On updating probabilistic graphical models in Bayesian Optimisation Algorithm

Mohamed El Yafrani
Aalborg University
Email: mey@mp.aau.dk

Marcella Scoczynski
UTFPR-Brazil
Email: marcella@utfpr.edu.br

Myriam Delgado
UTFPR-Brazil
Email: myriamdelg@utfpr.edu.br

Ricardo Lüders
UTFPR-Brazil
Email: luders@utfpr.edu.br

Inkyung Sung
Aalborg University
Email: inkyung_sung@mp.aau.dk

Markus Wagner
The University of Adelaide
Email: markus.wagner@adelaide.edu.au

Diego Oliva
Universidad de Guadalajara
Email: diego.oliva@cucei.udg.mx

***Abstract*—The Bayesian Optimisation Algorithm (BOA) is an Estimation of Distribution Algorithm (EDA) that uses a Bayesian network as probabilistic graphical model (PGM). During the evolutionary process, determining the optimal Bayesian network structure by a given solution sample is an NP-hard problem resulting in a very time-consuming process. However, we show in this paper that significant changes in PGM structure do not occur so frequently, and can be particularly sparse at the end of evolution. A statistical study of BOA is thus presented to characterise a pattern of PGM adjustments that can be used as a guide to reduce the frequency of PGM updates. This is accomplished by proposing a new BOA-based optimisation approach (FBOA) whose PGM is not updated at each iteration. This new approach avoids the computational burden usually found in the standard BOA. Inspired by fitness landscape analysis concepts, we perform an investigation in the search space of an NK-landscape optimisation problem and compare the performances of both algorithms by using the correlation between the landscape ruggedness of the problem and the expected runtime of the algorithms. The experiments show that FBOA presents competitive results with significant saving of computational time.**

***Index Terms*—Estimation of Distribution Algorithms, Probabilistic Graphical Models, Bayesian Networks, Model-based Metaheuristics**

I. Introduction

Probabilistic graphical models (PGMs) [1] combine graph and probability theory to represent structured distributions. They play an important role in many computationally oriented fields [2], including combinatorial optimisation, statistical physics, bioinformatics, machine learning, control theory and economics [3]. Being able to capture multivariate interactions between variables, Bayesian networks are PGMs widely used in evolutionary optimisation, especially in Estimation of Distribution Algorithms (EDAs) [4], such as Estimation of Bayesian Network Algorithm (EBNA) [5] and Bayesian Optimisation Algorithm (BOA) [6].

Finding the optimal structure of a Bayesian network is considered NP-hard problem [7]. The use of structure learning methods has been extensively studied in [8], [9], [2], resulting in a range of algorithms for various settings.

The main contributions of this paper are as follow. First, we consider the Bayesian Optimisation Algorithm (BOA) [6] and experimentally show that the changes in the PGM follow a certain pattern. Second, this pattern is used to propose an alternative version of BOA, called Fast BOA (FBOA) that decides, following a probability distribution, whether to adjust[1] the PGM or to resample from the previous one. To achieve our goals, we start by analysing the behaviour of BOA and the similarities between consecutive Bayesian networks. Additionally, inspired by the basis of Fitness Landscape Analysis (FLA), we explore the impact of problem features on the search performance of BOA versions. FLA directly associates search with information extracted from the fitness space and has been applied to investigate the dynamics of evolutionary algorithms using models to predict their behaviours [10]. In this paper, the performance is estimated by a new runtime estimation metric which considers the computational complexity of evaluation and PGM generation. The results indicate that our model-based BOA is about three times faster than the standard BOA while producing quality solutions.

II. Bayesian networks as PGM

A Bayesian network is a mathematical structure developed to represent a joint probability distribution by considering a set of variables. Bayesian networks are among the most general probabilistic models for discrete variables used in EDAs [11], [12]. In this paper, we use Bayesian networks to model multinomial data with discrete variables, generating new solutions by using the particular conditional probability [13] described by Equation 1.

$$p(y_m^k|\mathbf{pa}_m^{j,B}) = \theta_{y_m^k|\mathbf{pa}_m^{j,B}} = \theta_{mjk} \tag{1}$$

where $\mathbf{Y} = (Y_1, ..., Y_M)$ is a vector representation of M random variables and y_m its m-th component; B is the structure and Θ a set of local parameters; $\mathbf{Pa}_m^B$ represents the set of parents of Y_m, which $\mathbf{pa}_m^{j,B} \in \{\mathbf{pa}_m^{1,B}, ..., \mathbf{pa}_m^{t_m,B}\}$ denoting a particular combination of values for $\mathbf{Pa}_m^B$, t_m is the total number of different possible instantiations of the parent variables of Y_m given by $t_m = \prod_{Y_v \in \mathbf{Pa}_m^B} s_v$, with s_v defining the total of possible values (states) that Y_v can assume. The parameter

[1] Adjusting a PGM here means generating a new PGM, either based on the previous one or not.

