An Automatic Algorithm Configuration based on a Bayesian Network

Marcelo Branco do Nascimento
Univ Fed of São Paulo
São José dos Campos, Brazil
marcelo.branco01@unifesp.br

Antonio Augusto Chaves
Univ Fed of São Paulo
São José dos Campos, Brazil
antonio.chaves@unifesp.br

***Abstract*—The parameter tuning process is one of the main tasks in the development of metaheuristics. The appropriate parameter can assist in finding good solutions to combinatorial optimization problems. However, finding a good parameter setting is a hard task. It involves understanding how the relationship between parameters connects to the problem scenario. This article proposes a method to automatically tune parameters of metaheuristics, called the Bayesian Network Tuning (BNT). Our goal is to develop an efficient method in terms of solution quality and computational time, which can find configurations that support metaheuristics in solving optimization problems. In order to evaluate this method, a Biased Random-Key Genetic Algorithm (BRKGA) was implemented to solve the Bin Packing Problem. The BRKGA was tuned with our method and other tuning methods found in the literature. A comparison of the results shows that the proposed method found good solutions and was competitive in relation to the other tuning methods.**

***Index Terms*—Heuristics methods, Parameter tuning, Bayesian Network**

I. Introduction

The proper choice of parameter setting is an important step in the development of optimization algorithms that are generally designed to be flexible and robust for different problem scenarios. But, some metaheuristic algorithms have a set of parameters on which they are extremely dependent. Usually, these parameters are not robust and require a specific configuration according to the problem or even an instance of the problem under analysis.

Selecting optimal parameter settings is often done manually or heuristically by tweaking parameters at runtime, but without the proper experimentation to support or confirm the intuitive choice. An inappropriate selection of parameters can result in various drawbacks, like stagnation on the local optimum and a long search time. The task of finding these appropriate settings to a metaheuristic is called parameter tuning [1], [2].

Many strategies have been proposed in the literature to find the best parameters of a metaheuristic. A simple strategy is brute force. This strategy performs the same number of experiments for each configuration of a solution space. Only part of the information is stored, using a single vector with the performance estimation of each candidate configuration at different instances, whose size corresponds to the total number of configurations. Therefore, it is very intuitive to solve parameter tuning problems, but can be very costly in most cases.

Parameter tuning automation has several implications, such as the reduction in computational time and the possibility of finding better results more often than when the algorithm is executed with manual tuning.

The search for best parameters has led to several studies to find algorithmic alternatives in order to automate this process. These tuning strategies may be grouped according to when the parameters are adjusted:

- *offline tuning*: metaheuristic parameters such as chromosome size, crossover and mutation rates in a genetic algorithm are set up before initializing the algorithm and remain immutable during the search process.
- *online tuning*: metaheuristic parameters are configured adaptively during the search process of the algorithm.

The offline tuning has a high computational cost, but because it is applied to each instance or instance group of a problem, quality solutions can be guaranteed. The online tuning strategy is intended to generally solve a single, typically large and complex instance of a problem [3]. Both strategies are attractive. However, this article focuses on scenarios where a large number of instances of a problem are evaluated, so that the acquired data can be compared with the results of other parameter tuning algorithms in the literature. Thus, an offline strategy was adopted in our research.

Metaheuristics for parameter tuning offers the advantage of robustness when applied over unknown scenarios. The approach of metaheuristics for parameter adjustments is appropriate to avoid information specialization, since metaheuristics are independent of the target algorithm (other metaheuristic or algorithm that is tuned).

An offline metaheuristic tuning framework has a standard strategy in which the parameter domains and starting values are passed to the configurator, which, in turn, passes different parameter settings to run in the target algorithm. A set of instances are tested by the target algorithm in order to return the cost solution to the configurator, to calculate the cost of the data and update the settings, as shown in Fig. 1.

The quality of solution cost is important for analyzing tuning performance. It is usually measured by the objective function of the problem, while the quality of the algorithm is determined by a metric defined by the configurator (tuning method).

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

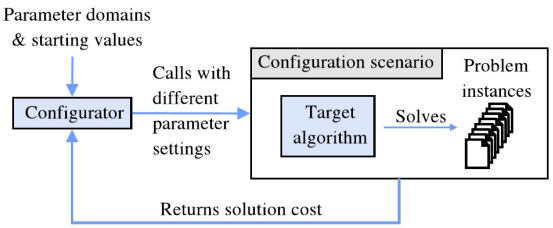


Fig. 1. Metaheuristic tuning framework [2].

