Characterizing and Analyzing the Relation Between Bin-Packing Problem and Tabu Search Algorithm

V. Landero[1(✉)], David Ríos[1], Joaquín Pérez[2], L. Cruz[3], and Carlos Collazos-Morales[4]

[1] Universidad Politécnica de Apodaca, Nuevo León, Mexico
{vlandero, drios}@upapnl.edu.mx

[2] Departamento de Ciencias Computacionales, Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), AP 5-164, Cuernavaca 62490, Mexico
jperez@cenidet.edu.mx

[3] División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero (ITCM), Cd. Madero, Mexico

[4] Vicerrectoría de Investigaciones, Universidad Manuela Beltrán, Bogotá, Colombia
carlos.collazos@docentes.umb.edu.co

Abstract. The relation between problem and solution algorithm presents a similar phenomenon in different research problems (optimization, decision, classification, ordering); the algorithm performance is very good in some cases of the problem, and very bad in other. Majority of related works have worked for predicting the most adequate algorithm to solve a new problem instance. However, the relation between problem and algorithm is not understood at all. In this paper a formal characterization of this relation is proposed to facilitate the analysis and understanding of the phenomenon. Case studies for Tabu Search algorithm and One Dimension Bin Packing problem were performed, considering three important sections of algorithm logical structure. Significant variables of problem structure and algorithm searching behavior from past experiments, metrics known by scientific community were considered (Auto-correlation Coefficient and Length) and significant variables of algorithm operative behavior were proposed. The models discovered in the case studies gave guidelines that permits to redesign algorithm logical structure, which outperforms to the original algorithm in an average of 69%. The proposed characterization for the relation problem-algorithm could be a formal procedure for obtaining guidelines that improves the algorithm performance.

1 Introduction

There exits a great variety of problems as constraint satisfaction, decision, optimization, forecasting, classification, clustering, sorting; where it has found that in certain problem instances the performance of a solution algorithm is very good and in other is very bad [1–4]. This phenomenon has been observed by broad range of disciplines: computational complexity theory, operations research, data mining, machine learning, artificial

O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12249, pp. 149–164, 2020.
https://doi.org/10.1007/978-3-030-58799-4_11

intelligence and bioinformatics [5]. In the real life situations, there is not an algorithm that outperforms others algorithms in all circumstances [6, 7] on some problem domain. The related works in the majority of cases have focused in building predictive models [8–22] for giving the best solution to new problem instances. Supervised or/and unsupervised learning algorithms have been used by the majority of related works for building that models. However, the built models are difficult to interpret the principal structure and relations between significant variables. These are used for prediction and there is no understanding of why an algorithm is better adequate to solve a set of problem instances than another (phenomenon). A few related works have tried explaining it [23–28]. Table 1 shows some of the latest related works; it emphasizes the information (variables) utilized (problem, behavior, and logical structure of algorithm) and the analysis main purpose. As it can be seen, not all information has been included. Also, it is necessary one formal formulation of phenomenon as a problem statement and one formal procedure for characterize the relation between problem-algorithm, that permits analyze deeply and solve the formulated problem, understanding the phenomenon.

Table 1. Related works

Work	Problem	Algorithm		Analysis purpose
		Behavior	Logical structure	
[13]	✓		✓	Prediction
[14]	✓		✓	Prediction
[19]	✓	✓		Prediction
[22]	✓			Prediction
[26]	✓			Explanation
[27]	✓	✓		Explanation
[28]	✓	✓		Explanation
This paper	✓	✓	✓	Explanation

In this paper, firstly, a nomenclature and the problem statement are formally formulated to previous questioning, where this formulation facilitates the proposal of a formal characterization of relation problem-algorithm (Sect. 2). It permits analyze and understand the relation of One Dimension Bin-Packing problem and Tabu Search algorithm and solve the problem statement performing cases of study. Three parts of algorithm logical structure, significant variables for characterizing structure and space of problem, algorithm searching behavior are considered, and significant variables for characterizing the algorithm operative behavior were proposed (Sect. 3). The models obtained by proposal characterization allowed to redesign Tabu Search algorithm for each case of study (Sect. 3). The results of proposed redesigns performance are analyzed and validated by means a statistical test in Sect. 4. The conclusions and future work are described in Sect. 5.

2 Problem Statement and Proposed Solution

Let the next nomenclature,

$\boldsymbol{I} = \{i_1, i_2, \ldots, i_m\}$ a set of instances of problem or problem space.

$\boldsymbol{F}$ = the features space, it represents the mapping of each problem instance to a set characterization variables.

$\boldsymbol{A} = \{a_1, a_2, \ldots, a_n\}$ a set of algorithms.

$a_s = \ a_s \in \boldsymbol{A}$, where a_s is the algorithm to analyze in each study case for relation problem-algorithm.

$\boldsymbol{B}$ = the features space, it represents the mapping of behavior of algorithm a_s to a set characterization variables.

$\boldsymbol{D} = \{D_1, D_2, \ldots, D_n\}$ a partition of $\boldsymbol{I}$, where $|\boldsymbol{A}| = |\boldsymbol{D}|$.