θ_{mjk} represents the conditional probability that variable Y_m takes its $k-$th value (y_m^k), knowing that its parent variables have taken their j-th combination of values ($\mathbf{pa}_m^{j,B}$). This way, the parameter set is given by $\Theta = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_m, ...\boldsymbol{\theta}_M\}$, where $\boldsymbol{\theta}_m = (\theta_{m11}, ..., \theta_{mjk}, ..., \theta_{m,t_m,s_m})$ and M is the total number of nodes in the Bayesian network.

The parameters of Θ and B are usually unknown, and the literature presents two possibilities to estimate them: Maximum Likelihood Estimate (MLE) and the more general Bayesian Estimate [14]. In this work, we address the last method.

In terms of Bayesian network structures learning process, there are mainly three different approaches: score-based learning, constraint-based learning, and hybrid methods [15].

Score-based techniques apply heuristic optimisation methods to sort structures by selecting the one that maximizes the value of a scoring metric, like the K2 metric [8] in Equation 2.

$$p(B|P) = \prod_{m=1}^{M} \prod_{j=1}^{t_m} \frac{(s_m - 1)!}{(N_{mj} + s_m - 1)!} \prod_{k=1}^{s_m} (N_{mjk})! \quad (2)$$

where N_{mjk} is the number of observations in the data set P for which Y_m assumes the k-th value given the j-th combination of values from its parents.

In this paper we consider the K2 algorithm, which is a commonly-used, score-based greedy local search technique that applies the K2 metric (Equation 2). It starts by assuming that a node, in a (pre-defined) ordered list, does not have any parent, then at each step it gradually adds the edges that increase the scoring metric the most, until no edge increases the metric anymore.

III. Revisiting Bayesian optimisation algorithms

In this section, we provide basic concepts used throughout this article and also present some contributions (Sub-sections III-A and III-B) that will support the analysis performed later on.

Among the most general probabilistic models for discrete variables used in EDAs, Bayesian networks play a central role. They are adopted by traditional algorithms like BOA [6] (considered in this work), hBOA [16], EBNA and EBNAK2 [5].

A. Tracing PGM adjustment

In a standard BOA implementation, the PGM is updated in each iteration. If two or more consecutive PGMs are very similar, then the algorithm performs a pointless time consuming task. This is particularly costly for the problem addressed in this paper, as a PGM update is done many times for the multiple ruggedness levels and the multiple landscapes of the NK-models, besides it can be an issue for most of PGM-based approaches.

Aiming to quantify similarities between consecutive PGMs, we use in this work the Structural Hamming Distance (*SHD*) [17] as a metric. At each iteration of BOA, we store the *SHD* value between two consecutive PGM structures, the current and the previous one, generating, at the end of evolution, an *SHD* vector as large as the total number of iterations that were necessary to achieve the stopping criteria.

In order to extract a common pattern across all multiple runs (R) of the algorithm, we need to aggregate these *SHD* vectors. As the resulting *SHD* vectors can be of different sizes due to the convergence speed of each run, we use a methodology to normalise the vector size, which is described in the following steps:

1) Define the normalised size L as the maximum size among all the *SHD* vectors (i.e. the maximum number of iterations among all R runs).
2) For all the *SHD* vectors with size smaller than L, fill-in the gaps by adding a '*' character in a uniform manner.
3) To obtain the final aggregated normalised vector, joint all *SHD* vector obtaining a matrix of size $R \times L$. Then average each column, ignoring '*' elements.
4) Normalise the vector elements into the range $[0, 1]$.

At the end we have the aggregated normalised information stored in a vector ($\overline{SHD}$), which describes in average the pattern of PGM adjustments followed by BOA among R multiple runs. We believe that this approach is a better fit to aggregate vectors of different sizes compared to interpolation.

