Average Time Complexity of Estimation of Distribution Algorithms

C. González, A. Ramírez, J.A. Lozano, and P. Larrañaga

Department of Computer Science and Artificial Intelligence,
University of the Basque Country, San Sebastián, Spain
{ccpgomoc, ccploalj, ccplamup}@si.ehu.es
juerware@hotmail.com

Abstract. This paper presents a study based on the empirical results of the average first hitting time of Estimation of Distribution Algorithms. The algorithms are applied to one example of linear, pseudo-modular, and unimax functions. By means of this study, the paper also addresses recent issues in Estimation of Distribution Algorithms: (i) the relationship between the complexity of the probabilistic model used by the algorithm and its efficiency, and (ii) the matching between this model and the relationship among the variables of the objective function. After analyzing the results, we conclude that the order of convergence is not related to the complexity of the probabilistic model, and that an algorithm whose probabilistic model mimics the structure of the objective function does not guarantee a low order of convergence.

1 Introduction

The most important questions concerning Evolutionary Algorithms (EAs) are how efficiently an EA will optimize a given objective function and which classes of objective functions can be optimized efficiently. Time complexity is a key issue in the analysis of EAs. It shows how efficiently an algorithm can face a large problem.

Recently, a new kind of EAs called Estimation of Distribution Algorithms (EDAs) [1] has appeared. In recent research on EDAs [2, 3], important questions related to the first hitting time have been raised. Does the first hitting time decrease when the complexity of the probabilistic model used by the algorithm increases? Is the algorithm's capability of detecting and exploiting the (in)dependencies of the objective function related to efficiency? In order to deepen our knowledge on these questions, we offer empirical results of the average first hitting time for some instances of EDAs: UMDA, TREE and EBNA$_{BIC}$ (see Section 2) applied to one example of linear, pseudo-modular and unimax functions. Our aim is to compare them.

The rest of this paper is organized as follows. Section 2 introduces EDAs and the instances that will be used in this work. The functions studied are introduced in Section 3. Section 4 explains the experiments carried out and presents the results. Finally, we draw conclusions in Section 5.

J. Cabestany, A. Prieto, and D.F. Sandoval (Eds.): IWANN 2005, LNCS 3512, pp. 42–49, 2005.

2 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) [1, 4] are based on GAs and constitute an example of stochastic heuristics based on populations of individuals, each of which encodes a possible solution to the optimization problem. These populations evolve in successive generations as the search progresses, organized the same way most EA heuristics are. To generate new individuals EDAs estimate and sample the joint probability distribution of the individuals selected. In Figure 1 a pseudocode for a general EDA can be seen. Unfortunately, the bottleneck of EDAs lies in estimating this joint probability distribution. To avoid this problem, several authors have proposed different algorithms where simplified assumptions concerning the conditional (in)dependencies between the variables of the joint probability distribution are made. The particular algorithms used in this paper have been chosen to provide one instance of each level of complexity of the probabilistic model.

EDA
  $D_0 \leftarrow$ Generate M individuals (the initial population) randomly
  Repeat for $l = 1, 2, \ldots$ until the stopping criterion is met
    $D_{l-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to the selection method
    $p_l(\mathbf{x}) = p(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the joint probability distribution for an individual to be one of the individuals selected
    $D_l \leftarrow$ Sample M individuals (the new population) from $p_l(\mathbf{x})$

Fig. 1. Pseudocode for the EDA approach

UMDA. The Univariate Marginal Distribution Algorithm (UMDA) was proposed by Mühlenbein [4]. UMDA uses the simplest model to estimate the joint probability distribution of the selected individuals at each generation, $p_l(\mathbf{x})$. This joint probability distribution is factorized as a product of independent univariate marginal distributions, which are usually calculated by maximum likelihood estimation.

TREE. Baluja and Davies [5] proposed an algorithm called COMIT. This algorithm uses a probabilistic model that considers second-order statistics. The dependency structure between the variables forms a tree. The tree structure of the probability distribution of the individuals selected at each generation is estimated using the algorithm proposed by Chow and Liu. The parameters, given the structure, are calculated by maximum likelihood estimation. In the original work, COMIT applied a local optimizer to each individual generated. We have eliminated this step in order to carry out a fair comparison. In other words, this paper refers to TREE as an algorithm that does not apply a local optimizer.

