Evaluating the Max-Min Hill-Climbing Estimation of Distribution Algorithm on B-Functions

Julio Madera[1](✉) and Alberto Ochoa[2]

[1] Department of Computer Science, University of Camagüey, Camagüey, Cuba
julio.madera@reduc.edu.cu
[2] Instituto de Cibernética, Matemática Y Física, La Habana, Cuba

Abstract. In this paper we evaluate a new Estimation of Distribution Algorithm (EDA) constructed on top of a very successful Bayesian network learning procedure, Max-Min Hill-Climbing (MMHC). The aim of this paper is to check whether the excellent properties reported for this algorithm in machine learning papers, have some impact on the efficiency and efficacy of EDA based optimization. Our experiments show that the proposed algorithm outperform well-known state of the art EDA like BOA and EBNA in a test bed based on B-functions. On the basis of these results we conclude that the proposed scheme is a promising candidate for challenging real-world applications, specifically, problems related to the areas of Data Mining, Patter Recognition and Artificial Intelligence.

Keywords: Estimation of distribution algorithms · B-functions
Bayesian networks · Dependency learning · Evolutionary optimization

1 Introduction

Nowadays research on Estimation of Distribution Algorithms (EDAs) is a well-established branch of evolutionary computation [3]. The amount and quality of papers reporting both theoretical [1, 4–6] and practical works increases every year, while practitioners have begun to consider the possibility of using EDAs for their challenging real-world applications [2, 3, 8]. However, before EDAs can really impact the optimization market still many theoretical issues have to be addressed and solved.

In this contribution, which is only aimed to present our algorithm, we have focused in just two aspects of the problem: the scalability and efficacy issues. By this we mean the ability of any EDA scheme to preserve its good algorithmic properties as the number of variables of the cost function increases. Let the number of function evaluations and the average time to get to the optimum, play the role of the mentioned properties. We shall show that the algorithm we are about to introduce on average scales better than several of the most famous existing Bayesian EDAs. To do that, we present a set of experiments that support our claim. Also, we will argue that these results extend to a very large class of problems.

We have chosen for our initial study the Bayesian Optimization Algorithm (BOA) [7] and the Estimation of Bayesian Networks algorithm (EBNA) [3]. These algorithms use a scoring metric to explore the space of Bayesian networks and pick up one that is somehow optimal for modeling the search distributions of a binary optimization problem. For the sake of brevity, we do not discuss any of these algorithms neither make a general presentation about EDAs or Bayesian networks. The interested reader is referred to the literature [1, 3, 4, 6].

The novel optimizer is designed for integer problems. It combines the use of a scoring metric and independence tests in its learning algorithm. Learning is accomplished with the Max-Min Hill Climbing (MMHC) algorithm, which was introduced in [11] and implemented in [10]. The MMHC algorithm has been shown to possess good properties for machine learning problems. According to the reported experimental evaluations, it outperforms on average its predecessors with regard to computational efficiency, scalability and quality of the learned networks. Besides it can be easily parallelized. It is obvious that all these properties are of great value for EDA optimization.

Once one has a MMHC implementation there are two things that one is tempted to do immediately: designing the and afterwards its parallel version. On one hand, we obtain scalability with respect to the amount of functions evaluations. On the other hand, time scalability is added with the parallel version. We have accomplished both tasks but, in this work, we will report only the sequential algorithm.

The outline of the paper is as follows. In the next section the MMHC algorithm is discussed following the presentation made in [9, 11] This section is ended with the introduction of our MMHCEDA algorithm. This is followed by the numerical results (Sect. 3) which are presented using the B-functions benchmark [4, 6]. Finally, some remarks on future works and the conclusions are given.

2 The Max-Min Hill-Climbing Algorithm

The so-called Max-Min Hill Climbing (MMHC) algorithm is a two-stage procedure that combines constraint-based and search-and-score methods for learning Bayesian networks. In stage I, the algorithm computes a collection of candidate sets, PC(T), which contain the parents and children of each problem's variable, T. In stage II, it performs a local search –constrained by the PC(T) sets– and looks for the solution in the space of directed acyclic networks.