Related work of offline parameter tuning includes iRace presented by López-Ibáñez et al. [4] based on race methods, with support for numeric and symbolic parameters. It uses a Model-Based run method along with Friedman's test to select good settings. If the test value exceeds a significance level, then the null hypothesis is rejected, so there is evidence that at least one candidate configuration performs better. Eiben and Nannen [5] present the REVAC (Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters). This method tries to show the relevance of each parameter using Shannon's entropy from information theory and a probability distribution is formed based on the parameter values. The method then tries to estimate the best parameters based on this distribution and on maximizing Shannon's entropy. Adenso-Díaz and Laguna [6] present the Calibra. It uses an orthogonal matrix from Taguchi $L_9(3^4)$ for sampling values. This matrix is based on a factorial model [7], which tries to faithfully represent, through little data, a whole set of significant information in a space of solutions. It is a strategy using continuous numerical parameters along with a method of estimating the relevance of each, also through Shannon's entropy. ParamILS was proposed by Hutter et al. [2], being a modified version of the ILS for parameter tuning. This algorithm uses the neighborhood information of the configuration found in the current iteration, to improve the solutions. This neighborhood considers the variation of only one parameter at a time. Roman et al. [8] propose an approach using Bayesian Optimization. The algorithm uses Mallow models and kernel function (Matern $5/2$). At each iteration, this function is applied to the configuration population to create probabilistic models. Barbosa and Senne [1] propose the Heuristic Oriented Racing Algorithm (HORA). It uses Design of Experiments along with a running method that employs a non-standard test.

Many offline tuning methods try to find good parameters in an acceptable runtime. An analysis of the problem context and the target algorithm can indicate possible relationships of the parameters with the scenario under study. This can assist in the search for good parameters. However, performing this analysis can be a complex task and demand a lot of effort on the part of the algorithm designer, when elaborating metaheuristics for different problem scenarios. Thus, several researchers have proposed different techniques for parameter tuning, developing new metaheuristics, or adapting classic strategies. Among the different techniques covered in the literature, we highlight the Estimation Distribution Algorithms (EDAs) [9]–[11].

EDAs are approaches that manage to create a statistical model of the problem scenario automatically, taking into account the relationships between the elements involved (parameters, strategies, among others) in the treated context. They use statistical models to represent a set of instances. For our research, BOA (Bayesian Optimization Algorithm) [12] was the EDA chosen. The benefits of its application are great, especially the treatment between symbolic parameters. It is important to note that the technique used by BOA, which creates Bayesian networks to find the best solution, is different from Bayesian optimization techniques (use Gaussians) [13].

This article proposes an efficient offline parameter tuning method using Bayesian Networks, taking into account the computational time and the solution quality, in order to develop a method to assist metaheuristics in solving optimization problems. This method was called BNT (Bayesian Network Tuning), since BOA is used for parameter tuning. The contribution of this research results in the reduction of computational cost in relation to the number of iterations necessary for the execution by the target metaheuristic and in a more detailed statistical analysis of the scenario which takes into account additional information from the generated stochastic models.

To evaluate the BNT, we use it to tune a Biased Random-Key Genetic Algorithm (BRKGA) [14] designed to solve the Bin-Packing Problem (BPP) [15]. Different groups of instances of BPP are tested to analyze the results. The validation of the BNT method consists in the comparison with other tuning methods found in the literature, such as Calibra, ParamILS and iRace. A statistical analysis shows the robustness of the BNT.

This article is organized as follows. In Section II we present the proposed method for offline parameter tuning using Bayesian Networks. Section III details the BRKGA method and its parameters. Section IV presents a validation test for the proposed method using a BRKGA for solving the Bin Packing problem, highlighting how BNT is applied to tuning the parameters of the BRKGA and the computational results. Section V concludes this article.

II. Bayesian Network Tuning

In order to establish relations of complex dependencies between parameters we proposed the Bayesian Network Tuning (BNT), which use Bayesian Networks for parameter tuning. Considering that each node of a Bayesian Network represents a parameter and each edge represents a connection between parameters, we can define a BNT solution as a parameter configuration that has the maximum joint probability distribution function in relation to the network. Algorithm 1 shows a pseudo-code of the BNT.

A scenario problem (Sc) consists in the description of the parameters of the target algorithm, the number of training instances, and the type of seed (fixed or random). In addition to the problem scenario, the BNT input data has the number of iterations (T), the size of the promising population (P_c) and the size of the population (P_b). Although these parameters need to be set by users, this process is not complex and some

Algorithm 1 Bayesian Network Tuning

Input: number of iterations T, size of promising population $|P_c|$, size population $|P_b|$.
Output: parameter settings θ, Bayesian Network graph Ω.

procedure BNT(T, $|P_c|$, $|P_b|$)
  Read scenario problem Sc
  Construct population P_b
  Calculate cost ϕ for each solution
  while $T \neq 0$ **do**
    Order population by solution cost
    Select the most promising solutions (P_c)
    Ω = CREATEBAYESIANNETWORK(P_c, Sc)
    Generate new solutions from Ω
    Update population P_b
    Set $T = T - 1$
  end while
  return tuple (θ, Ω).
end procedure

Algorithm 2 Create Bayesian Network

Input: population P_c, scenario problem Sc
Output: Bayesian Network graph Ω

procedure CREATEBAYESIANNETWORK(P_c, Sc)
  Set Ψ = max number of edges
  Set np = number of parameters of Sc
  while $\Psi \neq 0$ **do**
    for $k \leftarrow 1, ..., np$ **do**
      Compute all gains of add edge in Ω using K2
    end for
    Set ψ = get edge with maximum gain
    if not $\psi \in \Omega$ **then**
      Add new edge in Ω
      Set $\Psi = \Psi - 1$
    else
      Set $\Psi = 0$
    end if
  end while
  return Ω
end procedure

default values can be used to run the BNT. We recommend to set $T = 5 \times$ number of tuning parameters, $P_b = 10 \times$ number of tuning parameters, and $P_c = 0.3 \times P_b$. Users can also set a maximum runtime for the BNT.

