Effective Structure Learning for Estimation of Distribution Algorithms via L1-Regularized Bayesian Networks

Hua Xu¹,²,³*, Jiadong Yang¹,²,³, Peifa Jia¹,²,³ and Yi Ding¹,²,³

1 Department of Computer Science and Technology, Tsinghua University, Beijing, China
2 Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing, China
3 State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing, China

* Corresponding author E-mail: xuhua@tsinghua.edu.cn

Received 26 Jul 2012; Accepted 25 Oct 2012

© 2013 Xu et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract Estimation of distribution algorithms (EDAs), as an extension of genetic algorithms, samples new solutions from the probabilistic model, which characterizes the distribution of promising solutions in the search space at each generation. This paper introduces and evaluates a novel estimation of a distribution algorithm, called L1-regularized Bayesian optimization algorithm, L1BOA. In L1BOA, Bayesian networks as probabilistic models are learned in two steps. First, candidate parents of each variable in Bayesian networks are detected by means of L1-regularized logistic regression, with the aim of leading a sparse but nearly optimized network structure. Second, the greedy search, which is restricted to the candidate parent-child pairs, is deployed to identify the final structure. Compared with the Bayesian optimization algorithm (BOA), L1BOA improves the efficiency of structure learning due to the reduction and automated control of network complexity introduced with L1-regularized learning. Experimental studies on different types of benchmark problems show that L1BOA not only outperforms BOA when no prior knowledge about problem structure is available, but also achieves and even exceeds the best performance of BOA that applies explicit controls on network complexity. Furthermore, Bayesian networks built by L1BOA and BOA during evolution are analysed and compared, which demonstrates that L1BOA is able to build simpler, yet more accurate probabilistic models.

Keywords Estimation of Distribution Algorithms, Evolutionary Computation, Bayesian Optimization Algorithm, L1-Penalized Regression, Bayesian Networks

1. Introduction

Estimation of distribution algorithms (EDAs) [1], also called probabilistic model building genetic algorithms (PMBGAs) [2], have been recently identified as a novel paradigm in the field of evolutionary algorithms (EAs). Unlike traditional EAs, such as a simple genetic algorithm (sGA) [3], EDAs replace classical genetic operators (e.g., mutation or crossover) by building and sampling probabilistic models of promising candidate solutions.
EDAs ensure an effective mixing and reproduction of promising partial solutions, thereby solving GA-hard problems with linear or subquadratic performance in terms of fitness function evaluations [4].

According to the modelling of interactions between variables of optimization problems, two families of EDAs have been proposed. The first one, called univariate EDAs, assume that variables in a problem are independent of one another and, hence only a product of independent univariate probabilities is served as the distribution of promising solutions. Univariate EDAs include the univariate marginal distribution algorithm (UMDA) [5], the compact genetic algorithm (CGA) [6], population-based incremental learning (PBIL) [7], etc. This class of EDAs is known for its simplicity and computational efficiency in model learning. However, the inability to solve problems where nonlinear interactions among variables exist prevents those algorithms from solving a broader class of problems. Later developments in EDAs leverage joint distributions of variable pairs or groups to model multivariate interactions and achieve a very good performance on a wide range of decomposable problems. These algorithms include the factorized distribution algorithm (FDA) [8], the extended compact genetic algorithm (ECGA) [9], the estimation of Bayesian network algorithm (EBNA) [10] and the Bayesian optimization algorithm (BOA) [11][12], to name a few.

As one of the state of the art multivariate EDAs, BOA [11][12] uses Bayesian networks [13] to capture the dependencies between the decision variables of the problem. Obviously, the structural accuracy of the Bayesian network learned during each generation significantly impacts the quality of the solutions and the efficiency of the evolution. Unfortunately, it is an NP-hard problem to extract the optimal structure of a Bayesian network given a set of data, regardless of the size of the data [14]. BOA uses a greedy strategy, called score + search, to learn the structure from promising solutions. The strategy performs a search through the space of possible structures with a scoring function, such as Bayesian information criterion (BIC) [15] or the Bayesian - Dirichlet (BD) metric [16], measuring the fitness of each structure. The structure achieving the highest score is then selected. It has been pointed out in [17] that BIC often results in overly simple models and requires large populations to learn a model that captures all necessary dependencies. On the other hand, BD tends to generate an overly complex network due to the existence of noises. Consequently, an additional parameter is added to specify the maximum order of interactions between nodes and to quit structure learning prematurely [12]. As noted in [18], the choice of the upper bound on the network complexity strongly affects the performance of BOA. However, the proper bound value is not always available for black box optimization.

In this paper, a new estimation of distribution algorithm, L1-regularized Bayesian optimization algorithm (L1BOA) is proposed. In spite of relying on Bayesian networks to model distribution of promising solutions, we first utilize L1-regularized logistic regression to identify the potential dependencies between variables firstly. The idea is to apply regression analysis for each variable on the other ones and the regression coefficients with relatively large absolute values are explained as edges in the Bayesian network. Since L1-regularized regression gives exactly zero regression coefficients to most irrelevant or redundant variables [19], it provides a powerful tool for finding the candidate parents of each variable, which leads to a sparse but nearly optimized network structure. Then, greedy search, which is restricted to candidate parent-child pairs, determines the final structure of the network. Due to the excellent properties of L1-regularized learning, L1BOA not only controls the complexity of the network automatically and efficiently, according to the nature of the problem to be solved, but also learns the network structure efficiently with a smaller population size.

To evaluate our new algorithm, two sets of experiments between L1BOA and BOA are performed on different types of test functions, where two cases of BOA, i.e., with and without artificial bounds on network complexity, are considered respectively. The results indicate that the bounds play a key role in the performance of BOA and L1BOA always achieves, even exceeds, the performance of BOA with optimal complexity bounds. Furthermore, we compare the accuracy and the complexity of probabilistic models (i.e., Bayesian Networks) built by L1BOA and BOA based on the perfect models of the test functions. The empirical results of model investigations show that L1BOA is able to build simpler yet more accurate models and there is a positive correlation between performance of the algorithm and accuracy of the model it builds.