According to the reported experimental evaluations, MMHC learning outperforms on average its predecessors with regard to computational efficiency, scalability and quality of the learned networks. In what follows we give just a short overview of the algorithm; we refer the reader to [9, 11] and the references therein for a detailed treatment of the topic.

The computation of the candidate sets is accomplished using a local algorithm for causal discovery called Max-Min Parents and Children (MMPC, algorithm 1).

Given the variable T and the sample database D, MMPC uses also a two-phase scheme to discover the PC(T) [11]. In the forward phase, variables enter sequentially PC(T) by use of a heuristic function that selects the variable that maximizes the association with T conditioned on the subset of the current estimate of PC(T) that minimizes that association (hence the name of the algorithm). All variables with an edge to or from T and possibly more will enter PC(T). All false positives are removed in the second phase.

Algorithm 1. $PC(T) = MMPC(T, D)$.

```
Phase 1 (Forward)

1. $PC(T) = \emptyset$
2. REPEAT
    a. $PC(T) = PC(T) \cup X$ (where X maximizes a heuristic function based on independence tests)
3. UNTIL All variables are conditionally independent of T given some subset of $PC(T)$

Phase 2 (Backward)

1. Remove from $PC(T)$ any variable independent of T given some subset of $PC(T)$
```

The above-mentioned association is a conditional measure that captures the strength of the dependence with respect to D and is denoted by $dep(X_i, X_j|Z)$. MMPC runs χ^2 independence tests with the G^2 statistic in order to decide on the conditional (in)dependence of X_i and X_j given Z and uses the p-value of the test as the association. Our current implementation of the algorithm follows [10]. Under the conditional independence assumption:

$$G^2 = 2\sum\nolimits_{ijk} N_{ijk} ln \frac{N_{ijk}N_k}{N_{ik}N_{jk}} \tag{1}$$

where N_{ijk} denotes how many times X_i and X_j take, respectively, their i-th and j-th values, whereas the set of variables Z takes its k-th value. N_{ik}, N_{jk} and N_k are defined similarly. Assuming that there are not structural zeros, G^2 behaves asymptotically as χ^2 with:

$$df = (|X_i| - 1)(|X_j| - 1)\prod\nolimits_{X_k \in Z} |X_k| \tag{2}$$

degrees of freedom. Here, |X| denotes the cardinality of X.

Once we have an efficient algorithm to learn the local structure of the Bayesian network (MMPC) we can learn the overall network. MMHC starts from a fully disconnected network and uses a greedy hill climbing to look for the Bayesian network that optimizes a given scoring metric (our current implementation is based on the Bayesian information criterion). MMHC uses three local operators: *DeleteArc*, *InvertArc* and *AddArc*. The search is constrained because the addition operator obeys the following restriction: $T \rightarrow Y$ can be added only if $Y \in PC(T)$. It is worth noting, that the $PC(T)$ set is unique only if the distribution of data is faithful to a Bayesian network. The functions of the experimental section of this paper, the B-functions, are such a kind of functions.

Algorithm 1. $MMHC(D)$.

```
2. FORALL variables $X$ DO
   a. $PC(X) = MMPC(X, D)$
3. Start from a fully disconnected network ($BN$)
4. REPEAT
   a. Greedily apply hill-climbing operators: DeleteArc, InvertArc and AddArc ($X \rightarrow$
      $Y$ can be added only if $Y \in PC(X)$)
5. UNTIL no improvements can be obtained
6. Output the computed Bayesian network structure ($BN$)
```

2.1 An EDA Based on the MMHC Algorithm

MMHCEDA is just an EDA whose learning step is based on the MMHC algorithm [4, 5]. Its simplest version is shown in algorithm 3. Among other things, this simple scheme does not include mutation nor elitism. This is enough for our current presentation. The analysis of more elaborated algorithms is left for future papers. Note that although one can use any selection scheme, we have chosen the method of truncation selection for all the simulations of this paper.

