, instead of mapping the real values into discrete symbols, as other real-coded ECGA do [7, 10, 17], the proposed method models each cluster using a joint probability distribution and use MDL applied to continuous distribution to decide between candidate models. Similarly to rBOA [1, 2], our algorithm estimates the model parameters using several mixtures of models, one for each cluster of variables identified during the model building step. Overall, our real-coded ECGA can be either viewed as an extension of ECGA for real domains based on mixtures of models or, alternatively, as a simplification of the real-coded BOA to the case of a probabilistic model based on marginal product models (MPMs). The results for the scalability analysis we performed show that the number of evaluations required by our algorithm, to solve additively separable problems, scales up sub-quadratically with the problem size.

9 Acknowledgements

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center and under grant ISS-02-09199 at the National Center for Supercomputing Applications, UIUC. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

[1] C. W. Ahn. *Theory, Design, and Application of Efficient Genetic and Evolutionary Algorithms*. PhD thesis, NWC Lab, Department of Information and Communications Gwangju Institute of Science and Technology (GIST), Feb. 2005.

[2] C. W. Ahn. *Advances in Evolutionary Algorithms: Theory, Design and Practice*. Studies in Computational Intelligence. Springer-Verlag, Mar. 2006.

[3] C. W. Ahn and R. Ramakrishna. On the scalability of real-coded bayesian optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 2008. (in press).

[4] C. W. Ahn, R. S. Ramakrishna, and D. E. Goldberg. Real-coded bayesian optimization algorithm: Bringing the strength of boa into the continuous world. In K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. K. Burke, P. J. Darwen, D. Dasgupta, D. Floreano, J. A. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, and A. M. Tyrrell, editors, *GECCO (1)*, volume 3102 of *Lecture Notes in Computer Science*, pages 840–851. Springer, 2004.

[5] P. A. Bosman. *Design and application of iterated density-estimation evolutionary algorithms*. PhD thesis, Utrecht University, TB Utrecht, The Netherlands, 2003.

[6] C.-H. Chen and Y.-P. Chen. Real-coded ecga for economic dispatch. In H. Lipson, editor, *GECCO*, pages 1920–1927. ACM, 2007.

[7] C.-H. Chen, W.-N. Liu, and Y.-P. Chen. Adaptive discretization for probabilistic model building genetic algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1103–1110, New York, NY, USA, 2006. ACM Press.

[8] C.-H. Chen, W.-N. Liu, and Y.-P. Chen. Adaptive discretization for probabilistic model building genetic algorithms. In M. Cattolico, editor, *GECCO*, pages 1103–1110. ACM, 2006.

[9] L. de la Ossa, K. Sastry, and F. G. Lobo. χ-ary extended compact genetic algorithm in C++. IlliGAL Report No. 2006013, University of Illinois at Urbana-Champaign, Urbana, IL, March 2006.

[10] L. Fossati, P. L. Lanzi, K. Sastry, D. E. Goldberg, and O. Gomez. A simple real-coded extended compact genetic algorithm. In *Proceedings of the 2007 Congress on Evolutionary Computation (CEC2007)*, Singapore, Sept. 2007. IEEE.

[11] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Urbana, IL, January 1999.

[12] G. R. Harik. Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, 1995. (Also IlliGAL Report No. 94002).

[13] G. R. Harik, F. G. Lobo, and K. Sastry. Linkage learning via probabilistic modeling in the ECGA. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, chapter 3, pages 39–61. Springer, Berlin, 2006. (Also IlliGAL Report No. 99010).

[14] J. Hartigan. *Clustering algorithms*. John Wiley & Sons, New York, 1975.

[15] T. Higuchi, S. Tsutsui, and M. Yamamura. Theoretical analysis of simplex crossover for real-coded genetic algorithms. In *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 365–374, London, UK, 2000. Springer-Verlag.

[16] P. Larrañaga and J. A. Lozano, editors. *Estimation of distribution algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

[17] M. Li, D. E. Goldberg, K. Sastry, and T.-L. Yu. Real-coded ecga for solving decomposable real-valued optimization problems. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2194–2201, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.

[18] J. A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature*, 4:178–187, 1996.

[20] M. Pelikan. *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithm*. Springer Verlag, Berlin, 2005.

[21] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm (Toward a New Generation of Evolutionary Algorithms)*. Springer Berlin / Heidelberg, 2005.

[22] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Linkage learning, estimation distribution, and Bayesian networks. *Evolutionary Computation*, 8(3):314–341, 2000. (Also IlliGAL Report No. 98013).

[23] M. Pelikan, D. E. Goldberg, and S. Tsutsui. Combining the strengths of bayesian optimization algorithm and adaptive evolution strategies. pages 512–519, 2002.

[24] M. Pelikan, D. E. Goldberg, and S. Tsutsui. Getting the best of both worlds: discrete and continuous genetic and evolutionary algorithms in concert. *Inf. Sci.*, 156(3-4):147–171, 2003.

[25] M. Pelikan, D. E. Goldberg, and S. Tsutsui. Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert. *Inf. Sci.*, 156(3-4):147–171, 2003.

[26] M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[27] J. J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[28] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2001. (Also IlliGAL Report No. 2002004).

[29] K. Sastry and D. E. Goldberg. Probabilistic model building and competent genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practise*, chapter 13, pages 205–220. Kluwer, 2003.

[30] S. Tsutsui and D. E. Goldberg. Simplex crossover and linkage identification: Single-stage evolution vs. multi-stage evolution. 2002.

[31] S. Tsutsui, M. Pelikan, and D. E. Goldberg. Evolutionary algorithm using marginal histogram in continuous domain. In *Optimization by Building and Using Probabilistic Models (OBUPM) 2001*, pages 230–233, San Francisco, California, USA, 7 2001.

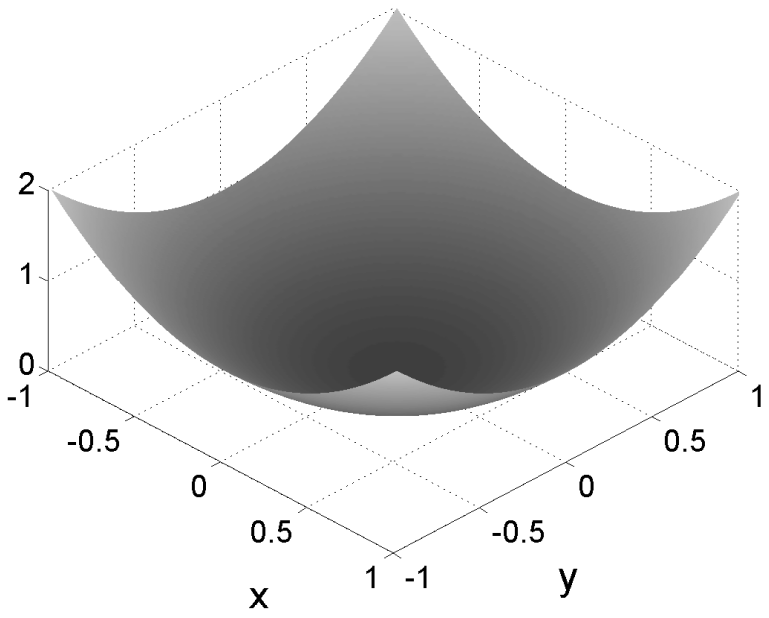


Figure 1: The sphere function of two variables.

