Applied Soft Computing 13 (2013) 2412–2432

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Regularized continuous estimation of distribution algorithms

Hossein Karshenas[a,*], Roberto Santana[b], Concha Bielza[a], Pedro Larrañaga[a]

[a] Computational Intelligence Group, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain
[b] Intelligent System Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizabal 1, 20080 San Sebastian-Donostia, Spain

ARTICLE INFO

Article history:
Received 19 April 2012
Received in revised form
26 November 2012
Accepted 28 November 2012
Available online 11 December 2012

Keywords:
Estimation of distribution algorithm
Regularized model estimation
Continuous optimization
High-dimensionality

ABSTRACT

Regularization is a well-known technique in statistics for model estimation which is used to improve the generalization ability of the estimated model. Some of the regularization methods can also be used for variable selection that is especially useful in high-dimensional problems. This paper studies the use of regularized model learning in estimation of distribution algorithms (EDAs) for continuous optimization based on Gaussian distributions. We introduce two approaches to the regularized model estimation and analyze their effect on the accuracy and computational complexity of model learning in EDAs. We then apply the proposed algorithms to a number of continuous optimization functions and compare their results with other Gaussian distribution-based EDAs. The results show that the optimization performance of the proposed RegEDAs is less affected by the increase in the problem size than other EDAs, and they are able to obtain significantly better optimization values for many of the functions in high-dimensional settings.

1. Introduction

Estimation of distribution algorithms (EDAs) [1–6] are a class of evolutionary algorithms based on estimating a *probability distribution model* for the space of possible candidate solutions to the given problem. This probabilistic model, which is learnt from a set of candidate solutions selected according to their quality, is used to generate new candidate solutions in the search space.

Assuming that the method used for generating new solutions is more likely to sample regions of the search space that have a higher probability, the ultimate goal of the model learning step in EDAs is to estimate probabilistic models that assign higher probabilities to a close neighborhood of optimal problem solutions (specified by the corresponding fitness function). It is almost impossible to estimate such a model directly at one go, especially for high-dimensional problems with a complex structure and high dimensionality.

Iterative model learning and factorized estimation of the probability distribution are two main techniques employed to facilitate model learning in EDAs. If the model is estimated across several generations, the algorithm can visit more regions of the search space and gradually improve its estimation as, due to the limitation of computational resources, algorithms have to work with a finite population of solutions. Techniques like univariate or bivariate factorization, or more generally, multivariate Bayesian network learning, which imposes a factorization over problem variables, are able to estimate the joint probability distribution as the product of simpler factors.

The use of probabilistic modeling in EDAs allows these algorithms to better exploit the information obtained up to the current stage of the search, in order to speed up convergence. Many of the probabilistic models employed in EDAs can also approximate the relationships or linkages between variables which is necessary for finding the optimal solutions to many problems. The successful application of EDAs to many real-world problems in different domains like: machine learning [7,8], bioinformatics [9–11], scheduling [12–14], industrial design and management [15,16], protein folding [17,18], software testing [19] and composite materials [20] have proved their usefulness in practice.

Despite promising performance for solving many real-world problems, there are still shortcomings in the behavior of EDAs that have made them the topic of active research. Several studies have tried to analyze the behavior of EDAs [21–26]. However, their results are mainly based on impractical assumptions or are limited to only specific problems. In continuous domains, especially, which is the scope of this paper, there are many difficulties with model estimation that prevent EDAs from exhibiting the expected behavior.

The ability of the chosen probabilistic model to fit the solutions of a given problem, which is referred to as model capacity [27], can greatly affect model estimation. Thanks to their analytical properties, Gaussian distributions have been the probabilistic model of

* Corresponding author. Tel.: +34 913363675; fax: +34 913524819.
E-mail addresses: hkarshenas@fi.upm.es, hosseinkarshenas@gmail.com (H. Karshenas), roberto.santana@ehu.es (R. Santana), mcbielza@fi.upm.es (C. Bielza), pedro.larranaga@fi.upm.es (P. Larrañaga).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.11.049

choice in most continuous EDAs [28–30,2,31]. However, a robust estimation of Gaussian distribution relies on acquiring adequate statistics that are often not available from the population of continuous EDAs. This will usually cause EDAs to fall into premature convergence (or rather stalemate). To overcome this shortcoming, techniques like variance scaling [32–34] or eigenvalue resetting [35,36] have been proposed in the literature.

Regularization techniques [37–40] are widely used in statistics and machine learning to obtain a more robust estimation of probabilistic models with lower prediction error. Regularized model estimation attempts to decrease the general prediction error of the estimated model by reducing the high variance caused for the prediction of new and unseen samples at the cost of introducing a little bias into the model [41,42]. The large-scale application of these techniques for model estimation, especially in high-dimensional problems where the number of samples is small compared with the number of variables, has proved useful.

Model estimation in EDAs has some characteristics that motivate the use of regularization techniques. Lack of adequate statistics can cause the estimated model to become highly biased to specific regions of the search space. This reduces its generalization ability which is an important factor when sampling the model. The use of regularization can reduce the generalization error of the estimated model in EDAs. Another important issue is the EDA scalability with regard to problem size. Estimating the probability distribution model of huge search spaces requires large population sizes. Since the model estimation and sampling parts of EDAs are very time-consuming, algorithm performance will decline steeply if population sizes are large, not to mention the memory constraints regarding large datasets. Being able to estimate a model of comparable quality using much smaller populations is a major requirement in these algorithms.

Very recently, regularization has been used in EDAs for discrete optimization. Yang et al. [43] used regularized regression in the context of a Bayesian optimization algorithm [44] to obtain a reduced set of candidate parents for each variable before searching for the correct Bayesian network structure. Luigi et al. [45] proposed the use of regularized logistic regression to learn the structure of the Markov network in the DEUM framework [46]. In a different context, Karshenas et al. [47] studied some of the methods for integrating regularization techniques into the model estimation of continuous EDAs.

This paper analyzes some of the methods to regularized model learning in EDAs and shows how they can be applied to continuous optimization in high-dimensional settings. The rest of the paper is organized as follows. Section 2 reviews some of the background material about continuous EDAs and regularization techniques, used in other sections. Section 3 discusses the incorporation of different regularization techniques into EDAs and studies their effect on model estimation using synthetic data. The results of applying the proposed algorithms on different well-known optimization functions are presented in Section 4. Finally, the conclusions and future perspectives are given in Section 5.

2. Background

2.1. Multivariate Gaussian distribution

A joint *multivariate Gaussian distribution* (MGD) for n random variables $X_1, \ldots, X_n$ is determined with two overall parameters: $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is an n-dimensional vector of mean values for each variable, and $\boldsymbol{\Sigma}$ is a $n \times n$ symmetric and positive semidefinite covariance matrix. The total number of individual parameters (free parameters) that have to be estimated in order to determine an MGD is $(n^2 + 3n)/2$, i.e. of $O(n^2)$ complexity.

Positive definite matrices are interesting since they are full-ranked and non-singular, implying that their inverse exists. The inverse of a positive definite covariance matrix, which is called the precision or concentration matrix, represents partial covariances between variables and any zero entry in this matrix implies that the corresponding two variables are conditionally independent given all other variables. Therefore, the zero pattern of the precision matrix directly induces the graphical structure of a Markov network. The positive definiteness of the covariance matrix also allows for a unique triangular decomposition, known as Bartlett or Cholesky decomposition [48], that can be used to generate samples from the corresponding MGD. These types of sampling algorithms have also been extended to work for MGDs with positive semidefinite matrices.

In many application domains, the covariance matrix of MGD is obtained with maximum likelihood (ML) estimation using a dataset of N samples, which is denoted with $\boldsymbol{S}$ (assuming row-wise vectors)

$$\boldsymbol{S} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathrm{T}}(\boldsymbol{x}_i - \overline{\boldsymbol{x}}), \tag{1}$$

where $\overline{\boldsymbol{x}}$ is the ML estimation of $\boldsymbol{\mu}$. However, the covariance matrices obtained with ML estimation (Eq. (1)) usually result in a poor generalization of MGD [42,49]. For many applications, the covariance matrix should be positive definite or at least the partial correlations between the variables should be known. Therefore, several techniques for improving the estimation of the covariance matrix or its inverse have been proposed, some of which will be discussed in the following sections.

2.2. Estimation of distribution algorithms

Algorithm 1 shows the basic steps taken by an EDA for optimization. The algorithm starts from an initial population (step 1), which is usually generated randomly, though other techniques are applicable. In each generation, after selecting a subset of solutions according to their fitness values, a probability distribution model $\hat{\rho}_g(\boldsymbol{x})$ is learnt from the selected solutions to encode the general characteristics of these solutions (step 6). A set of new candidate solutions to the optimization problem is then generated using a sampling algorithm, which is incorporated into the EDA population (steps 7 and 9). This procedure is repeated until one of the stopping criteria (e.g. maximum number of generations, optimal solution(s), population convergence) is met (step 4).

Algorithm 1. The basic steps of an estimation of distribution algorithm

```
     Estimation of Distribution Algorithm
     Inputs:
       A representation of solutions
       An objective function $f$
1    $P_0 \leftarrow$ Generate initial population according to the given representation
2    $F_0 \leftarrow$ Evaluate individuals of $P_0$ using $f$
3    $g \leftarrow 1$
4    while termination criteria are not met do
5      $S_g \leftarrow$ Select a subset of $P_{g-1}$ according to $F_{g-1}$ using a selection mechanism
6      $\hat{\rho}_g(\boldsymbol{x}) \leftarrow$ Estimate the probability of solutions in $S_g$
7      $Q_g \leftarrow$ Sample $\hat{\rho}_g(\boldsymbol{x})$ according to the given representation
8      $H_g \leftarrow$ Evaluate individuals of $Q_g$ using $f$
9      $P_g \leftarrow$ Incorporate $Q_g$ into $P_{g-1}$ according to $F_{g-1}$ and $H_g$
10     $F_g \leftarrow$ Update $F_{g-1}$ according to the solutions in $P_g$
11     $g \leftarrow g+1$
12   end while
     Output: The best solution(s) in $P_{g-1}$
```

2.3. Regularization

Consider a simple linear regression model for estimating the values of a response (output) variable Y, given a set of n predictor (input) variables $\boldsymbol{X}=(X_1, X_2, \ldots, X_n)$ in continuous domains

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon,$$

where ϵ is a homoskedastic zero-mean Gaussian noise: $\mathcal{N}(0, \sigma^2)$. Given a set of N observations of the form $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ is a value setting for the variables in $\boldsymbol{X}$ and y_i is the corresponding response value, ordinary least square (OLS) estimation of the response variable $(\hat{Y})$ tries to minimize the sum of squared errors between the predicted value and the actual value:

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

where $\boldsymbol{\beta}=(\beta_1, \ldots, \beta_n)$. Regularization techniques try to improve model estimations like Eq. (2) by introducing a penalization term, imposed on the values of model parameters, denoted by $J(\boldsymbol{\beta})$. For example, the regularized OLS estimation is

$$arg \min_{(\beta_0,\boldsymbol{\beta})} \left(\sum_{i=1}^{N} (y_i - (\beta_0 + \boldsymbol{\beta}\boldsymbol{x}_i^{\mathrm{T}}))^2 + \lambda J(\boldsymbol{\beta}) \right), \quad (3)$$

where $\lambda \geq 0$ controls the amount of penalization. Some of the popular regularization techniques are as follows.

- Ridge regression [50] with $J(\boldsymbol{\beta}) = \sum_{j=1}^{n} \beta_j^2$. This penalization term (also called an ℓ_2 regularization term) causes the regression parameters to shrink toward zero, although they do not become exactly zero.
- LASSO (Least Absolute Shrinkage and Selection Operator) [51] or ℓ_1 regularization with $J(\boldsymbol{\beta}) = \sum_{j=1}^{n} |\beta_j|$. This type of penalization term has the appealing property of setting some of the regression parameters exactly equal to zero, which will result in a behavior similar to that of variable selection and hence the name [39].
- Elastic net [37], which is a combination of the previous two terms, i.e. $J(\boldsymbol{\beta}) = \sum_{j=1}^{n} (\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$, where $\alpha \in (0, 1)$ controls the combination of the two regularization terms. Using elastic net penalty, there is no limitation on the choice of only a maximum of N variables as in the case of LASSO regularization. This kind of regularization is especially useful for problems with a large number of variables when only a small number of observations are available for model estimation (i.e. "large n, small N" or "$n \gg N$" problems).

An important parameter that affects the outcome of regularized model estimation is the value of the regularization parameter λ. Although in some cases there are theoretical proposals [52] for selecting the value of this parameter or for the bounds of an effective value, one should generally use a trial-and-error strategy to select the best value. A more general approach is to obtain the solutions for a range of possible λ values. The solutions thus obtained by varying λ values form the *regularization path* or the profile of the regularization technique. In all of the above penalization terms, since the intercept parameter (β_0) can be estimated by the average absolute value of responses (y_i) it is left out, and without loss of generality we can assume it is equal to zero.

Adding the regularization term changes how we compute the optimal model parameters (solutions of the model estimation) are computed. In the case of ridge regression, optimal values can be computed using a closed-form formula. However, the LASSO penalization will render the equation indifferentiable and thus there are no closed-form formulas for calculating the solutions. Nevertheless, there are numerical optimization algorithms that can very efficiently compute the solutions along the whole regularization path and with the same computational cost as that of ridge regression [39].

Another well-known regularization technique used for selecting between different model estimations is the least angle regression (LARS) [53]. This method can be considered as an improvement of forward stage-wise model selection [54]. At each step of the LARS algorithm, the coefficients of the selected variables are fitted until another variable (not yet selected) reaches the same level of absolute correlation with the current residual of model estimation. At this point, the new variable is added to the model, and the coefficients begin to fit in an equiangular direction between all of the selected variables. In this way, contrary to LASSO and forward stage-wise regression, variables are added to but never removed from the model.

The LARS algorithm can also be used, with a simple modification, to efficiently fit models for LASSO and elastic net estimation. The entire sequence of LARS steps with $n < N$ variables requires $O(n^3 + Nn^2)$ computations, which is the cost of an OLS model fitting n variables. If $n \gg N$, the computational cost is of order $O(N^3)$, since the algorithm will terminate at the saturated OLS fit after N variables are added to the model.

2.3.1. Regularized estimation of MGDs

Regularization techniques have been extensively applied and studied in estimating multivariate Gaussian distributions, and especially their covariance matrices. Here some of these techniques are briefly reviewed.