B. Runtime estimation

In many studies on fitness landscape analysis [18], [19], the number of fitness evaluations is used to estimate the runtime of a given stochastic algorithm.

Let $p_s \in (0; 1]$ be the probability of success of the algorithm and let T_f be the random variable measuring the number of function evaluations for unsuccessful runs.

After $(t-1)$ failures, each one requiring T_f evaluations, with the final successful run of T_s evaluations, the total runtime is then defined as $T = \sum_{i=1}^{t-1} T_f + T_s$, where t is the random variable measuring the number of runs. The random variable t follows a geometric distribution with parameter p_s. By taking the expectation and by considering independent runs for each instance, stopping at the first success, we have:

$$\mathbf{E}[T] = (\mathbf{E}[t] - 1)\mathbf{E}[T_f] + \mathbf{E}[T_s] \quad (3)$$

The expected runtime for successful runs $\mathbf{E}[T_s]$ is estimated as the average number of function evaluations performed by successful runs, and we note T_{max} the expected runtime for unsuccessful runs. As the expectation of a geometric distribution for t with parameter p_s is equal to $1/p_s$, the estimated runtime can be expressed as the following:

$$\mathbf{E}[T] = \frac{1 - \hat{p}_s}{\hat{p}_s} T_{max} + \frac{1}{t_s} \sum_{i=1}^{t_s} T_i \quad (4)$$

where t_s is the number of successful runs, T_i is the number of evaluations for successful run i.

In the context of BOA, it is clear that this approach is far from being accurate. In fact, the process of adjusting the PGM has a higher computational complexity [2] than those of evaluating a population of solutions, even for fast greedy algorithms such as K2. Therefore, we propose an alternative

[2]In terms of complexity theory, finding the optimal PGM structure is NP-complete. However, some real-world problems could require more wallclock time for calculating the objective value of a single solution, in which case surrogate models are often used.

approach specific to our case study based on the expected time complexity instead of the number of evaluations. More precisely, we take into consideration the complexity of both: the cost of the objective function and the cost of generating a new PGM using a given structure learning algorithm like K2.

Let us start by estimating the time complexity of PGM adjustments of BOA. We assume that it generates a Bayesian network based on a given probability distribution $\{p_j\}$. Let U_s and U_f denote the number of PGM adjustment for successful runs and unsuccessful runs respectively. We can define the number of PGM adjustment as $U = \sum_{i=1}^{t-1} U_f + U_s$. Thus, the expected number of PGM adjustment is defined by:

$$\begin{aligned}\mathbf{E}[U] &= \frac{1-\hat{p}_s}{\hat{p}_s} U_f + U_s \\ &= \frac{1-\hat{p}_s}{\hat{p}_s} \sum_{j=1}^{I_{max}} p_j + \frac{1}{t_s} \sum_{i=1}^{t_s} \sum_{j=1}^{I_i} p_j \end{aligned} \tag{5}$$

where I_{max} is the number of iterations of T_{max} and I_i is the number of iterations for successful run i. By noticing that the complexity of evaluating a population of μ solutions, each one represented by an N size vector, is μN and the complexity of the K2 algorithm is $2(N^5 + N^4)$, and by unifying the runtime unit to *number of elementary operation* instead of *number of evaluation*, we obtain the following runtime equation:

$$ert = \mu N \mathbf{E}[T] + 2(N^5 + N^4)\mathbf{E}[U] \tag{6}$$

with sampled population size $\mu = |S|$ and bitstring length N.

IV. Designing a model-based BOA

In this section, we start by analysing the similarity patterns between PGM based on the output of a standard BOA implementation on the NK-landscape model instances [20]. The outcome is then used to propose a model-based algorithm based on the BOA framework that adopts a strategy to reduce the frequency of PGM adjustments.

A. Analysis of PGM adjustment patterns

In Figure 1, we show the evolution of *SHD* values (solid lines) and objective values (dashes lines) for 5 randomly selected runs (each color is associated with a specific run), for each ruggedness level K. The experiments are conducted for different problem sizes ($N = 18$, $K = \{2, 4, 6, 8, 10, 12, 14, 16\}$) and the *SHD* curves (each one with a different convergence speed meaning different number of iterations until find the best solutions) have been fitted with polynomials of degree 6.