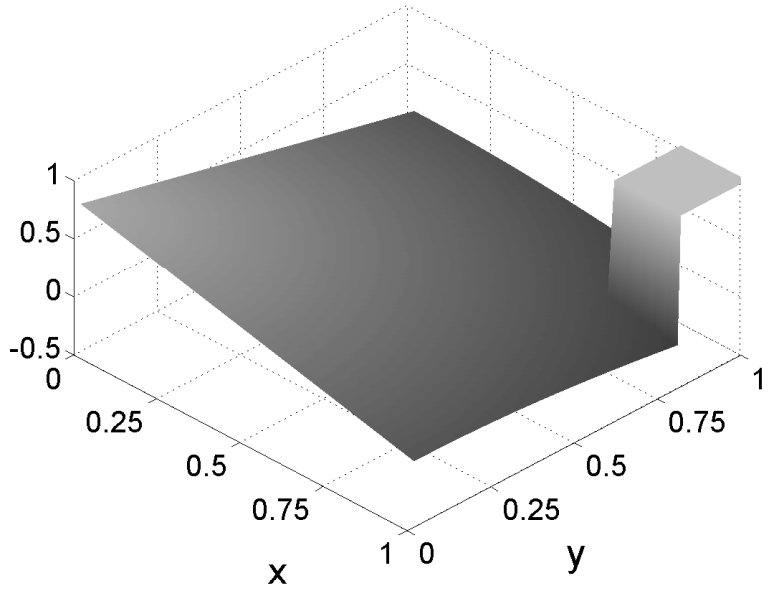
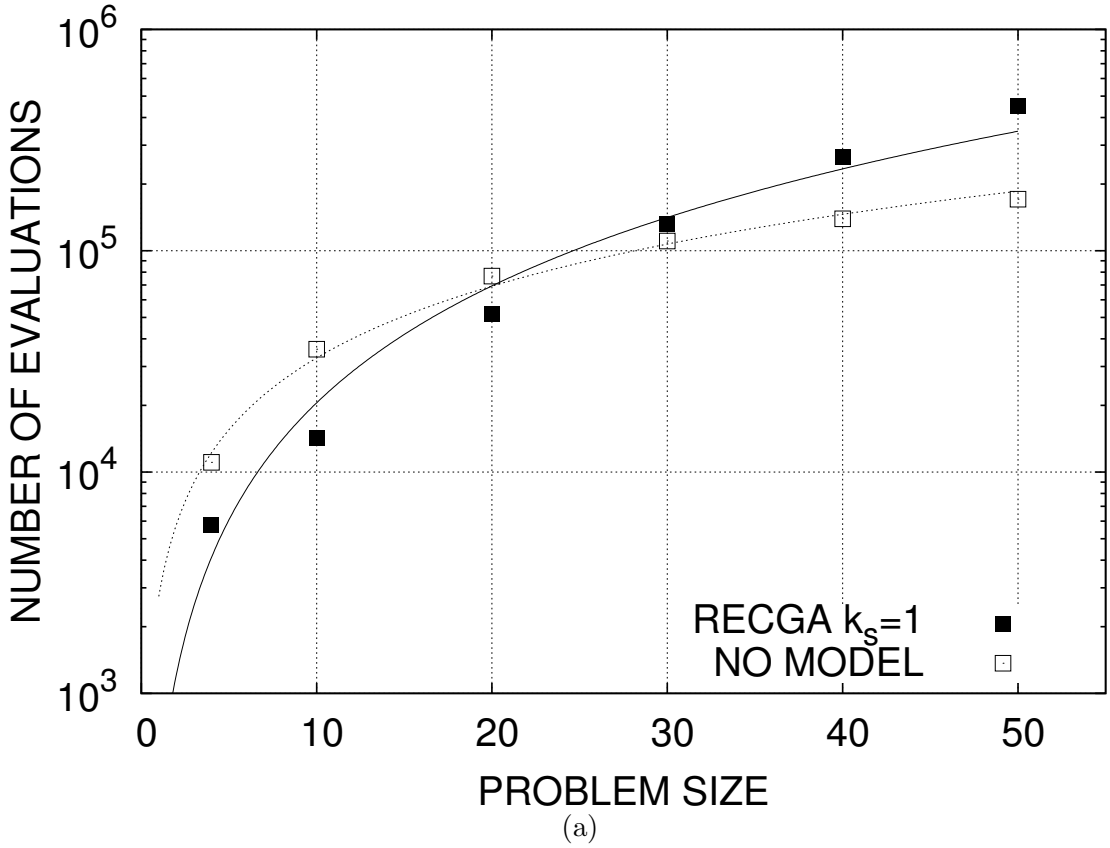
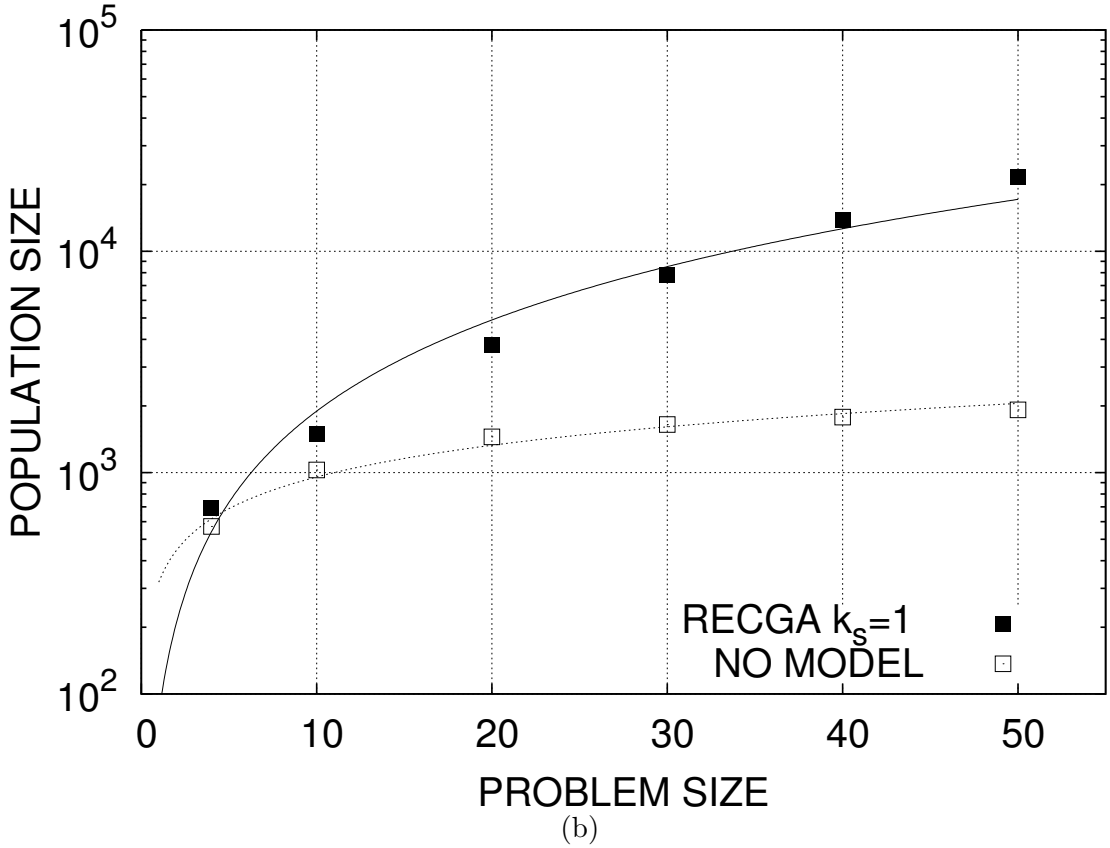


Figure 2: Bidimensional basis function for the real trap deceptive function.