Algorithm 3. Simple *MMHCEDA* algorithm.

```
1. Let $t \leftarrow 1$. Generate $N \gg 0$ random points
2. According to a selection method, construct a set, $CS$, of $M$ points
3. Find the structure of the modeling Bayesian network: $BN = MMHC(CS)$
4. Estimate the parameters of $p^s(x, t)$ using $BN$ and $CS$
5. Generate $N$ new points from $p(x, t + 1) \approx p^s(x, t)$
6. If the termination criteria are not met, make $t \leftarrow t + 1$ and go to step 2
```

3 Numerical Results

In this section we report our empirical comparison of the algorithms *MMHCEDA*, *BOA* and *EBNA* with the so-called B-functions (details will follow). The aim of the experiments is simple: to show that the *MMHCEDA* is well suited for optimization and that it is competitive with the best algorithms that have been reported so far.

3.1 B-Functions

The B-function benchmark was introduced in [5]. Let us start with the formal definition of B-functions.

Definition 1. (B-functions). Given $n > 0$, let $X_1, X_2, \cdots, X_n$ be arbitrary discrete random variables with joint probability mass function $q(X_1, X_2, \cdots, X_n)$. Denoting $x = (x_1, x_2, \cdots, x_n) \in X = X_1 \times X_2 \times \cdots \times X_n$ and assuming that $\hat{x}$ is one of the modes of $q(X)$, the parametric function:

$$Bf_{q,\eta}(x) = \frac{1}{\eta} \log\left(\frac{q(\hat{x})}{q(x)}\right) \tag{3}$$

is non-negative, additive, has a minimum at the point $\hat{x}$ and is called B-function. The temperature, $\frac{1}{\eta}$, and the distribution, $q(x)$ are called definition parameters of the B-function. It worth noting that this definition can be trivially adapted to deal with continuous variables [6].

Loosely speaking, we say that B-functions are important for research because if a "good enough" EDA algorithm is used to optimize (see Eq. 3), it will see search distributions were the probabilistic relationships existing in the definition distribution q (x) are approximated [6]. At this point we recall that the success of the *MMHC* learning algorithm is affected by the faithfulness of the data.

The B-functions have a number of good properties that make them interesting as a benchmark for EDAs (see [4]). For example, it is possible to construct efficient random function generators for some B-function classes. In the experiments of this paper we use the class of binary polytree B-functions, which have definition distributions with polytree structure (single-connected Bayesian network).

We start our experiments with the function FirstPolytree5 –one of the first polytree functions designed [6]. In this case the old practice of building large problems by concatenation of small ones was followed. However, the remaining functions of the paper were obtained using a random B-functions generator. The B-functions studied were named following [6], thus the reader is referred there for details. Here, we just give a short description of the functions.

BF2B30s4-2312: It stands for the random subclass of binary polytree functions that have 30 variables, each with a maximum of four parents. Each pair of adjacent variables – in the polytree – has mutual information in the interval $[0.2, 0.3]$ and the univariate probabilities lie either in $[0.15, 0.25]$ or in $[0.75, 0.85]$. Note that these intervals are mapped to the same entropic interval.

BF2B30s4-1245: Almost as before but this time the entropic interval corresponds to the probability interval $[0.45, 0.55]$ (very high entropy) and the mutual information of the edges belong to the interval $[0.1, 0.2]$.

BF2Bns0-0034: This class has not correlations at all, thus the fields containing the mutual information bounds and the connectivity constraint should be ignored. In this case the univariate probabilities lie either in $[0.35, 0.45]$ or in $[0.55, 0.65]$.