$\boldsymbol{X} = \{(a_q \in \boldsymbol{A}, D_q \in \boldsymbol{D}) \mid d(a_q(i)) > d(\alpha(i)) \; \forall \alpha \in (\boldsymbol{A} - \{a_q\}), \forall i \in D_q\}$,

is a set of ordered pairs (a_q, D_q), where each algorithm $a_q \in \boldsymbol{A}$ solved instances associated to D_q better than others algorithms (see function d as Expression 1). In the following sections each ordered pair (a_q, D_q) will be used to mean the domain region D_q (implicitly inferiority region $(D_q)^c$) of algorithm a_q.

$$d(a_q(i)) = \begin{cases} 1 & \text{if algorithm } a_q \text{has the rm } \textit{quality} \text{ among all the algorithms for instance } i \\ 1 & \text{if algorithm } a_q \text{has the smallest } \textit{time} \text{ among all the algorithms (when they have the same } \textit{quality}\text{)} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$\boldsymbol{P}$ = the performance space, it represents the mapping of performance of algorithm a_s, to a set of performance characterization variables; so too the value of function d $(a_s(i)) \; \forall i \in \boldsymbol{I}$.

2.1 Problem Statement

According mentioned in Introduction and before nomenclature, the next research question arises:

Why an algorithm a_s dominates in an instance's region D_s?

The problem of explaining formally why an algorithm outperforms others in solving an instances region and why not in others could be described formally as follows:

For a domination region $(a_s, D_s) \in \boldsymbol{X}$ and inferior region $(D_s)^c$, of an algorithm $a_s \in \boldsymbol{A}$, applied to problem instances $\boldsymbol{I}$, with problem features $\boldsymbol{F}$, algorithm features $\boldsymbol{B}$, find explanation model $\boldsymbol{R}$, which represents the relations between significant variables that characterize the problem instances, algorithm features, and provides solid foundations to explain why algorithm a_s is superior for solving instances in the domination region D_s and inferior for solving instances in subset $(D_s)^c$.

Figure 1 shows a possible graphical solution to above. The model $\boldsymbol{R}$, obtained by some mechanism $\boldsymbol{M}$ and sets $\boldsymbol{F}$, $\boldsymbol{B}$, $\boldsymbol{P}$ (described previously) as input; where values in $\boldsymbol{F}$ were obtained by variables a, b, c, d and values in $\boldsymbol{B}$ were obtained by variables v, x, y, z. The model $\boldsymbol{R}$ shows important and significant relations between variables that were significant and influence on the values from function d. In this example, b, c, x, y. The variables b and c characterize relevant and significant information about description and space of problem. The variables x and y characterize relevant and significant information from operative and searching behavior of analyzed algorithm.

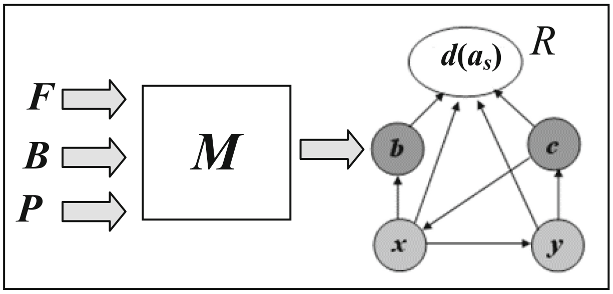

Fig. 1. Example of model R obtained by some mechanism M

The model $\boldsymbol{R}$ could be interpreted as one causal model, where it is Directed Acyclic Graph (DAG) over a set of vertices (variables) and a set of directed edges that connect vertices. These relations can be formally interpreted as causal (cause-effect) if the graph accomplish with the conditions: Causal Markov, Minimality and Faithfulness [29]. The causal relations (cause-effect) are estimated and the model is validated. The interest relations are analyzed and interpreted for obtaining explanations. If the model $\boldsymbol{R}$ accomplish with the above, it could be say that model $\boldsymbol{R}$ could formally answer the above principal question and problem statement. Nevertheless, there can be other models R_2 and R_3 with other variables that characterize: the structure and space of problem (d, e, h, j); the operative and searching of algorithm (w, z, n, v) where there are not influence variables for values of function d (see Fig. 2).

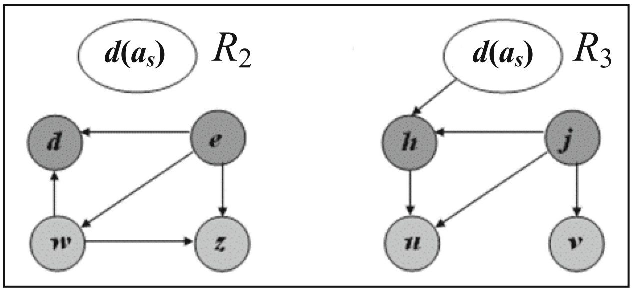

Fig. 2. Examples of models R_2 and R_3

The causal models R_2 and R_3 could not answer to principal question. Therefore, it is not easy to find one model $\boldsymbol{R}$ like Fig. 1. It will depend a lot on a mechanism $\boldsymbol{M}$ that can find from sets $\boldsymbol{F}$, $\boldsymbol{B}$, significant variables, whose values and its relation to values from function d allows build such model. The next section a characterization of relation of problem-algorithm is proposed, where its objective is to solve the before problem statement through case studies about the relation Bin-Packing problem and Tabu Search algorithm.