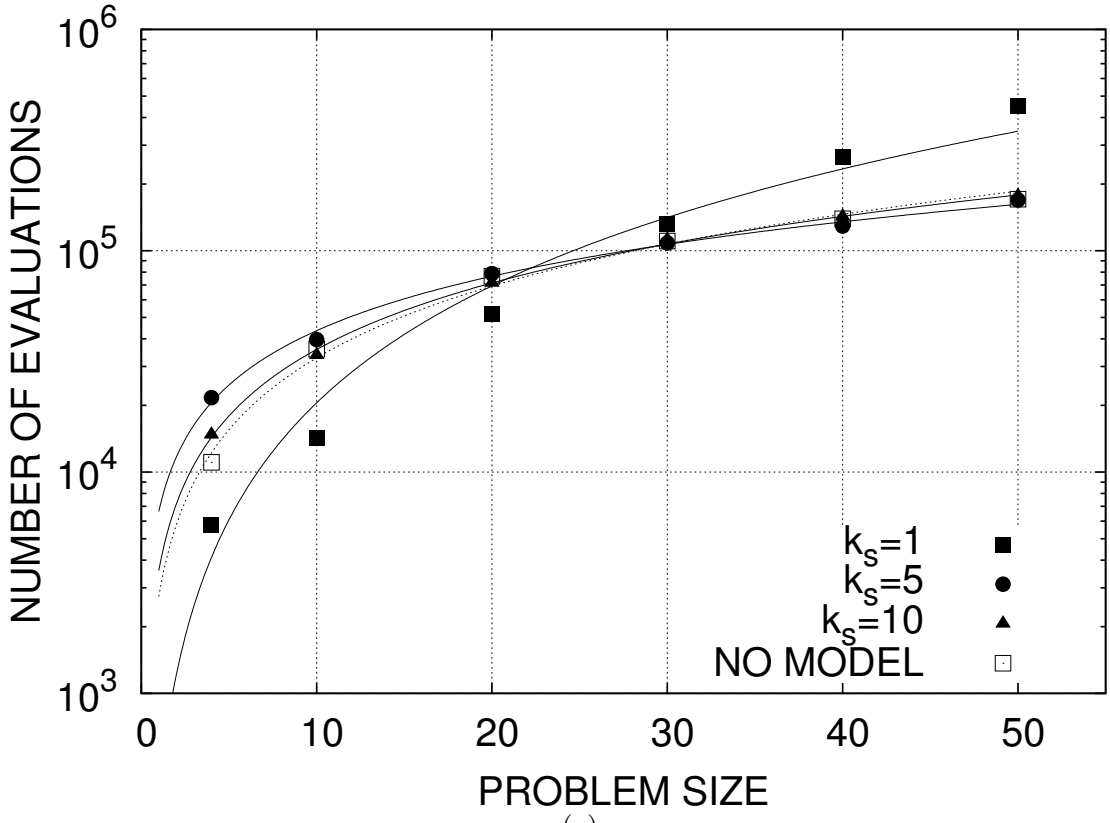


(a)

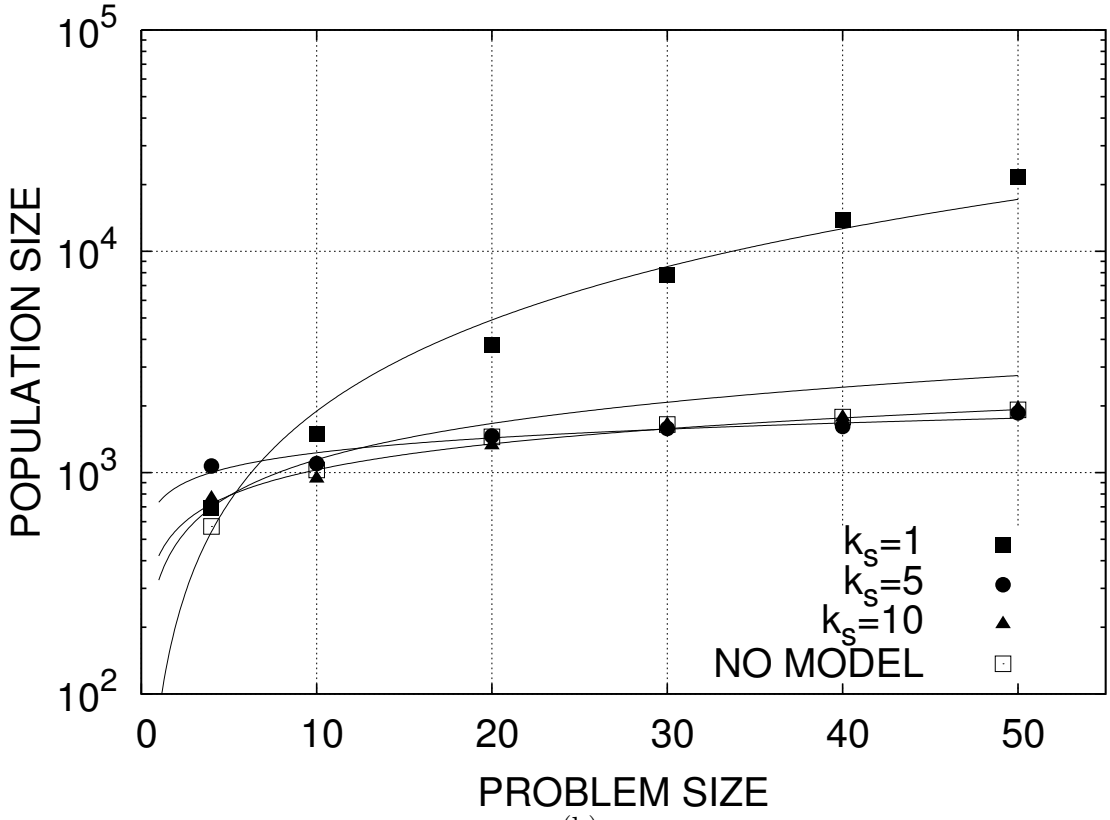


(b)

Figure 3: Real-Coded ECGA applied to the sphere function: (a) number of evaluations and (b) population size as a function of the problem size. Clustering parameters are set as in rBOA [1]: model selection applies k-means with one cluster (k_s=1) and model fitting applies RLA clustering with at most 100 clusters.

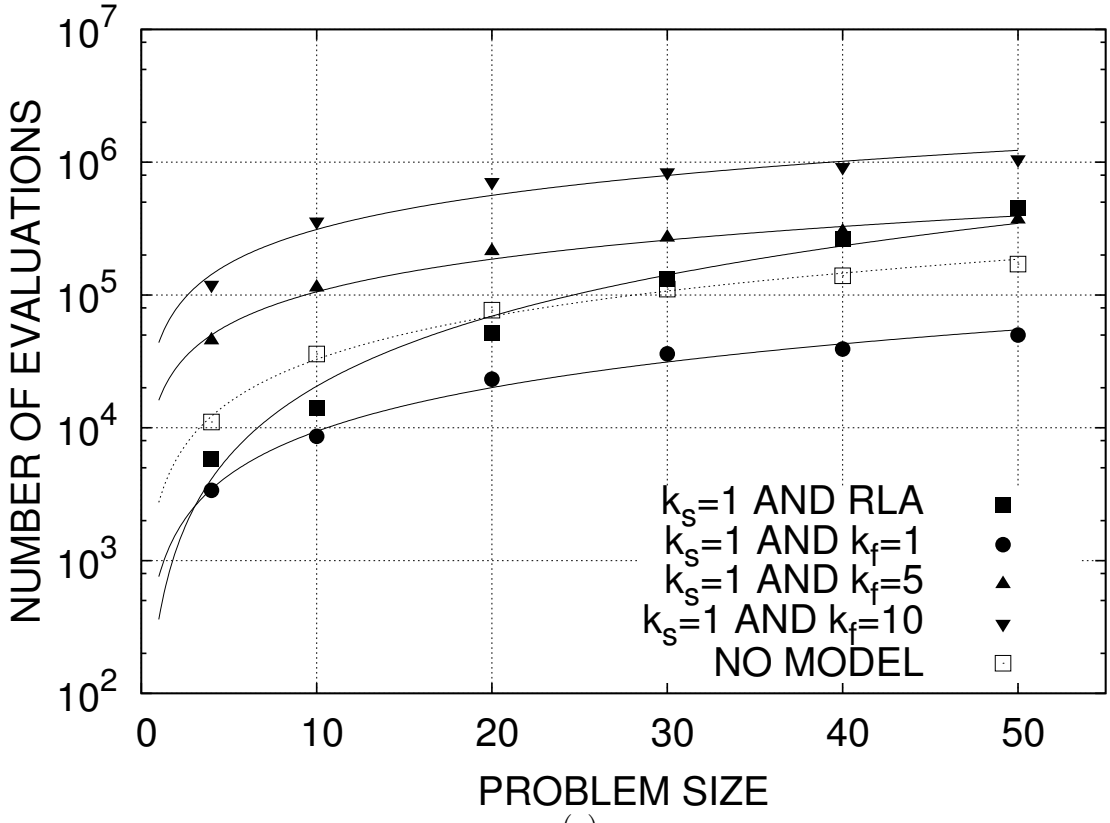


(a)