EBNA. The Estimation of Bayesian Networks Algorithm (EBNA) was introduced by Larrañaga et al. [6]. This algorithm allows statistics of unrestricted

order in the factorization of the joint probability distribution. This distribution is encoded by a Bayesian network that is learned from the database containing the individuals selected at each generation. Formally, a Bayesian network is a pair, $(S, \boldsymbol{\theta})$, representing a graphical factorization of a probability distribution. The structure, S, is a directed acyclic graph which reflects the set of conditional (in)dependencies among the variables, while $\boldsymbol{\theta}$ is a set of parameters for the local probability distributions associated with each variable.

In EBNA learning the probabilistic model at each generation of the algorithm means learning a Bayesian network from the individuals selected. There are different strategies to learn the structure of a Bayesian network. Here the 'score + search' method is used. In this method, given a database, D, and a Bayesian network whose structure is denoted by S, a value which evaluates how well the Bayesian network represents the probability distribution of D is assigned. Different EBNA algorithms can be obtained by using different scores. In this work we have used the BIC score (based on penalized maximum likelihood).

After defining the score, we have to set a search process to find the Bayesian network that maximizes the score given the set of individuals selected. As we need to find an adequate model structure as quickly as possible, a simple algorithm which returns a good structure, even if not optimal, is preferred. An interesting algorithm with these characteristics is Algorithm B. Algorithm B is a greedy search which starts with an arc-less structure and, at each step, adds the arc with the maximum score improvement. The algorithm finishes when there is no arc whose addition improves the score. The parameters of the Bayesian network are calculated by maximum likelihood.

3 The Functions Used

We optimize particular cases of three problem classes of pseudo-Boolean functions: linear, pseudo-modular and unimax functions. To briefly analyze each objective function we set a neighborhood structure: the neighbor solutions $N(\mathbf{x})$ of a given solution $\mathbf{x} \in \{0,1\}^n$ are composed of $\mathbf{x}$ and all points at Hamming distance 1.

The particular **linear function** analyzed in this work is $f(\mathbf{x}) = \sum_{i=1}^{n} i \cdot x_i$, $x_i \in \{0,1\}$. Clearly $(1,\ldots,1)$ is the only global maximum for this function, and the value of $f(\mathbf{x})$ at this point is $\frac{n(n+1)}{2}$. It should be stressed that this function can be optimized variable by variable. Taking this fact into account, a suitable probabilistic model that corresponds to the relationship between the variables of this function assumes that all the variables are independent. Therefore, UMDA seems to be a good candidate to optimize it.

The **pseudo-modular** function that we have used is $f(\mathbf{x}) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$, with $x_j \in \{0,1\}$. For this fitness function the only optimal solution is $(1,\ldots,1)$, and the value of $f(\mathbf{x})$ at this point is n. The variables of this function present a chain of pairwise dependencies, where x_i depends on x_{i-1}, $i = 1,\ldots,n-1$. This fact suggests that TREE is a suitable model to optimize it.

In the experiments carried out, we have used a well-known **unimax function**: the long path function [7]. It is important to stress that it only makes sense if n is odd. The optimal point in the long path problem is $(1, 1, 0, \ldots, 0)$ and the function value is 0. In long path problems, the relationship between the variables of the problem are not evident, and the EBNA$_{BIC}$ algorithm, therefore, seems to be the most adequate one to optimize these functions.

4 Experimental Results

In order to find the average first hitting time of the EDAs introduced in Section 2, we have carried out experiments using those algorithms to maximize the different functions in Section 3. The empirical results for each algorithm and each objective function have been fitted to curves (in the least squares sense) in terms of problem size, using "*Mathematica*" program. We have measure the "goodness" of each fit with the coefficient of determination and the mean squared error. After comparing those quantities for various fits we have chosen the fit with greater coefficient of determination and lower mean squared error. Finally, all this information has been graphically presented.

The population size, M, was fixed to avoid dependency of the first hitting time on this parameter (as done in previously published works [8]), thus we fixed M to the individual size n. The stopping condition is the same for each algorithm: they stop when they find the optimum for the first time. The best individuals are selected (truncation selection).