2.2 Proposed Characterization for Relation Problem - Algorithm

The relation between problem – algorithm can be formally characterized by the function $\boldsymbol{M}$ described by Expression 2. The domain is the set of parameters: the set $\boldsymbol{F}$, which is obtained by the function $\boldsymbol{f}$; the set $\boldsymbol{B}$, which is obtained by the function $\boldsymbol{b}$; the set $\boldsymbol{P}$, which is obtained by the function $\boldsymbol{p}$.

$$\boldsymbol{R} = (\boldsymbol{M} \cdot v \cdot e \cdot s \cdot t)(f(\boldsymbol{I}), b(\boldsymbol{A}), p(\boldsymbol{a_s}, \boldsymbol{A})) \tag{2}$$

The function $\boldsymbol{f}$ develops the process of mapping each problem instance from set $\boldsymbol{I}$ to a set variable that characterize the problem structure and space. The function $\boldsymbol{b}$ develops the process of mapping the behavior of each algorithm from set $\boldsymbol{A}$ to a set of variables that characterize the searching and operative behavior. The function p develops the process of mapping the performance of algorithm a_s to a set of variables that characterize the time, quality and its domain over other algorithms in $\boldsymbol{A}$ (described in nomenclature). The function $\boldsymbol{t}$, described by Expression 3, firstly builds the dataset $\boldsymbol{C}$, where the tuples represent the problem instances and the columns are the values that characterize the problem (set $\boldsymbol{F}$), the behavior, performance and domination of algorithm a_s. A discretization process is performed on the continuous dataset $\boldsymbol{C}$. The codomain of the function t is the discretized dataset $\boldsymbol{T}$. The function s (Expression 4) performs a process of selecting significant variables from discretized dataset $\boldsymbol{T}$. The codomain of this function is the discrete dataset S.

$$T = t(f(\boldsymbol{I}), b(\boldsymbol{A}), p(\boldsymbol{A})) \tag{3}$$

$$s(T) = S \tag{4}$$

The function e, described by Expression 5, consists in a learning process for obtaining causal relations from dataset S. The codomain of this function is the set of relations E. The function v, described by Expression 6, consists in estimating the causal relations in set E. The codomain of this function is the estimations set V. The codomain of function M is the causal model R of a set significant variables S, which represents a set of causal relations E and a set of estimations of these relations.

$$e(S) = E \tag{5}$$

$$v(E) = V \tag{6}$$

3 Characterizing and Analyzing the Relation Bin-Packing Problem and Tabu Search Algorithm: Case Studies

3.1 Description of Framework

The framework consists of characterizing and analyzing the relation between One Dimension Bin-Packing (BPP) problem and Tabu Search algorithm through proposed function ***M***; deepening in the problem structure, solutions space, the algorithm internal logical structure, its behavior operational, behavior during search and performance. Two types of instances for problem BPP were considered. The first, *instances* 1 (set ***I*** of nomenclature), for characterization and analysis process; and the second, *instances* 2 (different to *instances* 1) for prediction process. Each type of instances has 324 instances (this sample size has given good results in past experimentations). These were collected randomly from repositories [30, 31].

The internal logical structure of Tabu Search algorithm consists in four important parts. In this paper, the focus will be in three parts (control parameter, initial solution, search methodology), where the last part (stop criterion) is considered as a future work. Each case of study consists of comparing two variants, which only are different by one change in some fundamental part of its internal logical structure (see Table 2). In this paper, the size of Tabu list (control parameter) can be defined in a static (S) or dynamic way (D). It is to say, the size of Tabu list is defined as: 7 [32] or $\sqrt{n}$, where n is the number of the objects or items of the problem instance. The initial solution can be generated randomly (S) or by means deterministic procedure (H). For generating the neighborhood of a solution, one method can be considered (O) or several methods (M), which were proposed in [33]. The stop criterion was considered the same for all variants, it happens after 4000 iterations (divergence). Table 3 shows the cases of study.

Table 2. Variants of tabu search algorithm

Variants	Tabu list		Initial solution		Neighborhood	
	S	D	R	H	O	M
TB1		✓		✓		✓
TB2	✓			✓		✓
TB3		✓	✓			✓
TB4		✓		✓	✓	

3.2 Variables for Problem and Algorithm

Problem Variables. There are two considered variables for characterizing the problem structure, which are proposed in [16]. The first b characterizes the proportion of the total size of the objects that can be assigned to one container. The second f characterizes the proportion of objects where its weight w_i is factor of the container capacity.

Table 3. Cases of study

Case of study	Variants	Internal logical structure
1	TB1 and TB2	Tabu List Size
2	TB1 and TB3	Initial Solution
3	TB1 and TB4	Neighborhood

The problem solutions space is characterized by variable *os*, it has been one significant variable in past experimentations [28]. A sample of *ms* (the value 100 has given good results) randomly generated solutions before algorithm execution is built. This variable characterizes the variability of fitness function *f*(*x*) [34] of these *ms* solutions.

