

Effects in the Algorithm Performance from Problem Structure, Searching Behavior and Temperature: A Causal Study Case for Threshold Accepting and Bin-Packing

V. Landero[1(✉)], Joaquín Pérez[2], L. Cruz[3], Tania Turrubiates[4], and David Ríos[1]

[1] Universidad Politécnica de Apodaca (UPAP), Apodaca, Nuevo León, Mexico
vanesa.landero.najera@gmail.com, drios@upapnl.edu.mx
[2] Departamento de Ciencias Computacionales, Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), AP 5-164, 62490 Cuernavaca, Mexico
jperez@hotmail.com
[3] División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero (ITCM), Cd. Madero, Mexico
lcruzreyes@prodigy.net.mx
[4] Instituto Tecnológico Superior de Álamo Temapache, Veracruz, Mexico

Abstract. A review of state of art reveals that the characterization and analysis of the relation between problem-algorithm has been focused only on problem features or on algorithm features; or in some situations on both, but the algorithm logical is not considered in the analysis. The above for selecting an algorithm will give the best solution. However there is more knowledge for discovering from this relation. In this paper, significant features are proposed for describing problem structure and algorithm searching fluctuation; other known metrics were considered (Autocorrelation Coefficient and Length) but were not significant. A causal study case is performed for analyzing causes and effects from: Bin-Packing problem structure, Temperature, searching behavior of Threshold Accepting algorithm and final performance to solving problem instances. The proposed features permitted in the causal study to find relations cause-effect; which gave guidelines for designing a Threshold Accepting self-adaptive algorithm. Its performance outperforms to original algorithm in 74% out of 324 problem cases. The causal analysis on relevant information from problem, algorithm (both) and algorithm logical could be an important guideline to discover rules or principles over several problem domains, which permit the design of self-adaptive algorithms to give the best solution to complex problems.

1 Introduction

The majority of reviewed related works from some disciplines as combinatorial optimization, machine learning, artificial intelligence have characterized the relation between problem and algorithm focusing only on problem information, some examples [1–4]; or algorithm, some examples [5–7]; other only one problem and algorithm information [8–12]. This relation has been analyzed in the majority cases to select the

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 152–166, 2019.
https://doi.org/10.1007/978-3-030-24289-3_13

most indicated algorithm [2, 11, 13–15]; or predict the problems hardness [16, 17]; or built hyper-heuristics [18]; or adjust control parameter [6, 9, 12, 19–23]. The objective principal is to give the best solution to some problem.

However, these not include both information about: problem (problem structure) and algorithm (behavior in its search trajectory); the algorithm logical design; for analyzing deeply causes and effects between problem-algorithm; it is to say, why this relation becomes successful in some problem instances and why not in other, what problem feature are related to algorithm feature, which is the algorithm logical design; and how they relate in some way for obtaining the best solution on instances set. These causal relations found over several problems domains could give guidelines for discovering theories that permit to self-adapt the algorithm logical, depending on problem structure, searching behavior and give the best solution to any problem.

In order to be able to give small, but relevant steps according to above, firstly, starting for a specific domain; in this paper, firstly, a reviewing of state of art from disciplines combinatorial optimization, machine learning, artificial intelligence about information type and approaches for characterizing and analyzing the relation between problem-algorithm is presented in Sect. 2, and a reflection about this long trajectory of these related works is performed. In Sect. 3, a framework is proposed for characterizing the One Dimension Bin-Packing problem and Tabu Search algorithm. In Sect. 4, a causal study case is presented, where the relation between One-Dimension Bin-Packing problem and Tabu Search algorithm is analyzed, the found causes and effects between proposed features from problem and algorithm, considering the algorithm logical design, gave knowledge that permitted design a Threshold Accepting self-adaptive algorithm. In Sect. 5, an analysis of performance results of original and self-adaptive algorithms is performed; where a statistical test was applied. Finally, in Sect. 6, conclusions about the performed work are presented, as well as research future works.

2 Reviewing State of Art: Information and Analysis Approaches

This section emphasizes two main issues, information type and the approaches considered to analyze the relation between problem and algorithm. Table 1 presents some investigations of both. The column 2 indicates if the analysis was based on problem features (F). The column 3 indicates if the analysis was based on algorithm features (A) and column 4 indicates if the analysis considered some algorithm logical area. The last column indicates the analysis approach performed. The analysis approach in this paper is referred to procedure for discovering knowledge from experimentation data, which permits the analysis of the relation between problem and algorithm. Some of these: Unsupervised Learning (UL), Supervised Learning (SL), Regression Analysis (RA), Functions of Probability Distribution (FD), Exploratory Analysis (EA), Causal Analysis (CA).