In Algorithm 1, the BNT initially creates a population of solutions P_b. These solutions are generated by selecting a value for the parameters at random, within a set of possibilities described in Sc. After creating the initial population, the solutions are sorted by cost ϕ. This cost is based on the objective functions found by the target algorithm with the value parameters of a solution. The most promising solutions (P_c) are selected, for $|P_c| < |P_b|$. We consider that solutions with minimum cost are promising. The Bayesian Network construction process (Algorithm 2) begins with the $|P_c|$ solutions selected. After building the Ω network, the next steps of BNT are to generate new solutions based on sampling the probability distribution of its topology and update the worst solutions of the population P_b with $0.3 \times |P_b|$ new solutions. The BNT ends when it reaches the maximum number of iterations (T).

We use a metric to create the Bayesian Network that evaluates the structure of the dependencies between the parameters that best fits the data for a given problem to add new edges to the network. It is considered a complex problem to find the best network structure for a set of parameters whose runtime can vary proportionally according to the number of available parameters in the problem. Among the existing metrics we can highlight the K2 metric [16]. One of the advantages of this metric is that it does not depend on a priori information of the data to build the network, i.e. there is no information initially about which dependencies exist between the parameters.

The K2 metric performs a greedy search between the data and creates a gain matrix with the given values. The edges that have the highest score are added to the network. This strategy makes it possible to analyze the joint probability of the network, calculating on each generation a priori knowledge about the structure and conditional probabilities of the network. An example of a gain matrix can be seen in Figure 2. Each row and column defines a parameter. Value (score) in Line 1 and Column 2, for example, represents the gain that adds an edge between the two selected parameters. The quality of each edge directly influences the estimation of values. This metric returns a cost to know which edges are to be inserted into the network.

	1	2	3	4
1	-1.0	0.0687	0.0605	0.0136
2	0.0002	-1.0	0.0932	0.0324
3	0.0804	0.0001	-1.0	0.0234
4	0.0027	0.0008	0.0772	-1.0

Fig. 2. Example of gain matrix to four parameters

The Bayesian network is constructed during the tuning process. In Algorithm 2, we set the maximum number of edges (Ψ) that are added in each iteration of the BNT. The value used was $3 \times$ number of parameters. The gain matrix is generated using the K2 metric and a new edge with maximum gain (ψ) is inserted in the Ω network, if and only if this edge does not exist on Ω. This algorithm stops when the maximum number of edges are added or when a new edge is not added in Ω.

The generation of new solutions in an iteration of the BNT begins by defining the order in which the nodes (parameters) are explored in the Ω network. In the BNT the exploration begins at the first independent node found and continues following the order of the parameters defined in the Sc scenario.

Figure 3 shows an example of a population with four solutions (vectors), four parameters (X_i), and the Ω network structure. This information is used to calculate the probabilities of each parameter in relation to the discrete values defined

in the Sc scenario. The main function of the Ω network is to guide what values new solutions may have considering the probability scenario and the population. The conditional probabilities for all parameters and values are calculated. The parameter values are inferred in new solutions from these probabilities.

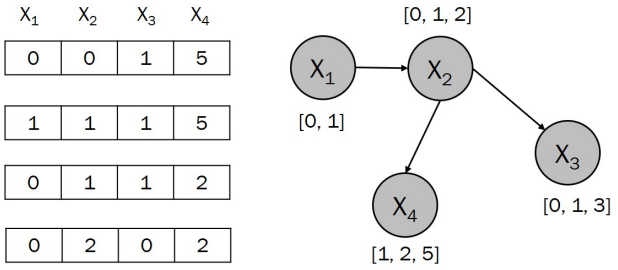


Fig. 3. Example of generation of new solutions

In the example shown in Fig. 3, X_1 is an independent node and assumes a value of 0 or 1. Thus, we calculate the probabilities of X_1: $P(X_1 = 0) = 0.75$ and $P(X_1 = 1) = 0.25$. The next step is to generate a vector of accumulated probabilities, $Vec_{P(X_1)} = [0.75, (0.75 + 0.25)] = [0.75, 1.00]$. Therefore, the value of X_1 in a new solution is defined from a random number in the range $[0, 1]$. For example, a random value of 0.8 indicates that X_1 assumes 1 in the new solution. The algorithm continues from parameter X_2, which depends on the value of X_1. The probability of X_2 is $P(X_2|X_1)$. If X_1 in the new solution is 1, then X_2 considers $P(X_2|X_1 = 1)$, and the process is repeated. We observe that X_3 and X_4 can assume values that are not in the population. In this case, we add a perturbation to increase the diversity in which at least one occurrence of the missing value is added to the probability vector.