Next, in order to set up the number of individuals selected and the type of elitism used, we illustrate how a new population is created. Once the population D_{l-1} is created, the $n/2$ best individuals of D_{l-1} are used to estimate the joint probability distribution $p_l(\mathbf{x})$. After that, $n-1$ individuals are sampled from $p_l(\mathbf{x})$, obtaining D_l^*. Finally the new population D_l is created by selecting the n best individuals from $D_{l-1} \cup D_l^*$. This way, we make sure that the best individual in population D_{l-1} will not get lost. To ensure that the algorithm goes through the optimum, the maximum likelihood estimation of parameters is modified via

EDA
  $D_0 \leftarrow$ Generate $M = n$, individuals (initial population) randomly
    Define: $f_{max} = max\{f(\mathbf{x}) : \mathbf{x} \in \{0,1\}^n\}$
  Repeat for $l = 1, 2, \ldots$ until $f_{D_l} = f_{max}$
    $D_{l-1}^{Se} \leftarrow$ Select the $n/2$ best individuals from D_{l-1}
    $p_l(\mathbf{x}) = p(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the joint probability distribution for an individual to be one of the individuals selected, using Laplace correction when estimating the the parameters
    $D_l^* \leftarrow$ Sample $n-1$ individuals from $p_l(\mathbf{x})$
    $D_l \leftarrow$ Select the n best individuals from $D_{l-1} \cup D_l^*$
    Define $f_{D_l} = max\{f(\mathbf{x}) : \mathbf{x} \in D_l\}$

Fig. 2. Pseudocode for the EDA approach used in the experiments

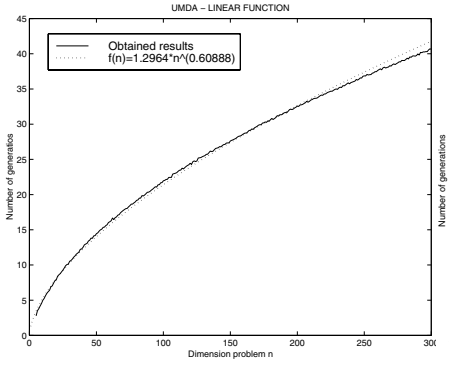

Fig. 3. UMDA - Linear F

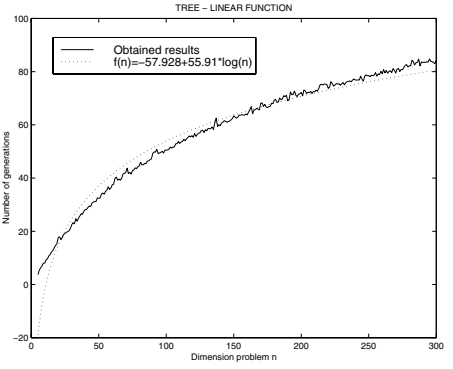

Fig. 4. TREE - Linear F

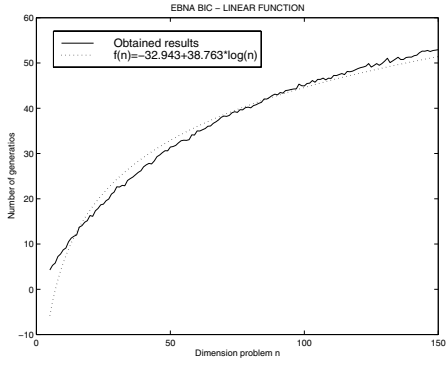

Fig. 5. EBNA$_{BIC}$ - Linear F

	Fitting Curve	Order
UMDA	$1.2964 \cdot n^{0.60888}$	$O(n^{\varepsilon})$ $0.6<\varepsilon<1$
TREE	$-57.928+55.91 \log(n)$	$O(\log n)$
EBNA$_{BIC}$	$-32.943+38.763 \log n$	$O(\log n)$

Fig. 6. Fitting curves for the different EDAs in the optimization of the linear function and their order

Laplace correction. A pseudocode for a general algorithm used in the experiments can be seen in Figure 2.

We run each algorithm 1,000 times for each objective function and each problem dimension, each time recording the generation in which the optimum was reached for the first time. Due to the computational cost associated with the learning of a Bayesian network at each iteration, the problem dimension of the EBNA algorithm is lower than in the rest. The problem dimensions used for the UMDA and TREE algorithms ranged from 5 to 300 in the linear function, from 5 to 150 in the pseudo-modular function, and from 5 to 65 in the unimax function. For EBNA$_{BIC}$ they ranged from 5 to 150 in the linear and pseudo-modular functions and from 5 to 65 in the unimax function.