Regularized Neighborhood Selection [38]. The aim of this method is to discover the conditional independence relationships of the inverse covariance matrix from a set of observations. It computes the set of potential neighbors for each variable using regularized regression on other variables. The neighborhood set of a variable X_j is defined as the smallest subset of variables so that, when given, X_j is conditionally independent of all remaining variables. 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 final structure. An AND strategy requires both of the variables to be present in each other's whereas an OR strategy will insert the dependency as soon as one of the variables appears in the other's neighborhood set. Based on a number of assumptions, they have shown that this method can asymptotically obtain consistent sparse models for high-dimensional data. A similar technique has also been applied to obtain the DAG structure of a Gaussian Bayesian network [55,56].

Covariance Shrinkage [49]. In shrinkage estimation an unrestricted high-dimensional model $\boldsymbol{S}$ (e.g. the ML estimation of the covariance matrix) is shrunk toward a restricted lower-dimensional target model $\boldsymbol{T}$ with fewer parameters (e.g. a diagonal covariance matrix where all off-diagonal elements are set to zero):

$$\boldsymbol{W} = \lambda\boldsymbol{T} + (1-\lambda)\boldsymbol{S}, \quad (4)$$

where $\lambda \in [0, 1]$ denotes the shrinkage intensity and is estimated using a closed formula [52]. Since there are many parameters in the high-dimensional model $(\boldsymbol{S})$ that need to be fitted, the variance of estimation will be high. On the other hand, the variance of the estimation of fewer parameters in the low-dimensional model $(\boldsymbol{T})$ is lower but is considerably biased with respect to the true model. It has been shown that the combination in Eq. (4) is a systematic way to obtain a regularized estimate of the model that outperforms each of the individual estimators in terms of both accuracy and statistical efficiency. Schäfer and Strimmer [49] compared the structure recoverability of this method and the previous technique [38] 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 the 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.

Graphical LASSO [57]. This method tries to maximize the regularized log-likelihood estimation of an MGD

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

where $\boldsymbol{\Theta}$ denotes the estimation of inverse covariance matrix and $\boldsymbol{S}$ is the empirical covariance matrix computed from the dataset. $det(\cdot)$ is the determinant operator, $trace(\cdot)$ gives the sum of the main diagonal elements of the input matrix, and $||\cdot||_1$ computes the sum of absolute values of the matrix entries. This problem can be solved by obtaining its derivative using element-wise sub-gradients. The block-wise optimization derived for this problem over blocks of rows (or columns) can be represented with a LASSO regularized OLS

$$min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \|\boldsymbol{\beta W}_{11}^{\frac{1}{2}} - \boldsymbol{s}_{12}\boldsymbol{W}_{11}^{-\frac{1}{2}}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{5}$$

where $\boldsymbol{W}$ is the regularized estimation of the covariance matrix and is partitioned as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_{11} & \boldsymbol{w}_{12}^{\mathrm{T}} \\ \boldsymbol{w}_{12} & w_{22} \end{bmatrix}.$$

In Eq. (5), $\boldsymbol{S}$ is similarly partitioned. After estimating the optimal $\boldsymbol{\beta}$ for each row (column) of the matrix, the covariance matrix estimation is updated by setting $\boldsymbol{w}_{12} = \boldsymbol{\beta W}_{11}$.

Instead of solving n separate regularized regression problems, the graphical LASSO algorithm couples and solves these problems together in the same matrix $\boldsymbol{W}$. In fact information can be shared across problems as the same matrix is used. The regularized neighborhood selection [38] can be seen as a graphical LASSO approximation related to the case where $\boldsymbol{W}_{11} = \boldsymbol{S}_{11}$.

3. Regularized model estimation in EDAs

The focus of this paper is on continuous domain optimization, modeled with Gaussian distributions. For this reason, we have considered two approaches employing regularization techniques:

- The first approach is based on obtaining a regularized estimation of the dependency structure between variables, and uses this structure to estimate the covariance matrix of the Gaussian distribution.
- The second approach applies techniques that directly obtain a regularized estimation of the Gaussian distribution.

The first approach, has to explicitly obtain a structure of variable dependencies. Structure estimation in this approach employs a three-step technique. In the first step, regularized regression models (Eq. (3)) are used to determine the dependencies between each variable and the other variables. These models estimate the value of a variable given the value of other variables. To favor sparser structures, which can be of considerable help for model estimation in high-dimensional problems and thus allow for better EDA scalability, only those regularization techniques that result in explicit variable selection (i.e. LARS, LASSO and elastic net) are considered in this step.

Since the whole regularization path is computed for the regression model of each variable, the regression solution (vector of regression coefficients) resulting in the lowest generalization error of the estimated model is selected in the second step. There are many metrics that can be used for selecting among different models, e.g. (k-fold) cross-validation, mean square error, Mallows' Cp statistic [58], Akaike information criterion (AIC) [59] or Bayesian information criterion (BIC) [60].

The variable (in)dependence information obtained from n (number of variables) independent regularized regression models is then combined in the third step to construct a single structure. Apart from the AND and OR strategies proposed by Meinshausen and Bühlmann [38], a third strategy, denoted “DAGS”, is to consider the set of neighbors of each variable as its possible Markov blanket and try to search for the corresponding directed acyclic graph (DAG) of a Bayesian network in a reduced search space constrained by this (in)dependence information [55,56].

Finally, the variable dependence structure learnt in this approach is used to obtain an estimation of the MGD. This will then be used in the sampling step of EDA for generating new solutions. If the resulting structure is a DAG, then the parameters of the respective Bayesian network can be obtained by, for example, ML estimation according to the given structure. If an undirected graph is learnt as the structure, then it can either be transformed to a DAG [61], be used as a dependency network [62], or be used as the zero pattern in the estimation of covariance matrix [63,64].

In this paper, a local greedy search algorithm [65] is used, along with the BIC scoring metric, to learn the DAG structure of a Gaussian Bayesian network from data in the context of the search + score methods and within a search space constrained by the variable dependence structure. When the learnt structure is an undirected graph (obtained from AND or OR strategies), a simple algorithm proposed by Hastie et al. [39] (section 17.3.1) is adopted to estimate the covariance matrix of MGD. This algorithm obtains a constrained log-likelihood estimation of the MGD by adding Lagrange constants for all independencies imposed by the given structure. The mean vector of MGD is obtained from ML estimation.

Unlike the first approach, the methods in the second approach are straightforwardly applicable in the context of EDAs (Algorithm 1). Here, both the methods of shrinkage estimation and graphical LASSO are considered for this approach and used to obtain the covariance matrix of MGD according to Eqs. (4) and (5). Like the first approach, the other parameter of Gaussian distribution (mean vector) is obtained using ML estimation.

Once an MGD is estimated using either two of the above approaches, new solutions are generated from the model by translating and rotating a normally distributed random vector according to, respectively, the mean vector and covariance matrix of the estimated MGD [47]. If the resulting model is a Bayesian network, the probabilistic logic sampling (PLS) algorithm [66] is used to generate new solutions.

3.1. Analyzing regularized model learning methods

To better examine the methods introduced in the two approaches for regularized model estimation, they are evaluated from different perspectives in this section, comparing their merits and disadvantages. For this purpose, a number of reference Gaussian models with a predefined variable dependence structure are synthesized. Table 1 shows the details of these reference models. In these models, which all have 100 variables (except the tri-variate model which has 99 variables), the dependent variables are assumed to be organized in different blocks. All variables within a block are correlated, whereas there is no inter-block dependency at all. The reference models differ as to the size of their blocks of dependent variables. They range from a block size of one (no dependence) in the univariate model to 20 in the vigint variate model.

These reference Gaussian models are used to generate a population of solutions that will be used for model estimation with

Table 1
Synthesized Gaussian distributions used as reference models.

Name	Block size	No. blocks	No. dependencies	No. independencies
Uni	1	100	0	10,000
Bi	2	50	100	9900
Tri	3	33	198	9603
Quad	4	25	300	9700
Quint	5	20	400	9600
Vigint	20	5	1900	8100

each of the regularization methods. Here, a combination of 3×3 methods from the first approach and two methods from the second approach, namely the shrinkage estimation and graphical LASSO, are considered for this study. Different combinations in the first approach result from considering each of the LARS, LASSO and elastic net regularization techniques with each of the AND, OR and DAGS merging strategies. In all of these combinations, the Mallows' Cp statistic is used for selecting the best regression solution in the regularization path of the model for each variable. The n/N ratio is fixed to 10 for all populations generated from the reference models to resemble model estimation in a high-dimensional problem.

3.1.1. True structure recovery

Recovering the true structure of the probabilistic model is one of the requirements for a good model estimation algorithm. The structural accuracy of the estimated models can reveal the learning algorithm capability to capture the interdependencies between problem variables. One of the ways to measure the accuracy of structures learnt in the regularized model estimation methods is to compute the confusion matrix entries, i.e. the number of *true positive* (TP), *false positive* (FP), *false negative* (FN) and *true negative* (TN) links of the model structure. Here we consider two well-known indicators computed using these measures:

- Sensitivity (TP/(TP + FN)): what percentage of the actual dependencies are correctly learnt.
- Specificity (TN/(TN + FP)): what percentage of the actual independencies are correctly learnt.

These indicators can give a clearer insight into the effects of different regularized estimation methods on the inclusion of spurious and excess links, or missing real dependencies. Figs. 1 and 2 show, respectively, the sensitivity and specificity of the model structures estimated with the methods in the first approach. Since the univariate model actually does not have any dependencies, it is not included in the analysis of structure sensitivity. All the results are averaged over 30 independent runs. When the resulting structure is a DAG, its undirected counterpart, obtained by removing the direction of the links, is considered for computing the indicators. This is because from a variable interdependency point of view, the important thing is the existence of a dependency not its direction.

The sensitivity results show that the proportion of true links captured by the methods in the first approach using different combinations of the regularization techniques and merging strategies drops as the size of the blocks of dependent variables increases. The OR strategy combined with all three regularization techniques has resulted in better sensitivity, as expected because it greedily adds links to the structure. On the other hand, the performance of the AND strategy changes with different regularization techniques from the sensitivity point of view. Whereas the LARS-AND combination is capturing more TP links than the LARS-DAGS combination, their sensitivity behavior comes very close to the LASSO technique, and the sensitivity of the AND strategy falls below the DAGS strategy when using the elastic net technique. The sensitivity results of all merging strategies decrease for elastic net, but this regularization technique appears to affect the AND strategy more than other strategies. One possible explanation for this behavior is the grouping effect of the elastic net technique, which causes several variables to be added to or removed from the regression model together. Therefore, when the AND strategy is used to couple all regression models, a group of variables is less likely to have a mutual relationship (as required by the AND strategy). Thus the resulting structure becomes sparser and may miss many TP links.

From the specificity point of view, the results show that the tendency of regularization techniques to obtain sparser models causes the structures learnt by the methods in the first approach to include very few spurious links. Increased block sizes do not seem to considerably affect these methods. Note at this point that, as shown in Table 1, the number of model independencies is larger with smaller block sizes. Therefore, normally one expects to see smaller specificity values for smaller block sizes. As block size increases, regularization techniques like LARS and LASSO tend to add more dependencies between the variables. However, as we saw with the sensitivity results, many of these links are incorrect and, thus, the number of FP links increases. Another point to be considered here is the total number of dependencies in a model compared with the sample size used for learning. This can affect the sparsity assumption based on which the consistency of some of the regularized model estimation methods in the first approach is shown [38].

When comparing the specificity and sensitivity results of different combinations of regularization techniques and merging strategies, they seem to be complementary. For example, the DAGS strategy results in poorer sensitivity results compared with the other two merging strategies combined with the LARS technique, but specificity is better. Whereas the sensitivity behavior of the AND strategy combined with the elastic net technique is worse than others, the same combination obtains better specificity results compared with all other combinations for all block sizes. Therefore, if a specific method tends to add more links to the structure (like the LARS-OR combination), the probability of recovering TP links will clearly increase, but so will the possibility of adding FP links. On the other hand, conservative methods like the ELNET-AND combination will hit fewer TP links but also add fewer spurious links to the structure.

Fig. 3 shows the sensitivity and specificity of the structures learnt by the methods in the second approach. Since these methods do not explicitly obtain a structure, the structure encoded in the covariance matrix of the estimated MGD is used to evaluate their ability to recover the true structure. This structure is obtained from the zero pattern in the inverse covariance matrix by introducing a link between every two variables whose respective entry in the inverse covariance matrix is not zero. The results show that the two methods in the second approach also follow a similar trend to the methods in the first approach: a decrease in the sensitivity and an almost uniform specificity as the size of the dependent variables block increases. These two methods discover more links than the methods in the first approach. Shrinkage estimation, especially, tends to add many links to the structure, and the increase in the number of dependent variables has less effect on this method. This results in a sensitivity of more than 50% for all block sizes. However, the specificity results show that many of the links that this method adds to the model are spurious and this method generally tends to obtain denser structures than other methods in either of the approaches. The graphical LASSO method behaves more like the methods in the first approach, and especially the LARS-OR and LASSO-OR combinations, as this method also uses a similar regularization mechanism by adding a LASSO penalization term to the ML estimation of the MGD.

3.1.2. Time complexity

The computational time needed by an algorithm is an important feature that can affect its range of application and how it is going to

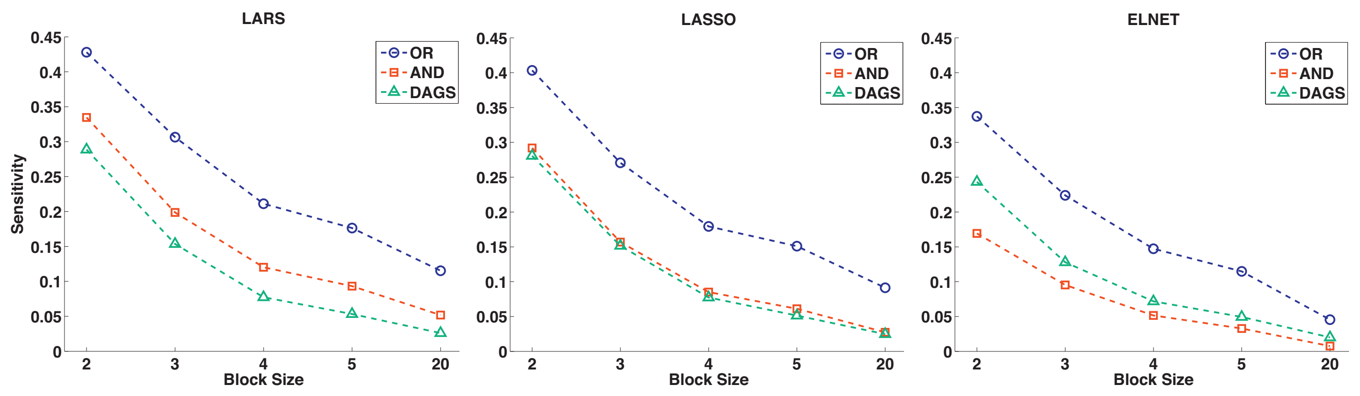


Fig. 1. Average sensitivity of the model structures learnt with the methods in the first approach.