The accumulated probabilities are calculated once for each iteration of the BNT and used to generate new solutions. BNT tends to conserve the values with higher probabilities in the new solutions, and the probability of these values are extracted from the best solutions of the population.

To simplify the use of BNT, we proposed an Application Programming Interface (API) for BNT. The API is implemented in C++ and is portable. The user only needs to inform the target algorithm and the training instances. The API is open source and can be downloaded from https://github.com/MarceloBRN/BayesianNetworkTuning.

III. Biased Random-Key Genetic Algorithm

The Biased Random-Key Genetic Algorithm (BRKGA) [14] is a class of algorithms that uses the evolutionary elements of genetic algorithms to solve combinatorial optimization problems in which solutions can be represented as permutation vectors. A random-key is a random real number in continuous interval [0,1].

In order to perform operations with these vectors, a solution of the optimization problem is mapped to a vector of random-keys. In addition, there is a decoder that has the function of re-mapping the vector of random-keys to find solution. The decoding process of BRKGA sorts by elements of key vectors to generate a permutation corresponding to the indexes of ordered elements.

After the creation of the population with P random-key vectors of size n, the Darwinian principles of elitism are applied. The most adapted individuals in the population are more likely to survive in next iterations, in addition to perpetuating their genetic material for future generations.

In each generation, the individuals of the population are divided into elite (P_e) and non-elite vectors ($\bar{P_e}$). All elite vectors are copied to the next generation population. Thus P_m mutants vectors, with randomly generated keys, are introduced into the population. Finally, $|P| - |P_e| - |P_m|$ offspring vectors are added. These offspring vectors are generated by the combination of one parent of the elite and one parent of the non-elite vectors which are randomly chosen. Each offspring vector is created by the parameterized uniform crossover [17].

The crossover scheme chooses two vectors, b (elite) and c (non-elite), to be parents of a (offspring), where the i^{th} component of the offspring vector $a[i]$ receives the i^{th} key $b[i]$ with probability $\rho_e > 0.5$ and $c[i]$ with probability $1 - \rho_e$. The set of offspring will most likely inherit the features of the keys of the elite parent. An important factor is that this behavior is independent of the optimization problem.

The BRKGA parameters can assume quantitative values of many different levels and ranges. The offline tuning method must perform a parameter setup for each training instances. Therefore, it is computationally costly to work with real numbers.

The first thing to do is to infer a set of values for each parameter. Prasetyo et al. [18] suggest ranges of values for BRKGA parameters, as shown in Table I. For our research, at this stage a uniform distribution was applied in the intervals to discretize the parameters.

TABLE I
Parameters and recommended values of BRKGA

Parameter	Description	Recommended value
P	size of population	$k * n \| k \in \{1, ..., 5\}$
P_e	size of elite population	$0.10P \leq P_e \leq 0.25P$
P_m	size of mutant population	$0.10P \leq P_m \leq 0.30P$
ρ_e	elite allele inheritance probability	$0.50 < \rho_e \leq 0.80$

IV. Study case of BNT

To validate the efficiency of the proposed BNT method in relation to other tuning methods in the literature, we analysed one optimization problem to be solved by a metaheuristic. In this research, the problem selected was the Bin Packing Problem (BPP) [15] and the metaheuristic chosen to be calibrated was the BRKGA. In this section we present a description of the BPP and an explanation of how BRKGA is applied to BPP, because it is important for understanding the analysis of the tuning method results as well as for assessing the impact of tuned parameters on BRKGA performance.

A. Bin Packing Problem

The Bin Packing Problem (BPP) is a combinatorial NP-hard problem [19]. Several recent studies can be found in the literature with BPP and its variants [15], [20]–[22]. This article discusses the one-dimensional (1D) version.

The BPP 1D consists of distributing a list of items with associated weights in a set of bins in order to minimize the number of bins used. Bins have a fixed capacity and all items need to be packed without violating the capacity constraint.

Regardless of the BPP objective function, there are some classic approaches to finding BPP solutions. Among the most known, we can mention: First Fit and Best Fit. The First Fit method analyzes the bins in ascending order and places the item in the first free bin that has enough space to allocate it. A new bin is added only if the item does not fit in any of the existing bins. The Best Fit technique checks bins in ascending order and places the item in the bin that is free and has the smallest possible space to allocate it. A new bin is added if the item does not fit into any of the available bins [23].

Figure 4 illustrates the behavior of the two approaches for handling an item vector. The value at each position of the vector represents the weight of each item. The optimal solution occupies a maximum of 3 containers. It can be seen that in Containers 2 and 3, First Fit and Best Fit allocated items in different ways. This example shows that the allocation methodology can influence the search for an optimal solution. After some empirical tests we decided to use the First Fit strategy in the BRKGA. Although, this configuration could also be a parameter of the tuning method.