The remainder of this paper is organized as follows. In Section 2, some background materials are introduced. Section 3 describes L1BOA in detail. The experimental setup is illustrated in Section 4, followed by Section 5, which compares the performances of L1BOA and BOA. In Section 6, probabilistic models (i.e., Bayesian Networks) built by L1BOA and BOA are investigated. Section 7 discusses the merit of L1BOA. Finally, the paper is summarized in Section 8.

2. Related Work

2.1 Bayesian Networks

Bayesian networks [13] are powerful graphical models that combine probability theory with graph theory to encode probabilistic relationships between variables. A
Bayesian network is defined by its structure and corresponding parameters. The structure is represented by a directed acyclic graph where the nodes correspond to the random variables to be modelled and the edges correspond to conditional dependencies. (Throughout our paper, the terms node and variable are used interchangeably.) The parameters are represented by a set of conditional probability tables (CPTs), which specify the conditional probabilities for each variable with all possible instances of the parent variables. More formally, the factorization of the joint probability distribution in a Bayesian network is defined as

$$ p(X) = \prod_{i=1}^{n} p(X_i | X_{\text{pa}_i}) $$

(1)

where $X = (X_1, X_2, ..., X_n)$ denotes a vector of random variables, $X_{\text{pa}_i}$ refers to the set of parents of $X_i$ and $p(X_i | X_{\text{pa}_i})$ is the conditional probability of $X_i$ given the parents $X_{\text{pa}_i}$.

### 2.2 Bayesian Optimization Algorithm

As one of the multivariate EDAs, the Bayesian optimization algorithm (BOA) [11][12] generates a population of candidate solutions by building and sampling Bayesian networks. After the initialization of a population at random, with a uniform distribution over all possible solutions, the population is then updated for a number of generations. In each generation, BOA works as follows: (1) promising solutions are selected from the current population by means of a selection method, such as tournament or truncation selection; (2) the structure of a Bayesian network that models the population of promising solutions is constructed; (3) the conditional probabilities of the Bayesian network are identified according to promising solutions; (4) the offspring population is generated by sampling the built Bayesian network; (5) replacement scheme is executed to incorporate new solutions into the original population. The above five steps are repeated until some termination criteria are satisfied.

To build a Bayesian network from the promising population, both the structure and the CPTs have to be learned. BOA adopts a greedy strategy to learn the structure of Bayesian networks. More specifically, a certain score of the network is employed to measure the quality of Bayesian network structures. At the beginning, the network is initialized as an empty network with no edges. Then the edge that improves the network score most is added in each step, until no further improvement is possible. Additionally, the number of edges that end in any node is usually limited to some artificial bound to control the complexity of the final structure [12]. After the structure is learned, the calculation of CPT for each variable corresponds to counting the number of times that each possible value of the variable appears, given all possible values of its parents in the promising population.

### 2.3 Graphical Model Learning using L1-Regularization

Traditionally, learning the structure of graphical models is viewed as a combinatorial search problem. In recent years, solving the problem using L1-regularized learning has been proposed [20, 21, 22, 23, 24]. In such a way, the structural learning is embedded within the problem of parameter estimation. To prevent overfitting, regularization is a common technology, which adds an extra term in the regression (linear or logistic) to penalize models using a large number of features. In a variety of regularization technologies, it is known [19] that L1-regularization, in which the penalty term is formed as the absolute values of the parameters, tends to lead to sparse models, where many weights have value zero. Features that have a weight of zero do not induce dependencies between the variables they involve. Consequently, it is able to estimate parameters and select variables simultaneously. Moreover, it has been shown that L1-regularization has a logarithmic sample complexity [25], which means that the number of samples required learning a proper structure grows only in proportion to the logarithm of the problem size. For EDAs that means a small population size for global convergence, as will be shown in our experiments. Therefore, L1-regularization provides a competitive alternative in learning the structure of graphical models.

### 3. Structure Learning for L1BOA

This section presents our method of learning structure for L1BOA. As outlined in Algorithm 1, L1BOA consists of generations including the same five steps as BOA [11][12] (see Section 2.2). The main peculiarity of L1BOA, with respect to BOA, lies in the procedure of learning Bayesian networks by means of L1-regularization (see line 5 in Algorithm 1), which will be discussed in detail as follows.

**Algorithm 1 L1BOA**

1. $t \leftarrow 0$
2. $P_t \leftarrow \text{Generate initial random population}$
3. **repeat**
4. $s \leftarrow \text{Select promising individual in } P_t$
5. $B \leftarrow \text{L1-regularized Bayesian networks learning from } S$
6. \hspace{1cm} Learn the condition probability tables of $B$ from $S$
7. $T \leftarrow \text{Sample from } B$
8. $P_{t+1} \leftarrow \text{Replace } P_t \text{ with } T$
9. $t \leftarrow t + 1$
10. **until** $t \geq t_{\text{max}}$ or population converge

There are two steps in learning the structure of Bayesian networks via L1-regularization. In the first stage, each
variable is applied by L1-regularized logistic regression on the other ones and the non-zero elements in regression coefficients identify the candidate parents of the variable. In the second stage, a greedy search is only restricted to edges in candidate parent-child pairs. Before describing the two steps above, we first discuss how to form the problem of learning the structure of a Bayesian network as a penalized logistic regression problem.

