Technical Report: UPM-FI/DIA/2011-1

Regularized Model Learning in Estimation of Distribution Algorithms for Continuous Optimization Problems

Hossein Karshenas Roberto Santana
Concha Bielza Pedro Larrañaga

Computational Intelligence Group, School of Computer Science
Technical University of Madrid, Madrid, Spain

January 2011

Abstract

The effect of applying regularization techniques on the model learning of estimation of distribution algorithms is studied in this paper. Three different algorithms using different regularization techniques are proposed and their modeling accuracy and algorithm performance are studied on a number of test functions. The results are compared against other state of the art algorithms and the influence of some algorithm parameters on the behavior of the proposed algorithms are investigated. The performance results of applying the algorithm on a high dimensional problem are also presented. The obtained results suggest that regularization can help to derive more accurate models and is a promising technique for further research in order to obtain better optimization results.

Keywords: Estimation of Distribution Algorithms, Regularization, Continuous Optimization Problems

1 Introduction

First introduced in [21], Estimation of Distribution Algorithms (EDAs) are a class of evolutionary algorithms that take advantage of the probabilistic models to guide the search further in the space of candidate solutions for a problem [17, 19, 22, 24]. In a typical EDA, a probabilistic model is learnt from the set of selected solutions. The information encoded in the model are then used to generate new solutions for the next generation of the algorithm.

The probabilistic model and its related learning and sampling algorithms, which are the EDAs main point of difference from other evolutionary algorithms, play an important role in efficiency and performance of these algorithms. In the context of continuous optimization, Gaussian (normal) distributions are the models of choice and many Gaussian-EDAs have been proposed in the literature (e.g. see [16, 18, 27, 32]).

However these algorithms usually use the maximum likelihood estimation for computing model parameters, which may result in poor performance on difficult high dimensional problems. As a remedy for this shortcoming, several improvements have been proposed in the literature like mixture learning [3], variance scaling [4, 26, 40] or eigenvalue resetting [6, 36]. In this paper regularization techniques [10, 14, 20, 41] are used for obtaining a better estimation of the probabilistic models in EDAs.

The basic idea of regularization is very simple: a specific penalization term is applied on the values of the model parameters, to *regularize* the model estimation process. In most of the cases regularization is applied to regression formulas of the model parameters which makes it especially suitable for continuous problems. It can be viewed as a way to construct more accurate models with higher generalization ability. An appealing property of the regularization process is that while it will guide the estimation to more accurate models, it also favors sparser ones. This property is important when trying to learn models from high dimensional data.

Recently Yang et al. [37] have also used of regularization techniques for learning the Bayesian network structure in Bayesian Optimization Algorithm (BOA) in discrete domains. Using regularized regression, they obtain a reduced parent set for each variable before starting the search in the space of directed acyclic graphs, thus reducing the structure search space to a great extent. The presented results show that the algorithm is able to obtain competitive results in comparison to original BOA.

Although continuous optimization problems have many similarities with their discrete counterparts, the model learning task for obtaining better optimization performance turns out to be significantly harder in this domain. One of the objectives of this paper is to show how different regularization techniques can be applied for model learning in continuous EDAs. Therefore, different regularization methods are described and the advantages and limitations of using these techniques for optimization performance of the proposed EDAs are studied.

The rest of this paper is organized as follows. In the next section some of the major regularization techniques are briefly reviewed. A number of methods applicable for learning a regularized probabilistic model are also discussed. In Section 3 the proposed EDAs based on regularization techniques are introduced and discussed. The experiments and their results are presented in Section 4. Section 5 concludes the paper and gives some future lines of research. The mathematical notations used in this paper are explained in the appendix.

2 A Review on Regularization

A typical regression model is used for estimating a response (output) variable Y given a set of n predictor (input) variables $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ in continuous domains. One of the very first regression models which is still frequently used due to its simplicity is linear regression where the estimation of the response variable, $\hat{Y}$, is given by

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

and it can be used to predict the response value for new samples. The typical way of learning this linear regression model (estimating the regression parameters) given a set of N observations and their corresponding responses $(\mathbf{x}^i, y_i)$, is to perform a least squared

error estimation, trying to minimize the sum of squared errors between the predicted value and the actual value

$$\arg\min_{(\beta_0,\boldsymbol{\beta})}\left(\sum_{i=1}^{N}(y_i-\hat{y}_i)^2\right)=\arg\min_{(\beta_0,\boldsymbol{\beta})}\left(\sum_{i=1}^{N}\left(y_i-\left(\beta_0+\mathbf{x}^i\boldsymbol{\beta}^T\right)\right)^2\right)$$

However this kind of estimation has often shown poor results due to low prediction accuracy and bad interpretability [34]. The decomposition of the expected prediction error shows the large variance of prediction accuracy for this type of model fitting despite its low bias. The interpretability of the models is especially important when dealing with a large number of variables where only a strong subset of dependences are required to appear in the model.

To tackle the shortcomings of least squared model fitting, some methods have been proposed. One of the techniques is called *ridge* regression [15]. In this technique a penalization term, imposed on the values of the regression parameters, is added to the least squared error estimation

$$\arg\min_{(\beta_0,\boldsymbol{\beta})}\left(\sum_{i=1}^{N}\left(y_i-\left(\beta_0+\mathbf{x}^i\boldsymbol{\beta}^T\right)\right)^2+\lambda\sum_{j=1}^{n}\beta_j^2\right)$$

This penalization term (which is also called an L2 regularization term) causes the regression parameters to shrink towards zero, although they do not become exactly zero. The parameter $\lambda \geq 0$ controls the amount of shrinkage. From a prediction accuracy point of view the use of penalization term in the least squared error, will reduce the estimation variance at the cost of introducing some bias.

LASSO (Least Absolute Shrinkage and Selection Operator) [34] is another penalization technique for regularizing the estimation process which is based on the absolute values of the regression parameters

$$\arg\min_{(\beta_0,\boldsymbol{\beta})}\left(\sum_{i=1}^{N}\left(y_i-\left(\beta_0+\mathbf{x}^i\boldsymbol{\beta}^T\right)\right)^2+\lambda\sum_{j=1}^{n}|\beta_j|\right)$$

This type of penalization term which is also called L1 regularization, has the appealing property of setting some of the regression parameters exactly equal to zero, which will result to a behavior similar to that of variable selection and hence the name [14].

Another type of penalization term is formed by combining the previous two terms which is called the *elastic net* penalty

$$\arg\min_{(\beta_0,\boldsymbol{\beta})}\left(\sum_{i=1}^{N}\left(y_i-\left(\beta_0+\mathbf{x}^i\boldsymbol{\beta}^T\right)\right)^2+\lambda\left(\sum_{j=1}^{n}\left(\alpha\beta_j^2+(1-\alpha)|\beta_j|\right)\right)\right)$$

Here α controls the combination of the two regularization terms. This kind of regularization is especially useful in situations that a high dimensional problem is given but there is a small number of observations to be used for model estimation, usually called a "large n, small N" or "$n \gg N$" problem [41]. For such problems, the estimation algorithm regularized with elastic net penalty, is not bounded to select a maximum of N variables, which is the case for LASSO regularization.

Briefly speaking, ridge regularization shrinks the parameters corresponding to correlated predictor variables, towards each other (and at the same time towards zero). On the other hand, LASSO regularization will tend to shrink all but one of the correlated predictor variables to zero [10]. Thus combining these two types of techniques in the elastic net regularization, we can use the advantages of both, causing a grouping effect where correlated variables will be in or out of the model together.

One of the important issues that must be addressed when applying regularization technique for regression, is how to select the correct value of the regularization term (represented by λ). Larger values of this parameter (λ) will impose a higher penalization on the model parameters and therefore more parameters will be shrunk towards zero whereas smaller values (those close to zero) allow larger values for the regression coefficients.

Although for some regularization techniques, there are theoretical propositions for selecting the value of this parameter or the bounds for an effective value, in general one should use a trial and error strategy to select the best value. A more general approach is to obtain the solutions for the regularized regression relations with all possible λ values. These solutions which are obtained along the path of varying λ values form the *regularization path*, profile or degrees of freedom of the regularization technique. Having the entire path of solutions, methods such as k-fold cross-validation or C_p type statistic can be applied for selecting one of the models.

In all of the aforementioned penalization terms the intercept parameter (β_0) is left out since it can be estimated by the average absolute value of responses (y_i), and without loss of generality we can assume it is equal to zero. However, computing the optimal model parameters (the regression coefficients) is not the same for these regularized estimation techniques. In the case of ridge regression the optimal value of the parameters can be computed using a closed formula, but LASSO penalization brings a nonlinearity to the solutions that cannot be computed easily (the LASSO regression is not directly differentiable). Nevertheless, there are numerical optimization algorithms that can compute the solutions along the whole regularization path very efficiently and with the same computational cost as that of ridge regression [14].

Yet another efficient regularization technique is Least Angle Regression (LAR) [7]. This method can be considered as an adaptation and improvement of the Forward Stagewise Regression [13] which adds the variables one by one to the model. LAR uses the (absolute) correlation between the current residual (the difference between actual and predicted values of the response variable) and the parameter variables. From a geometrical point of view, this corresponds to the angle between parameter variables and the response variable.

Starting from all coefficients set to zero, the algorithm selects the variable that has the greatest correlation with the residual and adds it to the active set (the set of selected variables). At each step the coefficients related to the variables in the active set are moved towards their joint least squared values. This will cause the absolute correlation between these variables and the residual to decrease. As soon as another (not included in the active set) variable's correlation with residual equals that of the variables in the active set, it will be added to active set and its related coefficient will start to evolve along other active set coefficients. This process is repeated until all of the parameter variables are added to the model.

The LAR technique has a speed advantage over LASSO and forward stage-wise re-

gression since the variables are only added to the model and never removed. Hence it will reach the full least squared solution, using all variables, in n steps. For LASSO, and to a greater extent for stage-wise regressions, variables can leave the model, and possibly re-enter later, multiple times. Thus they may take more than n steps to reach the full model (if $N > n$).