Some related works considered problem features for building models, using machine learning, specifically supervised learning (SL), to learn patterns identified from data, model and use it to make predictions about who would be the most appropriate algorithm to solve better a new problem instance. Several algorithms have

Table 1. Relevant causes

Work	Problem	Algorithm searching	Algorithm logical	Analysis approach
[2]	✓			UL
[13]	✓			SL
[24]	✓			SL
[25]	✓			FD
[26]		✓		EA
[27]			✓	EA
[28]			✓	EA
[29]		✓	✓	EA
[7]	✓		✓	SL
[3]	✓			UL-SL
[9]	✓		✓	RA
[19]	✓		✓	SL
[10]	✓	✓		EA
[6]			✓	SL
[21]	✓		✓	SL
[11]	✓	✓	✓	SL
[30]	✓	✓		SL
[12]	✓		✓	SL
[22]			✓	RA
[23]			✓	EA
This paper	✓	✓	✓	CA

been utilized by some related works: Case-base reasoning [1], decision trees [2, 3, 24], Support Vector Machine (SVM) [13].

Other related works in the context of model-based algorithm portfolio considered only the algorithm performance as information for analyzing. It is to say, the performance of all algorithms is measured by means some function, so too, this performance is characterized and adjusted to a model, either by a regression model [16] or a probability distribution model [14]. Other related works in the context of feature-based portfolio, considered some problem features for building a model, applying supervised learning [25]. The model can be utilized in an off-line way or an on-line way [31], where model is updated and system changes to another algorithm to give the best solution to new problem instance.

Other related works analyze relations between problem features and algorithm performance by means an unsupervised learning model, and identify key problem features. Some methods from this approach have been utilized by some related works, which are K-means clustering [2], Self-organizing maps [3].

Some other related works obtain knowledge by means exploratory analysis [7, 28], supervised learning [6], which permits to configure the algorithm in a way that it can produce the better results. There is no clearly reasons for what kind of problem instances, a algorithm configuration will produce better results. It is to say, what is the

problem structure that permits to one configured algorithm to give the best results. Montero considers this information very relevant to adjust the algorithm logical [32]. There exists some related works that included problem features for analyzing the parameter control and algorithm performance by means supervised learning [12, 19, 21], as well as, this approach in the context of algorithm portfolio [9]. The searching methodology of algorithm is also analyzed with problem features and algorithm performance [29] by supervised learning, using the method k-nearest neighbor.

There exists related works that considered problem and algorithm features in their work by supervised learning [30], Exploratory Analysis [10]. However, the algorithm logical area could have given relevant knowledge for understanding more deeply the relationship that exists between problem and algorithm.

As it can be observed in Table 1, the majority of related works not includes in their works features from problem, from algorithm and the logical design. A model is built by means method Random Forest in [11], where features from problem and algorithm are considered, as well as, information of algorithm logical area. However, due nature of built model structure, which it is used for predicting the best algorithm to solve a problem instance, it is difficult to interpret the learned knowledge. It is to say, to identify the principal causes and effects about the algorithm performance, and understand better the relation between problem and algorithm; why an algorithm is best for a certain set of instances of a problem and why not in another.

3 Proposed Framework: Causal Analysis and Proposed Features

In this paper, a causal study case is presented for analyzing causes and effects that there exists between problem features, algorithm features, algorithm logical and algorithm performance; having as framework the One Dimension Bin-Packing problem and Threshold Accepting algorithm. Past works [33–35] have suggested that causal analysis is important for obtaining relevant knowledge, structured as causes and effects, from relationship problem-algorithm. As an extension from these related works, we proposed significant features for obtaining a new characterization from problem structure and algorithm searching fluctuation; which also permit to discover latent knowledge structured as a formal causal model. So too, such discovering gives rules for designing a new self-adaptive Threshold Accepting algorithm that outperforms to original algorithm for solving instances of One Dimension Bin-Packing problem.

In this section, a brief description of causal analysis is presented; where the literature nomenclature will be used for describing proposed features and causal study case.

3.1 Causal Analysis

This approach consists, in general terms, of identifying the principal causes of the behavior of some phenomenon, representing them in the form of relations cause-effect in a causal model. A causal model consists of a Directed Acyclic Graph (DAG) over a set $V = \{V_1, \ldots, V_n\}$ of vertices, representing features of interest, and a set E of directed edges, or arrows, that connect these vertices.