The first observation we can make is that BOA converges faster for small values of K. The explanation of this behaviour is straightforward: these instances are easier to solve and therefore require less computational effort to solve. More importantly, we can see that significant improvements in the objective values do not necessarily correspond to significant changes in the PGM. This is our first indicator that adjusting the PGM at each iteration might not be the best strategy to adopt.

Figure 2 represents the evolution of $\overline{SHD}$ for different problem sizes ($N = \{10, 12, 14, 16, 18\}$) and $L = 500$. The number of iterations L and values have both been normalised using the method proposed in Section III.

Looking at the patterns in Figure 2, one would be tempted to use the Boltzmann criterion to decide whether to adjust the PGM or resample from the previous one. A good argument not to use the Boltzmann criterion is that the patterns in Figure 2 do not necessarily embed a Boltzmann distribution. In fact, we can identify a small, but significant, peak towards the end of the evolutionary process. This means that the latest generated PGMs are different from their predecessors. In other terms, the algorithm tends to diversify the search process before convergence by sampling from new probability distributions.

B. FBOA: the proposed algorithm

Our proposed approach is summarised in Algorithm 1. It follows the same steps performed by a standard BOA, the exceptions are the use of a probability vector in line 3 and the decision rule in line 8.

In line 3, FBOA creates the PGM adjustment probability vector (which corresponds to the $\overline{SHD}$ vector for $N = 18$ in our experimental study). The PGM adjustment at a given iteration t is done with a probability p_t as shown in line 8. We will refer to this algorithm as FBOA, which stands for Fast Bayesian Optimisation Algorithm.

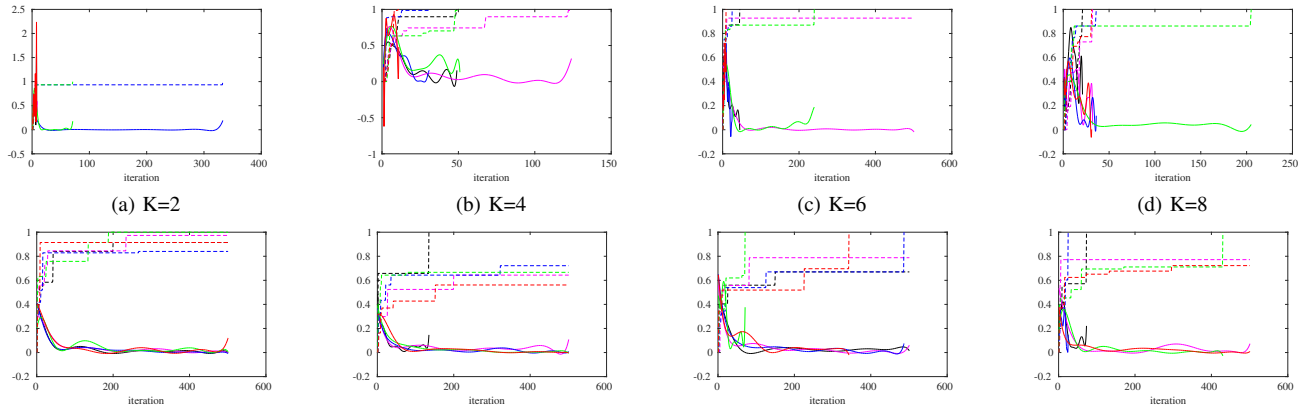
Algorithm 1 FBOA algorithm

1: $t \leftarrow 1$
2: $M^{(1)} \leftarrow initial\_model()$
3: $p_t \leftarrow PGM\_probability\_vector()$
4: **repeat**
5: $\quad S \leftarrow sample\_solutions(M^{(t)})$
6: $\quad F \leftarrow evaluate(S)$
7: $\quad P \leftarrow best\_solutions(S, F)$
8: $\quad$ **if** $random(0, 1) < p_t$ **then**
9: $\quad\quad M^{(t+1)} \leftarrow adjust\_model(P)$
10: $\quad$ **else**
11: $\quad\quad M^{(t+1)} \leftarrow M^{(t)}$
12: $\quad$ **end if**
13: $\quad t \leftarrow t + 1$
14: **until** termination criterion met

V. Experiments and Results

In this section, we are interested on finding out the ability of FBOA (algorithm presented in Section IV-B) to obtain similar performance in comparison with BOA. In particular, we investigate the estimated runtime of FBOA and BOA necessary to compute the optimal solution on particular NK-landscapes instances.