The basic LAR algorithm with its angle bisection geometry, can also be used, with a simple modification, to efficiently fit LASSO and stage-wise models. Therefore an algorithmic implementation framework based on LAR, called LARS, is developed that includes fast implementation of many regularization techniques and other useful tools. The entire sequence of LARS steps with $n < N$ variables requires $O(n^3 + Nn^2)$ computations, which is the cost of a least squared fit on n variables. For the case where $n \gg N$ the computation cost is of the order $O(N^3)$, since the algorithm will terminate at the saturated least squared fit after N variables are added to the model.

2.1 Regularized Learning of Probabilistic Graphical Models

Apart from the methods discussed above, regularization techniques have also been applied to other, not necessarily linear models. In this subsection some of the methods proposed in the literature for learning a regularized probabilistic model, which will be used for model learning purposes in this paper, are briefly reviewed.

Mainly, these techniques consider estimating the structure of the inverse covariance matrix (precision matrix) of a Gaussian distribution. Given a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ represents the vector of mean values and Σ is the symmetric positive definite covariance matrix, then the inverse covariance matrix $\Theta = \Sigma^{-1}$ contains the partial covariances between the variables. That is, any zero entry in the inverse covariance matrix means the corresponding two variables are conditionally independent given all other variables and therefore are not connected with a direct link in the structure.

Usually in many application domains, the unbiased empirical estimation of the covariance matrix (based on its maximum likelihood estimation – MLE), denoted with S, is used for the analysis and processing of the data

$$\mathrm{S} = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}^i - \bar{\mathbf{x}})^T (\mathbf{x}^i - \bar{\mathbf{x}})$$

where $\bar{\mathbf{x}}$ is the MLE of the vector of mean values computed from data (of size N). However, this type of estimation may not necessarily result in an accurate and reliable computation of covariance matrix:

- The estimated covariance matrix may not be positive definite.
- The estimated covariance matrix may not be well-conditioned or full-ranked (and therefore is not invertible).

This can be explained as the covariance matrix obtained with MLE can be the best estimator in terms of actual fit to the data, but in terms of generalization, especially in high dimensional context, it is not able to estimate the population covariance (Σ).

Therefore obtaining a good estimation of the covariance matrix (or its inverse) remains a challenging problem and many researches have been done on this topic. The following techniques are some of these methods.

2.1.1 Regularized Neighborhood Selection [20]

The aim of this method is to discover the conditional independence restrictions of the inverse covariance matrix from a set of observations. It computes the set of possible neighbors of each variable using regularized regression on other variables. The neighborhood set of a variable X_j, is the smallest subset of variables that when given, X_j is conditionally independent of all remaining variables.

In this method the model is computed using n distinct regularized regression relations, which are the linear regression of each variable on the rest of variables

$$\arg\min_{(\beta_0,\boldsymbol{\beta})} \left(\sum_{i=1}^{N} (x_j^i - (\beta_0 + \sum_{\forall k,k\neq j} \beta_k x_k^i))^2 + \lambda \sum_{\forall k,k\neq j} |\beta_k| \right)$$

The zero coefficients will correspond to zero patterns in the inverse covariance matrix of the data. Meinshausen and Bühlmann [20] used this technique to select the set of potential neighbors of each variable for estimating the structure of the inverse covariance matrix. Since the dependence between two variables is computed in two different regression formulas, they used two different strategies to decide about the existence of such dependency in the structure. An AND strategy requires both of the variables to be present in the estimated neighborhood set of the other, while an OR strategy will insert the dependency as soon as one of the variables appears in the neighborhood set of the other. Based on a number of assumptions, they have shown that this method can asymptotically obtain consistent sparse models for high-dimensional data.

Similar technique has also been used by others to obtain a regularized estimation of a directed graphical model. Schmidt et al. [31] used this technique to obtain the Markov blanket of each variable in order to reduce the search space of directed acyclic graphs (DAGs). They also applied this variable selection technique to restrict the space of possible variable orderings in an order-based search for learning the structure of a Bayesian network. Vidaurre et al. [35] combined this regularized variable selection with a greedy equivalence search method to search for the correct structure of a Bayesian network in the space of equivalence classes.

2.1.2 Covariance Shrinkage [29]

In shrinkage estimation two models are considered: one is an unrestricted high-dimensional model U which should be shrunk and the other is a restricted lower-dimensional model T with fewer parameters that is considered as the shrinkage target. Since there are many parameters in the high-dimensional model that need to be fitted, its estimation will show a high variance. On the other hand, the estimation of fewer number of parameters in the low-dimensional model has a lower variance but at the price of considerable bias from the true model.

The linear shrinkage approach suggests that instead of choosing anyone of these two extreme models, they can be combined with a weighted average to give a better estimation

$$U^\star = \lambda T + (1-\lambda)U$$

where $\lambda \in [0,1]$ denotes the shrinkage intensity. This combination offers a systematic way to obtain a regularized estimate of the model $U^\star$ that outperforms each of individual estimators U and T, both in terms of accuracy and statistical efficiency [30]. Another important advantage of this method is that the shrinkage intensity parameter (λ) is not an open parameter and can be determined with an analytic approach.

The shrinkage estimation approach can be applied for obtaining a better estimation of covariance matrix. The unrestricted high-dimensional model in this case will be the empirical covariance matrix (S). Different target matrices are possible but in order to have a compromise between simpler targets and those with many parameters, it is assumed that the target restricted model (T) will be a diagonal covariance matrix where all off-diagonal elements are set to zero. This type of target matrix will only cause the covariances to be shrunk and will leave the variances intact

$$w_{ij} = \begin{cases} s_{ii} & \text{if } i = j \\ q_{ij}\sqrt{s_{ii}s_{jj}} & \text{if } i \neq j \end{cases}$$

where W denotes the shrinkage estimation of covariance matrix. It can be seen that the estimation of covariances is reformulated using the shrinkage estimation of the correlation matrix, Q, computed as follows

$$q_{ij} = \begin{cases} 1 & \text{if } i = j \\ (1-\lambda)r_{ij} & \text{if } i \neq j \end{cases}$$

using the empirical estimation of correlation matrix R.

The optimal value of the shrinkage intensity λ always exists and can be uniquely computed when its required parameters are estimated consistently. However consistency is an asymptotic property and for practical applications that deal with a finite population, the optimal value can be estimated by using the unbiased empirical estimates of the required parameters. For the shrinkage estimation of the correlation matrix, the estimated shrinkage intensity $\hat{\lambda}$ will be

$$\hat{\lambda} = \frac{\sum_{i \neq j} \widehat{Var}(r_{ij})}{\sum_{i \neq j} r_{ij}^2}$$

For finite sample sets, this estimation may lead to values out of $[0,1]$. To avoid over-shrinkage or negative shrinkage, the estimated intensity value is forced to be in this interval: $\max(0, \min(1, \hat{\lambda}))$.

In the above relation, $\widehat{Var}(r_{ij})$ is the unbiased empirical variance of the entries (r_{ij}) in the empirical correlation matrix. Let

$$z_{ij}^k = \left(\frac{x_i^k - \bar{x}_i}{s_{ii}}\right)\left(\frac{x_j^k - \bar{x}_j}{s_{jj}}\right)$$

and $\bar{z}_{ij} = \frac{1}{N}\sum_{k=1}^{N} z_{ij}^k$ be its mean. Then

$$\widehat{Var}(r_{ij}) = \frac{N}{(N-1)^3}\sum_{k=1}^{N}\left(z_{ij}^k - \bar{z}_{ij}\right)^2$$

Schäfer and Strimmer [29] compared the structure recovery ability of this method with that of Meinshausen and Bühlmann method [20] when using LASSO regularization on a number of synthesized covariance matrices. The results presented there suggest that the true positive rate of the models built with covariance shrinkage approach is considerably higher than those built with the neighborhood selection method which tends to insert a lot of spurious dependencies to the structure.

2.1.3 Graphical LASSO [9]

This method tries to maximize the penalized log-likelihood of Gaussian distribution model given a data set of size N

$$\max_{\Theta} \ \log \prod_{i=1}^{N} \mathcal{N}(\mathbf{x}^i; \bar{\mathbf{x}}, \Theta^{-1}) - \rho \left\|\Theta\right\|_1$$

which can be converted to the following optimization problem by setting $\lambda = \frac{2}{N}\rho$

$$\max_{\Theta} \ \log\det(\Theta) - \mathrm{trace}(\mathrm{S}\Theta) - \lambda \left\|\Theta\right\|_1$$

In this relation $det()$ computes the determinant of its argument, $trace()$ gives the sum of the elements on the main diagonal of matrix and $\|.\|_1$ is the sum of absolute values of the elements in the matrix.

The optimum parameters for this relation can be computed by obtaining its sub-gradient with respect to Θ and setting it to zero

$$\mathrm{W} - \mathrm{S} - \lambda\Gamma = 0$$

where W is the estimation of covariance matrix (Σ), and matrix Γ contains the element-wise sub-gradients of the absolute value of Θ

$$\gamma_{ij} = \begin{cases} sign(\theta_{ij}) & \text{if } \theta_{ij} \neq 0 \\ \in [-1, 1] & \text{if } \theta_{ij} = 0 \end{cases}$$

The above relation can be solved by optimizing over each row (or column) of W in a block coordinate wise fashion when partitioning the matrices as follows

$$\mathrm{A} = \begin{bmatrix} \mathrm{A}_{11} & \mathbf{a}_{12}^T \\ \mathbf{a}_{12} & a_{22} \end{bmatrix}$$

The problem will then be reduced to solving a LASSO regularized least squared problem

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \left\| \mathrm{W}_{11}^{\frac{1}{2}} \boldsymbol{\beta} - \mathrm{W}_{11}^{-\frac{1}{2}} \mathbf{s}_{12} \right\|_2^2 + \lambda ||\boldsymbol{\beta}||_1$$

and setting $\mathbf{w}_{12} = \boldsymbol{\beta}\mathrm{W}_{11}$. This procedure will be repeated by permuting the rows (or columns) of the matrices so that the target row (column) is always the last. At each stage it estimates the optimal coefficient vector $\hat{\boldsymbol{\beta}}$ and updates the estimation of the corresponding row and column of the covariance matrix W. An alternative view to the permutation of rows or columns of the matrix is shown in Figure 1. This process is repeated until the estimated parameters of covariance matrix W converge.