These graphs can be causally interpreted if they have the features: Causal Markov condition, Minimality condition, Faithfulness condition. These features make the connection between causality and probability [36]. Causal modeling has four phases. The first phase is structure learning, where its objective is to find the principal structure of causal relations between several relevant features and a data set; it is to say, a causal graph G is found that represents the causal structure $G = (V, E)$; there exist several algorithms of structure learning, one of them is PC algorithm (Peter&Clark). The set E is described as:

$E = \{C_1, C_2, \ldots, C_n\}$, where each $C_i \in E$ is a set of ordered pairs,
$C_i = \{(v_i, y_1), (v_i, y_2), \ldots, (v_i, y_n)\}$, it is
$C_i = \{(v_i \in V; y_k \in V) \mid v_i \neq y_k, y_k$ is a direct cause of v_i relative to V and there is a directed edge from y_k to v_i in $G\}$

The second phase is estimation of the found causal relations; one of algorithms that perform it is Counting algorithm [37]. The third phase is the interpretation of causal model; the causal relations with higher magnitudes are analyzed and interpreted. The fourth phase is the causal model validation.

3.2 Characterizing One Dimension Bin-Packing Problem

The problem One Dimension Bin-Packing (BPP) is considered as combinatorial optimization problem NP-hard [38]. It consists to pack k items into minimum number of bins or containers mb; subjects to, all items packed in one bin or container does not exceeds the container capacity c (it is for all bins). The fitness function $f(x_i)$ for one solution x_i of One Dimension Bin-Packing problem is described in [39].

In this paper propose characterize the problem structure by means new feature vc. It is the coefficient of normalized pearson variation, from object sizes that are not multiple of container size; it considers the information obtained by feature f, proposed by [2]. The feature vc is described by expressions 1 and 2, where fr_i is the frequency of size s_i, dv is standard deviation of frequencies set Fre and me is the mean.

So too, in this experimentation, the features b and os are considered, which were significant in past experimentations [33–35]. The feature b describes the use of container; the proportion of total size that can be assigned to a container with capacity c. The feature os measures the variability of a sample of solutions generated randomly.

$$Fre = \{fr_i \mid s_i \, mod \, c \neq 0\} \tag{1}$$

$$vc = \frac{dv}{me}.(1 - f) \tag{2}$$

The set V begins to be built with the values of these features for each problem instance (see Expression 3). For set V, the rows represent the problem instances and columns are the values of these variables; where m is the total of problem instances.

$$V = \{\{b_1, vc_1, os_1\}, \{b_2, vc_2, os_2\}, \ldots, \{b_m, vc_m, os_m\}\} \tag{3}$$

3.3 Characterizing the Threshold Accepting Algorithm: Searching Fluctuation

The Threshold Accepting algorithm is described in [40]. In general terms, it starts generating a feasible initial solution. An iterative process is performed until freezing is reached. During this process, a neighborhood solution y is generated from one actual solution x; if the difference between fitness function values for these solutions $f(y) - f(x)$ is less than temperature value T, then solution y is the new actual solution. The temperature value T is decremented by freezing factor μ.

The term of searching fluctuation or behavior, in past experimentations and this work, refers to above algorithm iterative process. It is to say, all generated solutions are seen as the path traced by algorithm in axis "x" and the fitness function value of each from these solutions is represented in axis "y". The algorithm searching fluctuation has been characterized by feature *tm* in past works [33–35]. The feature *tm* is the average of valley sizes identified for all algorithm runs; it has been significant in past experimentations for characterizing behavior of algorithms Tabu Search and Threshold Accepting.

In this paper, the algorithm searching fluctuation is also characterized by new features *pp*, *pn* and *dv*; where *pp* is the positive slope and *pn* is the negative slope from one valley identified during the iterative process. Figure 1 shows one identified valley; axis "y" is the value of fitness function $f(x_i)$ for each generated solution x_i (axis "x"), during algorithm iterative process. Also the positive (*pp* - left side) and negative (*pn* - right side) slopping of an identified valley are shown.