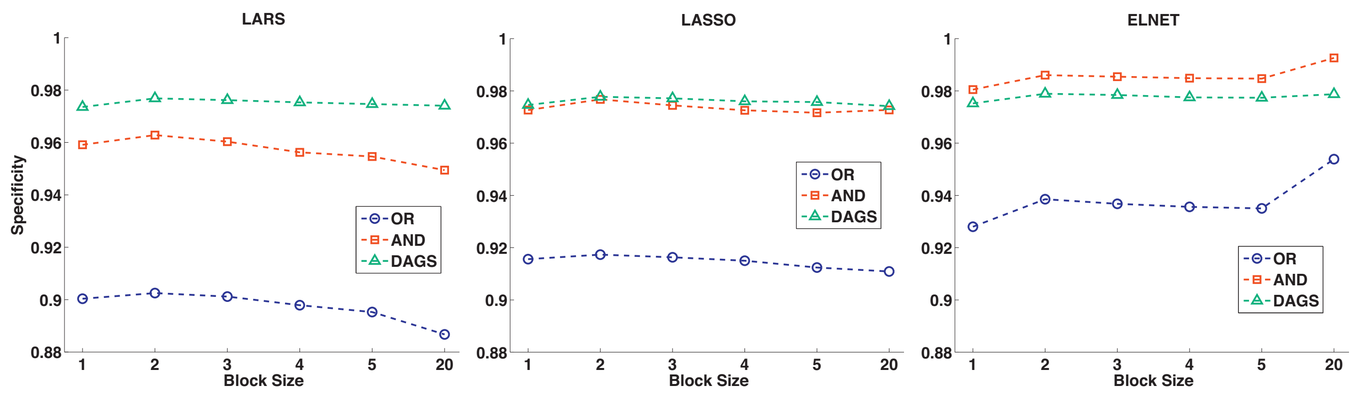


Fig. 2. Average specificity of the model structures learnt with the methods in the first approach.

be applied. In the case of EDAs, it is particularly critical since they make intensive use of learning methods. In this section, we examine the time complexity of the methods in each of the two approaches. The methods in the first approach have include three-step structure learning plus a parameter estimation of the target MGD. As already mentioned, the computational complexity of computing the whole regularization path of a regularized regression model for a variable with the LARS technique (which is also used as the base algorithm for the other two regularization techniques considered here) is $O(n^3+Nn^2)$. Note that in high-dimensional problems, which is the case in our study, it is far less than this. The cost of selecting the best regression solution from the regularization path is $O(lN)$, where l is the number of different solutions found for the regression model, and it is of order $O(n)$. These two steps are repeated for each of the n variables.

The AND and OR strategies used in the third step to merge the n models learnt for the variables are simple and only require $O(n^2)$ computations. However, the DAGS strategy is usually very costly and has a computational complexity of $O(kNn^2+kn^4)$, where k is the number of iterations that it takes the local greedy algorithm to search the constrained space. Finally, the algorithm used to estimate the covariance matrix according to the structure obtained by AND and OR strategies has a cost of $O(Nn^2)$.

The shrinkage estimation and graphical LASSO methods both have a total computational complexity of $O(Nn^2)$. The covariance matrix computation dominates the time requirement of

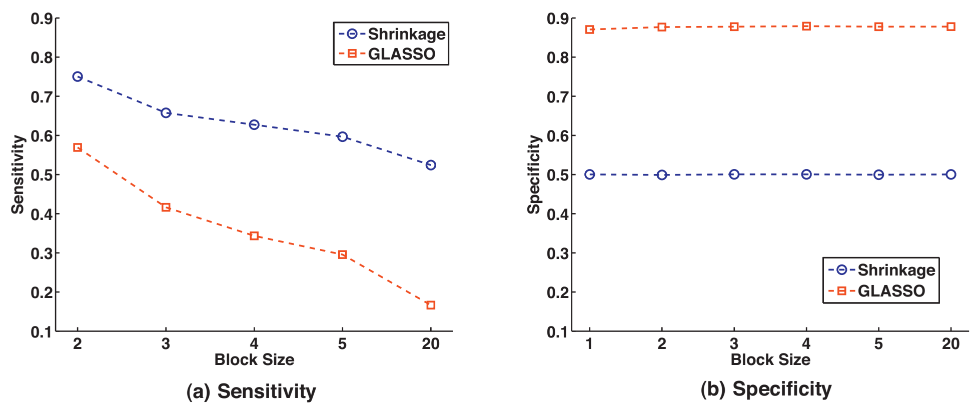


Fig. 3. Average accuracy of the model structures learnt in the second approach.

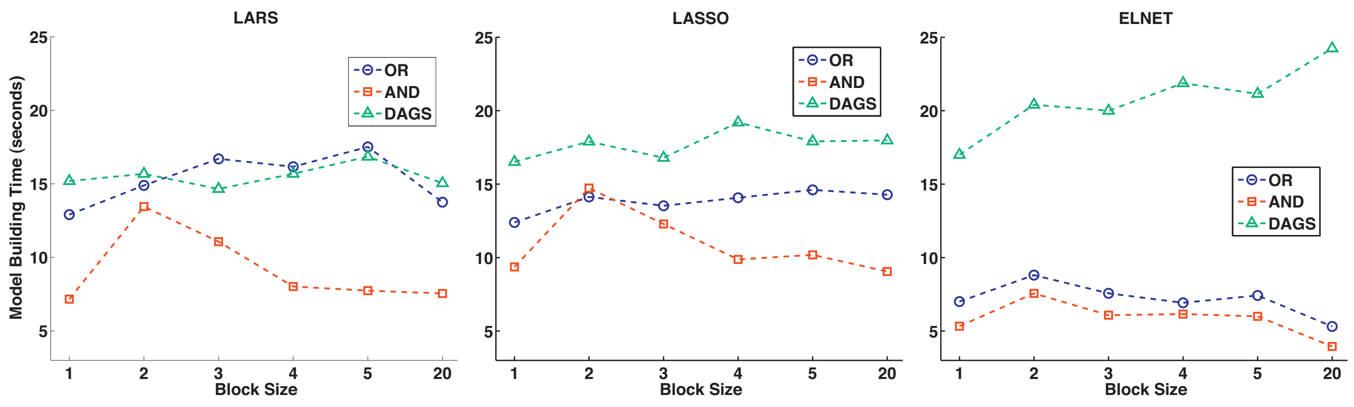


Fig. 4. Average model building time for the methods in the first approach.

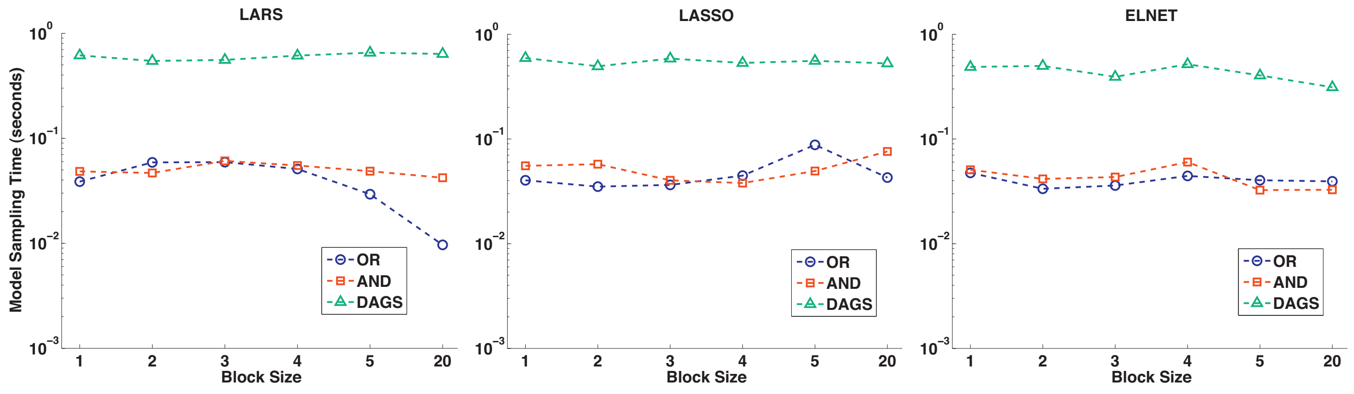


Fig. 5. Average model sampling time for the methods in the first approach.

computing the mean vector of the MGD, which thus does not influence the total cost of model estimation. The sampling algorithm used to generate M new solutions from the estimated MGD requires $O(n^3 + Mn^2)$ computations. This is also the case for the PLS algorithm when sampling a Bayesian network.

Figs. 4 and 5 show the total time that it takes all the methods in the first approach to learn a probabilistic model and then sample this model to generate a population of 1000 solutions. All the results are averaged over 30 independent runs. As expected, the DAGS strategy takes longer than the other merging strategies, though the choice of the regularization technique has a big influence on this strategy, with the LARS technique requiring less and elastic net technique more time. The regularization technique also has an impact on the AND and OR strategies combined with the elastic net technique, which takes less time than the other two regularization techniques. This can be explained by the sparser structures obtained by this regularization technique which in turn would lead to the estimation of fewer parameters. However, sparser structures have the opposite effect on the time requirement of the DAGS strategy. This can be traced back to the fact that the greedy local search algorithm is forced to reject many of the possible moves when searching a constrained space and it therefore takes longer to find a valid move.

The question is then whether searching in a constrained space can cause the learning time of a Bayesian network to decrease at all. Fig. 6 compares the model learning time required when searching an unconstrained space and a space constrained using the LARS technique. Both methods use the same greedy local search algorithm with a BIC scoring metric. Here, reference models with the increasing number of variables and equal size of dependent variables blocks, set to 5 (quint variate model), were sampled to generate populations of solutions, with a fixed n/N ratio of 10. These populations were then used for learning Bayesian networks. The results are averaged over 30 independent runs. The LARS-AND method is also included in the results for better comparison. We found that while, the model learning time of the constrained DAGS is larger, for smaller number of variables the computational time required by the unconstrained DAGS grows faster as the number of variables increase and becomes considerably larger than its constrained counterpart. This is because for larger number of variables, the search space becomes so huge that any kind of space-constraining information can serve as a heuristic to improve the search. For the difference in the estimation accuracy of these two methods, see [55,56].

As the model sampling times show (Fig. 5), it take longer to generate new solutions from a Bayesian network using the PLS algorithm than the algorithm used to sample an MGD. Although

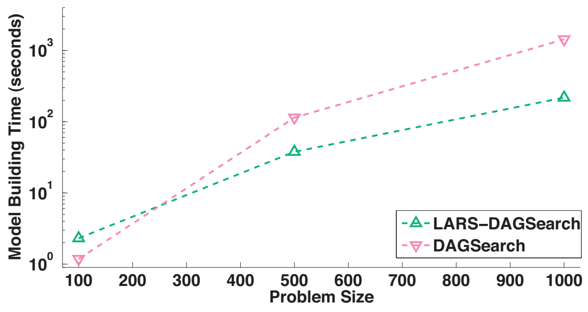


Fig. 6. Average model-building time for constrained and unconstrained DAG search algorithms on different problem sizes.

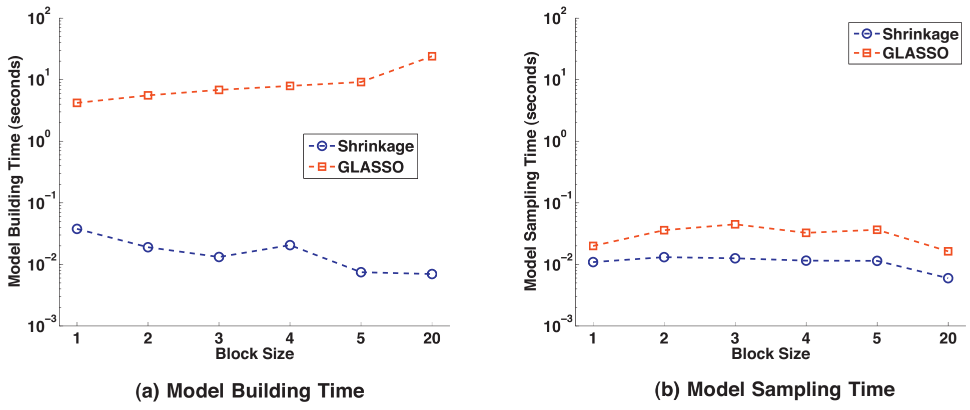


Fig. 7. Average model-building (left) and sampling (right) time for the methods in the second approach.

the same algorithm is used to sample the models learnt by the AND and OR strategies, the results show that the estimated models can affect the sampling time depending on the regularization technique and the block size. Generally, however both the model learning and sampling times do not appear to change considerably by increasing the size of dependent variables block, since all these models have the same dimension. Fig. 6 shows, on the other hand, that the increase in the number of variables raises the algorithm time requirement, as expected from the computational complexity analysis.

Fig. 7 shows the average model-learning and sampling times for the two methods in the second approach. Sampling times are computed for generating a population of 1000 solutions from the learnt model. The graphical LASSO method takes significantly longer to estimate the model than shrinkage estimation, and its time requirements closely follow those of the methods in the first approach, especially the LARS-OR and LASSO-OR combinations (whose structural accuracy is similar, too). Again, the size of blocks of dependent variables does not have a substantial effect on the computational time of these two methods, although further experiments have shown that graphical LASSO is highly sensitive to the violation of the conditions in the sparsity assumption. Section 3.1.4 discusses this issue in more detail. The time requirements of the shrinkage estimation method make it a perfect candidate for use within more complex algorithms like EDAs that require very fast-performing components.

3.1.3. Likelihood

Estimating an accurate structure is an important feature of a probabilistic model estimation method. For the purpose of model estimation and sampling in EDAs (Algorithm 1), however, it is more important to be able to generate solutions that are very close to the real problem solution, which is assumed to be representable with a probability distribution. One way to investigate this closeness is to compare the model estimated in EDA with the actual probabilistic model underlying the problem. Measures like Kullback–Leibler divergence [67] or Hellinger distance [68] can be used for this purpose. Another possibility is to evaluate the overall model estimation and sampling of EDAs in order to also take into account the inevitable model sampling error that is present in practice. This is also closer to the actual procedure enacted in each generation of an EDA. The evaluation measure computed in this study is the negative log-likelihood (NLL) of the reference Gaussian models (see Table 1) given the population of solutions generated from the estimated models.