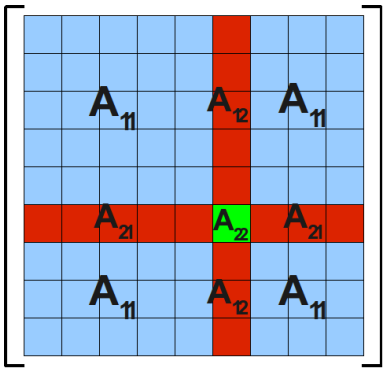


Figure 1: Partitioning the matrix for row (column) wise computation

Friedman et al. [9] used the path-wise coordinate descent algorithm [8] for estimating the optimal $\hat{\boldsymbol{\beta}}$. Let $\mathrm{V} = \mathrm{W}_{11}$ and $\mathbf{u} = \mathbf{s}_{12}$. This algorithm will cycle through all $n-1$ coefficients and update their estimation using the following update formula until convergence

$$\hat{\beta}_j \leftarrow \frac{SoftThreshold(u_j - \sum_{k \neq j} v_{kj}\hat{\beta}_k, \lambda)}{v_{jj}}$$

where $SoftThreshold(x, \tau)$ is the soft thresholding operator

$$SoftThreshold(x, \tau) = \begin{cases} x - \tau & \text{if } \tau < |x| \text{ and } x > 0 \\ x + \tau & \text{if } \tau < |x| \text{ and } x < 0 \\ 0 & \text{if } \tau \geq |x| \end{cases}$$

Instead of solving n separate regularized regression problems, in the graphical LASSO algorithm these problems are coupled together and solved using the same W. In fact the use of the same W allows sharing the information between problems. The neighborhood selection of Meinshausen and Bühlmann [20] can be seen as an approximation of graphical LASSO related to the case where $\mathrm{W}_{11} = \mathrm{S}_{11}$. Another feature of this algorithm is that allows a symmetric matrix of penalization parameters Λ to be used instead of a single parameter λ in order to penalize each element of the inverse covariance matrix differently.

3 Incorporating Regularized Learning into EDAs

Since the new solution generation of EDAs is solely dependent on the probabilistic model they learn, obtaining a better model can greatly affect their performance. Although the capacity of the chosen probabilistic model has a great impact on search success [1], but even for a specific model type, obtaining a more accurate approximation of the global structure of the solution space can help to generate better solutions. Therefore any attempt to improve the quality of the learnt models is of special importance.

Another major bottleneck of EDAs is their performance scalability as the problem size increases. In general the population size requirements of EDAs is more critical than other evolutionary algorithms as they try to capture problem regularities in a probabilistic model. Their performance and efficiency will rapidly degrade as problem size is increased, especially when they are applied to high dimensional problems. The situation can become

even worse for continuous domain problems, where EDAs should work with finite sample sizes to deal with infinite solution spaces. Therefore model learning techniques that can help EDAs to learn good models with smaller populations when applied to large problem sizes are strongly needed.

In this paper, the incorporation of three different regularization techniques to the model learning of EDAs is studied and their influence on obtaining more accurate models and better algorithm performance is examined. The resulting regularized EDA, which will be called R-EDA from here forth, can capture a sparser and more accurate set of dependencies between problem variables and use it to generate new solutions that are closer to those in the original population. Regularization also enables the algorithm to work with smaller population sizes to obtain models of almost the same quality. This will allow the algorithm to be applied to high dimensional problems.

3.1 Gaussian Distribution-based R-EDAs

Two of the regularization techniques considered for employing in the model learning of R-EDA are the covariance shrinkage method [29] and the graphical LASSO algorithm [9], and the resulting R-EDAs will be called CSR-EDA and GLR-EDA respectively. Both of these methods obtain a regularized estimation of the covariance matrix of a Gaussian distribution. Therefore a Gaussian distribution based EDA, namely the Estimation of Multivariate Normal distribution Algorithm (EMNA) [17], is used as the base EDA implementation for applying these regularization techniques.

EMNA assumes that the set of candidate solutions to the problem follow a joint Gaussian distribution, and therefore learns a multivariate normal distribution at each generation from a set of selected solutions. However this algorithm uses MLE to compute the covariance matrix of Gaussian distribution. Instead, the proposed R-EDAs use the regularized estimation of covariance matrix in this step. The MLE of covariance matrix (S) used by EMNA is passed as an input to these methods and the resulting regularized covariance matrix ($\hat{\Sigma}$) is returned (this matrix was shown by W in the description of these regularization techniques). Algorithm 1 shows the general outline of these R-EDAs, which only differ in the model learning step (line 6).

```
   Regularized Gaussian Estimation of Distribution Algorithm
 1 P_0 ← Generate initial population
 2 F_0 ← Evaluate Population(P_0)
 3 t ← 0
 4 while termination conditions are not satisfied do
 5     S ← Select a subset of promising solutions(P_t, F_t)
 6     M ← Learn regularized Gaussian distribution parameters(S)
 7     O ← Sample new solutions from Gaussian distribution(M)
 8     G ← Evaluate offspring(O)
 9     (P_{t+1}, F_{t+1}) ← Incorporate offspring(P_t, F_t, O, G)
10     t ← t + 1
11 end while
```

Algorithm 1: Basic Steps of CSR-EDA and GLR-EDA

In the sampling step, the resulting regularized estimation of Gaussian distribution

is used for generating new solutions. This is done by first decomposing the estimated covariance matrix $\hat{\Sigma} = \mathrm{A} \cdot \mathrm{A}^T$ (A is a triangular matrix). Then the following two steps are repeated to generate an offspring population:

1. A random vector $\mathbf{z}$ of normally distributed independent values is generated.
2. A new solution is obtained by applying the translation and rotation induced by the model to this vector: $\mathbf{y} = \bar{\mathbf{x}} + \mathbf{z} \cdot \mathrm{A}$ ($\bar{\mathbf{x}}$ is the vector of mean values computed in the model learning step).

Finally at the end of each generation an elitism approach is used to select the best solutions from the current population and the offspring population to be used as the next generation population.

3.2 Neighborhood Selection-based R-EDA

The third version of the algorithm uses a similar method to the LASSO regularized neighborhood selection technique [20] and will be called LR-EDA. First an auxiliary variable Z_j is computed from each variable X_j after normalizing all of samples $\mathbf{x}^i$ in the data set (to be in $[0, 1]$)

$$z_j^i = log(\frac{x_j^i}{1 - x_j^i})$$

A LASSO regularized linear logistic regression model is then fit for each auxiliary variable given the rest of variables ($\mathbf{X} \backslash X_j$), treating the variable values to be at the same time probability values.

The general outline of LR-EDA is very similar to that given in Algorithm 1 for the previous two R-EDA versions and it only differs in the way it learns and then samples the model. The other steps of the algorithm are identical. In the model learning step, instead of considering a Gaussian distribution as the probabilistic model, n separate regularized models are learnt for each variable. In fact, the model fitting of each variable returns the whole regularization path for different values of the regularization parameter (λ). From this path of solutions the one resulting in the minimum squared error is chosen.

In order to save some computational effort in the model learning step, especially when dealing with high dimensional problems, a sporadic model building scheme [25] is adopted for LR-EDA. In this method, instead of learning a new model for each variable at every generation of the algorithm, the models learnt in the previous generation are used in current generation when it is possible. Only those models that result in poor prediction accuracy according to the current selected population will be re-estimated. At every generation, the models of different variables are first sorted according to their prediction accuracy, and then only the lowest rc fraction of the models are re-estimated.

In the sampling step, a new solution is generated starting from a random value setting of the variables in unit interval $[0, 1]$. Then through a predefined number of sampling steps $nSteps$, the value of each variable is updated using its model approximation, applied to the current values of other variables, in a way that resembles the Gibbs sampling method. This process is repeated for generating each new solution. At the end of sampling step, the generated offspring is rescaled to be in the initial range of variables before normalization.

It is important to notice that the LASSO procedure used by LR-EDA allows the complexity of the model to be controlled by setting a constraint on the number of maximum

predictor variables to be included in the model. It is also possible to force specific variables to be included or excluded from the models, effectively allowing the incorporation of a priori knowledge about the problem domain. Similarly, it is possible to extract the problem structure by analyzing the coefficients of the regression models.

4 Experiments

As mentioned earlier, the implementation of GLR-EDA and CSR-EDA are based on the implementation of EMNA taken from MATEDA-2.0: The Matlab® EDA toolbox [28]. The graphical LASSO algorithm implementation in Matlab® provided by Mark Schmidt[1] is used for GLR-EDA. For CSR-EDA, the Matlab® implementation of covariance shrinkage method provided by Kevin Murphy[2] is used. The sampling of Gaussian distribution in GLR-EDA and CSR-EDA is done using the multivariate normal distribution (MND) randomizer of Matlab® Statistics toolbox which employs a *Cholesky* decomposition to decompose the covariance matrix.

To fit the model of each variable in LR-EDA, the Matlab® implementation of learning generalized linear models [10] provided by Hui Jiang[3] is used which applies the path-wise coordinate descent algorithm to obtain the entire regularization path. The re-estimation fraction (rc) of LR-EDA is set to 1/10 and number of sampling steps ($nSteps$) to 2.

All of the algorithms use the *truncation* selection algorithm for selecting the set of promising solutions and an *elitism* approach for replacing the offspring population in current population. Unless otherwise stated, the truncation threshold is set to 0.5 and the offspring population size is set to 0.5 of the main population size. The initial population is generated randomly within the range of problem variables. The termination condition is usually set to reach a maximum number of generations unless otherwise stated.