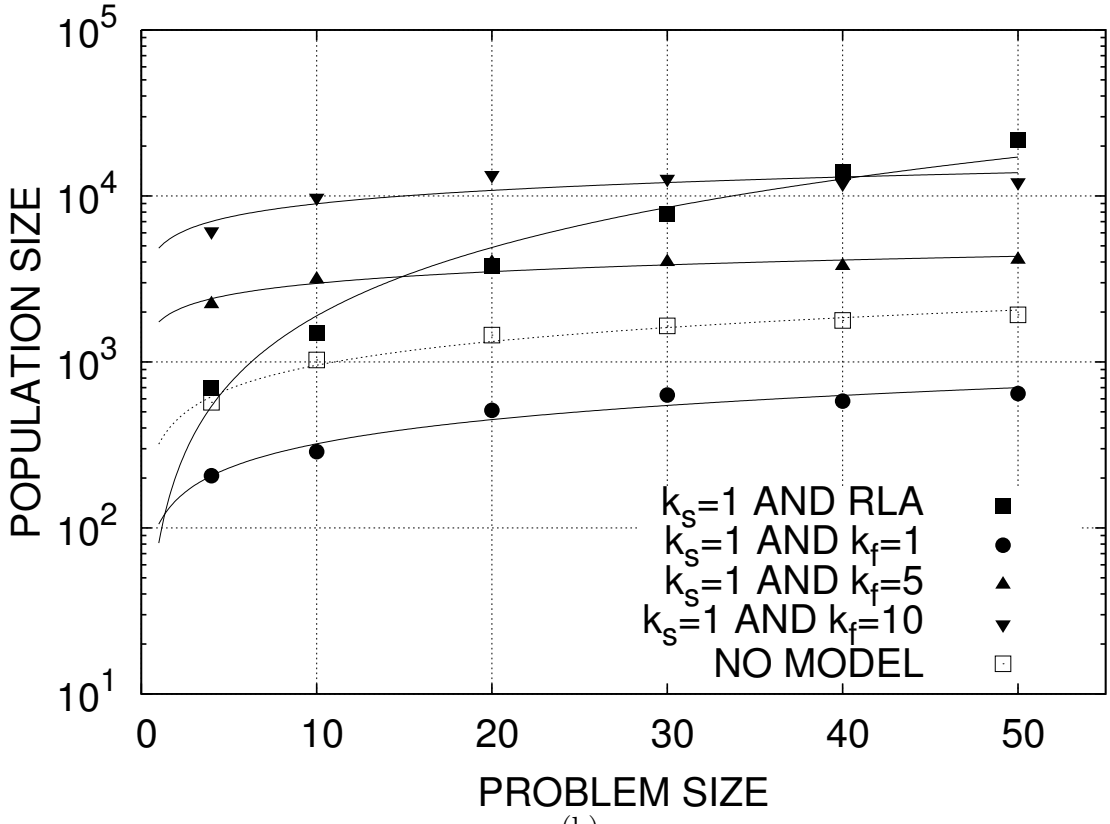


(b)

Figure 4: Our real-coded ECGA applied to the sphere function using different clustering algorithms for model selection: (a) number of evaluations and (b) population size as a function of the problem size.

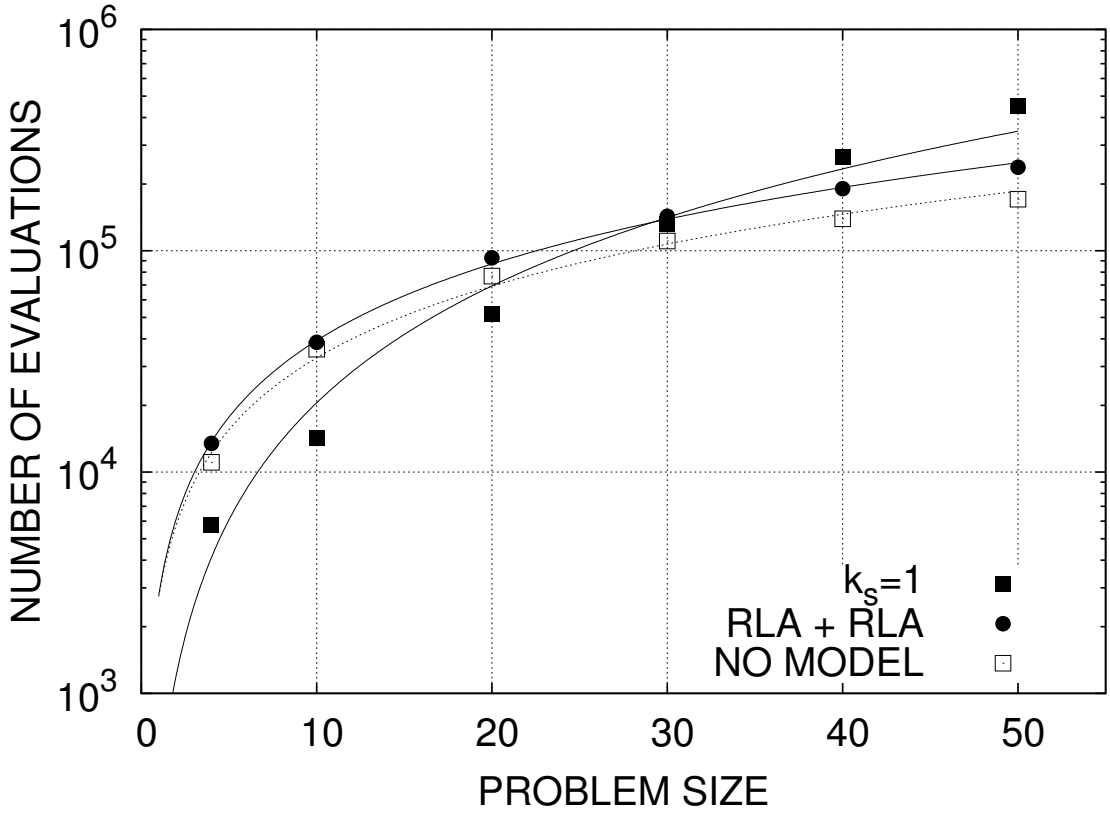


(a)