4.1 Summarizing the Results

After obtaining the results for each algorithm and each objective function, we have found a simple formula that approximates these numerical results, fitting a curve through the data obtained. Furthermore, we can offer a general idea con-

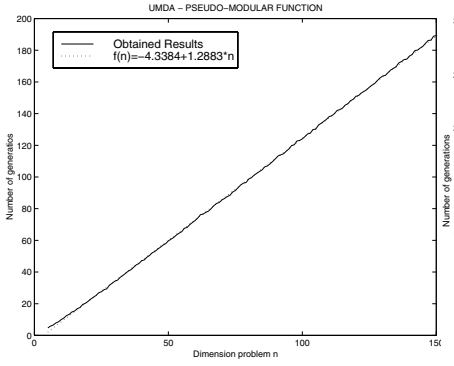

Fig. 7. UMDA - Pse. Mod. F

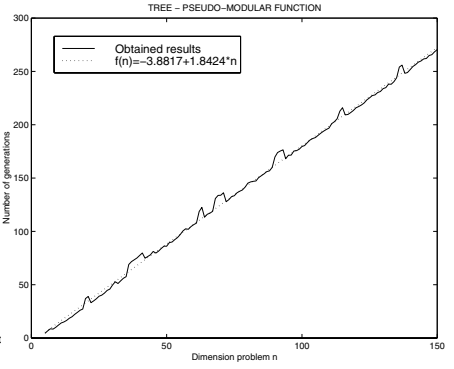

Fig. 8. TREE - Pse. Mod. F

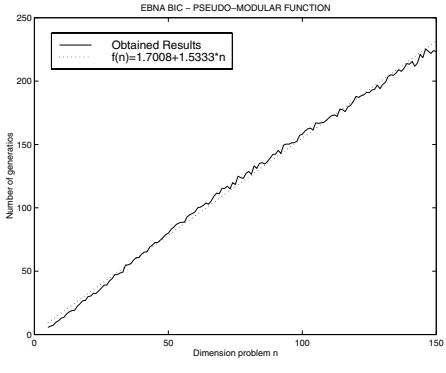

Fig. 9. EBNA$_{BIC}$ - Pse. Mod. F

	Fitting Curve	Order
UMDA	$-4.3384+1.2883 \cdot n$	$O(n)$
TREE	$-3.8817+1.8424 \cdot n$	$O(n)$
EBNA$_{BIC}$	$1.7008+1.5333 \cdot n$	$O(n)$

Fig. 10. Fitting curves for the different EDAs in the optimization of the pseudo-modular function and their order

cerning the expected time complexity for each algorithm. We show four figures for each function, the last of them being a table. The first three figures show the results obtained for an EDA when optimizing the objective function. The table entries are: (i) the curve that fits the results obtained for each EDA and (ii) the order of the fitting curve.

Linear Function. The results in the linear function can be seen in Figures 3 to 6. They show that UMDA has the best behavior ($O(n^{\varepsilon})$, $0.6 < \varepsilon < 1$). In this case, the probabilistic model of UMDA, that seemed to be the one that best corresponded with the relationship between the variables of the problem, obtains the best average first hitting time.

Pseudo-Modular Function. The results for the pseudo-modular function can be seen in Figures 7 to 10. As can be seen in Figure 10, all the algorithms have the same linear ($O(n)$) behavior. If we take into account the fitting curves, UMDA's results appear slightly better than the others. Here the correspondence between