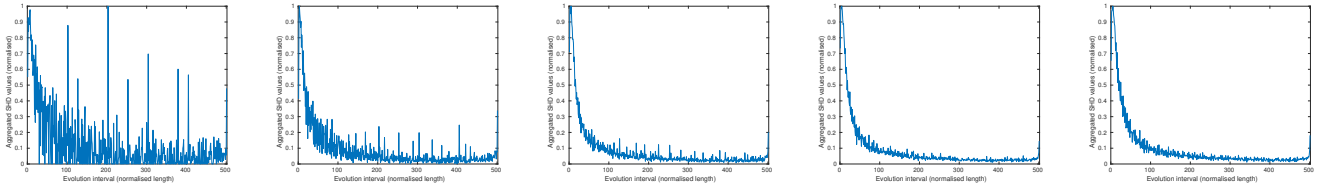
For both algorithms, we consider the population size $\lambda = |P| = 40$ and sample size $\mu = |S| = 100$. The addressed NK-landscapes have problem size $N \in \{10, 12, 14, 16, 18\}$ and epistatic degree $K \in \{2, 4, 6, 8, 10, 12, 14, 16\}$ (as applicable). As usually performed in FLA, we enumerate the solution space exhaustively for each N until the largest value $N = 18$ that can still be analysed with reasonable computational resources. A set of 10 different landscapes are independently generated at random for each N and K. Unless a near-optimal solution is found, a maximum number of evaluations $T_{\max} = 50000$ is used as a stopping criterion. Finally, each algorithm is executed $T_{\text{runs}} = 100$ times per instance.



(a) K=2 (b) K=4 (c) K=6 (d) K=8

(e) K=10 (f) K=12 (g) K=14 (h) K=16

Fig. 1: Illustration of the levels of similarities (*SHD*) in the generated Bayesian networks and the improvement in the objective values at each iteration. $N = 18$.



(a) N=10 (b) N=12 (c) N=14 (d) N=16 (e) N=18

Fig. 2: Bayesian network similarity patterns ($\overline{SHD}$) for different problem sizes. The curves mean values are (a) 0.125512; (b) 0.091312; (c) 0.083780; (d) 0.088092; (e) 0.097964.

A. Comparison of the optimisation results of BOA and FBOA

First, in Table I, we report the average gaps between the optimal objective values (obtained through enumeration) and the best objective values found by BOA and FBOA on average for each combination of K and N. The p-values of the last column of Table I have been obtained using Friedman test [21] as the results are not normally distributed according to the Shapiro-Wilk normality test [22]. The Friedman test has been computed with a confidence level of 99% indicating that there are statistically significant differences whenever p-value < 0.01. According to Table I, there are no statistically significant differences between BOA and FBOA for all instances. It means that BOA and FBOA performances cannot be differentiated based on the average gaps for all considered combinations of N and K.

Table II presents the average success rates for both BOA and FBOA for different settings of K and N. The values represent averages of the number of successful runs over the total numbers of landscapes and runs. The last row of a given N reports the average success rate over all K. We apply the McNemar Chi-Square statistical test [23] to compare BOA and FBOA approaches. This test is useful to show the difference between paired proportions and can determine whether there is marginal homogeneity between the two approaches.

There are a few statistically significant differences between success rates of the approaches when p-value < 0.01 (confidence level of 99%). They are highlighted in Table II. We can observe from Table II that there are no statistically significant differences for 26 of 30 NK-landscape configurations, except for those with $N = 16$ ($K = 8$ and $K = 10$), and

TABLE I: Average gap between the optimal and best objective values for BOA and FBOA