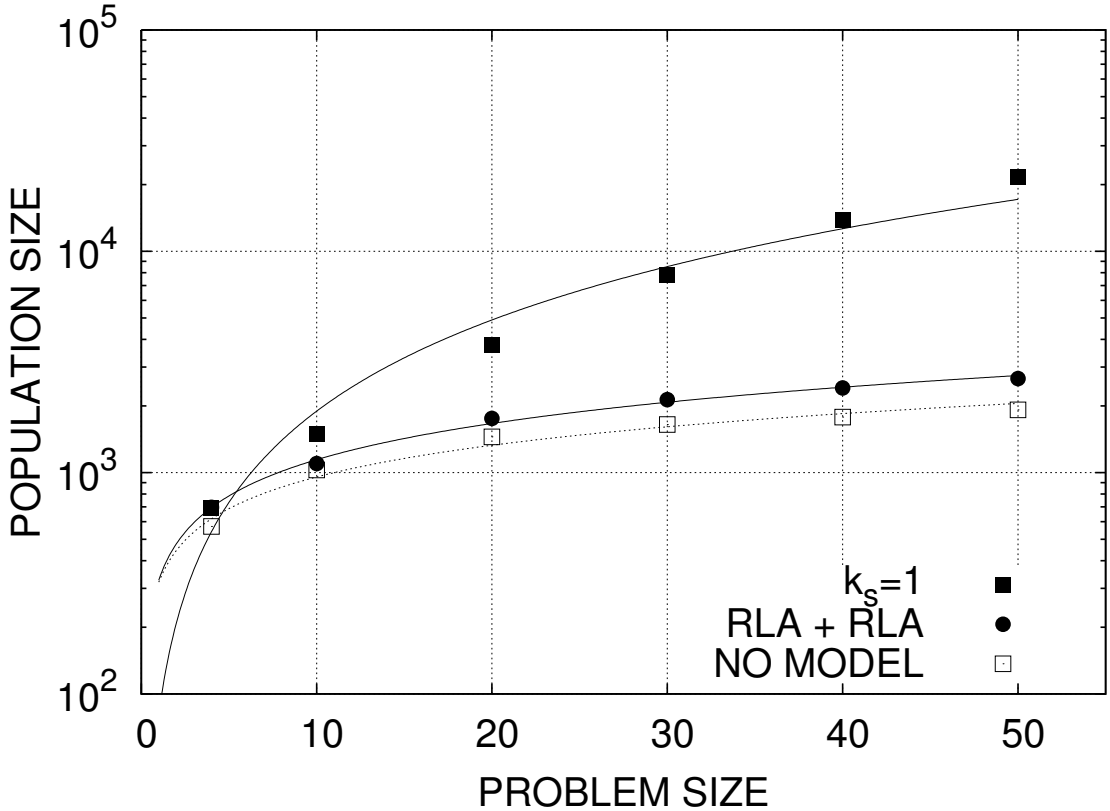


(b)

Figure 5: Our real-coded ECGA applied to the sphere function using different clustering algorithms for model fitting: (a) number of evaluations and (b) population size as a function of the problem size.

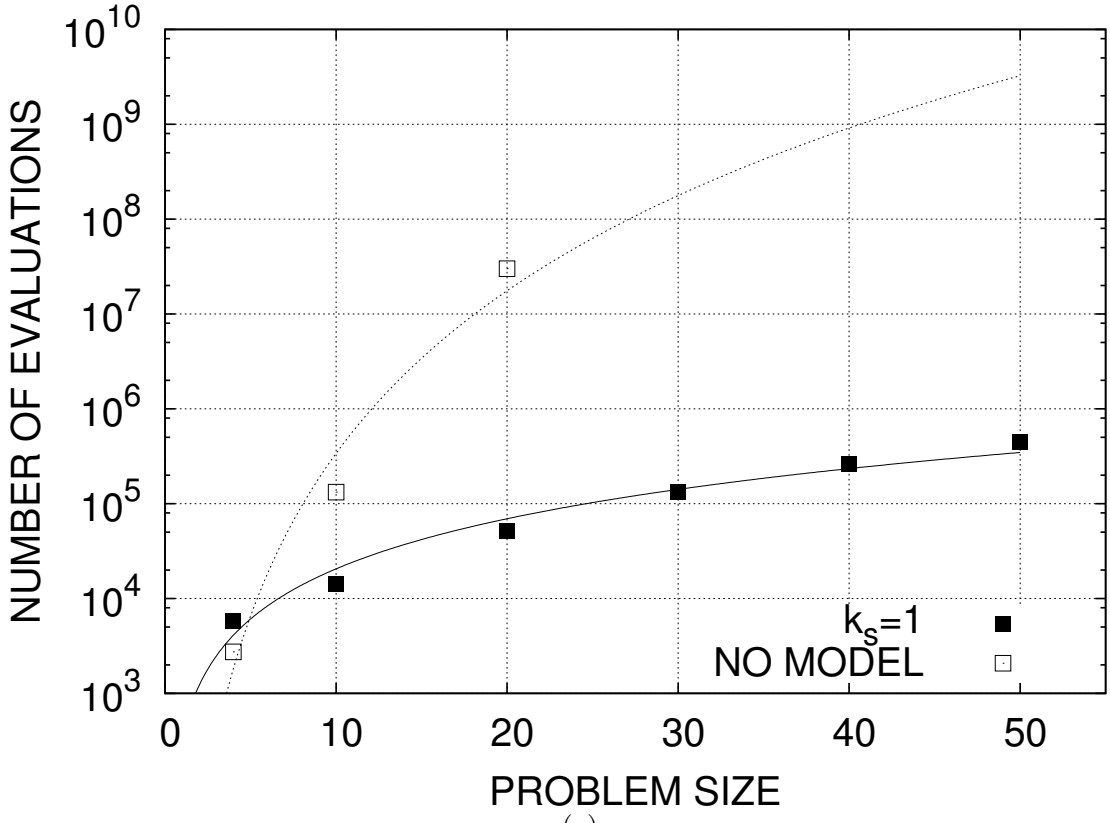


(a)

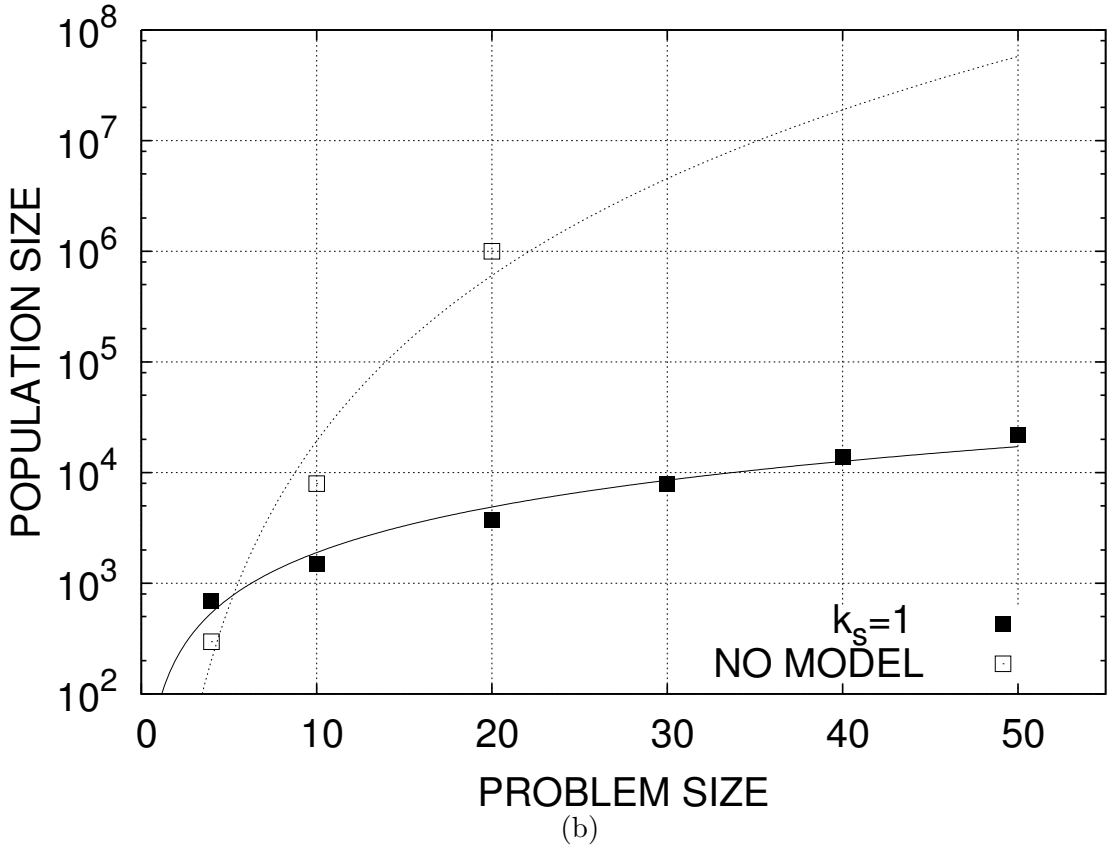


(b)

Figure 6: Our real-coded ECGA applied to the sphere function using the model selection and fitting strategies of rBOA [1] and a fully adaptive strategy: (a) number of evaluations and (b) population size as a function of the problem size.