4.1 Benchmark Functions

The two dimensional Deceptive Function [23] which has a simple known problem structure is used to test the modeling accuracy and algorithm performance of all proposed R-EDAs. This function is formed by the aggregation of 2-dimensional trap functions that have a local optimum with a large basin of attraction and an isolated global optimum (Figure 2)

$$f_{2D-deceptive}(\mathbf{x}) = \sum_{i=1}^{n/2} f_{2D-trap}(x_{2i-1}, x_{2i})$$

where

$$f_{2D-trap}(x,y) = \begin{cases} 0.8 - \sqrt{\frac{x^2+y^2}{2}} & \text{if } \sqrt{\frac{x^2+y^2}{2}} \leq 0.8 \\ -4 + 5\sqrt{\frac{x^2+y^2}{2}} & \text{otherwise} \end{cases}$$

This function has an exponential number of local optima ($2^{n/2} - 1$) which a pose a significant challenge to any optimization algorithm. The domain of all variables is the

[1] http://www.cs.ubc.ca/∼schmidtm/Software/L1precision.zip

[2] http://www.uni-leipzig.de/∼strimmer/lab/software/m-files/covshrink-kpm/covshrinkKPM.m

[3] http://biogibbs.stanford.edu/∼jiangh/glmnet_matlab.zip

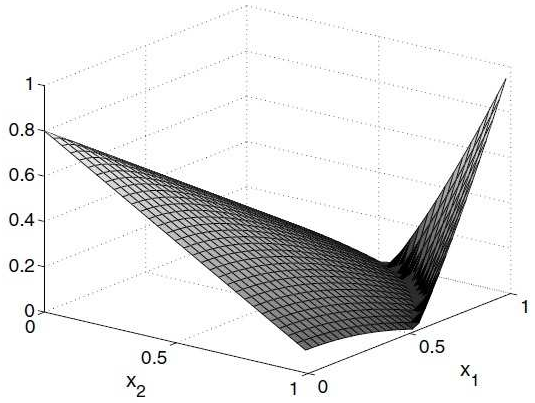


Figure 2: 2-dimensional continuous trap function. The figure is taken from [23].

unit interval ($[0, 1]$). The function is to be maximized and its optimum value is $n/2$ for a solution where all variables have a value of 1.

Two well known set of test functions are also used for some of the experiments in this study. The first set is the GECCO real-parameter Black-Box Optimization Benchmarking (BBOB) functions [12] which include 24 functions with different properties. All of these functions should be minimized and have a well defined target value to be reached by optimization algorithms. The second set is the CEC Large Scale Global Optimization (LSGO) functions [33] that include 20 functions with a problem size of 1000 variables. All of these functions that are considered as high dimensional optimization problems should be minimized. More specifically, the fifth function of this test set is used in this paper for studying the behavior of one of the proposed R-EDAs.

This function, called F_5, is a single-group shifted and m-rotated Rastrigin function of 1000 variables

$$F_5(\mathbf{x}) = 10^6 f_{rastrigin}(\mathbf{z}(P_1 : P_m) \cdot M) + f_{rastrigin}(\mathbf{z}(P_{m+1} : P_n) \cdot M)$$

where $\mathbf{x} \in [-5, 5]^n$ and $\mathbf{z} = \mathbf{x} - \mathbf{o}$ is the solution obtained by shifting the input solution $\mathbf{x}$ by a vector $\mathbf{o}$ which represents the global optimum solution. P is a random permutation of variable indices $\{1, 2, \ldots, n\}$. m is the group size used for producing different degrees of separability in the function and is set to 50. M is an $n \times n$ orthogonal matrix used for rotating the solutions. $f_{rastrigin}$ is the original Rastrigin function

$$f_{rastrigin}(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i) + 10)$$

4.2 GLR-EDA and The Influence of Regularization Parameter

The value of regularization parameter (λ) in GLR-EDA is left to be an open parameter. To study the influence of this parameter on the behavior of the algorithm, GLR-EDA is tested on GECCO BBOB set of functions, with different λ values. The experiment includes problem sizes of 5 variables as a low-dimension problem and 40 and 50 variables for a medium-scale problem. The population size is set to $20n$ (n is the number of variables) and the maximum number of generations is set to 500.

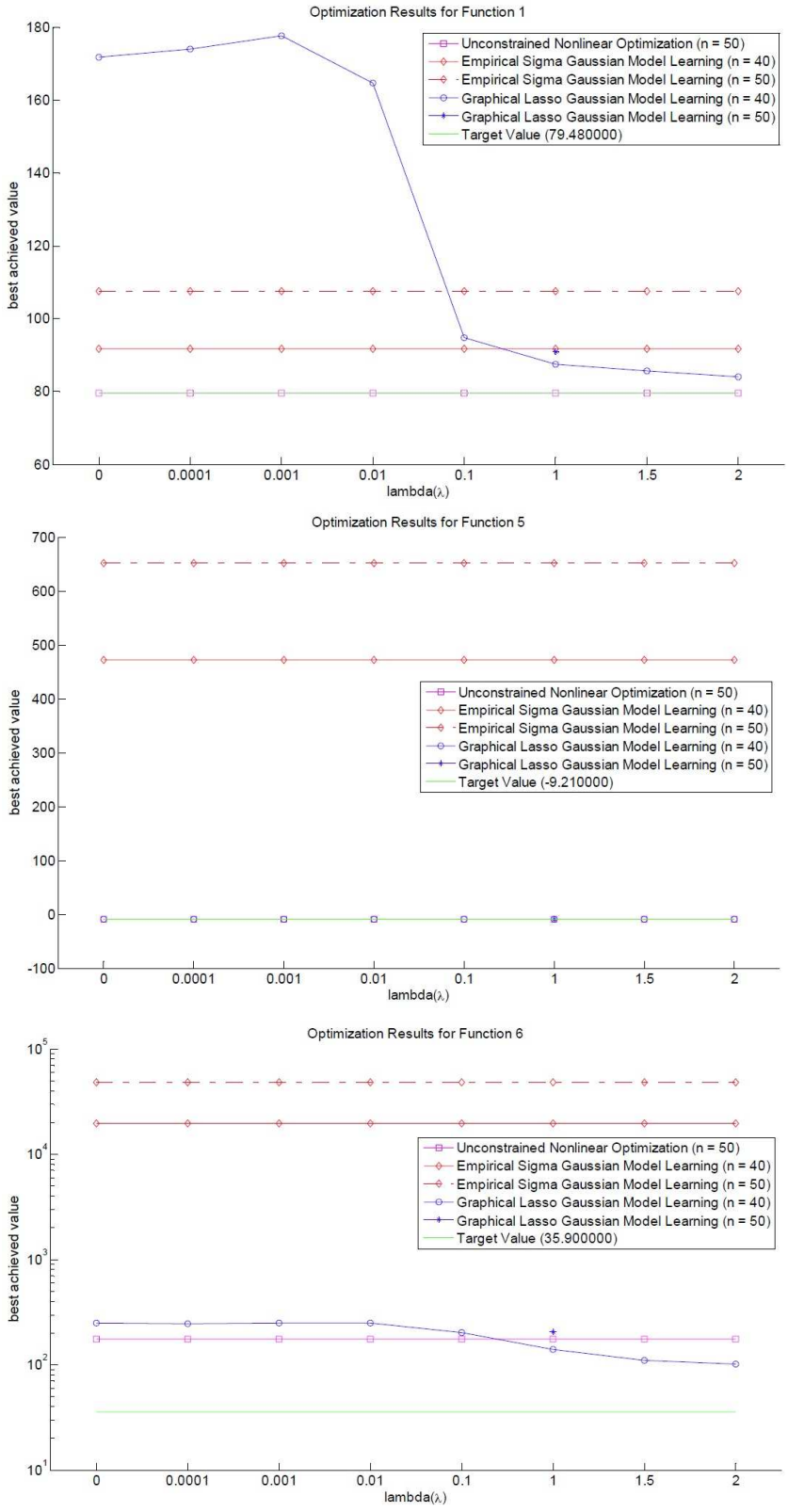


Figure 3: Best achieved values on some of BBOB functions. Here GLR-EDA is depicted with "Graphical Lasso Gaussian Model Learning" and EMNA with "Empirical Sigma Gaussian Model Learning".