N	K	**BOA**	**FBOA**	**p-value**
10	2	0.0336	0.0319	0.8319
10	4	0.0531	0.0488	0.8488
10	6	0.0232	0.0196	0.8196
10	8	0.0235	0.0231	0.8231
12	2	0.0383	0.0381	0.8381
12	4	0.0299	0.0303	0.8303
12	6	0.0297	0.0245	0.8245
12	8	0.0145	0.0145	0.8145
12	10	0.0252	0.0255	0.8255
14	2	0.0376	0.0342	0.8342
14	4	0.0302	0.0319	0.8319
14	6	0.0204	0.0213	0.8213
14	8	0.0253	0.0250	0.8250
14	10	0.0355	0.0347	0.8347
14	12	0.0272	0.0287	0.8287
16	2	0.0290	0.0316	0.8316
16	4	0.0293	0.0320	0.8320
16	6	0.0180	0.0191	0.8191
16	8	0.0309	0.0312	0.8312
16	10	0.0455	0.0452	0.8452
16	12	0.0291	0.0296	0.8296
16	14	0.0386	0.0366	0.8366
18	2	0.0252	0.0203	0.8203
18	4	0.0201	0.0193	0.8193
18	6	0.0232	0.0245	0.8245
18	8	0.0373	0.0364	0.8364
18	10	0.0349	0.0348	0.8348
18	12	0.0452	0.0443	0.8443
18	14	0.0424	0.0406	0.8406
18	16	0.0529	0.0511	0.8511

$N = 18$ ($K = 14$ and $K = 16$) with some advantage for BOA. By considering the average over all K for each N (last row of each N), there is no statistically significant differences between the approaches. It means that BOA and FBOA performances cannot be differentiated regarding the success rate among different values of N.

TABLE II: Average success rates for BOA and FBOA. Four cases with statistically significant differences are highlighted.

N	K	**BOA**	**FBOA**	**p-value**
10	2	1.0000	1.0000	0.9822
10	4	0.9990	0.9990	0.9822
10	6	0.9690	0.9740	0.9277
10	8	0.9950	0.9950	0.9821
10	all	0.9908	0.9920	0.9215
12	2	1.0000	1.0000	0.9822
12	4	0.9820	0.9830	0.9987
12	6	0.8980	0.9030	0.9249
12	8	0.8570	0.8550	0.9807
12	10	0.9280	0.9040	0.5910
12	all	0.9330	0.9290	0.8548
14	2	0.9990	1.0000	0.9898
14	4	0.9000	0.9300	0.4978
14	6	0.7140	0.6630	0.1778
14	8	0.5090	0.5350	0.4391
14	10	0.4790	0.4570	0.4925
14	12	0.4200	0.4870	0.0284
14	all	0.6702	0.6787	0.6210
16	2	0.9950	0.9960	0.9899
16	4	0.8620	0.8560	0.9040
16	6	0.7090	0.7390	0.4460
16	8	0.4300	0.2720	***0.0000***
16	10	0.3280	0.2600	***0.0057***
16	12	0.1960	0.1910	0.8389
16	14	0.2330	0.2180	0.5097
16	all	0.5361	0.5046	0.0512
18	2	0.9720	0.9760	0.9458
18	4	0.8340	0.7310	0.0100
18	6	0.6440	0.5550	0.0110
18	8	0.4260	0.4070	0.5328
18	10	0.1610	0.1090	0.0191
18	12	0.0480	0.0640	0.1564
18	14	0.0800	0.0430	***0.0012***
18	16	0.1010	0.0570	***0.0006***
18	all	0.4083	0.3678	0.0190

B. Comparison of the estimated runtimes of BOA and FBOA

In this section, we are interested on comparing the estimated runtimes of BOA and FBOA for different values of K for each problem instance. Using the new runtime metric ert calculated by Equation (6), the comparison is conducted under FLA basis since we investigate the correlation between landscape ruggedness parameter K and ert, using a linear regression model built according to Equation (7)

$$ert = \beta_0 + \beta_1.v_1 + e \tag{7}$$

with response variable ert, explanatory variable $v_1 = K$, and error e. A log-transformation is applied for both variables to better approximate linearity. The accuracy of the linear regression model is measured by the coefficient of determination r^2 which ranges from 0 to 1.

Table III shows the coefficient of determination for BOA and FBOA. The average distances between the respective linear regression curves are not log-transformed. According to Table III, well-adjusted linear regressions are obtained for all N but $N = 10$. For the best adjusted regression model $N = 18$ (greatest r^2 for both algorithms), FBOA is 3.8 times faster than BOA.