Fig. 8 shows the NLL values of the reference Gaussian models with different block sizes, computed from the populations generated using the models learnt with the methods in the first approach. Each of the generated populations had a size of 1000. For the first two regularization techniques, the NLL values obtained with the DAGS strategy on larger block sizes tend to infinity and are therefore not included in the figures. Merging strategies are ordered similarly for all regularization techniques, with the NLL of OR strategy turned out to be better and DAGS worse. This is consistent with the structural accuracy results and reveals the mixed effect of sensitivity and specificity analysis for the methods in the first approach. The NLLs computed for these methods grow visibly as block size increases, suggesting that as the number of dependent variables in the problem increases the estimated models and populations generated from these models move further from the reference models, from a log-likelihood point of view.

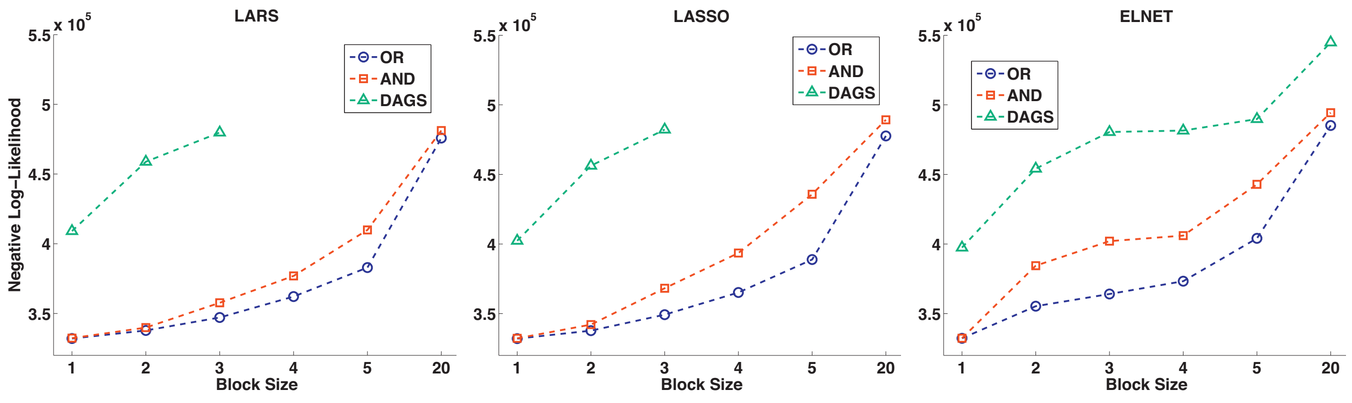


Fig. 8. Average negative log-likelihood of the reference Gaussian distributions obtained for model estimation methods in the first approach.

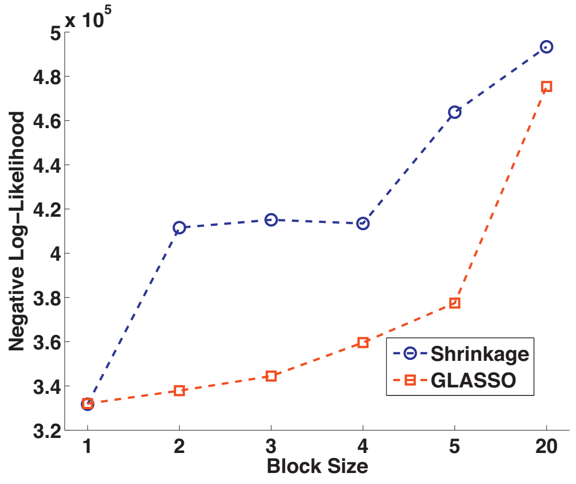


Fig. 9. Average negative log-likelihood of the reference Gaussian distributions obtained for the estimation methods in the second approach.

Fig. 9 shows the NLL results for the two methods in the second approach, where graphical LASSO results in better NLL values than the shrinkage method. Shrinkage estimation is able to capture more dependencies, but, at the same time, it adds many spurious links to the structure. As a result, the estimated probability distribution generates solutions that are less likely producible by the reference models. From the NLL values for graphical LASSO, and for LARS-OR and LASSO-OR used in the first approach, we also find that these methods behave similarly. Also, the NLL values computed for graphical LASSO and for the ML estimation of the MGD are almost equal, showing that, from a log-likelihood point of view, the addition of a LASSO penalization term does not have much impact on the probability estimation of the graphical LASSO method.

3.1.4. Regularization parameter of graphical LASSO

The regularization techniques employed in the first approach output the whole regularization path for varying values of the regularization parameter (λ) and then select the parameter resulting in the best regression solution according to a model selection metric. Also, the value of the shrinkage intensity in the shrinkage estimation method is analytically computed according to the learning data set. However, the regularization parameter of the graphical LASSO method is left open, as it is not practicable to compute the whole regularization path for this method, especially for high-dimensional problems.

To examine the influence of the regularization parameter, we apply the graphical LASSO method with different values of this parameter for model estimation. The model learning times and the corresponding NLL values computed for this method are shown in Fig. 10. Models are estimated from populations with increasing sizes, generated from the bivariate reference model by decreasing the n/N ratio from 10 to 0.2 (gradually getting away from high-dimensionality) in order to also investigate the effect of population size on this method. NLL values are computed using populations of size 1000 generated from the estimated models. All of the results are averaged over 30 independent runs.

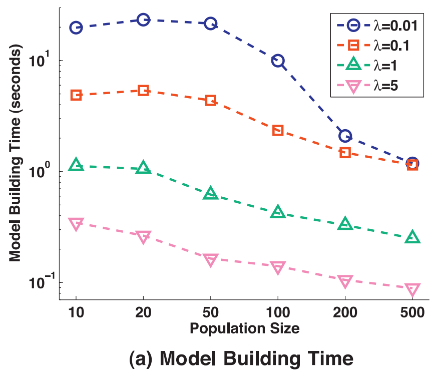
Clearly, whereas model estimation is faster with larger values of the regularization parameter, the solutions generated from these models are less likely from the reference model point of view. On the other hand, small values of this parameter (close to zero) will result in better NLL values, though at a higher computational cost. This observation suggests that, as reported in a previous study [47], the choice of a value for the regularization parameter of this method call for a trade-off between how good an estimation is and the time required to estimate the model.

The model learning times also show that, as population size grows the time required by the graphical LASSO method with a specific value of the regularization parameter gradually decreases. This is why it is expected to be more costly to estimate a model, with a specific dimension and block size, from populations with larger sizes. One possible explanation for this behavior is that with larger populations that better fulfill the sparsity assumption [38], more harmonious statistics can be collected from the population, allowing the method to converge faster. However, as the computed NLL values show, the population size increase does not affect the probability of the generated solutions from the reference model point of view. Based on these results, a 0.1 value is used for the regularization parameter of the graphical LASSO method throughout this paper.

3.2. RegEDA: an EDA based on regularized model estimation

So far we have studied some of the properties of the regularized model estimation methods in two approaches. The results of these analyses can be used as a guideline for employing these methods in EDAs. The regularized EDA (RegEDA) proposed in this paper utilizes the regularized model estimation methods in the course of optimization, trying to obtain a better estimation of the distribution of the problem solutions, in order to improve performance.

The constraints on time and computational resources, restrict the number of different regularized estimation methods that can



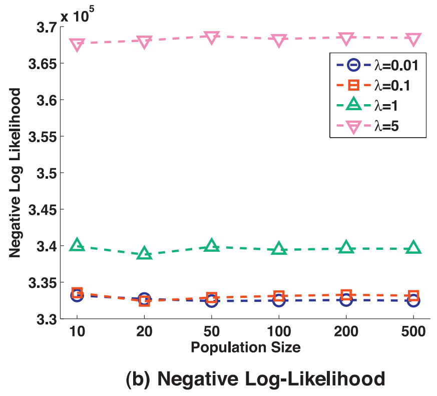


Fig. 10. Average model-building time (left) and negative log-likelihood of the bivariate Gaussian distribution (right) obtained for the graphical LASSO method with different regularization parameter (λ) values.

be tested and compared. For the experiments in the rest of the paper, some of the methods discussed in the previous sections were selected for use in RegEDA. We selected the LARS-OR and LARS-AND methods from the first approach. They appear to strike a better compromise between the computational time requirements and the quality of the estimated models, from both the structural accuracy and NLL points of view. The resulting algorithms are called "RegEDA-LARS-OR" and "RegEDA-LARS-AND", respectively. From the second approach, both the shrinkage estimation and graphical LASSO methods were selected to respectively build the "RegEDA-Shr" and "RegEDA-GL" algorithms. The properties of both these methods merit further investigation regarding optimization. In the next section, we examine the performance of these four versions of RegEDA in function optimization.

4. Experiments

In this section, the proposed RegEDAs are applied for continuous function optimization in order to investigate how regularized model estimation affects the optimization behavior and performance of EDAs when applied in a high-dimensional setting. The optimization results of these four versions of RegEDA are compared against another four Gaussian distribution-based EDAs. These four algorithms are:

- Continuous Univariate Marginal Distribution Algorithm (UMDA) [30].
- Estimation of Gaussian (Bayesian) Network Algorithm (EGNA) [30].
- Estimation of Multivariate Normal (distribution) Algorithm (EMNA) [2].
- Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [69].

4.1. Implementation details

All of the algorithms are implemented in Matlab®. The implementation of the LARS technique in RegEDA-LARS-OR and RegEDA-LARS-AND algorithms, is provided by K. Sjöstrand.[1] The implementations of the covariance shrinkage method and the graphical LASSO algorithm are provided by K. Murphy[2] and by H. Karshenas,[3] respectively.

The Matlab® implementations of UMDA and EMNA provided in MATEDA-2.0[4] [70] are used in the experiments. EGNA is implemented using the Gaussian–Bayesian network learning code provided by M. Schmidt and K. Murphy.[5] Since the PLS algorithm is used to sample the learnt Bayesian network, the algorithm is referred to as "EGNA-PLS" in the results presented in this section. Finally, the implementation of CMA-ES is provided by N. Hansen.[6]

4.2. Functions

The continuous optimization functions used for the experiments in this section are listed in Table 2. These optimization functions, defined on n input variables, have different properties that makes it possible to examine the performance of the tested optimization algorithms, in the presence of different problem features. The 2D fitness landscape of some of these functions, is shown in Fig. 11. The definitions of the functions are as follows.

Table 2
The optimization functions used in the experiments, their optimum solution ($\boldsymbol{x}^*$) and optimum function value (f^*). The number of variables is denoted with n.

	Name	Type	Domain	$\boldsymbol{x}^*$	f^*
1	Sphere	min	$[-5, 5]^n$	**0**	0
2	Ackley	min	$[-32, 32]^n$	**0**	0
3	Tablet	min	$[-7.5, 7.5]^n$	**0**	0
4	Cigar-Tablet	min	$[-7.5, 7.5]^n$	**0**	0
5	Michalewicz	min	$[0, \pi]^n$	–[a]	–[a]
6	Sum Cancellation	max	$[-0.16, 0.16]^n$	**0**	10^5

[a] Depends on n.

- Sphere

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2.$$

- Ackley

$$f(\boldsymbol{x}) = -a\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + e,$$

where a, b and c are the parameters of the function and are set to 20, 0.2 and 2π, respectively. e is Euler's number.

- Tablet

$$f(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2.$$

- Cigar-Tablet

$$f(\boldsymbol{x}) = x_1^2 + 10^4\sum_{i=2}^{n-1} x_i^2 + 10^8 x_n^2.$$

- Michalewicz

$$f(\boldsymbol{x}) = -\sum_{i=1}^{n} \sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right),$$

where m is the parameter of the function and is set to 10. The optimum value of the function is different for different numbers of variables.

- Sum Cancellation

$$f(\boldsymbol{x}) = \left(10^{-5} + \sum_{i=1}^{n}\left|\sum_{j=1}^{i} x_j\right|\right)^{-1}.$$

The Sphere function is a simple optimization problem without any interdependency between variables. Following the smooth downhill path will lead an optimization algorithm to the optimal solution of the function. The Ackley function has a rugged landscape, although some local regions of the search space can provide information about the global structure of the problem. The first

[1] http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3897/zip/imm3897.zip.
[2] http://www.uni-leipzig.de/~strimmer/lab/software/m-files/covshrink-kpm.zip.
[3] http://cig.fi.upm.es/components/com_phocadownload/container/GraphicalLasso.zip.
[4] http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/MATEDA.html.
[5] http://www.cs.ubc.ca/ murphyk/Software/DAGlearn/DAGLearn.zip.
[6] http://www.lri.fr/ hansen/cmaes_inmatlab.html.

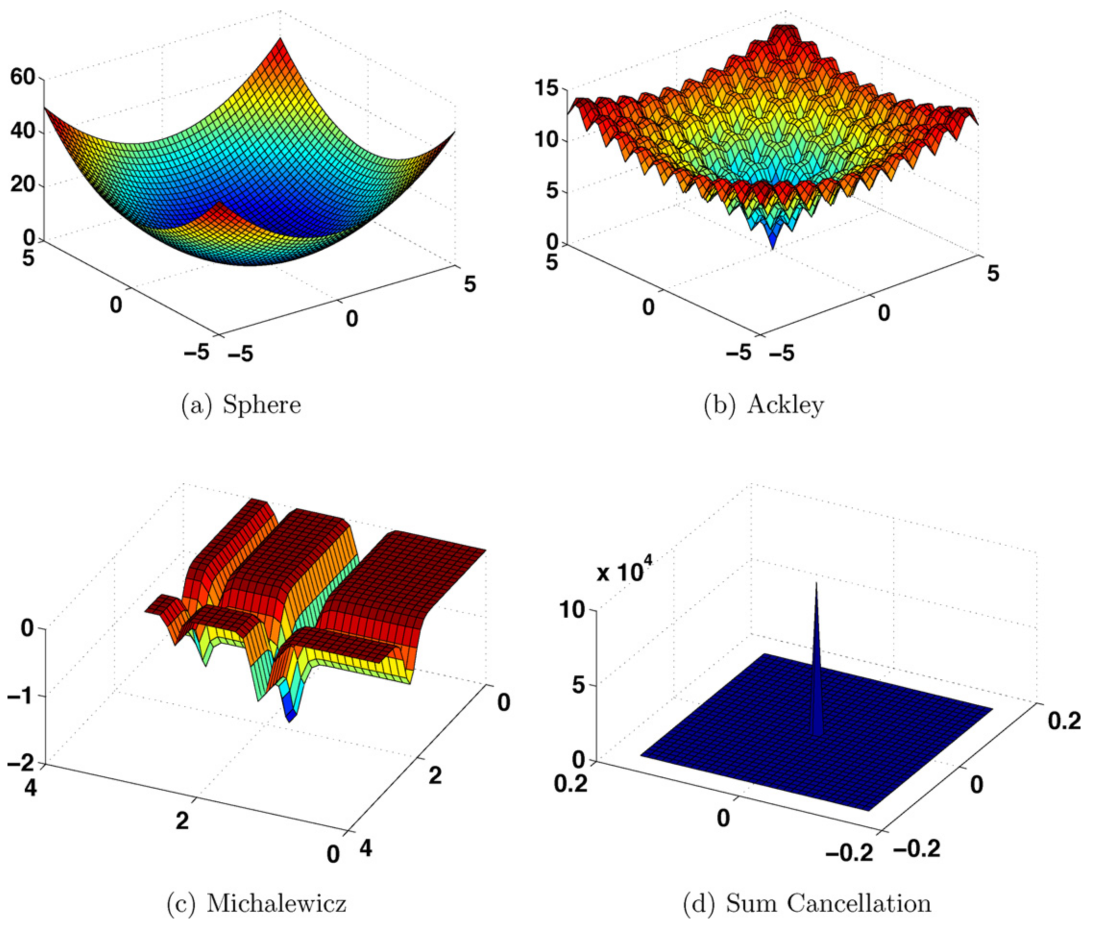


Fig. 11. Fitness landscape of some of the optimization functions in two dimensions.

variable of the Tablet function is scaled causing the optimization algorithms to be more sensitive to changes of this variable, and therefore the promising values of other variables may not be properly encoded in EDA model estimation. The Cigar-Tablet function extends the Tablet function by introducing three different levels of scalings for the variables. The Michalewicz function does not have a proper global structure and requires more exploration for detecting promising basins of attraction. The Sum Cancellation function is very similar to a needle in the haystack problem, especially for larger dimensions, and the function is not separable. Therefore, a very small search space is considered for this function.