3.1 Problem Setup

The goal of structure learning, given the promising population, is to find the Bayesian network B that maximizes the Bayesian information criterion (BIC) score [15], or equivalently minimizes the minimum description length (MDL) cost [26], defined as

$$MDL(B) = \sum_{i=1}^{n} L(X_i, X_{pa_i}, \theta_i) + \frac{1}{2} \frac{\hat{\theta}_i}{\log N}$$

(2)

where n specifies the number of variables (i.e., the length of the bits representing candidate solutions), N is the size of the training data set (i.e., the size of the promising population), \(L(X_i, X_{pa_i}, \theta_i)\) denotes the negative log-likelihood of variables \(X_i\) with parents \(X_{pa_i}\) and parameters \(\theta_i\), \(\hat{\theta}_i = \arg \min_{\theta_i} L(X_i, X_{pa_i}, \theta_i)\) refers to the maximum likelihood estimate of \(X_i\’s\) parameters and \(|\theta_i|\) is the number of free parameters (i.e., the number of non-zero elements in \(\theta_i\)) in the CPTs for variable \(X_i\). The negative log-likelihood of variables \(X_i\) is defined as

$$L(X_i, X_{pa_i}, \theta_i) = \sum_{x_{pa_i}} \log p(x_i|x_{pa_i}, \theta_i)$$

(3)

Here \(X_i\) and \(x_i\) are the instances of \(X_i = x_i\) and \(\Pi_j = x_j\) in the training data set, respectively. For binary problems, the logistic regression model [27] is usually used to turn linear predictions into probabilities. Given its parents \(X_{pa_i}\), the probability distribution of \(X_i\) is defined as follows.

$$p(x_i | x_{pa_i}, \theta_i) = \sigma(X_i(\theta_i^T X_{pa_i} + \theta_{0}))$$

$$= \frac{1}{1+\exp(-X_i(\theta_i^T X_{pa_i} + \theta_{0}))}$$

(4)

where \(X_i \in \{-1, +1\}\) and \(\theta_i\), which includes bias \(\theta_{0}\), and non-bias term \(\theta_i = (\theta_{i1}, \theta_{i2}, \ldots)\), denote the regression coefficients in the logistic regression model. And \(\sigma(a)\) is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

(5)

To compute the maximum of equation (3), while reducing the variance of the estimates and preventing overfitting, an L1 penalty term is added to form the L1-regularized logistic regression, which is defined as the following.

$$\hat{\theta}_i(\lambda, X_{pa_i}) = \arg \min_{\theta_i} L(X_i, X_{pa_i}, \theta_i) + \lambda |\theta_i|_1$$

(6)

where the parameter \(\lambda > 0\), called regularizer, governs the relative importance of the regularization on the non-bias term of regression coefficients, compared with the likelihood. L1-regularization has the property that if regularizer \(\lambda\) is sufficiently large, some elements in the non-bias term of \(\theta_i(i.e., \theta_i^*)\) are driven to zero, leading to a sparse model in which the corresponding variables in the model play no role [19]. Consequently, such a technique provides a powerful strategy to select candidate parents of each variable from \(X_{pa_i}\).

In equations (4) and (6), the variable \(X_i\) is applied to a logistic regression on the variables in \(X_{pa_i}\). Therefore, the absolute value of each element in a non-bias term of regression coefficients \(\theta_i\) reflects the strength of the dependency between the corresponding variable and \(X_i\). The greater the absolute value, the stronger the dependency. On the other hand, since a larger regularizer makes many of the elements in \(\theta_i\) zero, the set \(X_{pa_i}\), including the variables whose corresponding parameters \(\theta_i\) are non-zero (i.e., \(X_{pa_i} = \{X_j | j \in \varnothing\}\), \(\varnothing = \{z | \theta_z \neq 0, z \neq 0\}\)), could be regarded as the candidate parents of \(X_i\).

3.2 Candidate Parents Detection

From the discussion above, the problem of detecting the candidate parents is shifted to determine a suitable value of the regularizer \(\lambda\). In equation (6), candidate parents of each node could be seen as a function of \(\lambda\). The family of solutions as \(\lambda\) varies over \((0, +\infty)\) forms the regularization path. The stronger regularizer leads to more elements in the non-bias term of \(\theta_i\) zero, thus selecting fewer candidate parents. Figure 1 (a) represents the impact of the regularizer on the value of each element in \(\theta_i\) except bias \(\theta_{0}\), in which the values of different elements are represented by solid lines with different colours. The changes in the number of candidate parents and MDL score as the regularizer varies are represented in Figure 1 (b) and (c) respectively. The vertical dash lines in Figure 1 demonstrate that some elements in the regression coefficient are not equal to zero any more when the lines are lower than the corresponding value on the horizontal axis. In such a situation, it means that some new candidate parents emerge.
A naive approach available is to use the same \( \lambda \) for all nodes. However, such an approach has three drawbacks. First, different problems produce different types of dependencies between variables. Fixing the value of \( \lambda \) weakens the robustness of the algorithm. Second, as the scale of some separative problems increases, the structure of the whole problem changes, even if the dependencies between different variables remain the same. Third, in asymmetrical problems, some variables in real structure may depend on more variables than others. In fact, using a uniform value produces the same disadvantages as the one generated by treating the maximum income edges as a constant in BOA [11][12].

To determine \( \lambda \), cross-validation is the typical approach. However, the process is quite time consuming since the approach would have to be applied on each node. In [21] the number of non-zero elements in non-bias terms of \( \theta_i \) (i.e., the number of candidate parents) is piecewise constant against \( \lambda \), with kinks at each point where any element transitions from zero to non-zero, or vice versa (see Figure 1(b)). Hence, a searching approach with \( n \) steps along the regularization path [21] is utilized in this paper. The procedure begins with \( \lambda = \lambda_0 \), which is the minimum value making all elements in \( \theta_i \) zero except bias \( \theta_0 \). The value of \( \lambda_0 \) is equal to the maximum gradient magnitude when all non-bias parameters are zero (i.e., \( \theta_j = 0 \)), and the bias parameters, \( \theta_0 \), is at its optimal value, since all larger values of \( \lambda \) will have inactive non-bias parameters. Then a fixed step length \( \Delta \lambda = \frac{\lambda_k - \lambda_{k+1}}{n} = \lambda_0 / n \) is taken to decrease the value of \( \lambda \). In step \( k \), the maximum likelihood parameters \( \hat{\theta}_i \) are computed according to the current value of the regularizer \( k \). MDL score could be calculated according to equation (6). If the MDL score is improved, the set of variables, whose corresponding parameters in \( \hat{\theta}_i \) are non-zero, are recorded as the candidate parents of that node. Finally, the set achieving the minimum MDL is treated as the candidate parents.