3.2 MMHCEDA Scales with the Function FirstPolytree5

Our first experiment studies the numerical scalability of the *MMHCEDA* using the function $F_{FP5}(x)$ with 50, 100, 150 and 200 variables. FP5 is a separable additive decomposable function with blocks of length 5. In each block the FirstPolytree5 function is evaluated. It has the following property: its Boltzmann distribution with parameter $\beta \approx 2$ has a polytree structure with edges: $x_1 \rightarrow x_3, x_2 \rightarrow x_3, x_3 \rightarrow x_5, x_4 \rightarrow x_5$. The definition of the subfunction f_5^{Poly} is given in [6] whereas F_{FP5} is computed as follows:

$$F_{FP5}(x) = \sum\nolimits_{i=1}^{l} f_5^{Poly}(x_{5*i-4}, x_{5*i-3}, x_{5*i-2}, x_{5*i-1}, x_{5*i}) \tag{4}$$

The point $(0,1,0,0,1,\ldots,0,1,0,0,1)$ is the global maximum of this problem where the function takes the value $l * 1.723$, with $n = 5 * l$. The aim of the experiment was to find the critical (minimum) population size that guarantees 100% of success in 30 runs. We used the bisection method to compute this critical value.

Figure 1 shows the average number of function evaluations for the critical population size found. From this figure is easy to conclude that in this problem, *MMHCEDA* scales well with the number of variables.

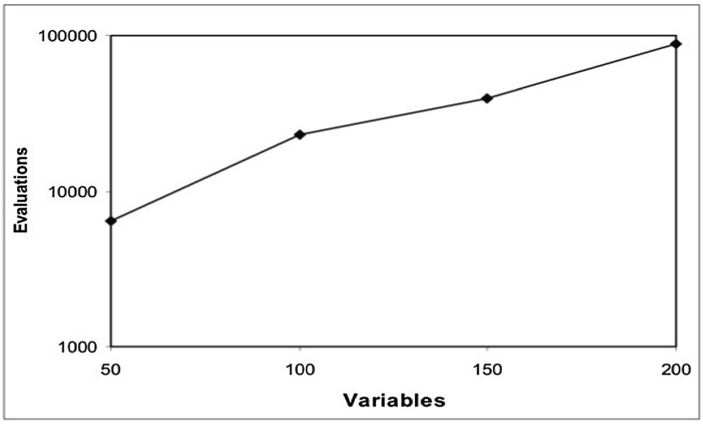

Fig. 1. Scalability of the $CBEDA_{MMHC}$ for the function $F_{FP5}(x)$.

3.3 Minimization of BF2B30s4-2312 and BF2B30s4-1245

In this experiment two B-functions, one with low (BF2B30s4-2312) and another with high univariate entropies (BF2B30s4-1245) were chosen. Later on, the percentage of convergence of each algorithm was obtained for a population size of 100 individuals. When a certain algorithm didn't converge 100%, its population was augmented with 100 individuals until arriving to a population of 500. Note that with the high entropy B-function were necessary population sizes of 1000, 2000, 4000 and 8000.

Table 1. Minimization of the function **BF2B30s4-2312**.

N	UMDA	BOA k=1	BOA k=2	BOA k=3	EBNA	MMHCEDA
100	97	30	0	0	100	46.6
200	100	100	50	23.3	97	100
300	100	100	90	46.6	100	100
400	100	100	100	90	100	100
500	100	10	100	100	100	100

As can be seen from Table 1, the optimization of the function BF2B30s4-2312 is an easy task for all studied algorithms. Again, BOA shows a very strong dependency on the parameter k. A more interesting observation is that *MMHCEDA* is more efficient than EBNA. We draw the attention of the reader to the following fact: with the function BF2B30s4-2312 the critical population sizes of EBNA and $CBEDA_{MMHC}$ were 100 and 200 respectively (see the Table 2). Despite this, $CBEDA_{MMHC}$ makes less functions evaluations than EBNA, so it converges faster and therefore discovers faster the important correlations of the problem.

Table 2. Minimization of the function **BF2B30s4-1245**.