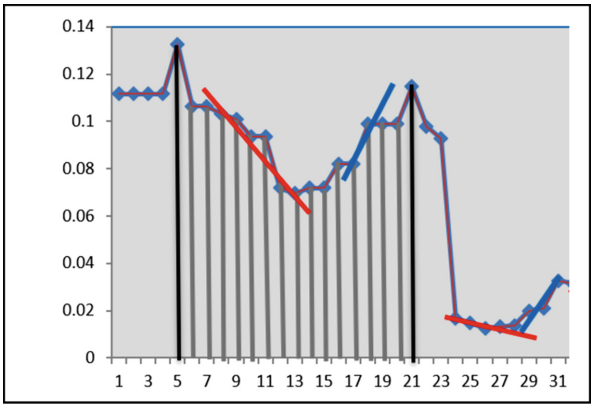

Fig. 1. Positive and negative slopes

The new features *pp* and *pn* are calculated by means of the approximation to tangent of a curve in one point; it is obtained by means of the secant defined by two points (one is where the tangent crosses and another is closed to curve). Figure 2 shows the slope of a curve in a point A, which is defined by the straight T. Also is shown how can be approximated this slope by means the secant; it is defined by points A and B.

When B is going to A, k and h going to 0, but its quotient is going to a determinate value; which is the slope of AT. Expression 4 defines the calculating of the slope of a curve in two points very closed ($x_0 + h$, x_0). Features *pp* and *pn* (Expressions 5 and 6) describe the average of the negative and positive slopping of the found valleys in all algorithm runs (*nrun*). The feature *vd* measures the dispersion of found valleys.

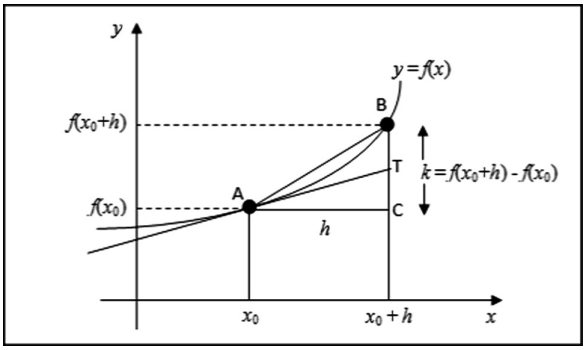

Fig. 2. Calculating a slope

$$slope = \frac{f(x_0 + h) - f(x_0)}{h} \tag{4}$$

$$pp = \frac{\sum_{run=1}^{nrun} pp_{run}}{ncor} \tag{5}$$

$$pn = \frac{\sum_{run=1}^{nrun} pn_{run}}{nrun} \tag{6}$$

The algorithms are executed in problem instances, during execution (considering that there is more than one algorithm that will resolve problem instances). These proposed features *tm*, *pp*, *pn* and *dv* are calculated for describing the behavior during searching of algorithm that will be object of study (see in Sect. 4). The set V continues to be built with the values of these features (Expression 7).

$$V = \{\{b_1, vc_1, os_1, tm_1, pp_1, pn_1, vd_1\}, \ldots, \{b_m, vc_m, os_m, tm_m, pp_m, pn_m, vd_m\}\} \tag{7}$$

3.4 Characterizing Algorithm Performance

After execution of each problem instance (1, 2, …, m), the algorithm performance is measured by features *time* and *quality*, described in expressions 8 and 9. The evaluations number of fitness function for feasible and infeasible solutions is considerate for *time*. The quality is the ratio between the best solution found by algorithm Q_f (final

number of containers) and theoretical solution Q_t (sum of object sizes divided by container capacity c), described by expression 10.

$$time = feasibles + infeasibles \tag{8}$$

$$quality = \frac{Q_f}{Q_t} \tag{9}$$

$$Q_t = \frac{\sum_{i=1}^{k} s_i}{c} \tag{10}$$

A set D is being built, mapping these performance features to each algorithm with the specific order as Expression 11; where n is the total of algorithms executed. The values of $quality_{11}$, $time_{11}$ meaning the performance of algorithm 1 for problem instance 1; the values of $quality_{12}$, $time_{12}$ meaning the performance of algorithm 2 for problem instance 1; the values of $quality_{m1}$, $time_{m1}$ meaning the performance of algorithm 1 for problem instance m; the values of $quality_{m2}$, $time_{m2}$ meaning the performance of algorithm 2 for problem instance m, and so on.

$$D = \left\{ \begin{array}{c} \{\{quality_{11}, time_{11}\}, \{quality_{12}, time_{12}\}, \ldots, \{quality_{1n}, time_{1n}\}\} \\ \{\{quality_{m1}, time_{m1}\}, \{quality_{m2}, time_{m2}\}, \ldots, \{quality_{mn}, time_{mn}\}\} \end{array} \right\} \tag{11}$$