Algorithm Variables. In this paper, the characterization of the algorithm operative behavior is proposed. It is characterized by two significant variables, the first is the number of feasible solutions found by algorithm per instance (variable *efac*), the second is the variance of these solutions (variable *evfac*). Also, two ways are considered by characterizing the algorithm behavior on trajectory during the search process (solutions generated during execution). The first way is using the concept fitness landscape with two known metrics, the autocorrelation coefficient (*coef*) and autocorrelation length (*long*), which were described in [35]. The second way is using significant variables proposed in past experiments [25, 28]. These are number of inflexion points *nc*, number of valleys *nv*, the average size *tm* of the valleys and its dispersion *vd* from algorithm search trajectory.

Performance Variables. The performance variables considered are *time* and *quality*, which are described in [28]. The first is the number of evaluations of the fitness function for feasible and infeasible solutions. The *quality* variable is the ratio of the best solution found by the algorithm (final number of containers) to the theoretical solution; this is the objects sizes divided by the containers capacity.

3.3 Characterizing the Relation Bin Packing Problem – Tabu Search Algorithm

The set of problem instances ***I*** (*instances 1*) is characterized by function *f*. This performs the calculation of problem variables described in before section on each problem instance in set ***I***. A set *F* is built as Expression 7. The algorithms in set ***A*** are executed on each problem instance, where the function *p* performs the calculation of performance variables described in before section and a set ***P*** is built. The function *b* performs the calculation of algorithm variables described in before section for this algorithm, a set ***B*** is built as Expression 8.

$$F = \{\{b_1, f_1, os_1\}, \{b_2, f_2, os_2\}, \ldots, \{b_m, f_m, os_m\}\} \tag{7}$$

The set *C* is obtained by function *t*, where its information is obtained from sets *F*, *B* and *P*. The Expression 9 shows an example of set *C*.

$$B = \left\{ \begin{array}{c} \{efac_1, evfac_1, nc_1, nv_1, tm_1, vd_1\}, \\ \{efac_2, evfac_2, nc_2, nv_2, tm_2, vd_2\}, \ldots, \\ \{efac_m, evfac_m, nc_m, nv_m, tm_m, vd_m\} \end{array} \right\} \tag{8}$$

$$C = \left\{ \begin{array}{c} \{b_1, f_1, os_1, efac_1, evfac_1, nc_1, nv_1, tm_1, vd_1, time_1, quality_1, d(a_s(1))\}, \\ \{b_2, f_2, os_2, efac_2, evfac_2, nc_2, nv_2, tm_2, vd_2, time_2, quality_2, d(a_s(2))\}, \\ \ldots, \\ \{b_m, f_m, os_m, efac_m, evfac_m, nc_m, nv_m, tm_m, vd_m, time_m, quality_m, d(a_m(1))\} \end{array} \right\} \tag{9}$$

The set *C* is discretized by the method MDL [36] and the set *T* is obtained. Then, the function *s* performs the method Correlation-based Feature Selection (CFS) [37]. It is important to emphasize that this selection depends of the values of set *T*, each study case is different due to the included variants and the sets *S* are different. The next sections describe which are these sets *S*, so too, the functions *e* and *v*.

Study Case 1
The set *S*, obtained by function *s*, is described by Expression 10. In Sect. 3.2 were mentioned two more common ways to characterize the trajectory traced by the algorithm during its execution: fitness landscape variables (*coef*, *long*) and our variables (*nv*, *tm*, *vd*).

$$S = \left\{ \begin{array}{c} \{f_1, os_1, nv_1, tm_1, vd_1, d(a_s(1))\}, \{f_2, os_2, nv_2, tm_2, vd_2, d(a_s(2))\}, \\ \ldots, \{f_m, os_m, nv_m, tm_m, vd_m, d(a_s(m))\} \end{array} \right\} \tag{10}$$

Therefore, we built another different set $\boldsymbol{S}_2$ with information *f*, *os*, *coef* and *long*. The value of function *d* is considered in two before cases. The function *e* performs the structure learning algorithm PC [29]. The causal inference software HUGIN (Hugin Expert, www.hugin.com) was used with a confidence level of 95%. The domain of function *e*, in a first time, was the set ***S***, obtaining the causal structure ***E*** (Expression 11); in a second time was the set $\boldsymbol{S}_2$, obtaining the causal structure $\boldsymbol{E}_2$ (Expression 12).

$$E = \left\{ \begin{array}{c} \{\{d, f\}, \{d, os\}, \{d, nv\}, \{d, tm\}\}, \{\{vd, os\}, \{vd, nv\}, \{vd, tm\}\}, \\ \{\{tm, f\}, \{tm, os\}\}, \{\{nv, os\}\} \end{array} \right\} \tag{11}$$

$$E_2 = \{\{\{d, f\}, \{d, os\}\}, \{\{os, coef\}, \{os, long\}\}, \{\{coef, long\}\}\} \tag{12}$$