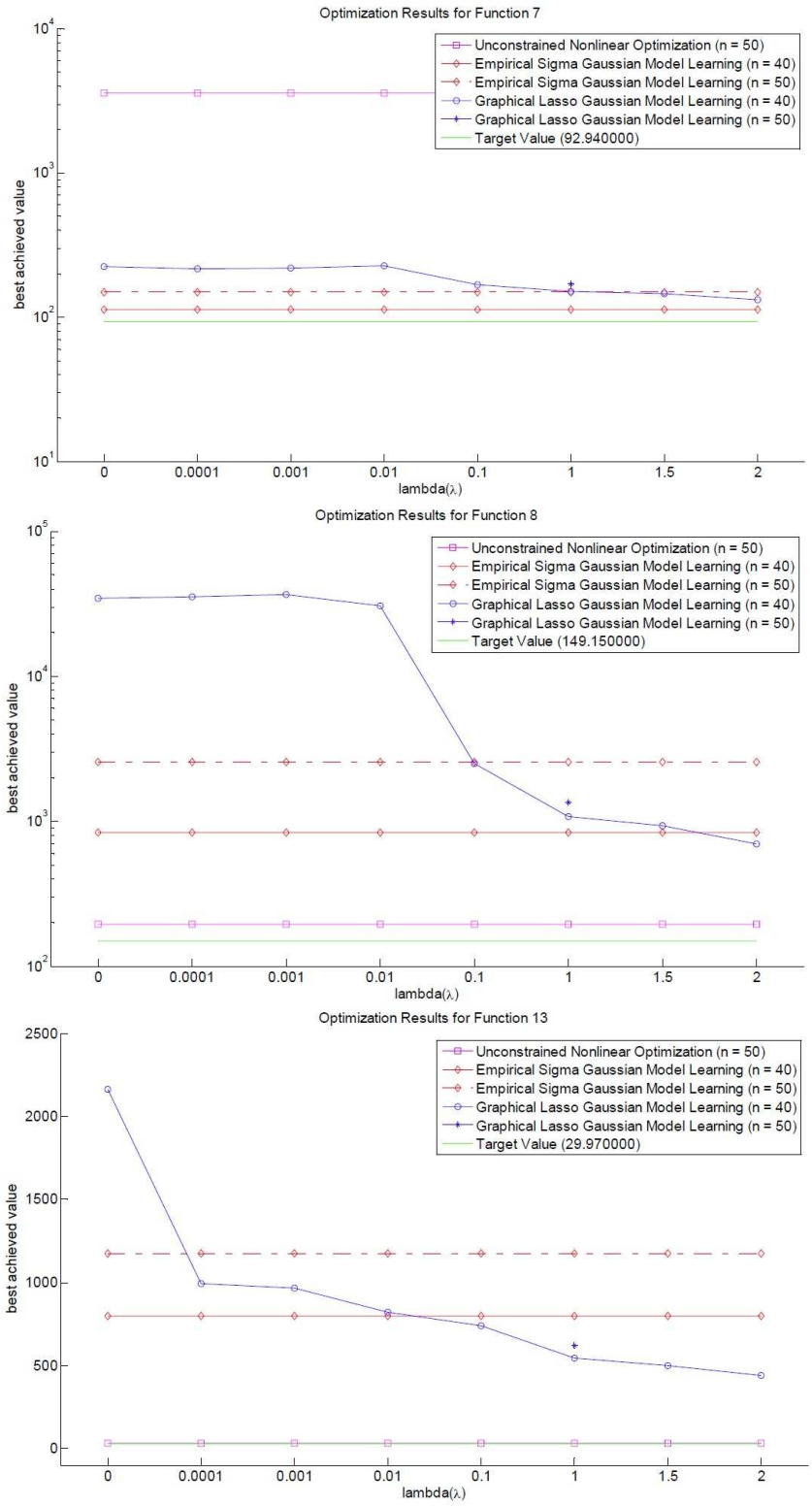


Figure 4: Continuation of Figure 3

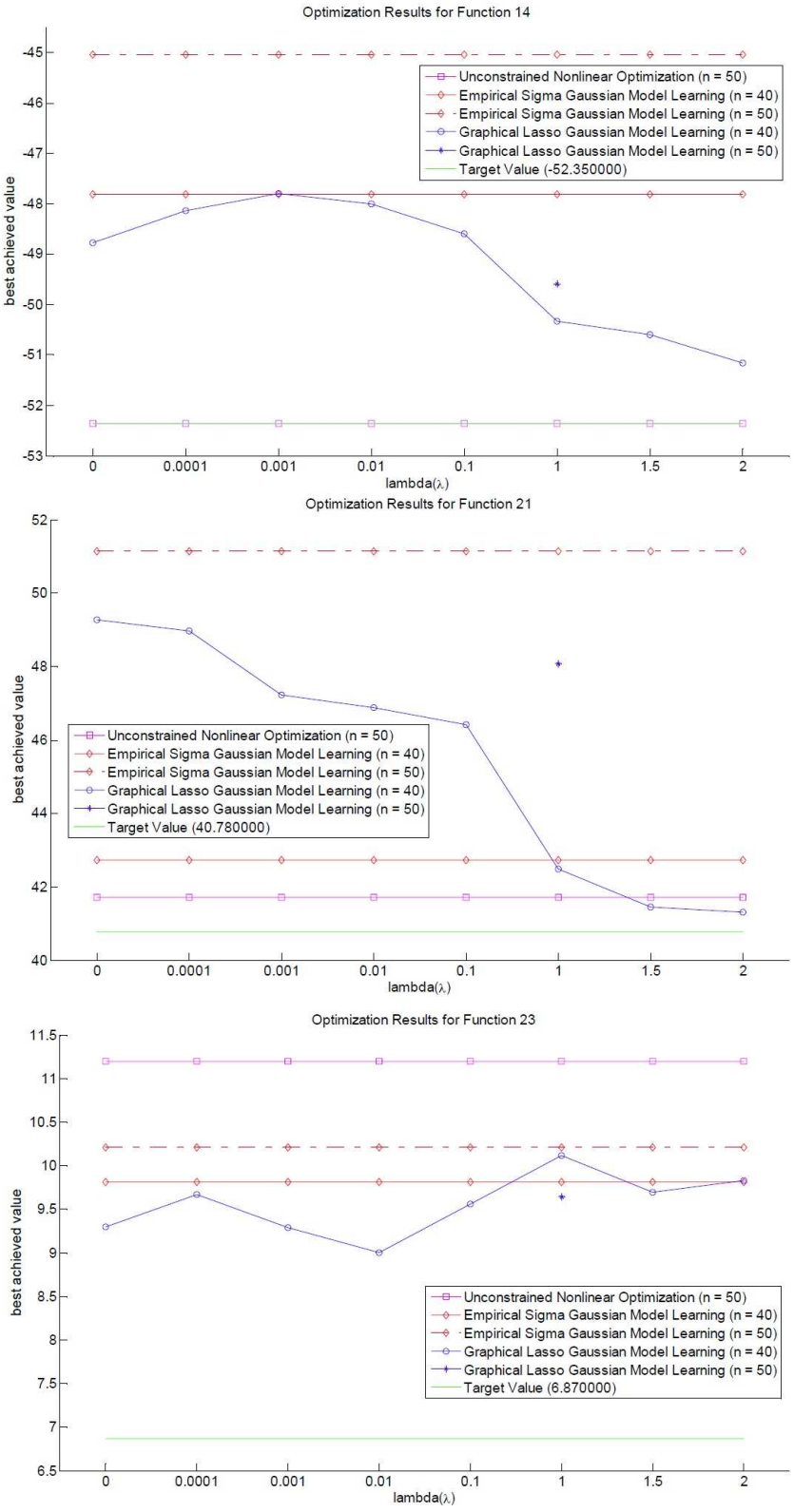


Figure 5: Continuation of Figure 3

To have an idea of how well GLR-EDA is performing with different λ values, EMNA and an unconstrained non-linear optimization (UNO) algorithm [5], employed as a local optimizer, are also included in the experiment. The parameter setting of EMNA is set identical to that of GLR-EDA. For UNO, its default parameter values are used. Since there are 24 different functions in the test set and for each function, different problem sizes are considered, therefore for brevity only the results obtained for some of the functions with medium-scale problem size are presented here (Figures 3 – 5).

The results obtained from this experiment suggest that although the behavior of GLR-EDA is different for each function, but in general for many of the functions the algorithm performance improves as the λ value increases. However there are functions that the best achieved value (BAV) is far away from the optimum value. The figures presented here also suggest that for some of the functions GLR-EDA is able find better BAVs than those found by EMNA or UNO whereas there functions that GLR-EDA is completely dominated by the other two in terms of BAV. A major problem of GLR-EDA is its large computational time requirement related to its covariance regularization step as the problem size increases. Therefore in the case of 50 variables only $\lambda = 1$ (which was suggested by smaller problem sizes to be a relatively good value) is used.

4.2.1 The Effect of Regularization Parameter on Model Accuracy

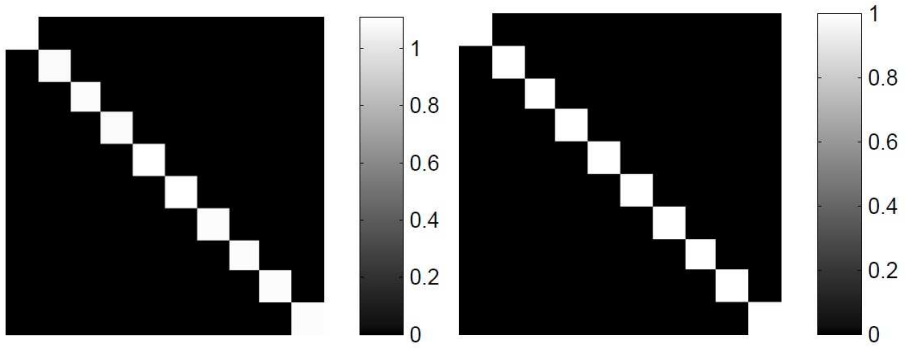
The previous experiment showed that GLR-EDA performance can be improved using a correct λ value setting. The large test set used in that experiment includes functions that exhibit different problem properties and therefore GLR-EDA behavior was not similar for all of these functions. Moreover for some functions the algorithm was not able to reach a reasonable neighborhood of the optimum value. Increasing the population size or the maximum number of generations was also tested but did not show to be fruitful (results not shown here).

Thus, to have a closer inspection of the regularized models learnt in GLR-EDA with different λ values and how they affect the optimization process, the algorithm is tested on the simpler 2D-deceptive function that has a known problem structure. To be able to easily study the learnt models, problem size is set to 10 variables. The population size is set to 200 and the maximum number of generations to 500, similar to the previous experiment.

In this experiment the individual traps of the 2D-deceptive function are formed on the variables on the two side of the solution up to those in the middle, i.e. each variable i in the solution string is coupled with variable $n - i + 1$ in a trap function. Figures 6 and 7 show the typical model structures learnt for this function with two different λ values, although other values have also been tested. The structures are presented with the inverse covariance matrix (precision matrix) and its related partial correlations matrix of the Gaussian distribution.