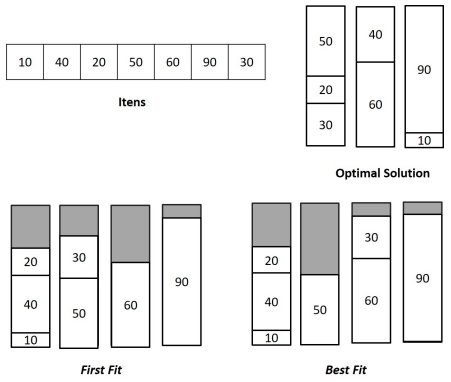


Fig. 4. Illustration of First Fit and Best Fit Strategies for a set of items

Since BPP 1D is a classic problem, several methodologies (exact and heuristic methods) was proposed to solve it [19], [24]–[28]. For BRKGA, some algorithms was presented about variants of BPP [29]–[31].

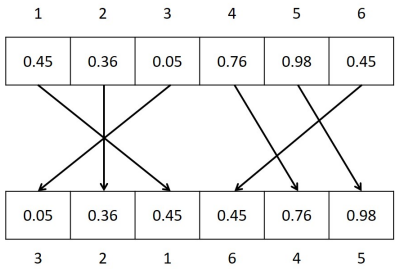
B. BRKGA design to BPP 1D

One of the challenges of solving BPP 1D is determining in which order items arrive to be allocated to bins. The main idea is to help the BRKGA in determining the item order. It can be established that each BRKGA vector is represented by a continuous number vector, whose vector size (n) corresponds to the number of items to be allocated in the bins.

To determine the order of arrival of the items, it is necessary to apply the decoder that will handle the inherent information of the random-key vector.

The decoder chosen for evaluation is based on the increasing ordering of the elements of this vector, with the detail that the order of indices must accompany this ordering, as shown in Fig. 5. The ordering of the indexes will determine the order of arrival of the items to the bins.

Fig. 5. BRKGA vector design for problem



In an instance of BPP 1D, items are usually arranged in a fixed sequential manner (not necessarily the order of arrival). The indexes found by the decoder correspond to the indexes of this sequence. For the example in Fig. 5 the sequence is $\{3, 2, 1, 6, 4, 5\}$.

After finding the order of arrival, we use the First Fit strategy to allocate the items. The fitness of a solution of the BRKGA is the number of used bins plus the lowest occupancy rate of the bins.

In addition to the recommended parameters of the BRKGA (P, P_e, P_m, and ρ_e), the maximum number of generations (n_{gen}) was also tuned in this research.

C. Description of Experiments

We compare our BNT with three classical tuning methods: Calibra [6], ParamILS [2], and iRace [32]. All tests were performed in the same execution environment on a computer with AMD Ryzen 7 3700X 3.60GHz configuration with 16GB of RAM, using Windows 10. Calibra only runs on the Windows Operating System. Therefore, we decided to use Windows in all the computational tests, so that there was no differentiation between the execution environments.

As shown in Table I, each parameter of the BRKGA has a range of recommended values. Since ParamILS and BNT work with discrete sets, these quantitative ranges were divided into values based on a uniform distribution, as shown below:

- size population $P = n * k | k \in \{1, 2, 3, 4, 5\}$
- elite set $P_e \in \{0.10, 0.125, 0.15, 0.175, 0.20, 0.225, 0.25\}$
- mutants set $P_m \in \{0.10, 0.125, 0.15, 0.175, 0.20, 0.225, 0.25, 0.275, 0.30\}$;
- $\rho_e \in \{0.50, 0.525, 0.55, 0.575, 0.60, 0.625, 0.65, 0.675, 0.70, 0.725, 0.75, 0.775, 0.80\}$;
- generations $n_{gen} \in \{200, 250, 300, 350, 400, 450, 500\}$;

where the value of k indicates a factor that multiplies the number of items in BPP instance (n). The result of this multiplication determines the population size of the tuned BRKGA. For example, if the number of items is 100 and the factor is 3, then the population will have 300 individuals.

The test instances were divided into Falkenauer (160 instances) [33], Hard28 (28 instances) [34], Scholl (1210 instances) [35], Schwerin (200 instances) [36] and Waescher (17 instances) [37]. In total, 1615 instances available in the literature[1] were tested. The optimal solutions of these instances were proven in [38]

Training time took an average of 12 hours for each tuning method. Cross-validation was used with a partition of 500 training instances that were randomly selected out of the set of 1615 instances. The set of test instances included all instances.

After each tuning process to find the best parameters configuration, 50 runs were made on the 1615 test instances for each of these configurations.

Another important issue to be considered is that in addition to the tuning methods running at the same computational time, a fixed deterministic seed was used in the tests to generate random numbers in each run. For example, for the first test, the seed was 1 for all instances; for the second test, the seed was 2, and so on. This scheme aims to capture possible information of similarity of behavior.

Table II shows the values of the BRKGA parameters found by the tuning methods. The best settings of iRace and BNT have a larger population size (k) compared to the settings of Calibra and ParamILS. However, the maximum number of generations (n_{gen}) obtained by BNT was the lowest. The other parameters were similar, except for Calibra, which found a small elite partition (P_e) and did not privilege information from elite parents. ($\rho_e = 0.5$).