The causal relations in ***E*** are estimated by function *v*, which performs the parameter learning algorithm Counting [29]. Figure 3(a) shows the causal model ***R***, containing the causal structure ***E*** and causal relations estimations ***V*** (due short space, the complete set ***V*** is not shown, only one part in Table 3). So too, Fig. 3(b) shows the causal model $\boldsymbol{R}_2$, containing the causal structure $\boldsymbol{E}_2$ and causal relations estimations $\boldsymbol{V}_2$. This did not yield relevant information about direct causes of the algorithm behavior and performance, in terms of regions of domination (D_s) and inferiority ($(D_s)^c$), using the correlation coefficient (*coef*) and autocorrelation long (*long*). Due before, the set $\boldsymbol{V}_2$ is not shown. Conversely, the causal model ***R*** (a) shows that the variables *f*, *os*, *nv* and *tm* are direct causes. The most important probabilities of direct causes of algorithm behavior and performance, in terms of regions domination and inferiority, are shown in Table 4 (one part of set ***V***). The internal logical structure of variant TB1 permits storage more solutions in Tabu List (big) being more restrictive for accepting a new neighbor solution. Variant TB1 is successful with instances where the sizes of objects or items that are multiples of the container, f, is in the range 2 [0.041, 1] and there is a variability of solutions space, os, in the range 3 [0.1695, 1].

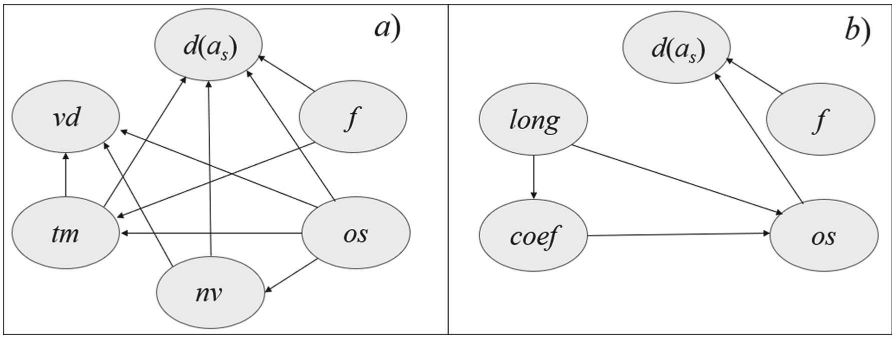

Fig. 3. Causal models ***R*** and ***R***$_2$

This indicates that there is more facility for generating different neighbor solutions and can be difficult that these can be in Tabu List due these are vastly different. Valleys were identified in the search trajectory of variant, the number of valleys, *nv*, is in the range 2 [0.061, 1] with sizes, *tm*, in the range 2 [0.3138, 1] (causal relation 1). This variant wins in quality. This variant has disadvantage with instances where the number of the sizes of the objects or items that are not multiple of container, *f*, is in the range 1 [0, 0.040] and the variability between solutions of problem space, *vo*, is in the ranges 1 [0, 0.054], 2 [0.055, 0.1694]. So too, a number of valleys were identified, *nv*, in the range 1 [0, 0.060] with sizes in the range 1 [0, 0.3137] (causal relations 2 and 3). The neighbors that can be generated from these instances will not be very different, generating a flat search trajectory; which can be Tabu, due the size of this list is big. Therefore, variant TB2 has advantage of this situation, due it uses a size of Tabu List more small (7) and is more flexible for accepting generated neighbor solutions. Variant TB1 lost in time.

Case of Study 2

In the development of this experimental test, function *s* could not select variables. Therefore, this case of study could not be performed.

Case of Study 3

The set *S*, obtained by function *s* is described by Expression 13.

Table 4. Relations Estimation, part of set ***V***

	%
P(d = 1 | f = 2, os = 3, nv = 2, tm = 2)	97.82
P(d = 0 | f = 1, os = 1, nv = 1, tm = 1)	70.78
P(d = 0 | f = 2, os = 2, nv = 1, tm = 1)	85.71

$$S = \left\{ \begin{array}{l} \{b_1, os_1, efac_1, evfac_1, nc_1, nv_1, tm_1, d(a_s(1))\}, \ldots, \\ \{b_m, os_m, efac_m, evfac_m, nc_m, nv_m, tm_m, d(a_s(m))\} \end{array} \right\} \tag{13}$$

Two different causal structures were built by function e (as case of study 1): the first ***E*** (Expression 14) with values of these variables and the second $\boldsymbol{E}_2$ (Expression 15) with *b*, *os*, *efac*, *evfac*, *coef* and *long*. So too, the function *d* is considered in two before cases (see Fig. 4). Part (a) of Fig. 4 shows the causal model ***R***, containing the causal structure ***E*** and causal relations estimations ***V***. The model ***R*** yields to more complete and relevant information about direct causes of the algorithm behavior and performance, in terms of regions of domination (D_s) and inferiority ($(D_s)^c$), considering information about problem structure. The causal relations found in ***E*** are estimated by function *v* and the most important probabilities of direct causes of algorithm behavior and performance, in terms of regions domination and inferiority, are shown in Table 5 (one part of set ***V***). The use of several methods for generating neighbor solutions (variant TB1) permits has advantage in problems where the total sum of sizes of objects or items is much greater than the capacity of the container and the arrangement of the objects can be very varied; it is to say, a great diversity of solutions can be generated.