**Algorithm 2 CandParents(Xi, Xpai)**

1. \( \lambda_0 \leftarrow \) The minimum value making all parameters in \( \hat{\theta}_i \) zero
2. \( \theta_i \leftarrow \arg \min_{\theta_i} L(X_i | X_{pai_i}, \theta_i) + \lambda_n \| \theta_i \|_1 \)
3. \( \text{MDL}_{\min} \leftarrow L(X_i | X_{pai_i}, \hat{\theta}_i) \)
4. \( k \leftarrow 0 \)
5. \( \Delta \lambda = \lambda_0 / n \)
6. while \( k < n \) do
7. \( k \leftarrow k + 1 \)
8. \( \lambda_k \leftarrow \lambda_{k-1} - \Delta \lambda \)
9. \( \theta_i \leftarrow \arg \min_{\theta_i} L(X_i | X_{pai_i}, \theta_i) + \lambda_k \| \theta_i \|_1 \)
10. \( \text{MDL} \leftarrow L(X_i | X_{pai_i}, \hat{\theta}_i) \)
11. if \( \text{MDL} < \text{MDL}_{\min} \) then
12. \( \mathcal{O} = \{ z | \theta_{iz} \neq 0, \theta_{iz} \neq 0 \} \)
13. \( \hat{X}_{pai_i} = [X_i | j \in \mathcal{O}] \)
14. end if
15. end while
16. return \( \hat{X}_{pai_i} \)

As noted before, the number of candidate parents is piecewise constant against \( \lambda \), and increases only at the value of \( \lambda \) where some elements in regression coefficients change from zero to non-zero (see the vertical dash lines in Figure 1). Therefore, the optimal MDL score must be achievable at these points and it is sufficient to check \( O(n) \) values of \( \lambda \) along the path to find the candidate parents of each variable, where \( n \) is the number of nodes in network. Furthermore, \( O(Nn^2) \) is needed for each step in the regularization path to compute maximum likelihood estimates of the logistic regression using iteratively.
reweighted least squares (IRLS) [28]. Hence, the complexity of computing the candidate parents of each node is \(O(Nn^2)\), where \(N\) denotes the size of the training data set. The procedure of finding candidate parents by tuning the regularizer is described in Algorithm 2.

3.3 Pruned Network Search

In equation (6), it is notable that \(X_{pa_i}\) also influences the detection of the candidate parents of each node. There are several different ways to construct \(X_{pa_i}\). One of the choices is to leverage the order search. Using hill climbing to find the best order of all nodes [29], \(X_{pa_i}\) could consist of the precedents before node \(X_i\) according to the ordering. In such a case, the cardinality of \(X_{pa_i}\) is equal to the number of precedents of \(X_i\), thus \(|X_{pa_i}| \leq n-1\). In this paper, we set \(X_{pa_i} = (X_1, X_2, ..., X_{i-1}, X_{i+1}, ..., X_n)\), thus \(|X_{pa_i}| = n-1\). This means that each node applies regression on the others to identify the candidate parents according to the procedure described in Section 3.2.

**Algorithm 3** L1-regularized Bayesian Networks Learning

1. Initialize an empty Bayesian network \(B\)
2. for each variable \(X_i\) in the problem do
3. \(X_{pa_i} \leftarrow (X_1, X_2, ..., X_{i-1}, X_{i+1}, ..., X_n)\)
4. \(X_{pa_i} \leftarrow \text{CandParents}(X_i, X_{pa_i})\)
5. end for
6. \(E \leftarrow \text{All edges within pairs } \{(X_j, X_i) | X_j \in X_{pa_i}\}\)
7. repeat
8. \(e \leftarrow \text{Select the edge from } E \text{ improving MDL most}\)
9. \(E \leftarrow E \setminus \{e\}\)
10. Add \(e\) into \(B\)
11. until MDL(\(B\)) is not improved or \(E\) is empty
12. return \(B\)

To add actual edges into the network, there is another issue to be considered. Since the dependencies may be asymmetric in the finite sample scenario (for example, the edge \(X_1 \rightarrow X_2\) may be found when regressing on \(X_1\) but not on \(X_2\)), two strategies, called ‘OR’ and ‘AND’ had to be chosen. In the former, \(X_j\) is selected as the candidate parent of \(X_i\) if either the dependency \(X_j \rightarrow X_i\) or \(X_j \leftarrow X_i\) is detected. In such a strategy, \(X_2\) is selected as the candidate parent of \(X_1\) and vice versa, since the dependency \(X_1 \rightarrow X_2\) is detected when regressions are applied for \(X_1\). The latter strategy chooses candidate parents when the both directions are identified, which means \(X_2\) is not selected as the candidate parent of \(X_1\), even though the dependency \(X_1 \rightarrow X_2\) is detected. Because the discrepancies between the two strategies are not remarkable [30] the ‘OR’ strategy is used in this paper. The reason lies in that the procedure is to find a set of variables with all the parents, children and co-parents, ideally only discarding false parents and our goal is to identify which edges to leave out, rather than which edges to include. Consequently, the cost of excluding a true edge is higher than including a spurious edge and each edge forms two candidate parent-child pairs to be considered in the greedy search.

Once the candidate parents of each node are known, the greedy search, which is restricted to these node pairs, is employed to make the final decision about which edges to include. Since L1-regularization is proved to hold a parsimonious property [31], the candidate detection usually reduces the search space quite a lot. The details about Bayesian networks structure learning via L1-Regularization are presented in Algorithm 3 and line 4 in Algorithm 3 has been described in Algorithm 2.