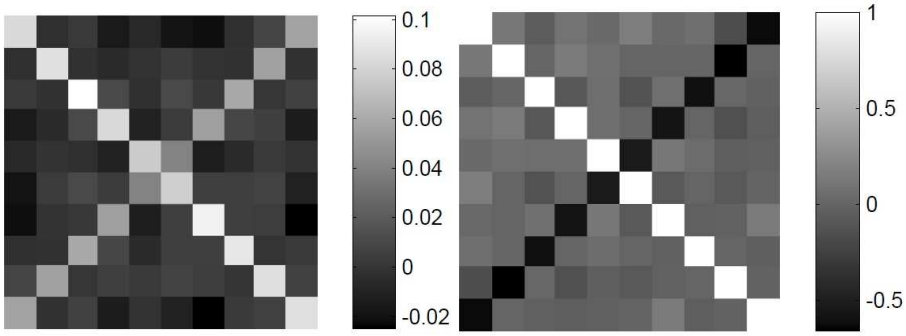
The models obtained in this experiment show that larger λ values placing a larger intensity penalization on the model parameters, will actually cause the models to become so sparse that eventually they do not capture any dependencies between the variables. In such situations the model learnt by GLR-EDA is actually reduced to a univariate Gaussian model, and therefore its behavior will become similar to that of continuous Univariate Marginal Distribution Algorithm (UMDAc) [16].

On the other hand, for λ values close to zero many unwanted spurious dependencies



(a) Precision matrix (b) Partial correlations matrix

Figure 6: Model learnt with $\lambda = 1$ for a 10 variable 2D-deceptive function



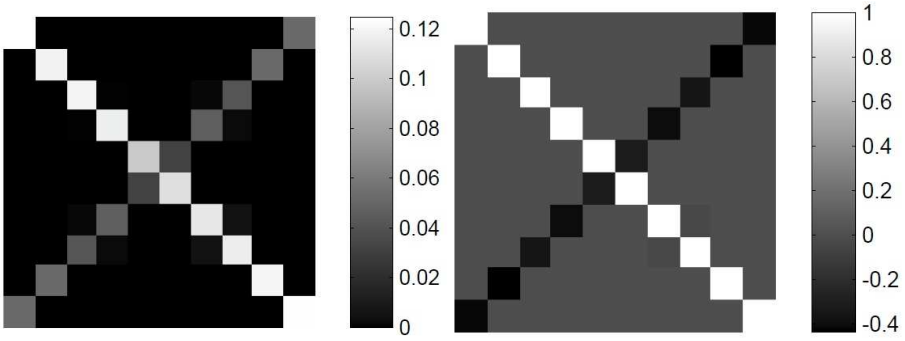
(a) Precision matrix (b) Partial correlations matrix

Figure 7: Model learnt with $\lambda = 0.001$ for a 10 variable 2D-deceptive function

appear in the model. This can explain why for larger problem sizes the computational time requirements of model learning with λ values close to zero, rapidly increases. To make a compromise between these two extremes, a dynamic λ value setting scheme is also tested. In this scheme, GLR-EDA starts from a small λ value (here an initial value of 0 is used) and gradually increments it in each generation. The incrementation step is set to 0.0001.

A typical model structure obtained with this method is depicted in Figure 8. It is evident that this model learning scheme is able to include the necessary dependencies while preventing many of the spurious links to enter the model. This good structure discovery is because small λ value at early generations of the algorithm allow more information about the dependencies (possibly spurious) to be included in the model. As this parameter is gradually incremented until the final generations it causes a fine-tuning effect on the model, removing many incorrect dependencies.

The results obtained from this experiment suggest that, using a similar λ value setting in all of the GLR-EDA generations may not necessarily result in the best modeling accuracy. To check how this increased modeling accuracy can influence the performance of the algorithm, GLR-EDA with dynamic λ setting was tested on 2D-deceptive function with different problem sizes. Other algorithm parameters such as population size and maximum number of generations are kept similar to those previous experiments. UMDAc and EMNA where also included in the experiment for comparison as they roughly



(a) Precision matrix (b) Partial correlations matrix

Figure 8: Model learnt with a dynamic λ value setting for a 10 variable 2D-deceptive function

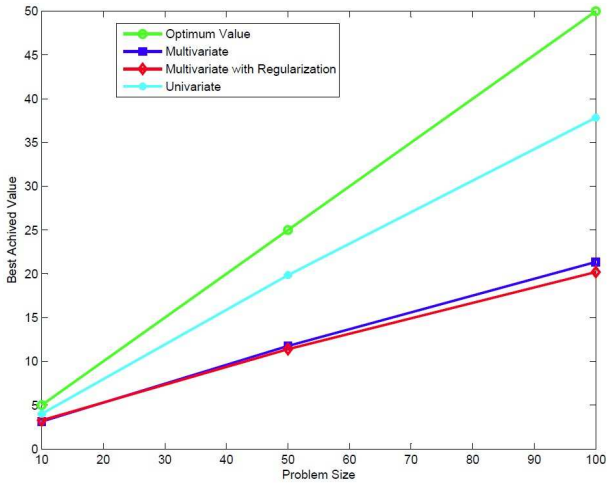


Figure 9: Performance results obtained for different EDAs on 2D deceptive function. Here UMDAc is depicted with "Univariate", EMNA with "Multivariate" and GLR-EDA with "Multivariate with Regularization".

represent special cases of the GLR-EDA. EMNA is similar to GLR-EDA with $\lambda = 0$ and UMDAc to a GLR-EDA with larger λ values, e.g. $\lambda = 1$ as was shown in the previous experiment.

Figure 9 illustrates the BAVs of these algorithms, averaged over 20 independent runs. It can be seen that as the problem size increases, the performance of GLR-EDA and EMNA degrades faster than UMDAc. This behavior of GLR-EDA implies that obtaining more accurate models may not necessarily result in better optimization performance of the algorithm. The simple model learning employed in UMDAc (ignoring any dependencies between variables), enable the algorithm to reach better BAVs. However this algorithm gets trapped in the huge number of local optima of this problem and is not able to reach the optimum value since its model is not using sufficient statistical information.

4.3 The Influence of Diversity Maintenance in CSR-EDA Performance

Unlike GLR-EDA, the regularization parameter in CSR-EDA is analytically computed using a closed formula. Therefore it is possible to apply it to different problems without needing to adjust the correct λ parameter. Furthermore, the initial experiments done on comparing GLR-EDA and CSR-EDA on the GECCO BBOB functions (the results are not displayed here), have shown a better performance of the latter algorithm. Nevertheless both of these algorithms are unable to obtain good optimization results for some of the functions and get stuck in local optima far away from global optimum value. One of the reasons for this behavior can be due to rapidly losing the population diversity, which can result in lack of good exploration of the search space.

As a solution for this problem, the influence of incorporating some diversity maintenance methods to CSR-EDA on algorithm performance is investigated. CSR-EDA is chosen for this purpose because, the same as GLR-EDA, it is using an MND as its probabilistic model, and population diversity loss is a common phenomenon in EDAs that use this type of modeling. However it has shown both better optimization performance and computational efficiency, and does not require specific parameter settings (e.g. λ). The diversity maintenance techniques considered are:

- Variance scaling: the variances along the diagonal of the covariance matrix are dynamically scaled to help a faster convergence [2].

- Reinserting initial solutions: if the initial population is sampled properly (e.g. a uniform distribution over all variables domain), it can be a very good source of knowledge about the fitness landscape of the problem. In this technique whenever the algorithm cannot improve the best found solution for a certain number of generations, a percentage of the current selected solutions will be replaced by some solutions from the initial population before model learning takes place. This method will be called second diversity forcing.

The behavior of CSR-EDA coupled with these diversity preserving techniques is tested on the 2D-deceptive function with different problem sizes. The algorithm performance is compared against one of the state of the art optimization algorithm, namely Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [11]. One of the issues well addressed in this algorithm, is its ability to prevent premature convergence and to bypass local optima. The default parameter setting is used for CMA-ES which uses a logarithmic population size with respect to the problem size. Therefore the population size of CSR-EDA is also set to $20 \log n$.

Figure 10 shows the experiment results, averaged over 20 independent runs. Since the algorithms use different population sizes and termination criteria, the required number of fitness evaluations for reaching BAVs is also depicted. It can be seen that although CSR-EDA is using a logarithmic population size with respect to problem size, but is still able to obtain improved BAVs in comparison to previous experiments. The incorporation of diversity maintenance techniques to CSR-EDA, improve the algorithm performance, though not significantly.

Although none of the algorithms is able to reach the optimum value, but CMA-ES is achieving better BAVs. Yet this algorithm is showing a less stable performance

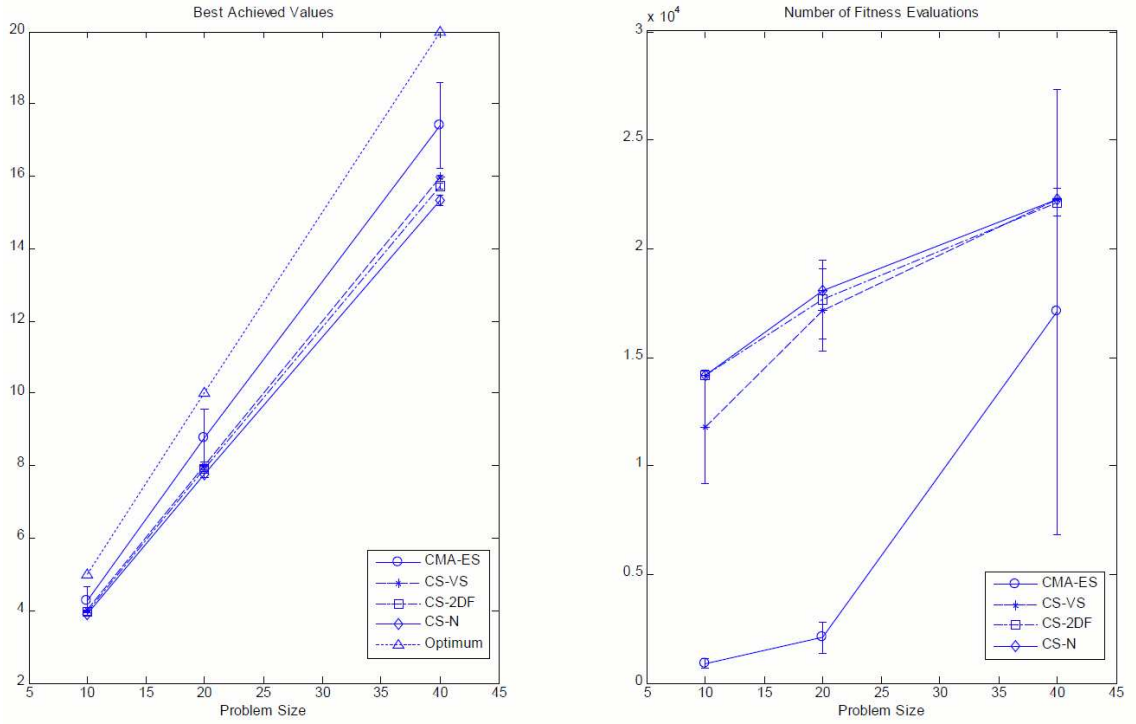


Figure 10: The best achieved value (left) and number of function evaluations (right) of CSR-EDA and CMA-ES on 2D-deceptive function. CSR-EDA with variance scaling is depicted as "CS-VS", CSR-EDA with second diversity forcing as "CS-2DF" and CSR-EDA without diversity maintenance as "CS-N".