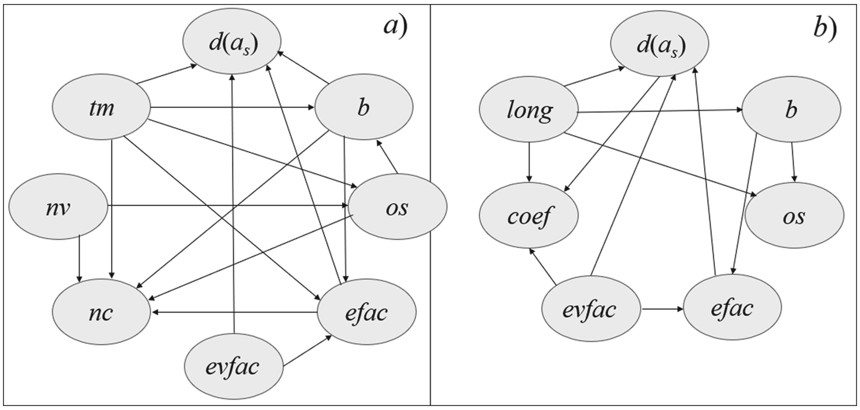

Fig. 4. Causal models ***R*** and $\boldsymbol{R}_2$

Table 5. Relations Estimation, part of set ***V***

	%
P(d = 1 | b = 1, $efac$ = 1, $evfac$ = 2, nv = 2)	98.78
P(d = 0 | b = 2, $efac$ = 2, $evfac$ = 1, nv = 1)	71.42
P(d = 0 | b = 3, $efac$ = 3, $evfac$ = 1, nv = 1)	100

$$E = \left\{ \begin{array}{c} \{\{d,b\},\{d,efac\},\{d,evfac\},\{d,nv\}\},\{\{b,os\},\{b,tm\}\}, \\ \{\{nc,b\},\{nc,os\},\{nc,efac\},\{nc,nv\},\{nc,tm\}\}, \\ \{\{efac,b\},\{efac,evfac\},\{efac,tm\}\},\{\{os,nv\},\{os,tm\}\} \end{array} \right\} \quad (14)$$

$$E_2 = \left\{ \begin{array}{c} \{\{d,efac\},\{d,evfac\},\{d,long\}\},\{\{efac,b\},\{efac,evfac\}\}, \\ \{\{coef,d\},\{coef,long\},\{coef,evfac\}\},\{\{os,b\},\{os,long\}\} \end{array} \right\} \quad (15)$$

Its internal logical structure enables intensify the search, generating a number of feasible solutions, *efac*, in the range 1 [0, 0.67], with a variability between feasible solutions, *evfac*, in the range 2 [0.1377, 1]; the search trajectory corresponds better to this problem structure and problem space, because it may enter and exit from a number of valleys, *nv*, in the range 2 [0.0297, 1] (relation 1). This variant wins in quality. The variant TB1 loses in time in problems where the total sum of sizes of objects or items is not much greater than the capacity of the container; it indicates that there is no variety to arrangement the objects; it is to say, there is a number of feasible solutions, *efac*, in the range 1 [0.68, 1] with a variability between feasible solutions, *evfac*, in the range 2 [0, 0.1376]. Therefore, there is no necessity for intensifying the search, the number of valleys, *nv*, is in the range 1 [0, 0.0296]. The variant may take longer to find solutions, where there is little to find (relations 2 and 3); its exhaustive attempt to generate neighbor solutions produces a cost in time and lost by this factor. The variant TB4 has advantage in this situation, because it only uses one method for generating neighbor solutions; it is very limited for searching in the problem space by its own structure and it is better adjusted to this kind of problems.

Redesign Proposals

The analysis of relation Bin-Packing problem and Tabu Search algorithm of each study case permit to find guidelines to redesign the algorithm. Figure 5 shows the proposal of redesign from case of study 1, marked by R1TB1.

It consists to automatically adjust its logical design, in terms of the Tabu list size. It is to say, if value f falls in the second interval and *os* falls in the third interval, the Tabu list size will be $\sqrt{n}$; otherwise it will be 7. So too, Fig. 5 shows the proposal of redesign from case of study 3, marked by R3TB1. It consists to adjust automatically its logical design, in terms of the way to build the neighborhood. It is to say, if value b falls in the first interval the neighborhood will be built by means several methods; otherwise, one method will be performed.

Tabu Search Algorithm

```
1   Begin
2     $x^*$ = a feasible initial solution; $x = x^*$
3     if ($f$==2 and $os$==3) or ($f$==1 and $os$==3) then $nLTabu = \sqrt{n}$;   } R1TB1
4     else $nLTabu = 7$;
5     $nLTabu = \sqrt{n}$;   } R3TB1
6     $tn = nLTabu$, Tenency of one solution in Tabu list
7     $LCANDI = \varnothing$; $LTabu = \varnothing$
8     Repeat
9       Building neighborhood of $x$ using swap (0, 1) or swap (1, 1) or   } R1TB1
10          swap (1, 2) or swap (2, 2) or swap (1, 0) or swap (2,1)
11      if $b$==1 then   } R3TB1
12        Building neighborhood $N(x)$ of $x$ using swap (0, 1) or swap (1, 1)
13        or swap (1, 2) or swap (2, 2) or swap (1, 0) or swap (2,1)
14      else Building neighborhood $N(x)$ of $x$ using swap (0, 1)
15      $LCANDI$ = list of candidate solutions, taken from neighborhood $x$
16      $y$ = the best solution of $LCANDI$ and $y \notin LTabu$
17      $LTabu = LTabu \cup \{(y, tn)\}$
18      For each $e \in LTabu$ the tenency of $e$ is decremented
19        If the tenency $e$ has expired Then $LTabu = LTabu - \{(e, tn)\}$
20      If $f(y) < f(x)$ Then $x = y$
21    Until 4000 iterations.
22  End
```