4. Experimental Setup

4.1 Test Functions

To compare the performance of LIBOA and BOA, the experiments on four additively decomposable test functions are performed. Function Onemax:

\[ f_{\text{onemax}}(x) = \sum_{i=1}^{n} x_i \]  

(7)

where \(x_i\) represents the \(i\)th bit of the input string with length \(n\). More 1s in the solution are included, the higher the fitness becomes. The optimum, equal to \(n\), is achieved in the string of all ones. In Onemax, there are no dependencies between variables.

Function Link-10:

\[ f_{\text{link10}}(x) = \sum_{i=1}^{n/10} \text{link10}(x_{10i-9}x_{10i-8}...x_{10i}) \]  

(8)

where

\[ \text{link10}(x_{10i-9}x_{10i-8}...x_{10i}) = 4 \sum_{j=1}^{4} t_j^{x_{10i-2j}} \]  

(9)

\[ t_j = \begin{cases} 1, & \text{if } x_{10i-2j} = 1 \text{ or } x_{10i-2j} = 0, \\ 0, & \text{otherwise} \end{cases} \]  

(10)

\[ u = \begin{cases} 1, & \text{if } x_{10i-1} \equiv 2 \mod(4) \text{ and } x_{10i-2j} = 0, \\ 0, & \text{otherwise} \end{cases} \]  

(11)

\[ w = \begin{cases} 10i, & \text{if } x_{10i-2j-1} = 1, \\ 0, & \text{otherwise} \end{cases} \]  

(12)
In Link-10, the input string is first partitioned into independent groups of 10 bits each. Equations (10), (11) and (12) are applied to each group. The contributions of all the groups are added together to form the fitness. The optimum of n-bit Link-10 is 0.6n, which is achieved in the string of all ones. The dependencies between variables in Link-10 are sparse and asymmetrical. More specifically, \( x_{10i} \) and \( x_{10i-1} \) depend on more variables than others in each group. Consequently, Link-10 is just an example of the asymmetrical problem mentioned in Section 3.2.

Function Uniformly Scaled Trap-4 [32]:

\[
 f_{trap_4}(x) = \sum_{i=1}^{n/4} \text{trap}_4(x_{4i-1}x_{4i-2}x_{4i-3}) 
\]

(13)

\[
 \text{trap}_4(u) = \begin{cases} 
 4, & \text{if } u=4 \\
 3-u, & \text{otherwise}
\end{cases}
\]

(14)

where \( u \) is the number of ones in the input string of 4 bits. The input string in uniformly scaled Trap-4 (uTrap-4) is also partitioned into disjointed groups. Each 4-bit group applies Trap-4 function, which is defined in equation (14).

The fitness is the sum of the contributions of each group. The optimum of n-bit uTrap-4, which is equal to \( n \), is achieved in the string of all ones. It necessitates that all bits in each group are treated together, because lower order statistics lead the search away from the global optimum.

Function Exponentially Scaled Trap-4 [33]:

\[
 f_{e_{trap_4}}(x) = \sum_{i=1}^{n/4} 3^i \text{trap}_4(x_{4i-1}x_{4i-2}x_{4i-3}) 
\]

(15)

Similar to uTrap-4, equation (14) is applied to each 4-bit group in exponentially scaled Trap-4 (eTrap-4). Rather than treating each group equally in uTrap-4, the contributions to the fitness of each building block are scaled exponentially. Such a scale gives each group distinguishable importance according to its position. Uniformly scaled functions resemble problems with subproblems of equal importance, while exponentially scaled functions represent those with subproblems of distinguishable importance.

### 4.2 Parameters Settings

In all experiments, both L1BOA and BOA choose truncation selection with \( ps = 0.5 \) as the selection strategy. This means that the best half of the current population is selected as promising individuals. We employ truncation selection because tournament selection leads to the models in BOA to be overly complex [34] and using truncation is much better for leading to accurate structural linkage information [35]. Since the choice of replacement strategy is not relevant to the performance when using truncation selection [35], we employ simple elitist replacement with \( pr = 0.5 \), in which the worst 50% of the population are replaced by newly generated solutions. Forward sampling of Bayesian networks [13] is adopted in both L1BOA and BOA to generate offspring in each generation. In addition, both algorithms are terminated when the population converges on the global optimum (success) or when the upper bound on the number of generations has been reached (i.e., \( t > t_{\text{max}} \) in Algorithm 1) without discovering the global optimum (failure). Unless otherwise specified, the maximum generations \( t_{\text{max}} \) is set as 200. The parameters described above are listed in Table 1.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>Truncation Selection</td>
</tr>
<tr>
<td>Sample</td>
<td>Forward Sample</td>
</tr>
<tr>
<td>Replacement</td>
<td>Elitist Replacement</td>
</tr>
<tr>
<td>Termination</td>
<td>Convergence or ( t &gt; t_{\text{max}} )</td>
</tr>
</tbody>
</table>

**Table 1. Parameters Settings**

5. Performance Comparison

In this section, two sets of experiments are conducted to compare the behaviours of L1BOA and BOA. The BD metric with K2 prior [16] is used to construct the Bayesian network in BOA in both sets, because BIC often results in overly simple models and requires large population to learn a model [17]. In the first set of experiments, the procedure of network structural learning in BOA is terminated only when the score is not improved. Then, the bound on the complexity of networks is treated as an additional criterion to end the network structural learning in BOA. When the number of edges added into the network exceeds the bound, the structural learning stops early. Furthermore, various values of bound are taken into account to investigate how that parameter impacts on the performance of BOA.