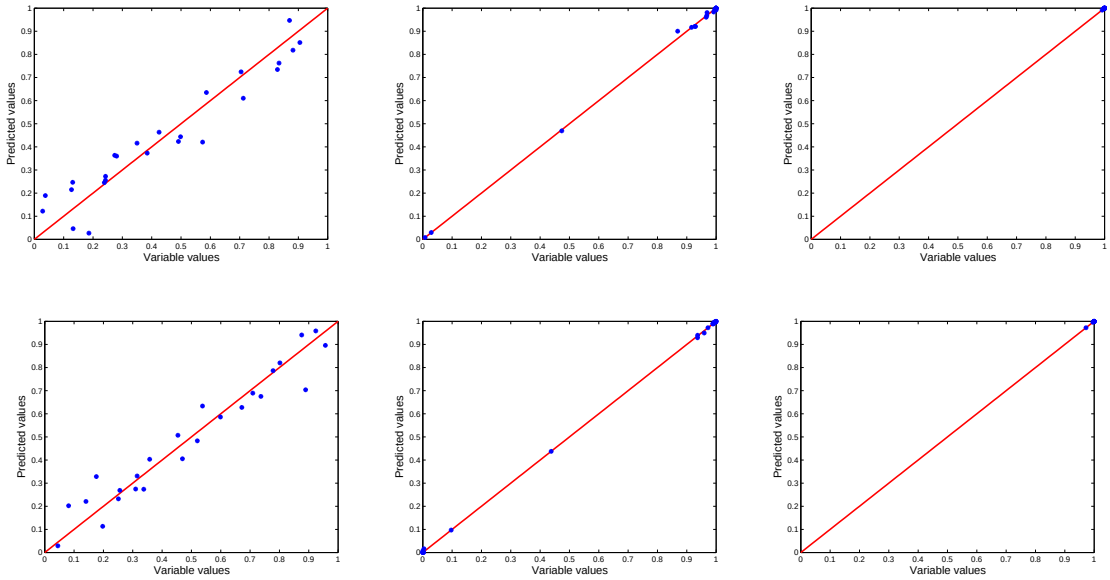
in comparison to CSR-EDA, both in terms of BAV and number of fitness evaluations. The results show that as the problem size increases, the required number of function evaluations for obtaining similar BAV increase more rapidly than that of CSR-EDA and with greater standard deviation.

4.4 The Modeling Accuracy and Performance of LR-EDA

Instead of using an MND for modeling the set of selected solutions, LR-EDA uses a LASSO regularized regression model for each of the variables. To investigate how accurately the model learnt for each variable can be used to predict its true value, LR-EDA is applied on the 2D-deceptive function with a problem size of 20 variables. The problem size is set to 25 samples. Figure 11 illustrates the prediction accuracy of two variables of a trap function at three different stages of LR-EDA evolution.

It can be seen that, although the initial population is distributed uniformly, the models learnt for the variables of a trap function at the first generation of the algorithm are able to give good predictions, close to the actual value of the variables, using the values of other problem variables. Furthermore the model learning of LR-EDA is perfectly acting alongside other parts of the algorithm while the generations advance, since the points are getting concentrated around local and global optimum values of each variable, and eventually only around the global optimum in final generations.

Figure 12 depicts another aspect of the model learning in LR-EDA. It shows the number of regression coefficients set to zero, averaged over the models learnt for all of



(a) Generation=1 (b) Generation=50 (c) Generation=100

Figure 11: Prediction accuracy of the models learnt for two variables of a 2D-deceptive trap function at different generations of LR-EDA. Each row corresponds to one of the trap function variables. The red diagonal line represents the correct prediction.

variables in the evolution path of LR-EDA. It can be seen that the sparsity of the models is increasing while the algorithm advance through generations. Considering the results presented in figures 11 and 12 together suggests that, as the prediction accuracy of each variable's model is improved in the evolution process, it is using fewer predictor variables.

To study how the model learning of LR-EDA can influence its performance, it is tested on the 2D-deceptive function with different problem sizes. The population size is set to $2.5n$ and the maximum number of generations to 200. The algorithm performance is compared with that of UMDAc, EMNA and CMA-ES. The parameters of UMDAc and EMNA are similar to those of LR-EDA. For CMA-ES the default parameter setting is used. Figure 13 shows the BAVs of these algorithms vs. the number of function evaluations, averaged over 20 independent runs.

It can be seen that, although LR-EDA initially produces bad optimization results in comparison to other algorithms, but it steadily and gradually improves the optimization results as the number of function evaluations increases, until it eventually outperforms the rest of the algorithms. Other algorithms, on the other hand, get stuck on some local optimum and no matter how many more function evaluations given, they can not improve their BAVs. The best contender after LR-EDA is CMA-ES which is able to rapidly reach a good BAV, but then gets stuck and can not move farther in the search space.

4.4.1 Behavior on High Dimensional Problems

Due to good performance and computational efficiency, LR-EDA is also applied to a high dimensional function with a problem size of 1000 variables. This function is the fifth function (F_5) of the CEC LSGO benchmark functions, and comprises two rotated

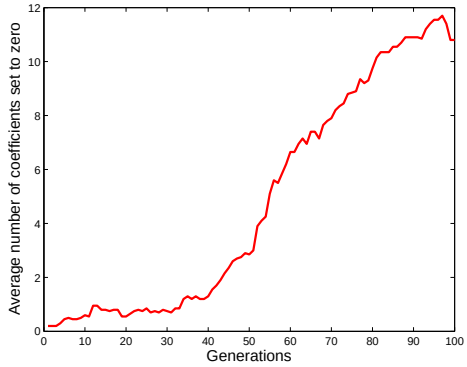
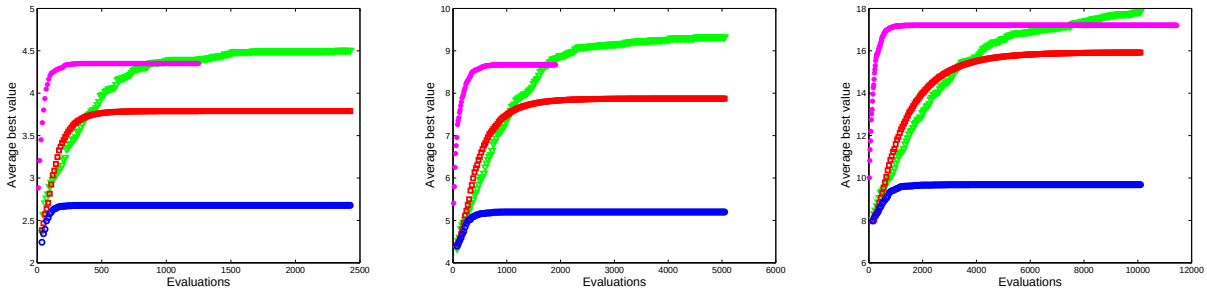


Figure 12: Average number of model coefficients set to zero at different generations of LR-EDA for a 2D-deceptive function.



(a) n=10 (b) n=20 (c) n=40

Figure 13: Best achieved values of LR-EDA (green), UMDAc (red), EMNA (blue) and CMA-ES (magenta) vs. the number of evaluations for 2D-deceptive function with different problem sizes. Shorter evolution curves mean the algorithm is terminated faster.

Rastrigin functions. The population size is set to 10000, the truncation threshold to 0.01 and the termination criterion is set to reach a maximum of 10^6 function evaluations. The sampling step of LR-EDA is modified to start from a perturbation of the selected solutions (10% of the variables are randomly modified) instead of starting from a random value setting. The re-estimation fraction is also increased to 1/5. These modifications are found to make the algorithm more appropriate to deal with large scale optimization problems.

The algorithm performance is compared with that of UMDAc, using the same algorithm parameters. Other EDAs, especially those using MND as their probabilistic model, can not be directly applied to such a large scale problem as they need tremendous computational time and space. Figure 14 shows the BAVs of these algorithms, averaged over 25 independent runs. It can be seen that, while UMDAc starts by achieving better optimization results, LR-EDA is able to eventually dominate it as UMDAc starts to get stuck in local optima. It is worth mentioning that LR-EDA was also tested on some other high dimensional functions of the CEC LSGO benchmark, but due to huge computational resources requirement, the experiments could not be completed.

The results obtained on this function are also compared against the results reported for

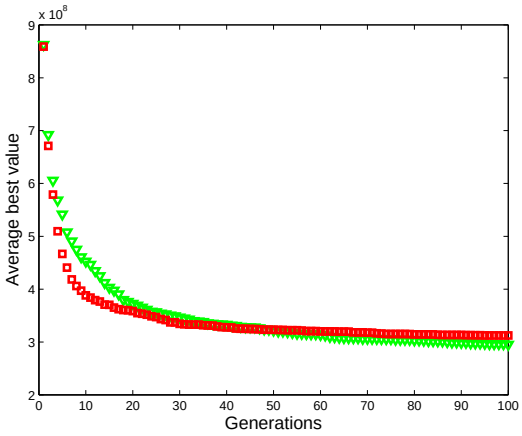


Figure 14: Best achieved values of LR-EDA (green) and UMDAc (red) vs. the number of generations for high dimensional F_5 function.