TABLE III: Coefficient of determination r^2 for BOA and FBOA, and average distance between linear regression curves

N	r^2		**Average distance**
	BOA	**FBOA**	
10	0.4650	0.5963	1.3998
12	0.7531	0.6987	2.5485
14	0.8643	0.8243	4.0443
16	0.8750	0.8360	3.6310
18	0.8949	0.8940	3.7509

Figure 3 shows both scatter plots and linear regression curves for different values of N in a log-log scale. Each combination of N and K has 10 values corresponding to ten different landscapes, and each landscape has a corresponding ert given by Equation (6). According to the regression curves of Figure 3, FBOA is always faster than BOA. It is worth noticing that runtimes increase faster for BOA than for FBOA.

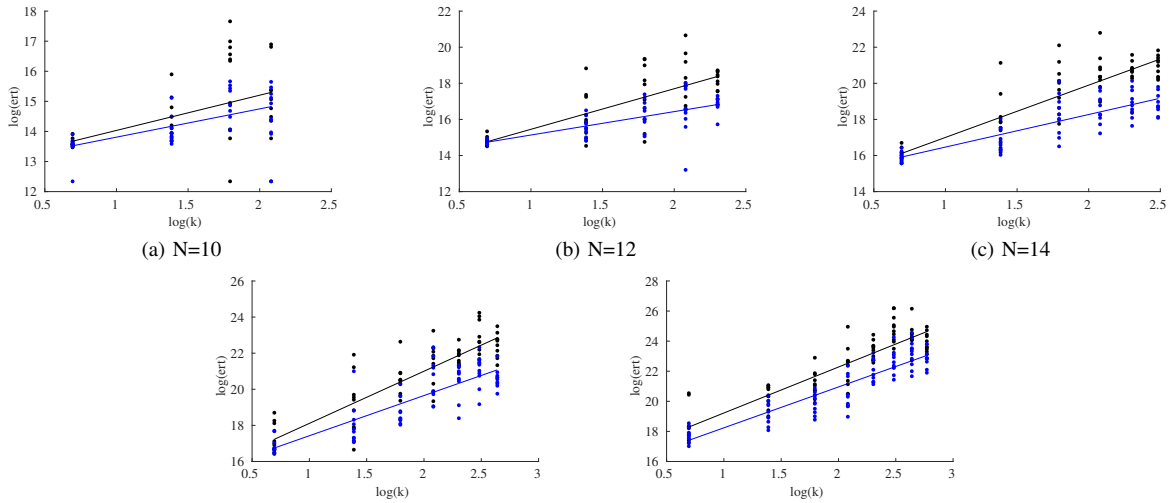
The results shown in Figure 3 and Table III might indicate that the computational effort saved with PGM adjustments could be beneficial as the problem difficulty increases (by increasing N and K). We do not claim that FBOA guarantees the best tradeoff between optimization results and runtime; this requires further investigation. However, the experiments show clearly that adjusting the PGM of FBOA less often than required by BOA yields a significant runtime improvement without noticing any statistically significant differences between gaps obtained from BOA and FBOA to the optimal solutions.

VI. Conclusions and future directions

In this study, we investigated similarity patterns between Bayesian networks of two consecutive BOA iterations using the Structural Hamming Distance (*SHD*) metric, i.e., a proxy to measure a similarity between two networks over the evolutionary process of BOA. The experiments clearly show patterns across a wide range of NK-landscapes in which SHD values decrease during BOA evolution. It means that the generation of a new Bayesian network at every iteration of BOA might not be necessary – except for an increase again towards the end that is necessary for exploitation.

Based on this observation, we proposed a faster alternative called FBOA which conducts adjustments of the PGM according to a probability computed as function of FBOA iterations. Furthermore, we tested the performance of FBOA in terms of solution quality and computational burden. The experiments performed with concepts of fitness landscape analysis demonstrate that FBOA provides solutions comparable to BOA in terms of quality while dramatically saving computational time.

For further improvement of FBOA performance, one option is to incorporate information about the objective improvement



(a) N=10 (b) N=12 (c) N=14

(d) N=16 (e) N=18

Fig. 3: Correlation between K and ert for BOA (black) and FBOA (blue) (in log-log scale)

into the decision by either adjusting the corresponding PGM or resampling it from the previous one. For example, we can employ self-adaptive methods that (i) enforce the generation of a new PGM or (ii) increase the probability of generating a new PGM, if the best found objective value is not improved by sampling the current PGM. In the future, we plan to explore the efficiency of such strategies when tackling real-world problems of large size. This is mainly to ensure the scalability of the proposed approach and its applicability in real-world situations.