5.1 Comparison between L1BOA and BOA without Complexity Control

The performance comparison between L1BOA and BOA without extra network complexity control is represented in this subsection. The problem sizes of three test functions are all set at 40 and population sizes are augmented by the power of 2. Two metrics are calculated to evaluate BOA and L1BOA: (1) the fitness, which is equal to the final result algorithms converging to; (2) the success rate (S.R.), which is set as the ratio of successful runs to total runs. The experimental results are depicted in Figure 2, where fitness and success rate are under the scale of left Y-axis and right Y-axis, respectively. As can be noted from Figure 2, the fitness of L1BOA is higher than that of BOA in all test functions with the same
population size. The gaps in fitness between L1BOA and BOA are remarkable. From the aspect of success rate, BOA is able to find the global optimum of Onemax, but needs a larger population than L1BOA. For function uTrap-4, although BOA could converge to the global optimum sometimes (about 60% of the time), it is not able to ensure the convergence in each run. This means that BOA could not find enough linkages in all the groups. A similar situation emerges with function eTrap-4. As the population rises, the success rate falls, which implies that BOA could not converge on the optimum of eTrap-4. Things are even worse when BOA deals with Link-10. It could hardly find the global optimum of Link-10 for the success rate nearly stays at zero. On the other hand, L1BOA could converge on the global optimum of Link-10 in all the runs with a relatively smaller population.

![Figure 2. Performance comparison between L1BOA and BOA without artificial control on network complexity.](image1)

![Figure 3. Performance comparison on required minimum population size between L1BOA and BOA with network complexity control.](image2)

From the results above, it could be concluded that network complexity control is necessary for BOA in addition to the K2 criteria that are normally used. Without the bound, BOA could not converge on the global optimal when handling problems with dependencies between variables. It is also obvious that L1BOA outperforms BOA without such a bound.

5.2.1 Comparison on Population Requirements

Here, we focus on the population requirements of L1BOA and BOA with different values of k. The ranges of the problem sizes are represented in Table 2. For each case of different functions, the RMPS to ensure the algorithm converges in all 30 runs is determined empirically by a bisection method [12]. Furthermore, the function evaluations are also compared when population sizes are set as the RMPS. The function evaluations of each case are calculated as the average over 30 runs with a 95% confidence interval.

<table>
<thead>
<tr>
<th>Function</th>
<th>Range of Problem Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onemax</td>
<td>(30, 40, 50, 60, 70, 80)</td>
</tr>
<tr>
<td>Link-10</td>
<td>(30, 40, 50, 60, 70, 80)</td>
</tr>
<tr>
<td>uTrap-4</td>
<td>(32, 40, 48, 56, 64, 72)</td>
</tr>
<tr>
<td>eTrap-4</td>
<td>(32, 40, 48, 56, 64, 72)</td>
</tr>
</tbody>
</table>

**Table 2. Problem Sizes of Test Function**

Figure 3(a) shows the empirical results of population requirements of Onemax. Obviously, BOA with k = 0 is able to rely on the lowest RMPS to converge on the optimum in all the instances. The RMPS of L1BOA are a little higher than those of BOA with k = 0, but remarkably they are lower than those in the other cases (e.g., BOA with k = 1, 2, 3). It is also notable that as the value of k increases, a larger population is needed to enable BOA to converge on the optimum. The same situation is observed in Figure 4(a), which represents the function evaluations of L1BOA and BOA on Onemax.
The comparison of the population requirements of Link-10 is depicted in Figure 3(b). The RMPS required by L1BOA are lower than those in all BOA cases. This grows when the value of \( k \) increases within BOA. It is worth pointing out that BOA with \( k = 0 \) needs a population and function evaluations one order of magnitude larger than the other cases to converge on the global optimum. Figure 4(b) demonstrates that the corresponding function evaluations of L1BOA are nearly the same as that of BOA with \( k = 1 \) and the trend of function evaluations of BOA with various bounds are identical to the situation of population requirements. Figure 3(c) and Figure 4(c) depict the experimental results of \( u_{Trap-4} \). BOA with \( k = 0, 1 \) is not able to achieve the global optimum of \( u_{Trap-4} \) since the problem necessitates that bits in each group are treated together. Because each group consists of four bits, \( k = 3 \) allows every bit to link with all the others in the same group. Consequently, only the cases with \( k \) equal to 2 and 3 are taken into consideration. It has been seen that the RMPS and the corresponding function evaluations of L1BOA are equal to those of BOA with \( k = 3 \) overall and are apparently lower than those of BOA with \( k = 2 \).

Groups in \( e_{Trap-4} \) are applied in the same function as \( u_{Trap-4} \). L1BOA is compared with BOA with \( k = 2, 3 \) in \( e_{Trap-4} \) as well. In Figure 3(d), L1BOA relies on the lowest RMPS to converge on the optimum in all cases. Within BOA, the RMPS of case \( k = 2 \) is less than that of \( k = 3 \), which is the opposite of the situation in \( u_{Trap-4} \). The same trend is also observed in Figure 4(d), which represents the function evaluations of L1BOA and BOA in \( e_{Trap-4} \) respectively. This is because the distinguishable importance of each group degrades the average order of statistics necessary to converge on the global optimal.

Furthermore, we want to test the null hypothesis that the number of evaluations of BOA with different complexity bounds are lower than that of L1BOA. If the hypothesis is accepted, it means that BOA outperforms L1BOA. Since the obtained results may present neither normal distribution nor homogeneity of variance, the Wilcoxon paired signed-ranks test [36], which is a non-parametric test, is executed according to the recommendations made in [37]. The results are presented in Table 3, which validates that L1BOA is able to achieve the performance of BOA with optimally controlled bounds on network complexity, when dealing with \( u_{Trap-4} \) and outperforms BOA when dealing with Link-10 and \( p_{Trap-4} \).

Given the results above, it can be concluded that the network complexity control significantly affects the performance of BOA. Different problems requires BOA to set different bounds to achieve its best performance. On the other hand, the performance of L1BOA is almost equal to, sometimes above, the best in all cases of BOA, when dealing with different functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>BOA(( k = 0 ))</th>
<th>L1BOA</th>
<th>( R^+ )</th>
<th>( R^- )</th>
<th>( p )-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( One_{max} )</td>
<td>BOA(( k = 0 ))</td>
<td>L1BOA</td>
<td>21</td>
<td>0</td>
<td>0.0894</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 1 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 2 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 3 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td>( Link-10 )</td>
<td>BOA(( k = 1 ))</td>
<td>L1BOA</td>
<td>3</td>
<td>18</td>
<td>0.0468</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 2 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 3 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td>( u_{Trap-4} )</td>
<td>BOA(( k = 2 ))</td>
<td>L1BOA</td>
<td>4</td>
<td>17</td>
<td>0.0711</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 3 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td>( e_{Trap-4} )</td>
<td>BOA(( k = 2 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
<tr>
<td></td>
<td>BOA(( k = 3 ))</td>
<td>L1BOA</td>
<td>0</td>
<td>21</td>
<td>0.0106</td>
</tr>
</tbody>
</table>

Table 3. Significance tests on numbers of function evaluations. The \( p \)-values are from one sided Wilcoxon paired signed-rank tests.