(a)



(b)

Figure 7: Our real-coded ECGA applied to the real deceptive function using the model selection and fitting strategies of rBOA [1] (solid dots) and no probabilistic model (empty dots): (a) number of evaluations and (b) population size.

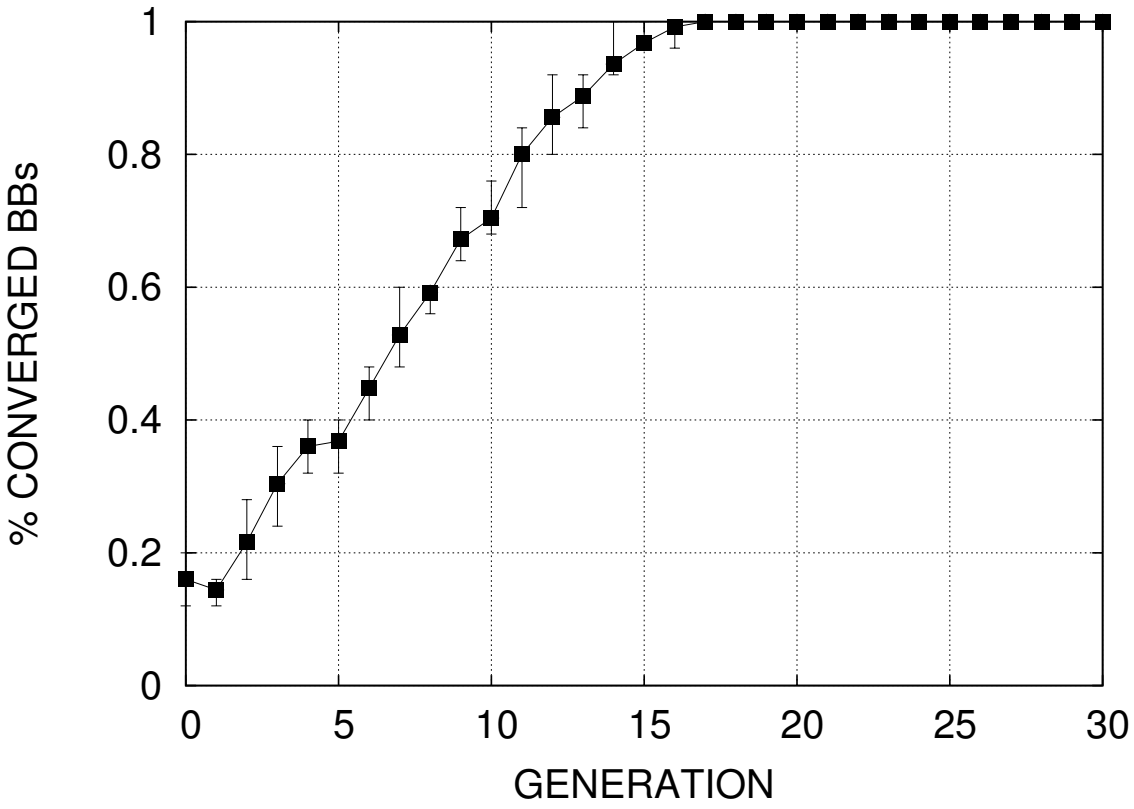
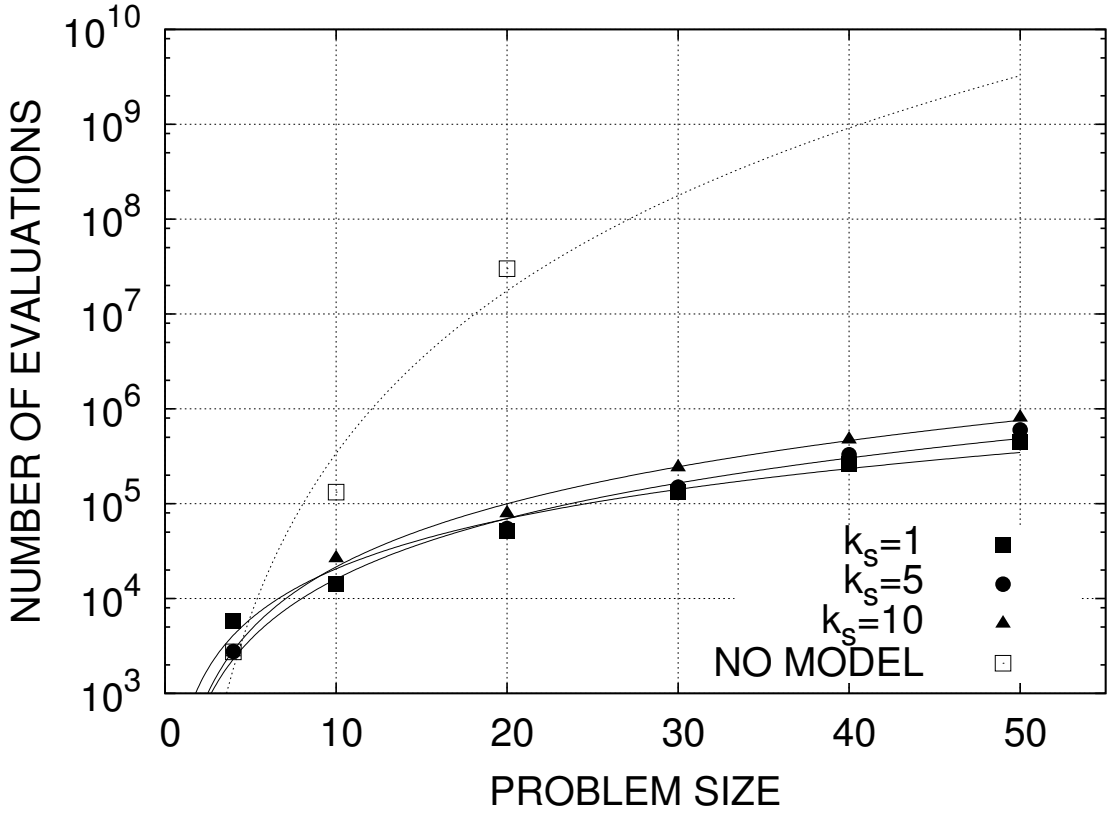
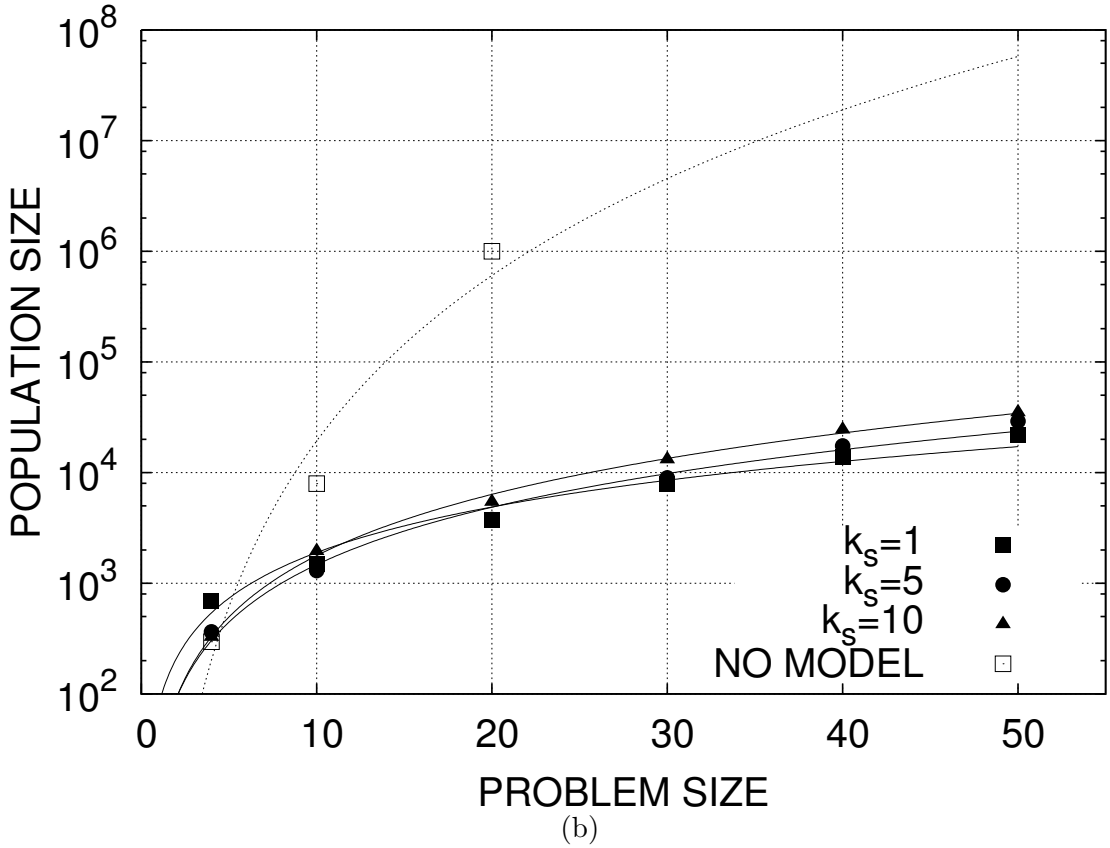