4.3. Experimental design

Five different dimensions are tested for each of the functions: 10, 20, 50, 100 and 200 variables. Population size is set to $N = 10\ln(n)$ for all of the algorithms in an attempt to emulate high-dimensional settings as the number of variables increase, starting with a minimum number of solutions. For each algorithm-function-dimension combination, 20 independent runs are performed. The initial population is randomly generated using a uniform distribution over the domain of variables. All algorithms terminate when the maximum number of generations, set to 500, is reached. Apart from this stopping criterion, when an algorithm gets stuck in a stalemate situation for 100 consecutive generations, it stops so as not to waste computational time. A stalemate situation is verified if the improvement in the fitness function is less than 10^{-8}. Fig. 12 gives an insight into the computational time requirements of the main steps of RegEDAs, which are compared with three other Gaussian-based EDAs used in the experiments of this section. The presented results are averaged over 500 generations of a run.

For all EDAs, solutions are selected using truncation selection with a $\tau = 0.5$ threshold. Except for CMA-ES, which completely replaces the population in each generation with new solutions, all other algorithms generate $N/2$ new offspring solutions in the sampling step. The newly generated solutions are repaired using a simple repairment strategy, where unacceptable values are replaced by a new value randomly chosen from the domain of the respective variable. An elitist replacement strategy is used to incorporate the offspring solutions into the population, where the best N solutions from the combination of offspring solutions and solutions in the population are selected to form the next-generation population. The initial standard deviation of CMA-ES is set to one third of each variable's domain, as it is suggested in [69]. This algorithm is

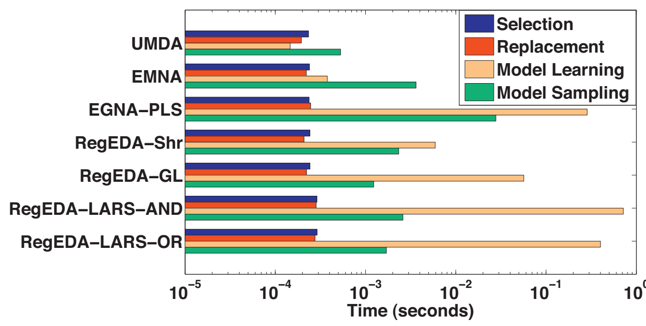


Fig. 12. Average time requirements for the main steps of RegEDAs and their comparison with other EDAs. The number of variables is 50.

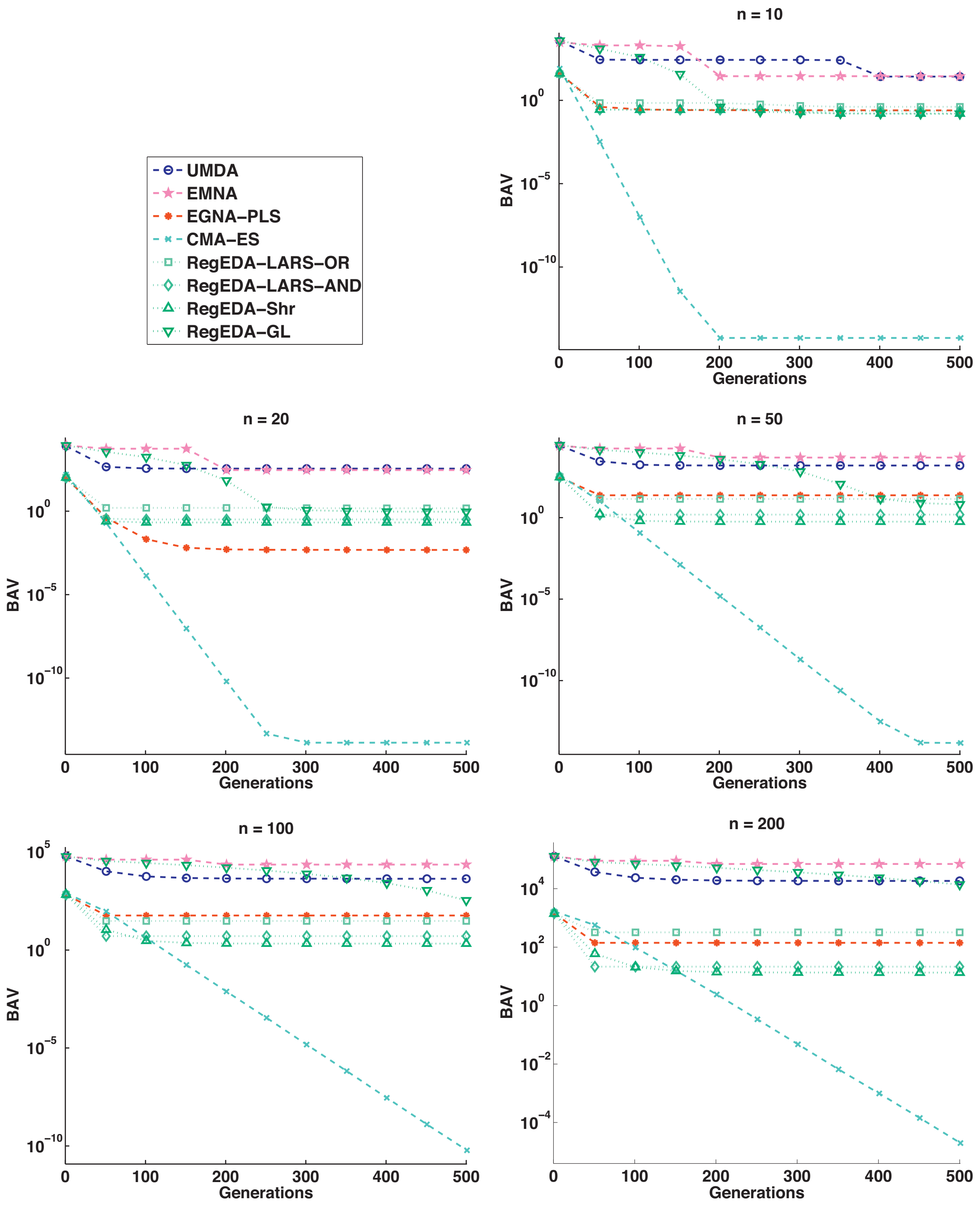


Fig. 13. Average BAVs for Sphere function.

considered to be in a stalemate situation if the improvement in the fitness function is less than 10^{-12} (which is less strict than others).

4.4. Results

Figs. 13–18 show the average best achieved values (BAVs) along the evolution path of RegEDAs and the other four EDAs, applied to the optimization functions. The presented results are averaged over the 20 runs performed. For runs that an algorithm terminates before reaching the maximum number of generations, the rest of evolution path is padded with the BAV of the last executed generation. For some of the functions, the BAVs are depicted on a logarithmic scale so as to better discriminate their performances.

The results for all functions show that, when considering the distance between BAVs and the optimal function value, the performance of all algorithms drops as the number of variables increases. However, this curse of dimensionality affects some algorithms (like CMA-ES and EGNA-PLS) a lot more than others. The optimization behavior of RegEDAs for most of the tested functions, suggests that these algorithms are less affected by this phenomenon.

The comparison of ML estimation in EMNA and UMDA with the regularized model estimation in RegEDAs better illustrates the difference in the performance of RegEDAs and how they are

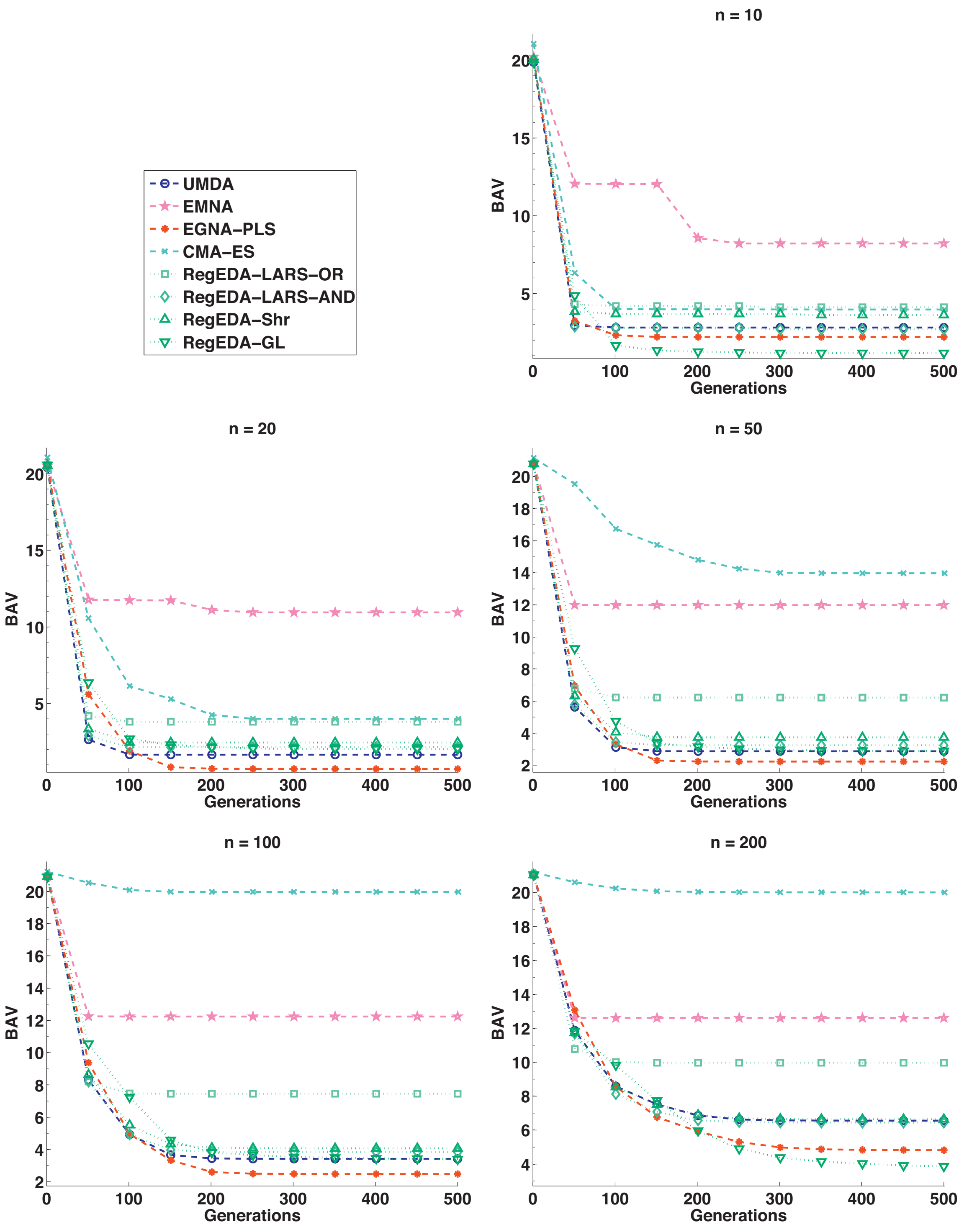


Fig. 14. Average BAVs for Ackley function.

affected by increases in the problem size. Since all other parts of the tested algorithms are the same (except for CMA-ES), the similarities and differences in the optimization performance of these algorithms can be attributed to the model estimation methods that they employ. For example, the performance of RegEDA-Shr and UMDA is very close for most of the functions, suggesting that the shrinkage estimation method is shrinking most of the off-diagonal entries in the covariance matrix to close-to-zero values. This property is especially useful when dealing with separable optimization problems. A comparison of the performances of RegEDA-Shr, and UMDA and EMNA for Sphere functions (Fig. 13), clearly shows that the model estimation employed in RegEDA-Shr is more efficient. This leads to a regularized combination of the models used in EMNA and UMDA.