Table 1: Optimization results obtained by different evolutionary algorithms for high dimensional F_5 function. All results are to be multiplied by 10^8.

	LR-EDA	DECC-G	DECC-G*	MLCC	UMDAc
Best	2.36	1.50	2.08	2.15	2.76
Mean	2.95	2.63	2.45	3.84	3.12
Worst	3.76	4.12	2.72	4.67	3.38

three other evolutionary algorithms that are used as references for CEC LSGO benchmark functions. These algorithms are: Cooperative Co-evolutionary Differential Evolution with Grouping (DECC-G) [38] and its informative version (DECC-G*), which requires the structure grouping information, and Multi-Level Cooperative Co-evolution (MLCC) [39].

It should be noted that the results reported for these algorithms are obtained using a maximum number of function evaluations three times larger ($3 \cdot 10^6$) than that used for LR-EDA and UMDAc. The results are presented in table 1. According to this table, DECC-G is achieving the best function value and, if DECC-G* is not considered, it also obtains the best mean value. However its performance is less stable, a behavior also seen for MLCC. LR-EDA is able to obtain comparative optimization results, considering the fact that it is using fewer number of function evaluations, and its mean achieved result is better than that of MLCC and UMDAc. LR-EDA and UMDAc are also showing a more stable performance.

5 Conclusions

The incorporation of regularization techniques in the model learning of EDAs was investigated in this paper. Regularized model learning can be seen as one of the promising alternatives to increase the accuracy of the learnt probabilistic models while keeping the costs, in terms of function evaluations, relatively low. It can also be considered as a solution to the scalability limitations of EDAs when dealing with high dimensional

optimization problems.

A number of potentially useful regularization techniques which can be used for learning a probabilistic model were identified and incorporated to EDAs to produce three different versions of the regularized-EDA (R-EDA). The proposed algorithms were tested on different test functions and in comparison with different algorithms in a series of experiments. The experiments consider both the accuracy of the learnt models and the performance of the algorithm. Although the obtained results were still preliminary, it was shown that regularized-EDAs are able to obtain better results for some of the functions in comparison to other evolutionary algorithms.

The first two versions of R-EDA, used a multivariate Gaussian distribution as their probabilistic model. They used regularization techniques to obtain a regularized estimation of the covariance matrix. It was shown that the regularization parameter plays an important role in the behavior of the GLR-EDA and it should be taken into consideration when applying this algorithm. Furthermore several learning strategies were considered to increase the accuracy of the regularized model learning in this algorithm, although this increase in model accuracy did not necessarily resulted in better optimization results.

The effect of incorporating diversity maintenance techniques into CSR-EDA was tested using a logarithmic population size setting with respect to problem size. LR-EDA was another version of the algorithm which uses LASSO regularized regression for obtaining the neighbor set of each variable. It can serve for addressing problems with many variables since the learning algorithm it uses for a single variable is very fast. However, since it may be necessary to learn a probabilistic model for each variable, efficiency gains could be lost. The application of the algorithm to a high-dimensional test function revealed its ability to obtain comparative results with the state of the art algorithms.

While all of these versions of the R-EDA have shown promising results in terms of model accuracy or algorithm performance but more work is needed to be done in order to obtain a more robust regularization based EDA. From the results presented in this paper, it can be said that Gaussian distribution based R-EDAs are more suitable for smaller problem sizes, as they need to estimate a covariance matrix in each generation. The performance of these algorithms is degraded as the ratio of population size over problem size (N/n) gets smaller. The model learning algorithm may also result in incorrect covariance matrix estimation in some cases when the estimated parameters do not converge in a reasonable amount of iterations. Applying CSR-EDA is easier than GLR-EDA as the value of regularization parameter is not required to be set for this algorithm. LR-EDA is using a simpler and more efficient model learning algorithm and is more appropriate for larger problem sizes.

There are many possibilities for improving the behavior of the proposed algorithms. Issues such as the role of the λ parameter selection method should be carefully investigated. Methods for further reducing the number of times the model learning is performed (as it is very expensive), and other procedures for sampling new solutions from the model should also be studied. One of the main barriers in applying regularization techniques to EDAs is the computational cost and stability of producing consistent results of these methods. Therefore finding better regularization techniques can be of great importance. Multi-response regularized regression techniques can be a promising alternative for this purpose. Finally, it should be emphasized that the use of regularization approaches in EDAs opens a completely different way to deal with large scale problems. Other regularization schemes will probably produce even better improvements.

References

[1] P. Bosman and J. Grahl. Matching inductive search bias and problem structure in continuous estimation of distribution algorithms. *European Journal of Operational Research*, 185(3):1246–1264, 2008.

[2] P. Bosman, J. Grahl, and F. Rothlauf. SDR: A better trigger for adaptive variance scaling in normal EDAs. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 492–499. ACM New York, 2007.

[3] P. Bosman and D. Thierens. Advancing continuous IDEAs with mixture distributions and factorization selection metrics. In *Proceedings of the Optimization by Building and Using Probabilistic Models (OBUPM) Workshop at the Genetic and Evolutionary Computation Conference (GECCO '01)*, pages 208–212. ACM, 2001.

[4] P. Bosman and D. Thierens. Adaptive variance scaling in continuous multi-objective estimation of distribution algorithms. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 500–507, ACM New York, 2007.

[5] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[6] W. Dong and X. Yao. Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms. *Information Sciences*, 178(15):3000–3023, 2008.

[7] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004.

[8] J. Friedman, T. Hastie, H. H
"ofling, and R. Tibshirani. Pathwise coordinate optimization. *Annals*, 1(2):302–332, 2007.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical LASSO. *Biostatistics*, 9(3):432–441, 2008.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[11] N. Hansen. The CMA evolution strategy: A comparing review. In Lozano et al. [19], pages 75–102.

[12] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report 6828, Institut National de Recherche en Informatique et en Automatique (INRIA), 2009.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data mining, inference, and prediction*. Springer Series in Statistics. Springer New York, 1st edition, 2001.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, 2nd edition, 2009.

[15] A. E. Hoerl and R. W. Kennard. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

[16] P. Larrañaga, R. Etxeberria, J. Lozano, and J. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In A. Wu, editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO '00) Workshop Program*, pages 201–204. Morgan Kaufmann, 2000.

[17] P. Larrañaga and J. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[18] P. Larrañaga, J. Lozano, and E. Bengoetxea. Estimation of distribution algorithms based on multivariate normal and Gaussian networks. Technical Report KZZA-1K-1-01, Department of Computer Science and Artificial Intelligence, University of The Basque Country, Donostia, Spain, 2001.

[19] J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, volume 192 of *Studies in Fuzziness and Soft Computing*. Springer New York Inc., Secaucus, NJ, USA, 2006.

[20] N. Meinshausen and P. Bühlmann. High-Dimensional Graphs and Variable Selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

[21] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberger, and H.-P. Schwefel, editors, *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 178–187. Springer, 1996.

[22] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, volume 170 of *Studies in Fuzziness and Soft Computing*. Springer, 1st edition, 2005.

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

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

[25] M. Pelikan, K. Sastry, and D. Goldberg. Sporadic model building for efficiency enhancement of the hierarchical BOA. *Genetic Programming and Evolvable Machines*, 9(1):53–84, 2008.

[26] P. Pošík. Preventing Premature Convergence in a Simple EDA Via Global Step Size Setting. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Proceedings of The 10th International Conference on Parallel Problem Solving from Nature (PPSN X)*, volume 5199 of *Lecture Notes in Computer Science*, pages 549–558, Springer, Berlin, 2008.

[27] S. Rudlof and M. Köppen. Stochastic hill climbing with learning by vectors of normal distributions. In *Proceedings of the 1st Online Workshop on Soft Computing (WSC-1)*, pages 60–70, 1996.

[28] R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armañanzas, and S. Shakya. Mateda-2.0: Estimation of distribution algorithms in MATLAB. *Journal of Statistical Software*, 35(7):1–30, 2010.

[29] J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1), 2005.

[30] J. Schäfer and K. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

[31] M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using L1-regularization paths. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07)*, volume 2, pages 1278–1283. AAAI Press, 2007.

[32] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, volume 1498 of *Lecture Notes in Computer Science*, pages 418–427. Springer, London, UK, 1998.

[33] K. Tang, X. Li, P. Suganthan, Z. Yang, and T. Weise. Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2009.

[34] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[35] D. Vidaurre, C. Bielza, and P. Larrañaga. Learning an L1-regularized Gaussian Bayesian network in the equivalence class space. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 40:1231–1242, 2010.

[36] M. Wagner, A. Auger, and M. Schoenauer. EEDA: A New Robust Estimation of Distribution Algorithm. Technical Report 5190, Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt, France, 2004.

[37] J. Yang, H. Xu, Y. Cai, and P. Jia. Effective structure learning for EDA via L1-regularized Bayesian networks. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, pages 327–334, ACM New York, 2010.

[38] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, 2008.

[39] Z. Yang, K. Tang, and X. Yao. Multilevel cooperative coevolution for large scale optimization. In *IEEE Congress on Evolutionary Computation (CEC '08)*, pages 1663–1670. IEEE Press, 2008.

[40] B. Yuan and M. Gallagher. On the importance of diversity maintenance in estimation of distribution algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05)*, pages 719–726, ACM New York, 2005.

[41] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

A Mathematical Notations

The following notations are used for representing the mathematical relations in this paper:

- Non-italic capital letters are used for representing matrices
- Bold capital letters are used for representing vectors of variables
- Bold small letters are used for representing vectors of values
- Italic capital letters are used for representing a variable
- Italic small letters are used for representing a variable value
- Variable indices are denoted as subscripts while instance indices are represented as superscripts