The function r considers information of algorithm a_t from set D for one problem instance i (consider a_t as algorithm that will be object of study). Expression 12 describes the function r_i. This function returns the performance scope of algorithm a_t compared to other algorithms for problem instance i. The set R contains the values for all problem instances (see Expression 13). The set V is finally described by the values of set R in the last column; with specific order described by Expression 14; rows represent problem instances and columns represent information from proposed features for describing problem structure, algorithm searching fluctuation, algorithm performance scope.

$$r_i(a_t, i) = \left\{ \begin{array}{c} 1, \, if \, (D(quality_{it}) = D(quality_{i\propto}) \, and \, D(time_{it}) > D(time_{i\propto})) \\ or \\ D(quality_{it}) > D(quality_{i\propto}), \forall \propto \neq t; quality_{it} \in D; quality_{i\propto} \in D; \\ 0, otherwise. \end{array} \right\} \tag{12}$$

$$R = \{r_1(a_t, 1), r_2(a_t, 2), \ldots, r_m(a_t, m)\} \tag{13}$$

$$V = \left\{ \begin{array}{c} \{b_1, vc_1, os_1, tm_1, pp_1, pn_1, vd_1, r_1\}, \\ \{b_2, vc_2, os_2, tm_2, pp_2, pn_2, vd_2, r_2\}, \ldots, \\ \{b_m, vc_m, os_m, tm_m, pp_m, pn_m, vd_m, r_m\} \end{array} \right\} \tag{14}$$

4 Causal Study Case of Relation Between Bin-Packing Problem and Threshold Accepting

The latent relation between One Dimension Bin-Packing problem (BPP) and Threshold Accepting Algorithm is analyzed by means causal analysis; considering the proposed problem and algorithm features. For considering the logical design of algorithm in the analysis, understanding the relation of this with the problem structure, algorithm behavior, algorithm performance; we considered in our analysis to study the important logical area of initializing temperature control parameter T. Two algorithms a_1 and a_2 were considered, which they are different only in this logical area (initializing temperature control parameter T). The procedure of algorithm a_1 is to fix T to 1. For algorithm a_2, a sampling method without replacement is performed over problem solutions space to fix the initial temperature T. The set of algorithms $\{a_1, a_2\}$ is applied to 324 instances of problem BPP, where each algorithm is executed 16 times in each instance (runs number *nrun*); it was observed a very small variance between best solutions found of these runs. The algorithm that will object of study was selected randomly, it is a_2. These 324 problem instances is considered as training set (*instances* 1) and were selected randomly from Beasley, Scholl and Klein repositories [41, 42]. So too, other different 324 problem instances were selected randomly from these repositories, which are considered as test set (*instances* 2).

4.1 Discovering Causal Structure

The values of set V, representing proposed features b, vc, os, tm, pp, pn, dv, are normalized by means of method min-max; after that, these values are discretized using the method MDL [43]. So too, with the objective of considering metrics known by scientific community, autocorrelation coefficient (ac) and autocorrelation length (al), described in [44], were calculated and discretized. A set V_2 is built almost the same as set V, with the only difference that it contains the ac, al metrics instead of features tm, pp, pn, dv proposed for characterizing the algorithm searching behavior. The set V_2 has b, vc, os, ac, al and set R as last column.

The method PC [36] was performed, using as input the sets V and V_2, for discovering causal structures E and E_2; where causal inference software Hugin [45] with a confidence level 95% was used. Expressions 15, 16 and Fig. 3(a, b) show the learned structures. E with tm, pp, pn describes better the direct causes of algorithm performance scope (in terms of algorithm searching behavior) than E_2 with known metrics ac, al. Therefore, E will be considered for continuing with study case.

$$E = \left\{ \begin{array}{c} \{(R,pn),(R,pp),(R,tm),(R,vc)\},\{(os,b),(os,pp)\}, \\ \{(pn,vc),(pn,vd)\},\{(pp,pn)\},\{(b,vc)\} \end{array} \right\} \tag{15}$$

$$E_2 = \left\{ \begin{array}{c} \{(R,vc)\},\{(b,R),(b,vc),(b,ac)\}, \\ \{(al,os),(al,ac)\},\{(os,b)\} \end{array} \right\} \tag{16}$$