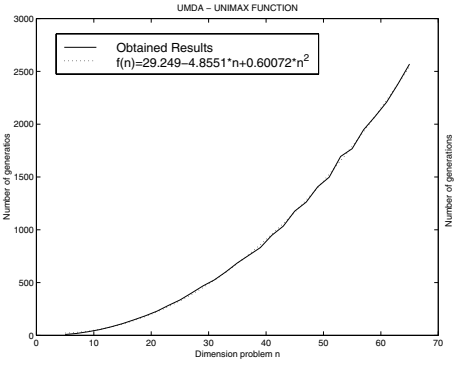

Fig. 11. UMDA - Unimax F

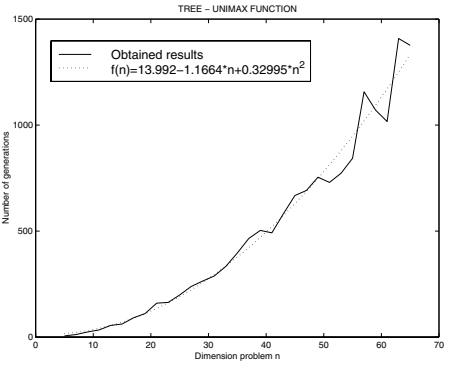

Fig. 12. TREE - Unimax F

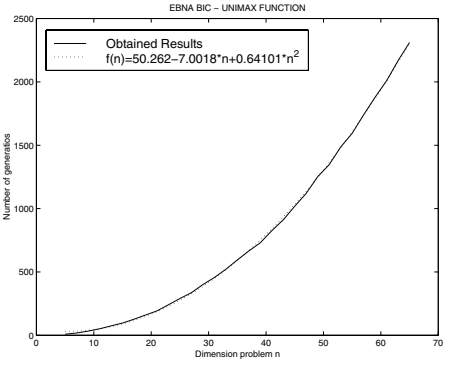

Fig. 13. EBNA$_{BIC}$ - Unimax F

	Fitting Curve	Order
UMDA	$29.249-4.8551 \cdot n+0.67002 \cdot n^2$	$O(n^2)$
TREE	$13.999-1.1664 \cdot n+0.32995 \cdot n^2$	$O(n^2)$
EBNA$_{BIC}$	$50.262-7.0018 \cdot n+0.64101 \cdot n^2$	$O(n^2)$

Fig. 14. Order of convergence and fitting curves for the different EDAs in the optimization of the unimax function

the probabilistic model used by the algorithm and the relationship between the variables of the problem does not imply a better average first hitting time.

Unimax Function. The results for the unimax function are given in Figures 11 to 14. As can be seen in Figure 14 all the algorithms have the same quadratic behavior ($O(n^2)$). Taking into account the fitting curves, TREE seems to have better behavior than the others. Again, the correspondence between the probabilistic model used by the algorithm and the relationship between the variables of the problem does not imply a better average first hitting time.

5 Conclusion

Based on empirical results, this work offers new information on the first hitting time of some EDAs applied to a number of objective functions. Two main conclusions have been reached. On one hand, the average optimization time in the experiments is not related to the complexity of the probabilistic model used by

the algorithm. A greater complexity of the probabilistic model of the EDA does not imply a greater efficiency or a low order of the first hitting time. On the other hand, EDAs whose probabilistic models reflect the relationship between the variables of the problem do not obtain a better order of the first hitting time.

In order to improve the statistical analysis carried out, in future research we will (i) run the algorithm more than 1,000 times, (ii) record at each generation the standard deviation and (iii) enlarge the range of n (to provide a more sensible fit).

Acknowledgments

This work was supported by the University of the Basque Country under grant 9/UPV/EHU 00140.226-15334/2003, and also by the Basque Government under grant Saiotek S-PE04UN25.

References

1. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers (2002)
2. Pelikan, M., Sastry, K., Goldberg, D.E.: Scalability of the Bayesian Optimization Algorithm. International Journal of Approximate Reasoning **31** (2002) 221–258
3. Mühlenbein, H., Mahnig, T.: Evolutionary Optimization and the Estimation of Search Distributions with Applications to Graph Bipartitioning. International Journal of Approximate Reasoning **31** (2002) 157–192
4. Mühlenbein, H., Paaß, G.: From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.P., eds.: Parallel Problem Solving from Nature, PPSN-IV. (1996) 178–187
5. Baluja, S., Davies, S.: Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of the Search Space. Technical Report CMU-CS-97-107, Carnegie Mellon University (1997)
6. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Combinatorial Optimization by Learning and Simulation of Bayesian Networks. In Boutilier, C., Goldszmidt, M., eds.: Proceedings of Uncertainty in Artificial Intelligence, UAI-2000, Morgan Kaufmann (2000) 343–352
7. Horn, J., Goldberg, D.E., Deb, K.: Long Path Problems. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature, PPSN III, Berlin and Heidelberg: Springer (1994) 149–158
8. He, J., Yao, X.: Drift Analysis and Average Time Complexity of Evolutionary Algorithms. Artificial Intelligence **127** (2001) 57–85