N	UMDA	BOA k=1	BOA k=2	BOA k=3	EBNA	MMHCEDA
100	3.33	0	0	0	13.3	6.67
200	3.33	30	10	0	16.6	23.3
300	3.33	46.6	23.3	13.3	30	40
400	3.33	53.3	43.3	30	30	66.6
500	3.33	50	56.6	43.3	26.6	53.3
1000	3.33	43.3	50	50	40	73.3
4000	3.33	43.3	50	43.3	33.3	100
8000	3.33	43.3	50	50	43.3	100

The B-function BF2B30s4-1245 turns out to be much harder, which confirms what was said in [4, 5] with regard to the entropy of the function. We have included the UMDA in our experiments to highlight this issue: it easily finds the optimum of the low entropy function but fails dramatically with the high entropy one. The good news is that the *MMHCEDA* performs well in both extreme cases. Neither BOA nor EBNA can do that, they fail even with a population of 8000 individuals. In contrast, our *MMHCEDA* already gets 100% success rate with 2000 individuals.

4 Conclusions and Future Work

The evaluated algorithm is based in recent advances in the area of learning of Bayesian networks from data. In particular, the algorithm scales well with high efficiency of data. We have wondered to what extent these features could mean a significant impulse with regard to the quality of the EDAs and their capacities to attack real world problems of large dimensions mainly problems from Data Mining, Pattern Recognition, and Artificial Intelligence fields. Indeed, the comparison of the *MMHCEDA* with the algorithms *BOA* and *EBNA* allows us to conclude that it is a competitive evolutionary optimization algorithm. Keeping in mind the nature of the chosen test functions, we conjecture that similar results will be obtained with many other classes of problems. The research reported here is just a first promising step and thus we continue working in this direction. We are currently developing a parallel version of the algorithm. Another line that we are already approaching is the creation of more efficient models of the learning type used in this report and consequently of new EDAs based on them.

References

1. Ding, C., Ding, L., Peng, W.: Comparison of effects of different learning methods on estimation of distribution algorithms. J. Softw. Eng. **9**(3), 451–468 (2015)
2. Gao, S., Qiu, L., Cao, C.: Solving 0–1 integer programming problem by estimation of distribution algorithms. J. Comput. Intell. Electr. Syst. **3**(1), 65–68 (2014)
3. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Optimization. Kluwer Academic, Boston (2002)
4. Madera, J.: Hacia una Generación Eficiente de Algoritmos Evolutivos con Estimación de Distribuciones: Pruebas de (In)dependencia +Paralelismo. Ph.D. thesis, Instituto de Cibernética, Matemática y Física, La Habana. Adviser: A. Ochoa (2009). (in Spanish)
5. Ochoa, A.: Opportunities for expensive optimization with estimation of distribution algorithms. In: Tenne, Y., Goh, C.-K. (eds.) Computational Intelligence in Expensive Optimization Problems. ALO, vol. 2, pp. 193–218. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-10701-6_8
6. Ochoa, A., Soto, M.: Linking entropy to estimation of distribution algorithms. In: Lozano, J. A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms. STUDFUZZ, vol. 192, pp. 1–38. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-32494-1_1
7. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: the Bayesian optimization algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, pp. 525–532. Morgan Kaufmann, San Francisco (1999)
8. Pérez-Rodríguez, R., Hernández-Aguirre, A.: An estimation of distribution algorithm-based approach for the order batching problem: an experimental study. In: Handbook of Research on Military, Aeronautical, and Maritime Logistics and Operations. IGI Global (2016)
9. Preetam, N., Hauser, A., Maathuis, M.H.: High-dimensional consistency in score-based and hybrid structure learning. arXiv preprint arXiv:1507.02608 (2018)
10. Scutari, M., Ness, R.: bnlearn: Bayesian network structure learning, parameter learning and inference. R package version 3 (2012)
11. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Mach. Learn. **65**(1), 31–78 (2006)