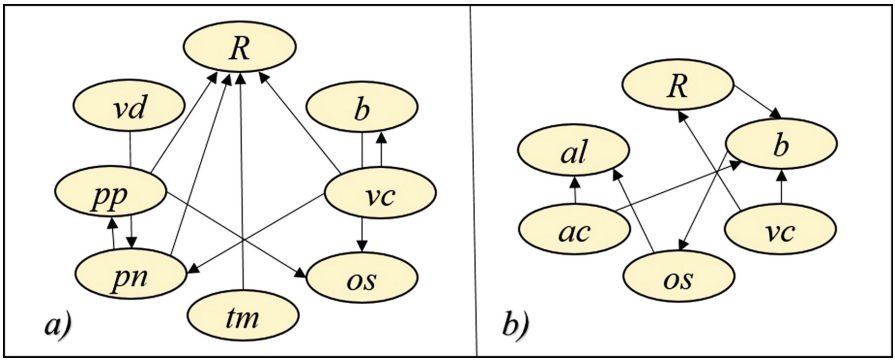

Fig. 3. Causal structures

The method Counting [36] was performed to estimate the causal relations from structure E. The function cR (Expression 17) is used to identify the direct causes of algorithm performance scope, in terms of set R. Table 2 shows the intervals of each feature and the major estimations for whose causal relations identified by function cR.

$$cR(E) = \{e_i = (v_i, y_k) |\ v_i = "R", \forall\ e_i \in C_i,\ C_i \in E\} \tag{17}$$

Table 2. Relevant causes

	%
P(R = 1 |vc = 2, pn = 3, pp = 2, tm = 2)	100
P(R = 1 |vc = 2, pn = 2, pp = 2, tm = 2)	100
P(R = 0 |vc = 1, pn = 1, pp = 1, tm = 1)	60

4.2 Understanding Discovered Knowledge: Design of Self-adaptive Algorithm

The interest algorithm a_2 corresponded better to problem instances where their structure had a variability of object size frequencies in the second interval; there are more possibilities for generating neighbor solutions, which can be accepted. The logical design forces the algorithm to begin with an initial temperature smaller than 1; it is taken of value of the objective function (values are between 0 and 1, being the optimal 1) of one solution from a sample of solutions generating randomly; being its searching very restrictive for accepting neighbor solutions. The algorithm behavior during its searching found valleys with sizes in the second interval, positive and negative slopping in the second and third interval; meaning that algorithm could enter and leave from these valleys, and does not stagnate in local optimal. The algorithm a_2 wins in time in the majority of instances; it found the best solution faster than algorithm a_1. However, a_2 had disadvantage with problem instances, where their structure had a

variability of object size frequencies in the first interval; there is not many possibilities for generating neighbor solutions, its behavior is almost flat (valley sizes, positive and negative slopping in the first interval); its condition of small temperature does not allow it to move much, ends faster than algorithm a_1 and lost in quality. The learned knowledge permit to design a self-adaptive threshold accepting algorithm (*sa*). It adjust automatically the temperature initial value; if feature *vc* has a value in the second interval, the temperature will be initialized from one solution taken from a solutions sample generated randomly; otherwise it will 1.

Self-Adaptive Threshold Accepting Algorithm (*sa*)

```
1  Begin
2     S = 100; size of neighborhood; μ = 0.85; freezing factor;
3     x = x*; initial feasible solution;
4     Calculate vc from parameters of problem instance
5     If (vc=2) Then
6        T = a solution chosen randomly from solutions space.
7     Else T = 1
8     Repeat
9        Repeat
10          For i = 1 to S
11             y = neighbor solution ∈ N(x)
12             If f(y) - f(x) < T Then x = y
13             Else solution y is rejected
14             i = i + 1
15       Until termal equilibrium is reached
16       T = μT
17    Until freezing is reached
18 End
```

5 Results Verification

Algorithm a_2 was executed in problem instances set *instances* 2; where sets D', R' were obtained. The set V', for algorithm a_2 also was obtained with proposed features and set R'. The set V' is normalized and discretized. The causal structure E is used with set V' for predicting values and to form the set R_p (predicted R values). The sets R' and R_p were compared. The causal model obtained an accuracy percentage of 84.88%, using the causal inference software NETICA [46]. The model validation indicates that knowledge learned and structured as E can be used to predict which algorithm will give the best solution to one new problem instance.