Figure 8: Our real-coded ECGA applied to the real deceptive function with 50 variables: percentage of converged building blocks. The vertical bars identify the minimum and the maximum percentages over 10 runs.

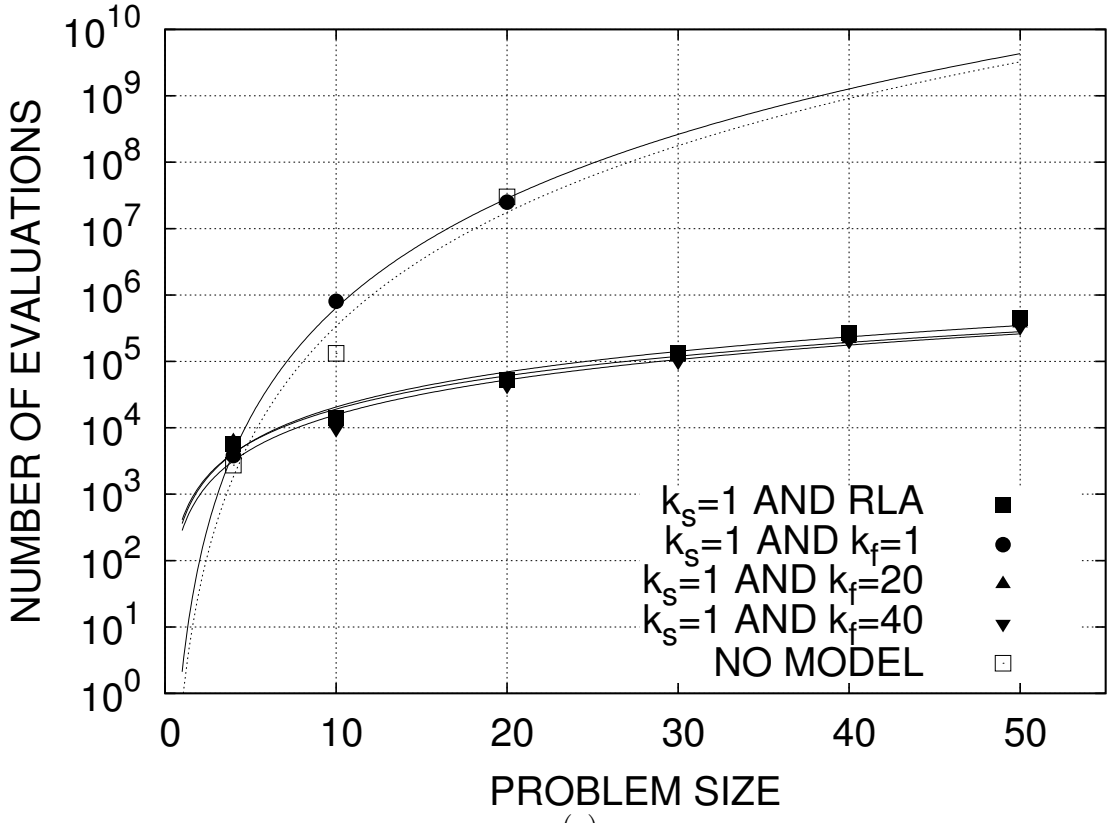


(a)

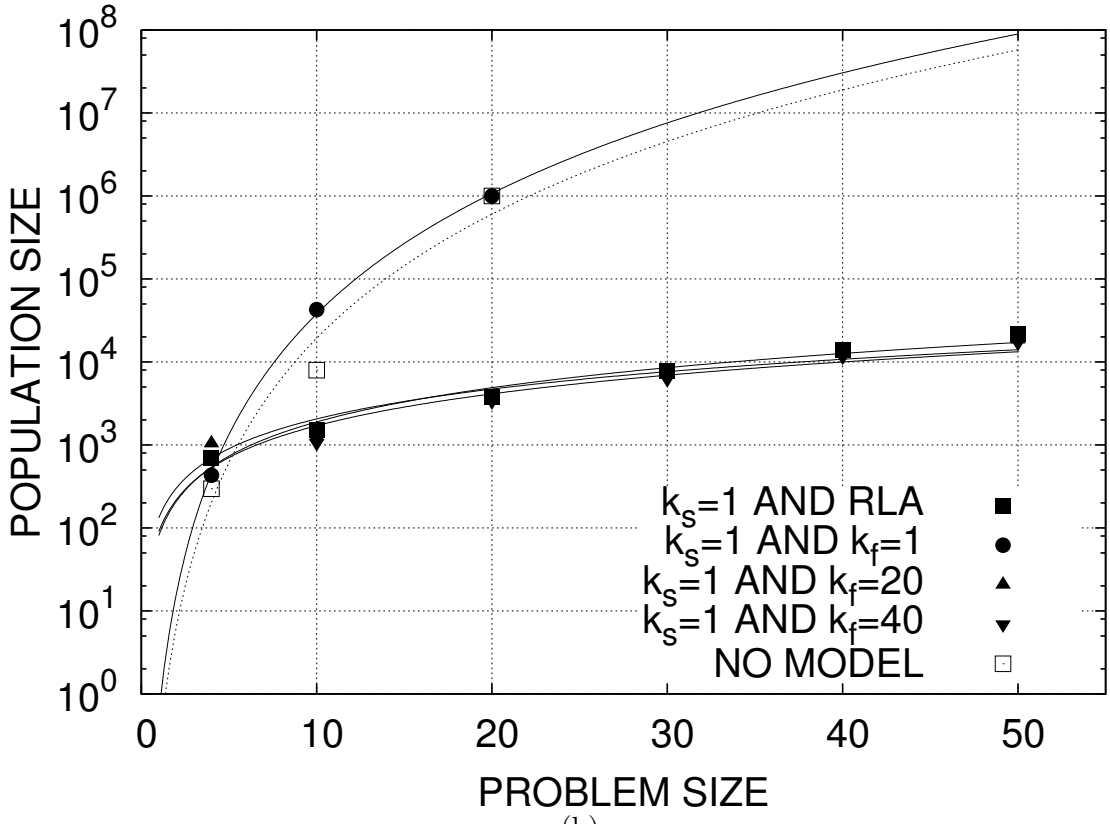


(b)

Figure 9: Our real-coded ECGA applied to the real deceptive function using different clustering algorithms for model selection: (a) number of evaluations and (b) population size.