Although we have proposed an alternative implementation of BOA, this work is mainly to show that the frequency of adjusting the PGM can be tuned to significantly save computational time.

REFERENCES

[1] D. Koller, N. Friedman, and F. Bach, Probabilistic graphical models: principles and techniques. MIT press, 2009.

[2] Y. Cheng, I. Diakonikolas, D. Kane, and A. Stewart, "Robust learning of fixed-structure bayesian networks," in NeurIPS, 2018, pp. 10 304–10 316.

[3] M. J. Wainwright, M. I. Jordan et al., "Graphical models, exponential families, and variational inference," Foundations and Trends® in Machine Learning, vol. 1, no. 1–2, pp. 1–305, 2008.

[4] H. Mühlenbein and G. Paab, "From Recombination of Genes to the Estimation of Distributions I. Binary parameters," in Parallel Problem Solving from Nature, ser. PPSN IV - Lecture Notes in Computer Science 1411. London, UK, UK: Springer-Verlag, 1996, pp. 178–187.

[5] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in Proceedings of the Second Symposium on Artificial Intelligence, ser. CIMAF'99. Havana, Cuba: Editorial Academia, 1999, pp. 332–339.

[6] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO'99, vol. I. San Francisco, CA: Morgan Kaufmann Publishers, 1999, pp. 525–532.

[7] D. Heckerman, D. Geiger, and D. Chickering, "Learning Bayesian networks: the combination of knowledge and statistical data," Machine Learning, vol. 20, no. 3, pp. 197–243, 1995.

[8] G. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," Machine Learning, vol. 9, no. 4, pp. 309–347, 1992.

[9] G. Bresler, "Efficiently learning ising models on arbitrary graphs," in Proceedings of the forty-seventh annual ACM Symposium on Theory of Computing (STOC). ACM, 2015, pp. 771–782.

[10] J.-P. Watson, L. D. Whitley, and A. E. Howe, "Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search." J. Artif. Intell. Res.(JAIR), vol. 24, pp. 221–261, 2005.

[11] P. Larrañaga and J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation. Netherlands: Springer, 2002, vol. 2.

[12] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques. Cambridge, MA: The MIT Press, 2009.

[13] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in Machine Intelligence and Pattern Recognition, vol. 5. Elsevier, 1988, pp. 149–163.

[14] E. Bengoetxea, "Inexact Graph Matching Using Estimation of Distribution Algorithms," PhD Thesis, University of the Basque Country, Basque Country, 2002.

[15] C. Yuan and B. Malone, "Learning Optimal Bayesian Networks: A Shortest Path Perspective," Journal of Artificial Intelligence Research, vol. 48, no. 1, pp. 23–65, 2013.

[16] M. Pelikan, "Analysis of estimation of distribution algorithms and genetic algorithms on NK landscapes," in Proceedings of the 10th annual conference on Genetic and Evolutionary Computation, ser. GECCO'08. Atlanta, Georgia: ACM, 2008, pp. 1033–1040.

[17] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," Machine Learning, vol. 65, no. 1, pp. 31–78, 2006.

[18] A. Liefooghe, S. Verel, F. Daolio, H. Aguirre, and K. Tanaka, "A feature-based performance analysis in evolutionary multiobjective optimization," in International Conference on Evolutionary Multi-Criterion Optimization. Guimaraes, Portugal: Springer, 2015, pp. 95–109.

[19] M. S. Martins, M. El Yafrani, R. Santana, M. R. Delgado, R. Lüders, and B. Ahiod, "On the performance of multi-objective estimation of distribution algorithms for combinatorial problems," in IEEE Conference on Evolutionary Computation, ser. CEC'18, 2018, pp. 1–8 in arXiv:1806.09 935.

[20] S. A. Kauffman, The origins of order: Self-organization and selection in evolution. USA: Oxford University Press, 1993.

[21] S. Siegel and N. Castellan, "The friedman two-way analysis of variance by ranks," Nonparametric statistics for the behavioral sciences, pp. 174–184, 1988.

[22] W. Conover, Practical Nonparametric Statistics, 3rd ed. New York: Wiley, 1999.

[23] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," Psychometrika, vol. 12, no. 2, pp. 153–157, 1947.