On the other hand, the self-adaptive Threshold Accepting algorithm was executed on set *instances* 1. Then, the experimentation results on algorithms (a_2, *sa*) were analyzed. The algorithm *sa* corresponded better to 239 problem instances from 324, representing the 74% percentage. In the instances with same *quality* for algorithms a_2

and *sa*, the time of algorithm *sa* is much less than a_2. It is to say, the differences are very big. Due the above, there is not necessity to verify significance on these differences. Then, the *quality* for both algorithms only is analyzed more deeply. The values of *quality* does not assume a normal distribution, therefore, we applied a non-parametric statistical test Wilcoxon of two dependent samples, using a confidence level of 95%. The null and alternative hypothesis H_0 and H_a were formulated, expressions 18 and 19; where μ_1 is the mean of quality performance results of algorithm a_2 and μ_2 is the mean of self-adaptive algorithm *sa*. The statistical software Dataplot [47] was used.

$$H_0 : \mu_1 - \mu_2 = 0 \tag{18}$$

$$H_a : \mu_1 - \mu_2 \neq 0 \tag{19}$$

The results of application of statistical test were: a calculated statistical value of 2.4802 and a theoretical statistical value of 1.96. Therefore, the null hypothesis H_0 is rejected and the alternative hypothesis H_a is accepted. There is a significant difference between means. The self-adaptive algorithm *sa* improves the performance of analyzed algorithm a_2.

6 Conclusions

In this paper, new features for characterizing problem structure and algorithm searching fluctuation were proposed. A causal study case could have performed with these proposed features; in the framework of Bin-Packing problem and Threshold Accepting algorithm, considering also information about algorithm logical area. These features were relevant, significant and permitted in the causal analysis to discover relations cause-effect from relationship between problem and algorithm. The causal structure obtained an accuracy percentage of 84.88% over another problem instances set (test data). The causal relations learned represent knowledge between problem structure, algorithm searching behavior, logical design (specifically in the logical area of initializing the temperature) and its performance. A self-adaptive Threshold Accepting algorithm was designed from the above discovered knowledge; it considers the problem structure to initialize the temperature. The performance results of original algorithm and self-adaptive algorithm were analyzed. The self-adaptive algorithm solved 74% of 324 problem instances better than analyzed algorithm. An application of the Wilcoxon statistical test, using a 95% of confidence level, over these results, indicates a significant difference. The proposed framework could be a methodological guideline for discovering knowledge and designing self-adaptive algorithms that adjust better to real problems. As next task, is interesting to analyze more about the knowledge discovered from relation problem-algorithm. It is to say, considering also the information of relations cause-effect; from proposed features *tm*, *pp*, *pn* (direct causes) and algorithm performance scope (effect); for constantly adjusting the algorithm logical during

execution, depending searching behavior. As future work is to study the relation between problem BPP and other algorithms Genetic, Ant Colony Optimization; and so on for other optimization problems.

References

1. Soares, C., Pinto, J.: Ranking learning algorithms: using IBL and Meta-learning on accuracy and time results. J. Mach. Learn. **50**(3), 251–277 (2003)
2. 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
3. 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
4. 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)
5. Le, M.N., Ong, Y.S., Jin, Y., Sendhoff, B.: Lamarckian memetic algorithms: local optimum and connectivity structure analysis. Memetic Comput. **1**(3), 175–190 (2009)
6. Cayci, A., Menasalvas, E., Saygin, Y., Eibe, S.: Self-configuring data mining for ubiquitous computing. Inf. Sci. **246**, 83–99 (2013)
7. Tavares, J.: Multidimensional knapsack problem: a fitness landscape analysis. IEEE Trans. Syst. Man Cybern. Part B **38**(3), 604–616 (2008)
8. Pérez, J., Cruz, L., Landero, V.: Explaining performance of the threshold accepting algorithm for the bin packing problem: a causal approach. Polish J. Environ. Stud. **16**(5B), 72–76 (2007)
9. Xu, L., Hoos, H., Leyton-Brown, K.: Hydra: automatically configuring algorithms for portfolio-based selection. In: AAAI, vol. 10, pp. 210–216 (2010)
10. 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. SCI, vol. 452, pp. 361–375. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-33021-6_29
11. Hutter, F., Xu, L., Hoos, H., Leyton-Brown, K.: Algorithm runtime prediction: methods & evaluation. Artif. Intell. **206**, 79–111 (2014)
12. 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)
13. 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. (2004)
14. Yuen, S., Zhang, X.: Multiobjective evolutionary algorithm portfolio: choosing suitable algorithm for multiobjective optimization problem. In: IEEE Congress on Evolutionary Computation (CEC), Beijing, China, pp. 1967–1973 (2014)
15. 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)
16. 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)