The conservative merging strategy used in RegEDA-LARS-AND model estimation causes fewer dependencies to be added to the model, leading to sparser structures. The fact that this algorithm behaves similarly to UMDA, and therefore RegEDA-Shr, for many of the functions, shows that the sparsity pattern of these structures is very similar to diagonal matrices in the presence of problem

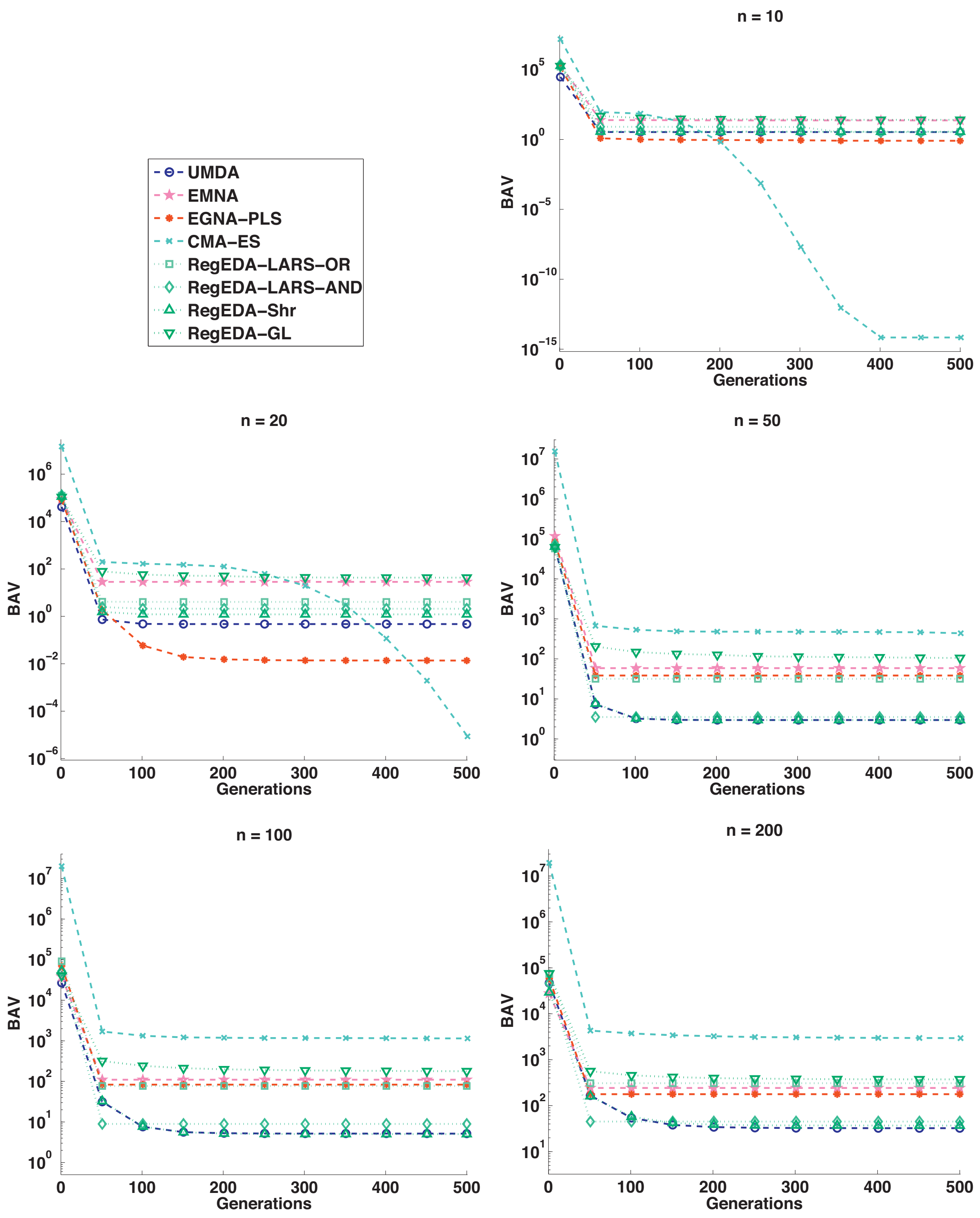


Fig. 15. Average BAVs for Tablet function.

separability. When the problem is not separable (like the Sum Cancellation function – Fig. 18), RegEDA-LARS-AND can, thanks to this regularized model estimation method, perform a more concentrated search by including only what are, according to the regularization technique, the more important links. Therefore this algorithm is able to outperform other algorithms with the increase in problem dimensionality.

The optimization performance of RegEDA-GL compared with EMNA is especially interesting in these experiments. The model estimation method in the two algorithms differs only as to the regularization term added to ML estimation of MGD in the graphical LASSO method. The results show that, for some of the functions (like Sphere and Ackley – Figs. 13 and 14), regularization improves the average performance of RegEDA-GL while, for some others (Michalewicz and Sum Cancellation – Figs. 17 and 18), this algorithm is not able to obtain such good BAVs as EMNA. Considering the properties of these functions, it appears that the use of regularization will result in better model estimation if the problem has a global structure, whereas lack of such information has a negative effect on the regularized model estimation used in RegEDA-GL.

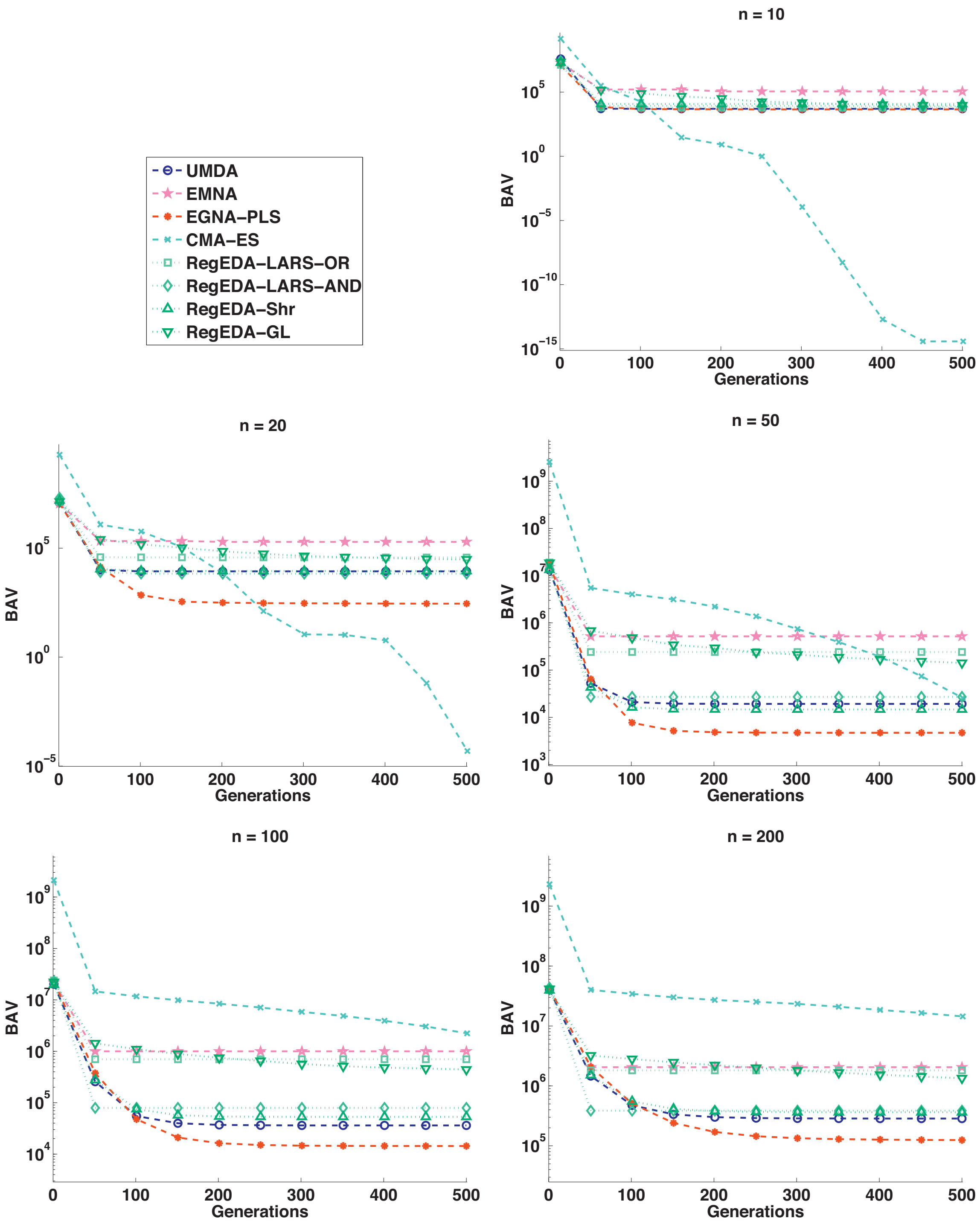


Fig. 16. Average BAVs for Cigar-Tablet function.

The similar optimization behavior of RegEDA-LARS-OR and EMNA for many of the functions suggests that the MGDs estimated with the greedy strategy employed in RegEDA-LARS-OR are very similar to those obtained by ML estimation. For example, a good option, in the absence of useful properties in some of the functions (like Michalewicz – Fig. 17), for use by the optimization algorithm, could be a coarse estimation of the search space. Model estimation in RegEDA-LARS-OR simulates this coarse estimation by adding many links to the structure considering all possible variable dependencies. However, the use of regularization to detect the dependencies, in the presence of specific problem properties (like separability) causes the algorithm to perform a finer search.

4.5. Discussion

There are some common points concerning the performance of the algorithms on the tested functions. First, most of the algorithms show a different optimization behavior from one function to another depending on the different features of these functions. This suggests that although no explicit rotation and translation is applied, the set of selected functions form a good benchmark for

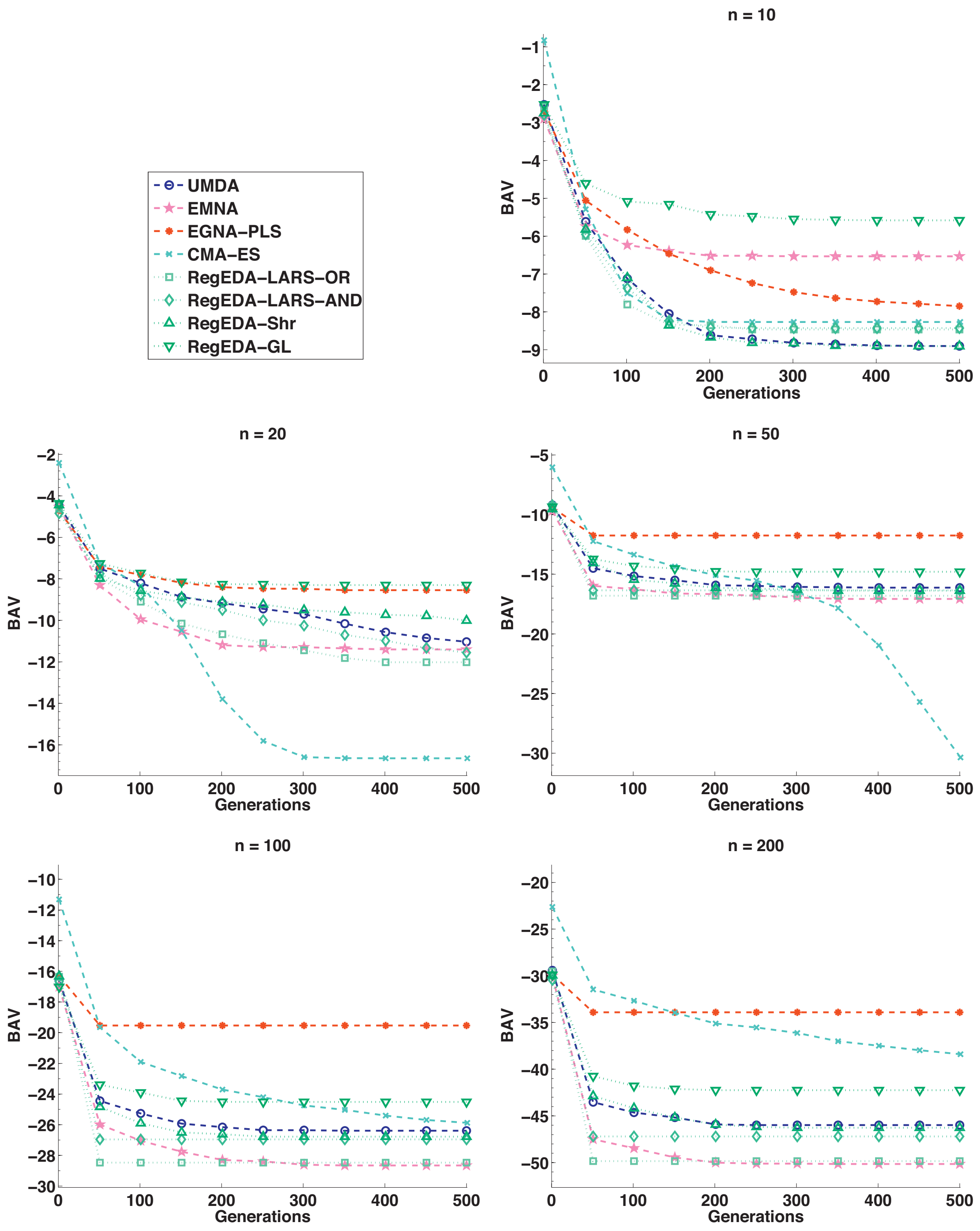


Fig. 17. Average BAVs for Michalewicz function.

the experiments. On the other hand, the Gaussian distribution-based model estimation used in the algorithms is invariant to these transformations. Second, the performance of all the algorithms is affected by the increase in the number of variables, from the viewpoint of the distance between BAVs and optimal function value. Some of the algorithms, like CMA-ES, appear to be able to improve their BAVs if given more time, but it is evident from the results that their performance is influenced considerably more than the RegEDAs proposed in this paper, when the number of variables increase.