TABLE II
VALUES OF THE PARAMETERS FOUND BY THE TUNING METHODS

	k	P_e	P_m	ρ_e	n_{gen}
Calibra	2	0.120	0.250	0.500	238
ParamILS	3	0.200	0.300	0.625	450
iRace	5	0.232	0.288	0.617	359
BNT	5	0.250	0.250	0.650	200

The average value of the objective function found in the 50 runs of the BRKGA was calculated. The standard deviation and mean of objective function were similar to all versions of BRKGA. In other words, the BRKGAs exhibited a well-defined behavior to solve the BPP and was always able to find good solutions with the parameters provided by the tuning methods.

Table III presents the average objective function for the five instance groups and Figure 6 shows the box-plot with the average solutions of all instances. Table IV reports the number of times the optimal solution was found at least once during the 50 runs by the tuned BRKGA (values on the left) and the total number of instances (values on the right). We observe that the solutions found by the BRKGAs were equal to or close to the global optimum found in the literature. The BRKGA with parameters found by ParamILS performed worse than the other BRKGAs for the instances of set Falkenauer and the BRKGA with parameters found by Calibra performed worse than the other BRKGAs for the instances of set Hard28. The BRKGAs configured with parameters found by BNT and iRace managed to find a greater amount of optimal solutions. These results show the robustness of the tuned BRKGAs.

[1]http://www.dcc.fc.up.pt/ fdabrandao/research/arcflow/results/

TABLE III
BEST SOLUTIONS FOUND BY THE BRKGAS

	Optimal solution	BRKGA			
		Calibra	ParamILS	iRace	BNT
Falkenauer	132.800	132.800	132.806	132.800	132.800
Hard28	70.786	71.214	71.179	71.179	71.179
Scholl	81.972	81.972	81.972	81.972	81.972
Schwerin	19.960	19.960	19.960	19.960	19.960
Waescher	17.412	17.471	17.471	17.471	17.471

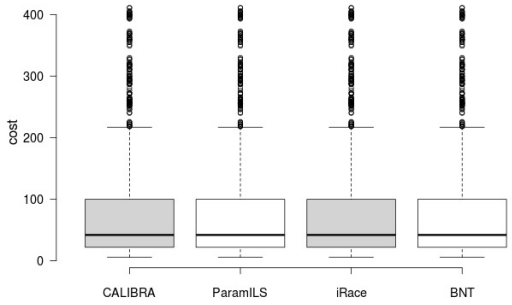


Fig. 6. Box-plot for the average solutions found by the BRKGAs

TABLE IV
NUMBER OF OPTIMAL SOLUTIONS FOUND BY BRKGAS

	BRKGA			
	Calibra	ParamILS	iRace	BNT
Falkenauer	160/160	159/160	160/160	160/160
Hard28	6/28	7/28	7/28	7/28
Scholl	1210/1210	1210/1210	1210/1210	1210/1210
Schwerin	200/200	200/200	200/200	200/200
Waescher	16/17	16/17	16/17	16/17
Total	1592/1615	1592/1615	1593/1615	1593/1615

Tables V and VI present the computational time results of the BRKGA configured with parameter values found by the tuning methods. A stop criterion was used to interrupt the BRKGA when it found the optimal solution of instances in which these values are known in the literature. We observe that the BNT results were very competitive with iRace at runtime. Although BRKGA with BNT parameters had a longer computational time in all groups, Table VI shows it was more stable to find optimal solutions when compared to the other methods because the standard deviation was the smallest.

However, it is necessary to confirm whether there is any statistical relevance in this information. To find this relevance, a normality test must first be applied to verify that the distribution of the values found by each group of instances follows

TABLE V
AVERAGE RUNTIME OF THE BRKGAS (IN SECONDS)

	BRKGA			
	Calibra	ParamILS	iRace	BNT
Falkenauer	0.351	0.376	0.354	0.445
Hard28	29.619	23.558	40.776	43.687
Scholl	0.059	0.066	0.085	0.087
Schwerin	0.021	0.031	0.052	0.054
Waescher	3.588	3.687	2.731	1.451

TABLE VI
STANDARD DEVIATION OF RUNTIME OF THE BRKGAS

	BRKGA			
	Calibra	ParamILS	iRace	BNT
Falkenauer	0.008	0.009	0.008	0.003
Hard28	0.246	0.296	0.361	0.148
Scholl	0.001	0.002	0.002	0.001
Schwerin	0.001	0.001	0.002	0.001
Waescher	0.005	0.011	0.057	0.006

the pattern of the normal distribution. Thus, the Shapiro-Wilk normality test [39] was applied to objective function and runtime samples. From this, Table VII was generated. If the p-value was less than 0.05, then the normal distribution hypothesis was rejected.