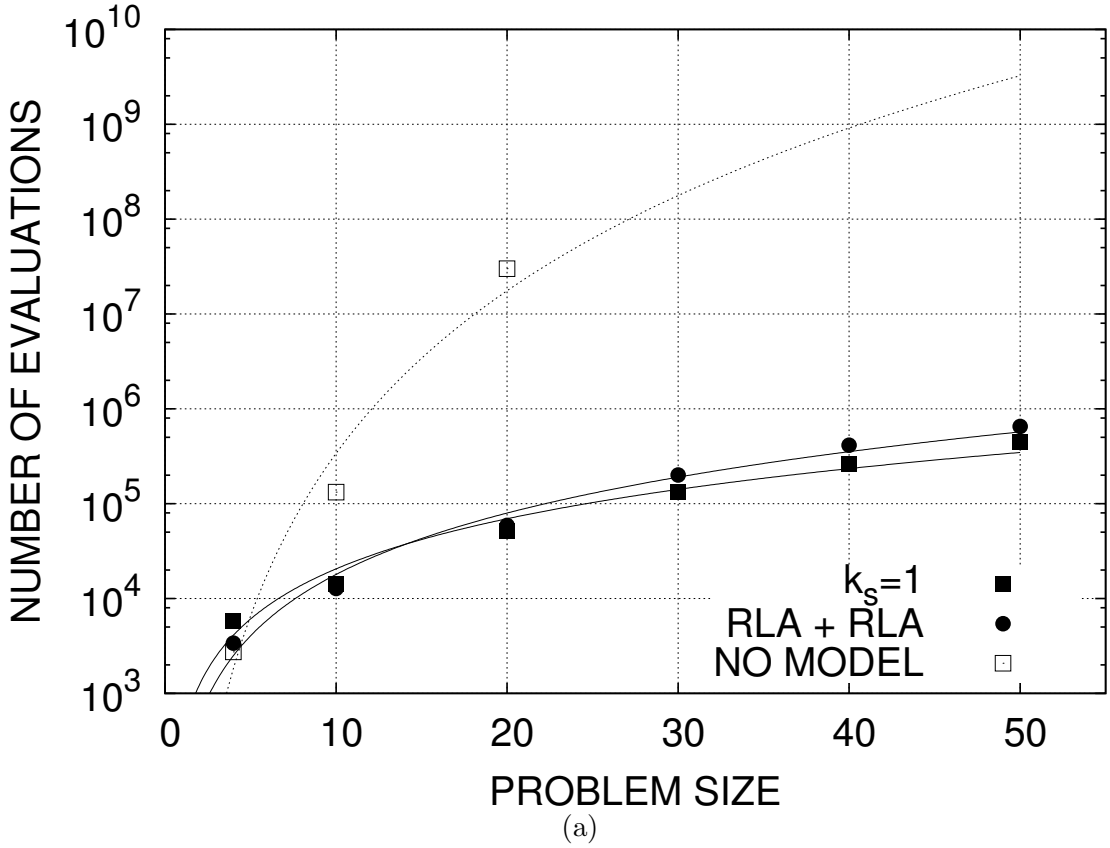


(a)

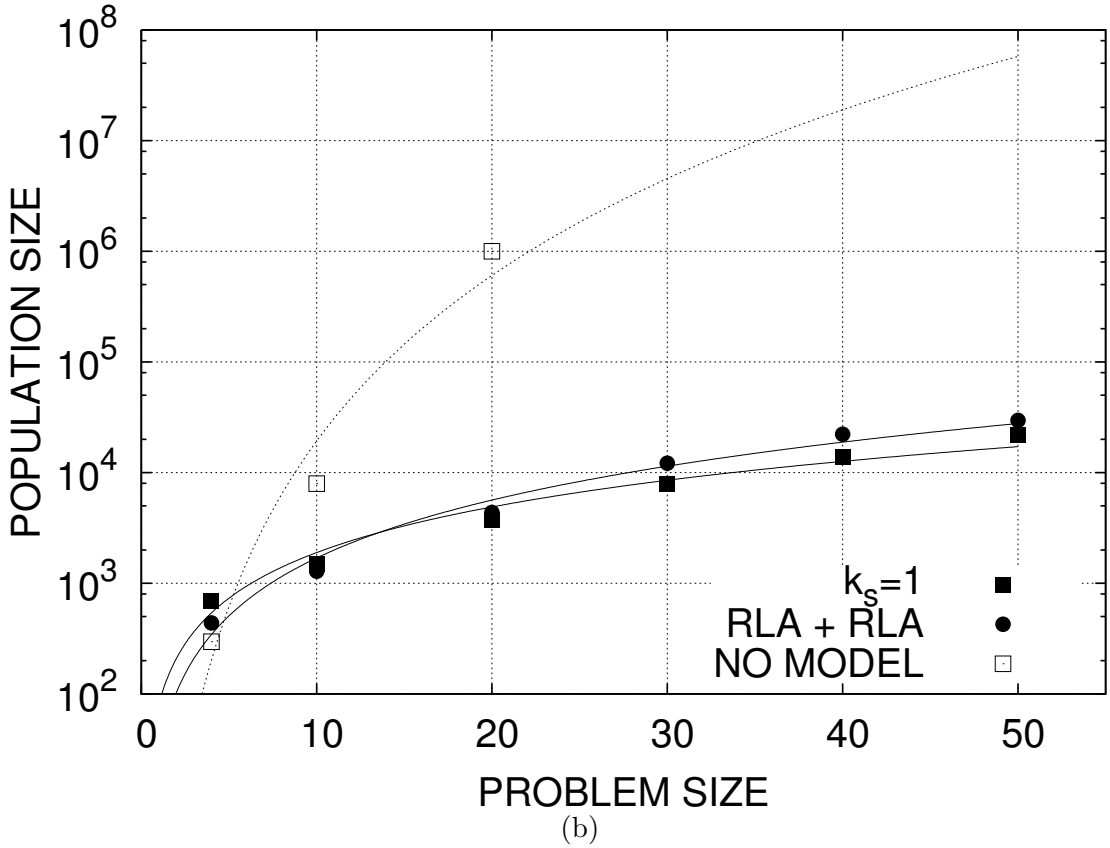


(b)

Figure 10: Our real-coded ECGA applied to the real deceptive function using a model selection based on one cluster and a model fitting based on 1, 20, and 40 clusters, or the adaptive RLA clustering: (a) number of evaluations and (b) population size.



(a)



(b)

Figure 11: Our real-coded ECGA applied to the real deceptive function using the model selection and fitting strategies of rBOA [1] and a fully adaptive strategy: (a) number of evaluations and (b) population size.