Fig. 5. Redesign proposals

4 Results Analysis

Table 6 shows the experimentation. Column 1 indicates the case of study; Column 2 indicates the prediction percentage of generated model on *instances 2*; for this, the causal inference software NETICA (Norsys Corporation, www.norsys.com) was utilized. Column 3 indicates an outperform percentage of proposed redesign with respect to original algorithm. The function ***M*** builds a causal model ***R*** that permits give answer to question and problem statement in explanation level for the cases of study 1 and 3. For the case of study 2, function *s* could not select variables that could be considered for building a causal model; therefore, this case of study could not be performed. One possible interpretation of this result, it may be that the method for generating the initial solution does not impact the algorithm performance. This had already been observed by [38] with the same Tabu Search algorithm for solving another problem, the Job-Shop Scheduling problem. The models generated in the cases 1 and 3 permit to make predictions about the algorithm performance in terms of domination or inferior region (D_s, $(D_s)^c$) with a percentage higher than 70% over a problem instances test set (*instances 2*) (see Table 6).

On the other hand, the causal relations found by the models generated permit to make redesign proposals; which improve the algorithm performance (see Table 6). To

Table 6. Experimentation results

Redesign proposal	% model prediction	Outperform percentage
R1TB1	79.01%	65%
R3TB1	72.14%	73%

validate the above, the two sample two-side Wilcoxon signed rank test was applied for significance levels 95% and 99% to verify the means of values of performance variable *quality* (it is not assuming a normal distribution). The Table 7 shows the null hypothesis conclusion (means are equal), where is rejected for two significance levels. It means, there is a significant improvement of redesign proposals.

Table 7. Null hypothesis conclusion

Redesign proposal	Significance level	Test statistic	Critical value	Conclusion
R1TB1	95%	8.2966	1.9600	Rejected
	99%	8.2966	2.5758	Rejected
R3TB1	95%	12.2664	1.9600	Rejected
	99%	12.2664	2.5758	Rejected

5 Conclusions

This paper formally formulates the research question "why an algorithm is the best for solving an instances subset and why not in other instances" as a problem statement to solve. This questioning is implicitly observed by scientific community in the solution process of problems such as decision, optimization, forecasting, classification, clustering, sorting. Nevertheless, its common objective, in the majority cases, is to predict the algorithm most adequate to solve a new problem instance. A few related works have tried to obtain explanations, without a formal formulation of this questioning as one problem to solve. As well as, it is necessary to explore more parts of algorithm logical design, analyze its relation to significant variables from problem and algorithm, including also significant variables of algorithm operative behavior.

Therefore, this paper also proposes one characterization of the relation problem-algorithm to solve the proposal problem statement through case studies of One Dimension Bin-Packing problem and Tabu Search algorithm. Significant variables were considered for characterizing the problem structure and space; the behavior of algorithm during its search trajectory and variables proposed for characterizing its operational behavior. As well as, three parts of the algorithm internal logical structure are considered in the analysis. The proposal characterization allowed to found causal models. Such models contribute to the justification for the use of an algorithm for solving a test instances subset, obtaining an average prediction percentage of 76%. Important relations between the algorithm performance, in terms of domination or inferior region and significant variables were identified from these causal models. Such obtained formal explanations permit to propose guidelines to redesign the algorithm internal logical structure for improving its performance. The redesign proposals outperform to original algorithm in an average of 69% out of 324 problem instances. The proposed characterization of relation problem-algorithm could be a formal procedure for obtaining guidelines that improves the algorithms performance. As future work is considered to extend this proposed characterization to other variants of the Tabu Search algorithm to further explore the internal logical structure; as well as to other algorithms (Genetic, Ant Colony Optimization, etc.), firstly to same problem and then to other optimization problems.