TABLE VII
SHAPIRO-WILK'S TEST FOR DATA PROVIDED BY TUNING METHODS

	BRKGA			
	Calibra	ParamILS	iRace	BNT
Falkenauer	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$
Hard28	$< 5.7 \times 10^{-6}$	$< 2.1 \times 10^{-5}$	$< 7.9 \times 10^{-6}$	$< 3.1 \times 10^{-6}$
Scholl	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$
Schwerin	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$
Waescher	0.566	0.688	0.868	0.769

As can be seen, only the data from the Waescher group indicated that they were arranged in a normal distribution. However, as the majority rejected the null hypothesis, a non-parametric test was chosen to analyze the degree of statistical significance among result sets. In this case, the chosen test was the Friedman Test with Nemenyi Post-hoc tests [40]. This test does not require the data distribution to be normal. Results are shown in Table VIII. This test also makes use of null hypothesis analysis for p-value less than 0.05. The Friedman test is applied to the sample sets to see if there are any difference between the samples. If the values are less than 0.05, the Nemenyi Post-hoc is applied to find out in which pairs the difference occurred. After applying the Friedman test for each group, the values were less than 0.05. Thus, the Nemenyi Post-hoc test can be applied to compare pairs of tuning methods.

The Nemenyi Post-hoc test works with two hypotheses. The first indicates rejection of the null hypothesis and shows only if the samples are different. The second hypothesis indicates which sample is statistically relevant. In Table VIII, results implied that the BNT were statistically relevant in relation to Calibra and ParamILS, except for the Waescher group. The values of BNT and iRace were less than 0.05, indicating that BNT was equivalent to iRace, except for the Hard28 group.

TABLE VIII
NEMENYI POST-HOC TEST FOR DATA PROVIDED BY TUNING METHODS

	BRKGA		
	BNT vs Calibra	BNT vs ParamILS	BNT vs iRace
Falkenauer	0	$< 2.12 \times 10^{-12}$	$< 5.52 \times 10^{-3}$
Hard28	$< 4.40 \times 10^{-7}$	$< 1.36 \times 10^{-13}$	0.468
Scholl	0	0	0
Schwerin	0	0	$< 1.29 \times 10^{-13}$
Waescher	0.067	0.997	0.712

An important feature of our proposed method is the Bayesian network generated by the BNT. In Figure 7, we can see the statistical dependence between parameters of the BRKGA to solve the BPP. We observed in this network that the parameter P_e is an independent parameter, but its value affects all other parameters. The n_{gen} and ρ_e parameters can be modified without impacting others. The k factor is directly influenced by the P_m parameter. The use of information on conditional probabilities makes it possible to observe how a value of one parameter affects the values of other parameters (global perspective), otherwise the parameter tuning becomes a task of local tuning (local perspective). A mapping of dependencies between parameters based on the best solutions helps the BNT to have a more detailed view of the parameter search space.

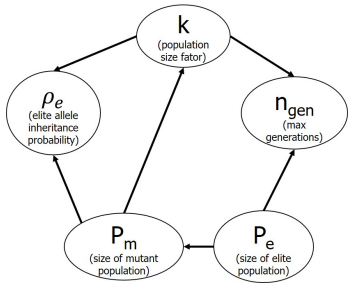


Fig. 7. Bayesian Network result of BNT

V. CONCLUSION

As can be seen in the results, the use of a Bayesian Network for the proposal of an offline tuning method, proved to be quite competitive in relation to some existing tuning methods in the literature. The tests demonstrated BNT's performance had statistical significance.

One of the great advantages of using our proposed BNT method is related to the reduction in number of iterations of the BRKGA, since the models induce patterns (information) to find the best solutions in the search space. Another advantage is that it has managed to establish the dependency between variables in the search for good solutions. These dependencies are very useful to assist the evolution of the algorithm over large numbers of parameters. ParamILS, for example, tends to have a slower evolution for very large parameter sets, since only one position of the vector is changed per stage. BNT, on the other hand, guides the search for solutions, showing that changing one parameter can affect the occurrence of a certain value in another parameter.

In future studies, new adjustments are needed to consolidate the BNT. For example, local search can be added to assist the Bayesian Network. New computational tests can be applied: tuning of very large number of parameters, set fixed number of generations or call of objective function for the tuned method, use different training time and distinct training instance sets, perform a comparison with other tuning methods, like EVOCA [41], and apply the BNT in other metaheuristics. We also plan to apply the BNT for unconstrained continuous optimization as well as other combinatorial optimization problems.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, FAPESP (grants 2018/15417-8), and CNPq (423694/2018-9 and 303736/2018-6).

REFERENCES

[1] E. B. d. M. Barbosa and E. L. F. Senne, "Improving the fine-tuning of metaheuristics: An approach combining design of experiments and racing algorithms," *Journal of Optimization*, vol. 2017, 2017.

[2] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "Paramils: An automatic algorithm configuration framework," *J. Artif. Int. Res.*, vol. 36, pp. 267–306, Sept. 2009.

[3] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*. Springer Publishing Company, Incorporated, 1st ed. 2005. 2nd printing ed., 2009.

[4] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[5] V. Nannen and A. E. Eiben, "Relevance estimation and value calibration of evolutionary algorithm parameters.," in *IJCAI*, vol. 7, pp. 975–980, 2007.

[6] B. Adenso-Diaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," *Operations Research*, vol. 54, no. 1, pp. 99–114, 2006.

[7] D. C. Montgomery, *Design and analysis of experiments*. John wiley & sons, 2017.