Figure 4. Performance comparison on the number of function evaluations between L1BOA and BOA with network complexity control.

Figure 5. Performance comparison on the number of generation with the population size increasing.
5.2.2 Time-Space Interrelationship

This section describes the behaviour of L1BOA and BOA when the population size increases. The sizes of these three problems are all equal to 40 and the population sizes are increased in proportion to the RMPS estimated in Section 5.2.1. The result of each experiment is calculated as the average over thirty runs with 95% confidence intervals.

Figure 5(a) and Figure 6(a) show the changes in generations and the corresponding function evaluations with increasing population sizes on 40-bit Onemax. In all test cases, the generations of L1BOA are lower than those of BOA no matter what value of k is taken. On the other hand, the generations of L1BOA and BOAs decrease as the population sizes rise. Although the generations of L1BOA is not as notable as those in BOA cases, its function evaluations nearly stay the same, while the increase in function evaluations of BOAs (e.g., k = 1, 2, 3) are much more remarkable.

The experimental results on 40-bit Link-10 are depicted in Figure 5(b) and Figure 6(b). The generations of L1BOA are higher than those of BOA, however, its function evaluations remain the lowest in all cases. When a population that is twice as large is used, the function evaluations only increase by one third, which is the lowest in all the cases.

As presented in Figure 5(c), the generations of L1BOA are lower than those of BOA with k = 2, 3 in all cases of 40-bit uTrap-4. Just like the results of Link-10, a decrease in generations is also obtained as the population sizes increase. The only difference is that the generations of L1BOA are reduced more dramatically. In Figure 6(c), when the population size is doubled, the function evaluations of L1BOA increases by less than 50%.

Figure 5(d) and Figure 6(d) show the changes in generations and the corresponding function evaluations when increasing population sizes in 40-bit eTrap-4 function. Although the generations of L1BOA are not the lowest, its function evaluations are always below those of the two cases of BOA when the population is enlarged. Furthermore, the generation’s reduction of L1BOA is more remarkable than the two cases of BOA and the function evaluations of L1BOA increase slower than those of two BOA cases.

From the results above, we argue that L1BOA performs as well as, sometimes better than, the best cases of BOA with a larger population available. And the function evaluations of L1BOA grow more slowly when enlarging population sizes.

6. Probabilistic Models Comparison

From the results represented in Section 5, it is already known that the performance of L1BOA is equal to, sometimes above that, of BOA with optimal artificial complexity control. In this section, we investigate the structure of Bayesian networks built by L1BOA and BOA with artificial complexity control. In the investigations, the accuracy and the complexity of the structure are involved. Before comparing Bayesian networks built by L1BOA and BOA, we firstly discuss the perfect models of the test functions according to the relationship between variables in these functions.

6.1 Perfect Models of the Test Function

The test functions used above are all additively decomposable problems. Thus, it is necessary that the probabilistic models encode all or nearly all conditional dependencies between corresponding pairs of variables within each partition. On the other hand, to maximize the mixing and minimize model complexity, it is desirable that we do not discover any dependencies between the different partitions [32] [38].

![Figure 7](image-url)
In Onemax, each bit serves as a building block (BB) and there are no dependencies between these BBs. Consequently, any dependencies in the model are illegal. The perfect model for Onemax is shown in Figure 7 (a). In Link-10, partitions apply equations (10), (11) and (12) to form the fitness. Therefore, the perfect model of Link-10 would obey these equations. According to equation (10), it is clear that there only exists one link for each variable $x_{10i-k}$ in the same partition when $k \geq 2$. (Dependencies only exist between variables if their indexes have the same quotient divided by 2.) On the other hand, equations (11) and (12) show that both $x_{10i-1}$ and $x_{10i}$ depend on a set of variables: the former depends on $[x_{10i-6}, x_{10i-6}, x_{10i-4}, x_{10i-2}]$ and the latter depends on $[x_{10i-9}, x_{10i-7}, x_{10i-5}, x_{10i-3}]$. Therefore, $x_{10i-1}$ and $x_{10i}$ depend on more variables than others. The perfect model of Link-10 in total is depicted in Figure 7 (b), which represents that the necessary dependencies for n-bit Link-10 are 1.2n. Since partitions in both $u$Trap-4 and $e$Trap-4 apply 4-bit trap function, the perfect models of them are identical. Trap-4 function, defined in equation (14), is fully deceptive [32]. Therefore, a “perfect” model for solving $u$Trap-4 and $e$Trap-4 would contain dependencies between each pairs of bits within the same partition but not contain any dependencies between the different trap partitions. Figure 7(c) depicts the perfect model of $u$Trap-4 and $e$Trap-4. The number of necessary dependencies for n-bit of these functions is 1.5n.

According to the discussion above, the distinction between dependencies that should be contained and avoided in the model is straightforward. Consequently, the dependencies that should be included in the perfect model are called necessary dependencies and the dependencies that should be excluded are noted as unnecessary dependencies.

6.2 Accuracy of Probabilistic Models

This subsection discusses the accuracy of the probabilistic models in L1BOA and BOA. Since the perfect model of Onemax contains no dependencies, the accuracies of models are not compared when BOA and L1BOA are dealing with Onemax. For other problems, according to the perfect models discussed above, we investigate the accuracies of dependencies in Bayesian networks built by L1BOA and BOA with explicit complexity control. The accuracies are measured by the average ratio of the number of necessary dependencies to the number of total dependencies in all generations during the evolution. For each experiment case (i.e., each test function with different problem sizes), the population size of BOA and L1BOA are set as the minimum population sizes estimated in Section 5.2.1. And the results of L1BOA and BOA for each case are the average over 30 runs.