17. Mull, N., Fremont, D.J., Seshia, S.A.: On the hardness of SAT with community structure. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 141–159. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_10
18. 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)
19. Pavón, R., Díaz, F., Laza, R., Luzón, M.V.: Experimental evaluation of an automatic parameter setting system. Expert Syst. Appl. **37**(7), 5224–5238 (2010)
20. Hoos, H.H.: Automated algorithm configuration and parameter tuning. In: Hamadi, Y., Monfroy, E., Saubion, F. (eds.) Autonomous Search, pp. 37–71. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21434-9_3
21. Yeguas, E., Luzón, M.V., 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)
22. Biedenkapp, A., Lindauer, M.T., Eggensperger, K., Hutter, F., Fawcett, C., Hoos, H.H.: Efficient parameter importance analysis via ablation with surrogates. In: AAAI, pp. 773–779 (2017)
23. Blot, A., Pernet, A., Jourdan, L., Kessaci-Marmion, M.-É., Hoos, H.H.: Automatically configuring multi-objective local search using multi-objective optimisation. In: Trautmann, H., et al. (eds.) EMO 2017. LNCS, vol. 10173, pp. 61–76. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54157-0_5
24. Guo, H., Hsu, W.H.: A learning-based algorithm selection meta-reasoner for the real-time MPE problem. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 307–318. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30549-1_28
25. Guerri, A., Milano, M.: Learning techniques for automatic algorithm portfolio selection. In: Burke, V.A. (ed.) Proceedings of the 16th Biennial European Conference on Artificial Intelligence, pp. 475–479. IOS Press, Valencia (2004)
26. Hoos, H.H., Smyth, K., Stützle, T.: Search space features underlying the performance of stochastic local search algorithms for MAX-SAT. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 51–60. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_6
27. Konak, A.: Simulation optimization using tabu search: an empirical study. In: Kuhl, M.E., Steiger, N.M., Armstrong, F.B., Joines, J.A. (eds.) Proceedings of Winter simulation Conference, pp. 2686–2692 (2005)
28. Chevalier, R.: Balancing the effects of parameter settings on a genetic algorithm for multiple fault diagnosis. In: Artificial Intelligence. University of Georgia (2006)
29. 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
30. 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)
31. Gagliolo, M., Schmidhuber, J.: Learning dynamic algorithm portfolios. Ann. Math. Artif. Intell. **47**(3–4), 295–328 (2006)
32. Montero, E., Riff, M.-C.: On-the-fly calibrating strategies for evolutionary algorithms. Inf. Sci. **181**, 552–566 (2011)
33. 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, Jacek 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

34. Pérez, J., et al.: A causal approach for explaining why a heuristic algorithm outperforms another in solving an instance set of the bin packing problem. In: An, A., Matwin, S., Raś, Z. W., Ślęzak, D. (eds.) ISMIS 2008. LNCS (LNAI), vol. 4994, pp. 591–598. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68123-6_64
35. Pérez, J., Cruz, L., Pazos, R., Landero, V., Pérez, V.: Application of causal models for the selection and redesign of heuristic algorithms for solving the bin-packing problem. Polish J. Environ. Stud. **17**(4C), 25–30 (2008). (ACS-AISBIS 2008)
36. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, 2nd edn. The MIT Press, Cambridge (2001)
37. Korb, K.: Bayesian Artificial Intelligence. Chapman and Hall, London (2004)
38. McGeoch, C.C.: Experimental analysis of algorithms. In: Pardalos, P.M., Romeijn, H.E. (eds.) Handbook of Global Optimization. Nonconvex Optimization and Its Applications, vol. 62, pp. 489–513. Springer, Boston (2002). https://doi.org/10.1007/978-1-4757-5362-2_14
39. Khuri, S., Schütz, M., Heitkötter, J.: Evolutionary heuristics for the bin packing problem. In: Artificial Neural Nets and Genetic Algorithms, pp. 285–288. Springer, Vienna (1995). https://doi.org/10.1007/978-3-7091-7535-4_75
40. Dueck, G., Scheuer, T.: Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J. Comput. Phys. **90**(1), 161–175 (1990)
41. Beasley, J.E.: OR-Library. Brunel University (2006). http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html
42. Scholl, A., Klein, R. (2003). http://www.wiwi.uni-jena.de/Entscheidung/binpp/
43. Fayyad, U., Irani, K.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. IJCAI 1022–1029 (1993)
44. Merz, P., Freisleben, B., et al.: Fitness landscapes and memetic algorithm design. New Ideas Optim. 245–260 (1999)
45. Hugin Expert. www.hugin.com
46. Norsys Corporation. www.norsys.com
47. Dataplot. www.itl.nist.gov/div898/software/dataplot/homepage.htm