[8] I. Roman, J. Ceberio, A. Mendiburu, and J. A. Lozano, "Bayesian optimization for parameter tuning in evolutionary algorithms," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pp. 4839–4845, IEEE, 2016.

[9] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and evolutionary computation*, vol. 1, no. 3, pp. 111–128, 2011.

[10] P. Larranaga, "A review on estimation of distribution algorithms," in *Estimation of distribution algorithms*, pp. 57–100, Springer, 2002.

[11] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, *Towards a new evolutionary computation: advances on estimation of distribution algorithms*, vol. 192. Springer, 2006.

[12] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pp. 525–532, Morgan Kaufmann Publishers Inc., 1999.

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[14] J. F. Gonçalves and M. G. Resende, "Random-key genetic algorithms," *Handbook of Heuristics*, pp. 1–13, 2016.

[15] L. Baumgartner, V. Schmid, and C. Blum, "Solving the two-dimensional bin packing problem with a probabilistic multi-start heuristic," in *International Conference on Learning and Intelligent Optimization*, pp. 76–90, Springer, 2011.

[16] G. F. 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.

[17] V. M. Spears and K. A. D. Jong, "On the virtues of parameterized uniform crossover," in *In Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 230–236, 1991.

[18] H. Prasetyo, G. Fauza, Y. Amer, and S. H. Lee, "Survey on applications of biased-random key genetic algorithms for solving optimization problems," in *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, pp. 863–870, IEEE, 2015.

[19] M. R. Garey and D. S. Johnson, "Approximation algorithms for bin packing problems: A survey," in *Analysis and design of algorithms in combinatorial optimization*, pp. 147–172, Springer, 1981.

[20] A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell, R. Ruiz, and J. Tamarit, "Matheuristics for the irregular bin packing problem with free rotations," *European Journal of Operational Research*, vol. 258, no. 2, pp. 440–455, 2017.

[21] T. O. Ojeyinka, "Bin packing algorithms with applications to passenger bus loading and multiprocessor scheduling problems," *Communications on Applied Electronics*, vol. 2, no. 8, pp. 38–44, 2015.

[22] R. D. l. Rosa, H. Castillo, J. C. Zavala, A. Martínez, H. Estrada, T. E. Simos, and C. Tsitouras, "Application of prime numbers to solve complex instances of the bin packing problem," in *AIP Conference Proceedings*, vol. 1648, p. 820006, AIP Publishing, 2015.

[23] J. Csirik and D. S. Johnson, "Bounded space on-line bin packing: Best is better than first," in *SODA*, pp. 309–319, 1991.

[24] S. Martello, "Knapsack problems: algorithms and computer implementations," *Wiley-Interscience series in discrete mathematics and optimization*, 1990.

[25] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[26] R. Hübscher and F. Glover, "Applying tabu search with influential diversification to multiprocessor scheduling," *Computers & operations research*, vol. 21, no. 8, pp. 877–884, 1994.

[27] J. V. De Carvalho, "Lp models for bin packing and cutting stock problems," *European Journal of Operational Research*, vol. 141, no. 2, pp. 253–273, 2002.

[28] C. János *et al.*, "A classification scheme for bin packing theory," *Acta Cybernetica*, vol. 18, no. 1, pp. 47–60, 2007.

[29] J. F. Gonçalves and M. G. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 17, no. 5, pp. 487–525, 2011.

[30] J. F. Gonçalves and M. G. Resende, "A biased random key genetic algorithm for 2d and 3d bin packing problems," *International Journal of Production Economics*, vol. 145, no. 2, pp. 500–510, 2013.

[31] A. Zudio, D. H. da Silva Costa, B. P. Masquio, I. M. Coelho, and P. E. D. Pinto, "Brkga/vnd hybrid algorithm for the classic three-dimensional bin packing problem," *Electronic Notes in Discrete Mathematics*, vol. 66, pp. 175–182, 2018.

[32] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 11–18, Morgan Kaufmann Publishers Inc., 2002.

[33] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pp. 1186–1192, IEEE, 1992.

[34] J. E. Schoenfield, "Fast, exact solution of open bin packing problems without linear programming," *Draft, US Army Space and Missile Defense Command, Huntsville, Alabama, USA*, 2002.

[35] A. Scholl, R. Klein, and C. Jürgens, "Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers & Operations Research*, vol. 24, no. 7, pp. 627–645, 1997.

[36] P. Schwerin and G. Wäscher, *A new lower bound for the bin-packing problem and its integration into MTP*. Martin-Luther-Univ. Halle-Wittenberg, Wirtschaftswiss. Fak., 1998.

[37] G. Wäscher and T. Gau, "Heuristics for the integer one-dimensional cutting stock problem: A computational study," *Operations-Research-Spektrum*, vol. 18, no. 3, pp. 131–144, 1996.

[38] F. Brandão and J. a. P. Pedroso, "Bin packing and related problems," *Comput. Oper. Res.*, vol. 69, p. 56–67, May 2016.

[39] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[41] M.-C. Riff and E. Montero, "A new algorithm for reducing metaheuristic design effort," in *2013 IEEE Congress on Evolutionary Computation*, pp. 3283–3290, IEEE, 2013.