In Figure 8, L1BOA is able to build more accurate models than BOA in almost all the test functions. Only in Link-10 with larger problem sizes, are the model accuracies of L1BOA slightly lower than those of BOA. On the other hand, the model accuracies of both L1BOA and BOA decrease with the rise of problem size. This is because the number of partitions grows larger when the problem size increases. As a consequence, the proportion of unnecessary dependencies across different partitions in all dependencies rises. It is also notable that with the same problem size, the model accuracy degrades when the scale of each partition changes from uniformity to exponentiation. The reason for the phenomenon is that exponentially-scaled problems are fundamentally different than uniformly-scaled problems [32].

6.3 Complexity of Probabilistic Models

In this subsection, the number of total dependencies in Bayesian networks in all generations during the evolution is investigated. The problem size of each function is set at 40 and for each case the population sizes of BOA and L1BOA are also set as RMPS estimated in Section 5.2.1. The results are the average over 30 runs and are represented in Figure 9 and 10, which depict the average number of dependencies in all generations and the number of dependencies in each generation respectively.

Figure 9 shows that the larger $k$ is, the more dependencies are involved when BOA deals with the same experimental case. We can also observe that the average total dependencies of each case are proportional to the value of $k$ generally. Furthermore, with the same problem size, the number of dependencies in the first few
generations of BOA (see Figure 10) is proportional to the value of \( k \) as well, no matter what test functions are handled. These phenomena demonstrate that the model complexity of BOA is completely dependent on the value of \( k \). Therefore, the explicit control for model complexity affects the efficiency of BOA significantly.

\[
\text{Figure 9. Performance comparison on the number of the dependencies during the evolution of the models.}
\]

\[
\text{Figure 10. Performance comparison on the number of the dependencies in models learned in each generation.}
\]

In Section 5, it is already illustrated that the performance of L1BOA is almost the same as BOA with \( k = 1 \) and \( k = 2 \) in function Link-10 and \( u\text{-Trap-4} \), respectively. The number of dependencies learned in the corresponding cases (see Figure 9) also reflects the same trend. On the other hand, the number of dependencies learned by L1BOA is smaller than that of BOA with \( k > 0 \) and \( k = 1 \), when coping with Onemax and the other functions, respectively. Although the gaps between the number of dependencies learned by L1BOA and BOA in different functions are not the same (the gaps in Onemax and \( u\text{-Trap-4} \) are more remarkable than those in Link-10 and \( u\text{-Trap-4} \)), the ratios between them are almost the same.

Additionally, it is also noticeable that the numbers of dependencies in the first several generations of L1BOA are totally different (see Figure 10). The reason behind these results lies in that during the evolution L1BOA controls the complexity of Bayesian networks automatically and effectively by means of L1-regularization.

7. Discussions

According to the results in Section 5, the complexity of Bayesian networks significantly impacts the performance of BOA. Without artificial control of the network complexity, BOA could not ensure finding the optimal solutions.

Furthermore, there are no uniform bounds to maximize its performance when dealing with different problems.

When variables are independent in the problem, such as Onemax, although larger bounds could not prevent BOA converging on the optimum, the performance degrades obviously. On the other hand, when variables in each separate partition depend (weakly or strongly) on each other, just like Link-10, \( u\text{-Trap-4} \) and \( e\text{-Trap-4} \), BOA could not achieve the optimum if the bound is not large enough. Unfortunately, when facing a black box optimization, such prior information about the problem is usually unavailable. Therefore, BOA may not exert its best performance, even worse in some situations, it may not achieve the global optimal.

The experimental results also show that L1BOA always reaches, sometimes exceeds, the best performance of BOA with various bounds on network complexity, when handling problems with different types and scales. Additionally, when the population size increases, L1BOA is able to maintain a competitive performance compared with BOA. The reason behind this phenomenon is that L1BOA detects candidate dependencies between nodes in Bayesian networks via L1-regularization and the number of the candidate parents of each node in Bayesian networks is able to be tuned to fit the specific problem without manual intervention. Consequently, L1BOA improves the robustness and the practicability of BOA.

Combined with the discussion in Section 5 and 6, there is a close connection between the behaviour of an algorithm and that of the Bayesian networks it builds. The higher the accuracy of Bayesian networks built, the better
performance gained. When there is an obvious difference between the accuracies of models, the gap between the performances is remarkable. Therefore, learning more accurate models may improve the performance of the algorithm.

8. Conclusions

This paper introduces and evaluates a new estimation of a distribution algorithm based on L1-regularized learning of Bayesian networks. In the proposed method, we apply L1-regularized logistic regression for each variable on the others. Since L1-regularized regression gives exactly zero regression coefficients to most irrelevant or redundant variables, these variables, whose corresponding parameters in regression coefficients are non-zero, are detected as candidate parents. Then the greedy search is restricted to the candidate parent-child pairs to identify the structure of Bayesian networks. Since the search is limited to a considerably small portion, the complexities of the network are controlled automatically and efficiently. Experiments on different types of benchmark problems show that L1BOA is better than BOA when no prior knowledge about the problem structure is available and achieves, even exceeds, the best performance of BOA with optimally controlled bounds on network complexity. Additionally, the probabilistic models (i.e., Bayesian networks) built by L1BOA and BOA are also analysed and compared, which demonstrate that L1BOA is able to build simpler, yet more accurate, models. Future areas for research include further evaluation of our proposed method in challenging real world scenarios. In addition, we intend to improve the strategy to tune the regularizer, in order to reduce the computation cost and find the global minimum of the MDL score more accurately.

9. Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 60875073, 61175110), National Basic Research Program of China (973 Program, Grant No. 2012CB316305)

10. References