References

1. Garey, M.R., Jhonson, D.S.: Computers and Intractability, a Guide to the Theory of NP-completeness. W. H. Freeman and Company, New York (1979)
2. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization, Algorithms and Complexity. Prentice Hall, Upper Saddle River (1982)
3. Rendell, L., Cho, H.: Empirical learning as a function of concept character. Mach. Learn. **5**, 267–298 (1990)
4. Lagoudakis, M., Littman, M.: Learning to select branching rules in the DPLL procedure for satisfiability. Electron. Notes Discrete Math. **9**, 344–359 (2001)
5. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. **41**(1), 1–25 (2009)
6. Wolpert, D., Macready, W.: No free lunch theorems for optimizations. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1996)
7. Vanchipura, R., Sridharan, R.: Development and analysis of constructive heuristic algorithms for flow shop scheduling problems with sequence-dependent setup times. Int. J. Adv. Manuf. Technol. **67**, 1337–1353 (2013)
8. Hutter, F., Xu, L., Hoos, H., Leyton-Brown, K.: Algorithm runtime prediction: methods & evaluation. Artif. Intell. **206**, 79–111 (2014)
9. Xu, L., Hoos, H., Leyton-Brown, K.: Hydra: automatically configuring algorithms for portfolio-based selection. In: Proceedings of the 25th National Conference on Artificial Intelligence (AAAI 2010), pp. 210–216 (2010)
10. Cayci, A., Menasalvas, E., Saygin, Y., Eibe, S.: Self-configuring data mining for ubiquitous computing. Inf. Sci. **246**, 83–99 (2013)
11. Pavón, R., Díaz, F., Laza, R., Luzón, M.: Experimental evaluation of an automatic parameter setting system. Expert Syst. Appl. **37**, 5224–5238 (2010)
12. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
13. Yeguas, E., Luzón, M., Pavón, R., Laza, R., Arroyo, G., Díaz, F.: Automatic parameter tuning for evolutionary algorithms using a Bayesian case-based reasoning system. Appl. Soft Comput. **18**, 185–195 (2014)
14. Ries, J., Beullens, P.: A semi-automated design of instance-based fuzzy parameter tuning for metaheuristics based on decision tree induction. J. Oper. Res. Soc. **66**(5), 782–793 (2015)
15. Yong, X., Feng, D., Rongchun, Z.: Optimal selection of image segmentation algorithms based on performance prediction. In: Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing, pp. 105–108. Australian Computer Society, Inc. (2003)

16. Pérez, J., Pazos, R.A., Frausto, J., Rodríguez, G., Romero, D., Cruz, L.: A statistical approach for algorithm selection. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 417–431. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24838-5_31
17. Nikolić, M., Marić, F., Janičić, P.: Instance-based selection of policies for SAT solvers. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 326–340. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02777-2_31
18. Yuen, S., Zhang, X.: Multiobjective evolutionary algorithm portfolio: choosing suitable algorithm for multiobjective optimization problem. In: 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, pp. 1967–1973 (2014)
19. Munoz, M., Kirley, M., Halgamuge, S.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. **19**(1), 74–87 (2015)
20. Leyton-Brown, K., Hoos, H., Hutter, F., Xu, L.: Understanding the empirical hardness of np-complete problems. Mag. Commun. ACM **57**(5), 98–107 (2014)
21. Cruz, L., Gómez, C., Pérez, J., Landero, V., Quiroz, M., Ochoa, A.: Algorithm Selection: From Meta-Learning to Hyper-Heuristics. INTECH Open Access Publisher (2012)
22. Wagner, M., Lindauer, M., Misir, M., et al.: A case of study of algorithm selection for the travelling thief problem. J. Heuristics, 1–26 (2017)
23. Pérez, J., Cruz, L., Landero, V.: Explaining performance of the threshold accepting algorithm for the bin packing problem: a causal approach. Pol. J. Environ. Stud. **16**(5B), 72–76 (2007)
24. Tavares, J.: Multidimensional knapsack problem: a fitness landscape analysis. IEEE Trans. Syst. Man Cybern. Part B Cybern. **38**(3), 604–616 (2008)
25. Pérez, J., et al.: An application of causality for representing and providing formal explanations about the behavior of the threshold accepting algorithm. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 1087–1098. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69731-2_102
26. Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) LION 2010. LNCS, vol. 6073, pp. 266–280. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13800-3_29
27. Quiroz, M., Cruz, L., Torrez, J., Gómez, C.: Improving the performance of heuristic algorithms based on exploratory data analysis. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Recent Advances on Hybrid Intelligent Systems, Studies in Computational Intelligence, vol. 452, pp. 361–375. Springer, Heidelberg (2013)
28. Landero, V., Pérez, J., Cruz, L., Turrubiates, T., Rios, D.: Effects in the algorithm performance from problem structure, searching behavior and temperature: a causal study case for threshold accepting and bin-packing problem. In: Misra, S., Gervasi, O., Murgante, B. (eds.) ICCSA 2019. LNCS, vol. 11619, pp. 152–166. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-24289-3_13
29. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, 2nd edn. The MIT Press, Cambridge (2001)
30. Beasley, J., E.: OR-Library. Brunel University (2006). http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html
31. Scholl, A., Klein, R.: (2003). http://www.wiwi.uni-jena.de/Entscheidung/binpp/
32. Glover, F.: Tabu search - Part I, first comprehensive description of tabu search. ORSA-J. Comput. **1**(3), 190–206 (1989)
33. Fleszar, K., Hindi, K.S.: New heuristics for one-dimensional bin packing. Comput. Oper. Res. **29**, 821–839 (2002)

34. Khuri, S., Schütz, M., Heitkötter, J.: Evolutionary heuristics for the bin packing problem. In: Artificial Neural Nets and Genetic Algorithms. Springer, Vienna (1995). https://doi.org/10.1007/978-3-7091-7535-4_75
35. Merz, P., Freisleben, B.: Fitness landscapes and memetic algorithm design. In: New Ideas in Optimization, pp. 245–260. McGraw-Hill Ltd., UK (1999)
36. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI, pp. 1022–1029 (1993)
37. Hall, M.A.: Feature selection for discrete and numeric class machine learning (1999)
38. Watson, J., Darrell, W., Adele, E.: Linking search space structure, run-time dynamics, and problem difficulty: a step toward demystifying tabu search. J. Artif. Intell. Res. **24**, 221–261 (2005)