Thirdly, the population size of all algorithms is equally set to be of a logarithmic order of the problem size, resembling a high-dimensional setting. Thus, for most of the functions and especially larger dimensions, almost all the algorithms get stuck in a stalemate situation in the early generations of the search. For the purpose of studying the effect of regularized model estimation, none of the algorithms (except CMA-ES, which uses specific variance scaling techniques) use any kind of explicit diversity preservation techniques. The fact that all other parts of the algorithms are the same was helpful for gaining a better understanding

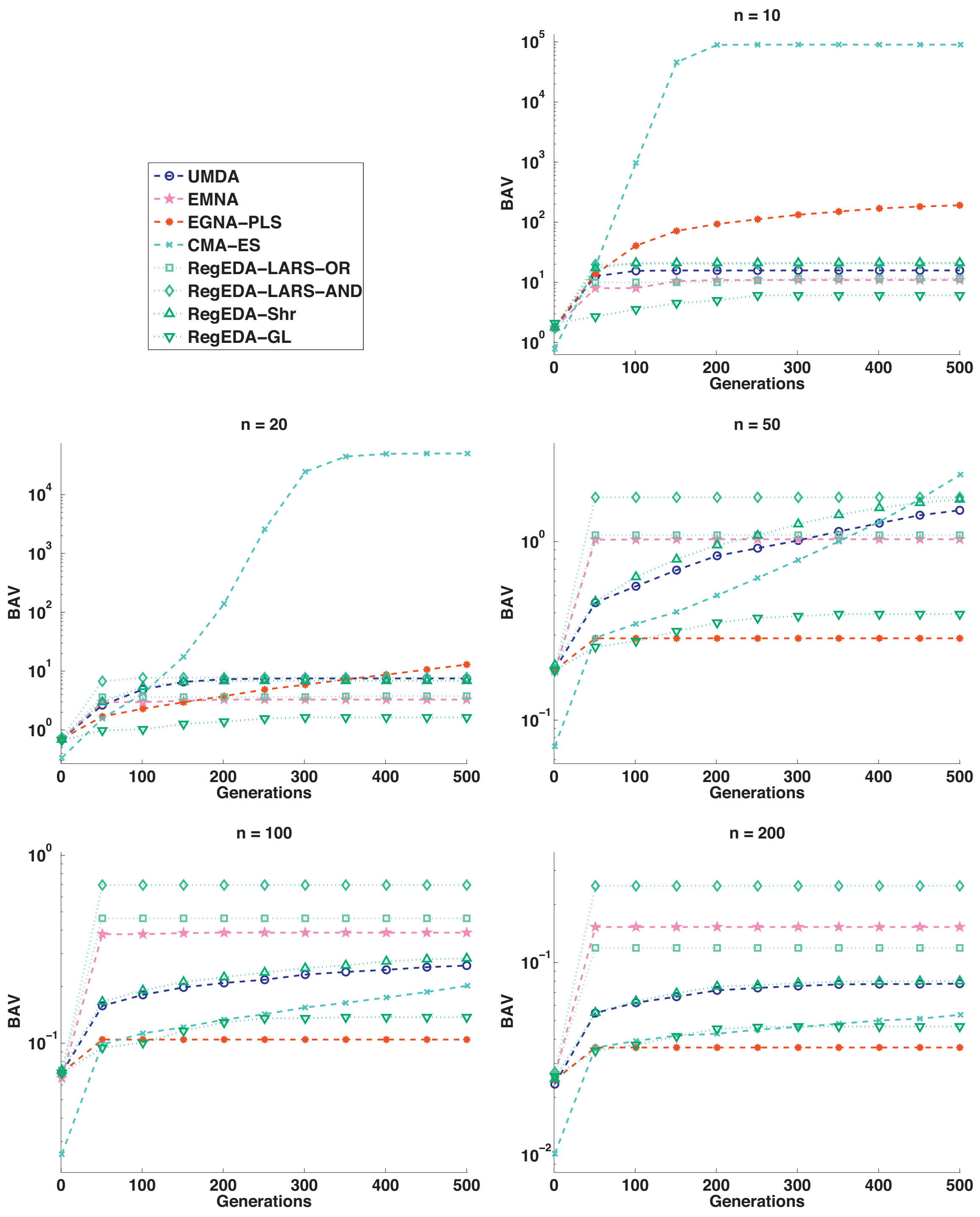


Fig. 18. Average BAVs for Sum Cancellation function.

of how different model estimation methods affect the optimization behavior.

Table 3 shows the statistical analysis results for the BAVs obtained by the algorithms on each dimension of each function. The non-parametric Friedman rank test [71] is used to check for the statistical difference in algorithm performance. The null hypothesis that all the algorithms have an equal average rank is rejected with a p-value less than 10^{-8} for all functions and all dimensions. The values shown in each entry of Table 3 are the final average BAVs obtained by the algorithm in the column, applied on the function-dimension in the row. The numbers in parentheses show the results of pairwise comparisons using Bergmann-Hommel's procedure as the post hoc test. The significance level for this test is set to 0.05. The first number shows how many algorithms are significantly worse than the algorithm in each column, and the second number shows how many algorithms are significantly better.

The results of statistical tests are evidently consistent with the average algorithm performance. For example, RegEDA-Shr and UMDA do not have a statistically different performance for most of the functions. Also whereas, CMA-ES is able to obtain significantly better BAVs for smaller dimensions of the functions, it rapidly becomes less proficient as the number of variables increases and

Table 3
The results of statistical tests on BAVs of the algorithms (refer to text for more explanation).

Function	Dim.	UMDA	EMNA	EGNA-PLS	CMA-ES	RegEDA-LARS-OR	RegEDA-LARS-AND	RegEDA-Shr	RegEDA-GL
Sphere	10	2.67e+01 (0, 6)	2.81e+01 (0, 6)	2.50e−01 (2, 1)	5.53e−15 (7, 0)	4.10e−01 (2, 1)	2.31e−01 (2, 1)	1.61e−01 (2, 1)	1.55e−01 (2, 1)
	20	3.58e+02 (0, 5)	2.90e+02 (0, 5)	4.76e−03 (4, 0)	1.33e−14 (6, 0)	1.50e+00 (0, 4)	3.22e−01 (3, 1)	2.10e−01 (3, 1)	9.15e−01 (2, 2)
	50	1.58e+03 (0, 4)	4.88e+03 (0, 5)	2.35e+01 (0, 3)	1.50e−14 (5, 0)	1.40e+01 (1, 2)	1.53e+00 (3, 0)	5.66e−01 (4, 0)	6.51e+00 (2, 1)
	100	4.35e+03 (0, 4)	2.30e+04 (0, 5)	5.78e+01 (1, 3)	6.02e−11 (5, 0)	3.05e+01 (2, 1)	5.13e+00 (4, 0)	2.12e+00 (4, 0)	3.44e+02 (0, 3)
	200	1.84e+04 (0, 5)	6.99e+04 (0, 5)	1.41e+02 (2, 2)	1.98e−05 (5, 0)	3.21e+02 (2, 2)	2.15e+01 (3, 0)	1.35e+01 (5, 0)	1.38e+04 (0, 3)
Ackley	10	2.81e+00 (1, 0)	8.22e+00 (0, 6)	2.20e+00 (2, 0)	3.97e+00 (3, 0)	4.12e+00 (0, 3)	2.70e+00 (1, 0)	3.62e+00 (1, 2)	1.17e+00 (3, 0)
	20	1.65e+00 (2, 0)	1.09e+01 (0, 6)	7.27e−01 (2, 0)	3.99e+00 (2, 0)	3.81e+00 (0, 3)	2.15e+00 (1, 0)	2.45e+00 (1, 0)	2.01e+00 (1, 0)
	50	2.87e+00 (3, 0)	1.20e+01 (0, 5)	2.22e+00 (4, 0)	1.40e+01 (0, 4)	6.22e+00 (0, 4)	3.24e+00 (3, 0)	3.73e+00 (1, 1)	2.89e+00 (3, 0)
	100	3.42e+00 (3, 0)	1.22e+01 (0, 5)	2.48e+00 (5, 0)	2.00e+01 (0, 5)	7.46e+00 (0, 4)	3.84e+00 (3, 1)	4.07e+00 (2, 1)	3.39e+00 (3, 0)
	200	6.56e+00 (2, 2)	1.26e+01 (0, 5)	4.82e+00 (5, 0)	2.00e+01 (0, 5)	9.97e+00 (0, 3)	6.45e+00 (3, 1)	6.64e+00 (2, 2)	3.87e+00 (6, 0)
Tablet	10	3.40e+00 (2, 1)	2.32e+01 (0, 6)	8.07e−01 (2, 0)	6.84e−15 (6, 0)	3.55e+00 (2, 1)	3.40e+00 (2, 1)	3.62e+00 (2, 1)	2.55e+01 (0, 6)
	20	4.73e−01 (2, 1)	2.84e+01 (0, 5)	1.36e−02 (5, 0)	8.63e−06 (6, 0)	4.07e+00 (1, 2)	2.10e+00 (2, 2)	1.21e+00 (2, 2)	4.30e+01 (0, 6)
	50	2.96e+00 (5, 0)	5.88e+01 (0, 3)	3.86e+01 (2, 3)	4.39e+02 (0, 5)	3.18e+01 (2, 3)	3.53e+00 (5, 0)	2.97e+00 (5, 0)	1.05e+02 (0, 5)
	100	5.17e+00 (5, 0)	1.10e+02 (1, 3)	8.32e+01 (2, 3)	1.15e+03 (0, 6)	7.69e+01 (2, 2)	8.97e+00 (4, 0)	5.04e+00 (5, 0	1.78e+02 (0, 5)
	200	3.23e+01 (5, 0)	2.43e+02 (1, 3)	1.78e+02 (2, 2)	2.95e+03 (0, 6)	3.08e+02 (1, 3)	4.51e+01 (4, 0)	3.69e+01 (5, 0)	3.74e+02 (0, 4)
Cigar-Tablet	10	5.01e+03 (1, 1)	1.13e+05 (0, 6)	4.53e+03 (1, 1)	3.77e−15 (7, 0)	6.04e+03 (1, 1)	7.80e+03 (1, 1)	1.20e+04 (1, 1)	9.46e+03 (0, 1)
	20	8.61e+03 (3, 1)	1.95e+05 (0, 5)	2.85e+02 (4, 0)	4.91e−05 (6, 0)	3.72e+04 (0, 4)	6.75e+03 (3, 1)	8.98e+03 (1, 2)	3.05e+04 (0, 4)
	50	1.93e+04 (3, 0)	5.17e+05 (0, 5)	4.71e+03 (5, 0)	2.57e+04 (3, 1)	2.41e+05 (0, 5)	2.71e+04 (3, 1)	1.47e+04 (3, 0)	1.42e+05 (0, 5)
	100	3.61e+04 (4, 0)	1.00e+06 (0, 4)	1.44e+04 (5, 0)	2.24e+06 (0, 5)	7.00e+05 (0, 4)	7.93e+04 (3, 1)	5.30e+04 (3, 0)	4.42e+05 (1, 2)
	200	2.85e+05 (4, 0)	2.05e+06 (0, 4)	1.25e+05 (6, 0)	1.44e+07 (0, 5)	1.82e+06 (0, 4)	3.86e+05 (3, 1)	3.64e+05 (3, 1)	1.34e+06 (1, 2)
Michalewicz	10	−8.90e+00 (2, 0)	−6.53e+00 (0, 5)	−7.85e+00 (1, 0)	−8.27e+00 (2, 0)	−8.47e+00 (2, 0)	−8.43e+00 (2, 0)	−8.91e+00 (2, 0)	−5.58e+00 (0, 6)
	20	−1.10e+01 (2, 1)	−1.14e+01 (2, 1)	−8.54e+00 (0, 5)	−1.66e+01 (7, 0)	−1.20e+01 (2, 1)	−1.16e+01 (2, 1)	−1.00e+01 (0, 1)	−8.30e+00 (0, 5)
	50	−1.61e+01 (2, 1)	−1.71e+01 (2, 1)	−1.18e+01 (0, 6)	−3.04e+01 (6, 0)	−1.68e+01 (2, 0)	−1.63e+01 (2, 1)	−1.64e+01 (2, 1)	−1.48e+01 (0, 6)
	100	−2.64e+01 (2, 2)	−2.87e+01 (5, 0)	−1.95e+01 (0, 6)	−2.59e+01 (1, 2)	−2.85e+01 (5, 0)	−2.70e+01 (2, 0)	−2.68e+01 (2, 2)	−2.45e+01 (0, 5)
	200	−4.60e+01 (2, 2)	−5.01e+01 (5, 0)	−3.39e+01 (0, 5)	−3.84e+01 (0, 5)	−4.98e+01 (5, 0)	−4.72e+01 (3, 0)	−4.63e+01 (2, 2)	−4.22e+01 (0, 3)
SumCan	10	1.58e+01 (1, 2)	1.09e+01 (0, 2)	1.91e+02 (4, 0)	9.00e+04 (6, 0)	1.16e+01 (1, 2)	2.05e+01 (1, 1)	2.11e+01 (1, 1)	6.07e+00 (0, 6)
	20	7.51e+00 (3, 0)	3.28e+00 (0, 5)	1.29e+01 (3, 0)	5.00e+04 (3, 0)	3.76e+00 (0, 5)	7.74e+00 (3, 0)	6.83e+00 (3, 0)	1.63e+00 (0, 5)
	50	1.50e+00 (2, 0)	1.03e+00 (1, 3)	2.87e−01 (0, 6)	2.38e+00 (4, 0)	1.09e+00 (1, 3)	1.77e+00 (4, 0)	1.73e+00 (4, 0)	3.93e−01 (0, 4)
	100	2.59e−01 (2, 2)	3.88e−01 (3, 0)	1.05e−01 (0, 6)	2.02e−01 (1, 3)	4.62e−01 (5, 0)	6.94e−01 (5, 0)	2.83e−01 (2, 2)	1.38e−01 (0, 5)
	200	7.77e−02 (2, 1)	1.53e−01 (3, 0)	3.63e−02 (0, 6)	5.36e−02 (1, 2)	1.19e−01 (2, 1)	2.50e−01 (6, 0)	8.01e−02 (2, 1)	4.66e−02 (0, 5)

ends up being significantly worse than many other algorithms. For larger dimensions, the statistical test ranks one of the proposed RegEDAs first or second (without any statistical difference from the first ranked algorithm) for all functions, except Sphere. The pairwise statistical comparisons also show that these algorithms are able to obtain statistically better BAVs than most other algorithms. Looking at the performance over all functions, RegEDA-LARS-AND appear to have a better overall performance than other RegEDAs for larger problem sizes.

5. Conclusions

This paper proposed and studied the use of regularized model estimation in EDAs for continuous optimization. It was argued that the use of regularization techniques can lead to a more robust model estimation in EDAs. This improves performance in high-dimensional settings, while keeping the population size and therefore the number of function evaluations relatively low.

Two approaches to regularized estimation of Gaussian distributions were introduced, and several alternative methods proposed in the statistics literature were discussed within each approach. It was shown that an important factor affecting regularized model estimation is the level of variable dependencies in the problem. Considering a high-dimensional setting, the different methods in these approaches were analyzed from several points of view: true structure recovery, time complexity, and likelihood. The results of these analyses helped to select some of the regularized model estimation techniques for use in RegEDA.

These different versions of RegEDA were applied to a set of continuous optimization functions, featuring different properties, and the results were compared with those of other Gaussian-based EDAs. The results show that the increase in problem dimensionality, with a logarithmic population size in the number of variables, affects the performance of the proposed RegEDAs less than other Gaussian-based EDAs. Specific problem properties can play a vital role in algorithm performance. The statistical analysis results show that RegEDAs are able to obtain BAVs that are significantly better than the other algorithms for larger dimensions of most functions.

Of all the versions of RegEDA, RegEDA-LARS-AND and RegEDA-Shr have proved to have a better average optimization behavior, with RegEDA-LARS-AND having statistically better overall performance for larger dimensions. The comparison of the behavior of RegEDA-LARS-OR and RegEDA-GL with EDAs using ML estimation (like EMNA) helped to clarify how the use of regularization can affect the optimization performance of EDAs. A further study of algorithm population diversity (the results are not shown here for brevity) revealed that RegEDA-GL is able to maintain a relatively diverse population along the whole evolution path by using regularized model estimation. We found that this property can help the algorithm to obtain better optimization results in the presence of specific function characteristics.

We mentioned that no explicit diversity preservation technique is used in the proposed RegEDAs for the purpose of studying the effect of regularization. Therefore, a future line of work would be to investigate the optimization performance of RegEDAs that do not get trapped in early generations. Possible choices for this purpose are variance scaling and the incorporation of gradient information, as in CMA-ES. Alternative approaches to regularized model estimation, especially for larger problem sizes, are another potential option for extending this work. A problem decomposition based on regularized learning of the variable dependence structure in order to reduce the computational time required for model estimation and also the application of regularization techniques to estimate other types of continuous probabilistic distributions are possible approaches in this regard.

EDAs have been used for solving many of the optimization tasks in real-world problems. Many of these problems are characterized by high-dimensionality which demands special considerations when applying an optimization algorithm. Problems like feature subset selection in machine learning [7], gene expression analysis in genomics [9,72] and optimal ordering of tables in text organization [73] usually involve optimization over a large number of variables. The use of regularization techniques in the proposed RegEDAs makes them promising methods for this type of problems. Another potential application is when a limited amount of computational resources (e.g. time and memory) is available for an optimization problem, and thus small population sizes should be used during evolution.

Acknowledgements

This work has been partially supported by TIN2010-20900-C04-04, TIN-2010-14931, Consolider Ingenio 2010-CSD 2007-00018, and Cajal Blue Brain projects (Spanish Ministry of Science and Innovation), the Saiotek and Research Groups 2007–2012 (IT-242-07) programs (Basque Government), and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

[1] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions I. B inary parameters, in: H.-M. Voigt, W. Ebeling, I. Rechenberger, H.-P. Schwefel (Eds.), Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV), vol. 114 of Lecture Notes in Computer Science, Springer, 1996, pp. 178–187.
[2] P. Larrañaga, J. Lozano (Eds.), Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Norwell, MA, USA, 2001.
[3] J. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms, vol. 192 of Studies in Fuzziness and Soft Computing, Springer, Secaucus, NJ, USA, 2006.
[4] M. Pelikan, K. Sastry, E. Cantú-Paz (Eds.), Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Studies in Computational Intelligence, Springer, Secaucus, NJ, USA, 2006.
[5] I. Segovia-Domínguez, A. Hernández-Aguirre, E. Villa-Diharce, The Gaussian polytree EDA for global optimization, in: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO'11), ACM, New York, NY, USA, 2011, pp. 69–70.
[6] P. Larrañaga, H. Karshenas, C. Bielza, R. Santana, A review on probabilistic graphical models in evolutionary computation, Journal of Heuristics 18 (5) (2012) 795–819.
[7] I. Inza, P. Larrañaga, R. Etxeberria, B. Sierra, Feature subset selection by Bayesian network-based optimization, Artificial Intelligence 123 (1–2) (2000) 157–184.
[8] I. Inza, P. Larrañaga, B. Sierra, Feature subset selection by Bayesian networks: a comparison with genetic and sequential algorithms, International Journal of Approximate Reasoning 27 (2) (2001) 143–164.
[9] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J. Flores, J. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, P. Larrañaga, A review of estimation of distribution algorithms in bioinformatics, BioData Mining 1 (6) (2008), http://dx.doi.org/10.1186/1756-0381-1-6.
[10] R. Santana, A. Mendiburu, N. Zaitlen, E. Eskin, J.A. Lozano, Multi-marker tagging single nucleotide polymorphism selection using estimation of distribution algorithms, Artificial Intelligence in Medicine 50 (3) (2010) 193–201.
[11] R. Armañanzas, Y. Saeys, I. Inza, M. García-Torres, C. Bielza, Y. van de Peer, P. Larrañaga, Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms, IEEE/ACM Transactions on Computational Biology and Bioinformatics 8 (3) (2011) 760–774.
[12] Y. Zhang, X. Li, Estimation of distribution algorithm for permutation flow shops with total flowtime minimization, Computers & Industrial Engineering 60 (4) (2011) 706–718.
[13] L. Wang, C. Fang, An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem, Computers & Operations Research 39 (2) (2012) 449–460.
[14] S.-H. Chen, M.-C. Chen, Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems, International Journal of Production Economics 141 (1) (2013) 24–33.
[15] J. Sun, Q. Zhang, J. Li, X. Yao, A hybrid estimation of distribution algorithm for CDMA cellular system design, International Journal of Computational Intelligence and Applications 7 (2) (2008) 187–200.
[16] S. Jiang, A. Ziver, J. Carter, C. Pain, A. Goddard, S. Franklin, H. Phillips, Estimation of distribution algorithms for nuclear reactor fuel management optimisation, Annals of Nuclear Energy 33 (11–12) (2006) 1039–1057.

[17] R. Santana, P. Larrañaga, J.A. Lozano, Side chain placement using estimation of distribution algorithms, Artificial Intelligence in Medicine 39 (1) (2007) 49–63.
[18] R. Santana, P. Larrañaga, J. Lozano, Protein folding in simplified models with estimation of distribution algorithms, IEEE Transactions on Evolutionary Computation 12 (4) (2008) 418–438.
[19] R. Sagarna, J.A. Lozano, Scatter search in software testing, comparison and collaboration with estimation of distribution algorithms, European Journal of Operational Research 169 (2) (2006) 392–412.
[20] L. Grosset, R. LeRiche, R. Haftka, A double-distribution statistical algorithm for composite laminate optimization, Structural and Multidisciplinary Optimization 31 (2006) 49–59.
[21] H. Mühlenbein, T. Mahnig, A. Ochoa Rodríguez, Schemata, distributions and graphical models in evolutionary optimization, Journal of Heuristics 5 (2) (1999) 215–247.
[22] M. Pelikan, E. Cantú-Paz, D.E. Goldberg, Bayesian optimization algorithm, population sizing, and time to convergence, in: L.D. Whitley, D.E. Goldberg, E. Cantú-Paz, L. Spector, I.C. Parmee, H.-G. Beyer (Eds.), Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'00), Morgan Kaufmann, 2000, pp. 275–282.
[23] C. González, J. Lozano, P. Larrañaga, Mathematical modelling of UMDAc algorithm with tournament selection: behaviour on linear and quadratic functions, International Journal of Approximate Reasoning 31 (3) (2002) 313–340.
[24] Q. Zhang, H. Mühlenbein, On the convergence of a class of estimation of distribution algorithms, IEEE Transactions on Evolutionary Computation 8 (2) (2004) 127–136.
[25] J. Grahl, S. Minner, F. Rothlauf, Behaviour of UMDA$_c$ with truncation selection on monotonous functions, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05), vol. 3, 2005, pp. 2553–2559.
[26] B. Yuan, M. Gallagher, Convergence analysis of UMDA$_C$ with finite populations: a case study on flat landscapes, in: F. Rothlauf (Ed.), Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09), ACM, New York, NY, USA, 2009, pp. 477–482.
[27] P.A.N. Bosman, J. Grahl, Matching inductive search bias and problem structure in continuous estimation of distribution algorithms, European Journal of Operational Research 185 (3) (2008) 1246–1264.
[28] M. Sebag, A. Ducoulombier, Extending population-based incremental learning to continuous search spaces, in: A. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V), vol. 1498 of Lecture Notes in Computer Science, Springer, London, UK, 1998, pp. 418–427.
[29] P.A.N. Bosman, D. Thierens, Expanding from discrete to continuous estimation of distribution algorithms: the IDEA, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J.M. Guervós, H.-P. Schwefel (Eds.), Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI), vol. 1917 of Lecture Notes in Computer Science, Springer-Verlag, London, UK, 2000, pp. 767–776.
[30] P. Larrañaga, R. Etxeberria, J. Lozano, J. Peña, Optimization in continuous domains by learning and simulation of Gaussian networks, in: A. Wu (Ed.), Proceedings of the Workshop on Optimization by Building and Using Probabilistic Models (OBUPM), Conference on Genetic and Evolutionary Computation (GECCO'00), Morgan Kaufmann, 2000, pp. 201–204.
[31] C. Ahn, R. Ramakrishna, D. 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. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell (Eds.), Proceedings of the Sixth Annual Conference on Genetic and Evolutionary Computation (GECCO'04), vol. 3102 of Lecture Notes in Computer Science, Springer, 2004, pp. 840–851.
[32] B. Yuan, M. Gallagher, On the importance of diversity maintenance in estimation of distribution algorithms, in: H.-G. Beyer, U.-M. O'Reilly (Eds.), Proceedings of the Sixth Annual Conference on Genetic and Evolutionary Computation (GECCO'05), ACM, New York, NY, USA, 2005, pp. 719–726.
[33] P.A.N. Bosman, D. Thierens, Adaptive variance scaling in continuous multi-objective estimation of distribution algorithms, in: H. Lipson (Ed.), Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07), ACM, New York, NY, USA, 2007, pp. 500–507.
[34] P. Pošík, Preventing premature convergenc in a simple EDA via global step size setting, in: G. Rudolph, T. Jansen, S. Lucas, C. Poloni, N. Beume (Eds.), Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN X), vol. 5199 of Lecture Notes in Computer Science, Springer, Berlin, 2008, pp. 549–558.
[35] M. Wagner, A. Auger, M. Schoenauer, EEDA: a new robust estimation of distribution algorithm, Tech. Rep. 5190, Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt, France, 2004.
[36] W. Dong, X. Yao, Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms, Information Sciences 178 (15) (2008) 3000–3023.
[37] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society, Series B (Methodological) 67 (2) (2005) 301–320.
[38] N. Meinshausen, P. Bühlmann, High-dimensional graphs and variable selection with the LASSO, Annals of Statistics 34 (3) (2006) 1436–1462.
[39] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics, second ed., Springer, New York, 2009.
[40] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software 33 (1) (2010) 1–22.
[41] C. Stein, Inadmissibility of the usual estimator for the mean of a multivariate normal distribution, in: J. Neyman (Ed.), Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, 1956, pp. 197–206.
[42] B. Efron, Maximum likelihood and decision theory, Annals of Statistics 10 (2) (1982) 340–356.
[43] J. Yang, H. Xu, Y. Cai, P. Jia, Effective structure learning for EDA via l1-regularized Bayesian networks, in: M. Pelikan, J. Branke (Eds.), Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO'10), ACM, New York, NY, USA, 2010, pp. 327–334.
[44] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA The Bayesian optimization algorithm, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'99), vol. 1, Morgan Kaufmann Publishers, Orlando, FL, USA, 1999, pp. 525–532.
[45] M. Luigi, M. Matteo, V. Gabriele, Introducing ℓ_1-regularized logistic regression in Markov networks based EDAs, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC'11), 2011, pp. 1581–1588.
[46] S. Shakya, DEUM: a framework for an estimation of distribution algorithm based on Markov random fields, Ph.D. Thesis, Robert Gordon University, 2006.
[47] H. Karshenas, R. Santana, C. Bielza, P., Larrañaga, Regularized model learning in estimation of distribution algorithms for continuous optimization problems, Tech. Rep. UPM-FI/DIA/2011-1, Computational Intelligence Group, School of Computer Science, Technical University of Madrid, Madrid, Spain, 2011.
[48] M. Bilodeau, D. Brenner, Theory of Multivariate Statistics, Springer-Verlag, 1999.
[49] J. Schäfer, 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), http://dx.doi.org/10.2202/1544-6115.1175.
[50] A.E. Hoerl, R.W. Kennard, Ridge regression: applications to nonorthogonal problems, Technometrics 12 (1) (1970) 69–82.
[51] R. Tibshirani, Regression shrinkage and selection via the LASSO, Journal of the Royal Statistical Society, Series B (Methodological) 58 (1) (1996) 267–288.
[52] O. Ledoit, M. Wolf, A well-conditioned estimator for large-dimensional covariance matrices, Journal of Multivariate Analysis 88 (2) (2004) 365–411.
[53] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of Statistics 32 (2) (2004) 407–451.
[54] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer-Verlag, New York, 2001.
[55] M. Schmidt, A. Niculescu-Mizil, K. Murphy, Learning graphical model structure using L1-regularization paths, in: Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07), vol. 2, AAAI Press, 2007, pp. 1278–1283.
[56] D. Vidaurre, C. Bielza, P. Larrañaga, Learning an l_1-regularized Gaussian Bayesian network in the equivalence class space, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 40 (2010) 1231–1242.
[57] J. Friedman, T. Hastie, R. Tibshirani, Sparse inverse covariance estimation with the graphical LASSO, Biostatistics 9 (3) (2008) 432–441.
[58] C.L. Mallows, Some comments on C_P, Technometrics 15 (4) (1973) 661–675.
[59] H. Akaike, A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (6) (1974) 716–723.
[60] G. Schwarz, Estimating the dimension of a model, Annals of Statistics 6 (2) (1978) 461–464.
[61] S.L. Lauritzen, Graphical Models, vol. 17 of Oxford Statistical Science Series, Clarendon Press, Oxford, 1996.
[62] D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, C. Kadie, Dependency networks for inference, collaborative filtering, and data visualization, Journal of Machine Learning Research 1 (2001) 49–75.
[63] S. Chaudhuri, M. Drton, T.S. Richardson, Estimation of a covariance matrix with zeros, Biometrika 94 (1) (2007) 199–216.
[64] J. Bien, R.J. Tibshirani, Sparse estimation of a covariance matrix, Biometrika 98 (4) (2011) 807–820.
[65] W. Buntine, Theory refinement on Bayesian networks, in: B. D'Ambrosio, P. Smets (Eds.), Proceedings of the 7th Annual Conference on Uncertainty in Artificial Intelligence (UAI'91), Morgan Kaufmann, San Francisco, CA, USA, 1991, pp. 52–60.
[66] M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling, in: J.F. Lemmer, L.N. Kanal (Eds.), Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence (UAI'86), vol. 2, Elsevier, 1986, pp. 149–163.
[67] S. Kullback, R.A. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1) (1951) 79–86.
[68] L. Le Cam, G. Lo Yang, Asymptotics in Statistics: Some Basic Concepts. Springer Series in Statistics, second ed., Springer, 2000.
[69] N. Hansen, The CMA evolution strategy: a comparing review, in: Lozano et al. [3], pp. 75–102.

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

[71] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[72] C. Bielza, V. Robles, P. Larrañaga, Estimation of distribution algorithms as logistic regression regularizers of microarray classifiers, Methods of Information in Medicine 16 (2008) 345–366.

[73] E. Bengoetxea, P. Larrañaga, C. Bielza, J. Fernández del Pozo, Optimal row and column ordering to improve table interpretation using estimation of distribution algorithms, Journal of Heuristics 17 (5) (2011) 567–588.