*Mathematical and Computational Applications*

MDPI

*Article*

# Causal Analysis to Explain the Performance of Algorithms: A Case Study for the Bin Packing Problem

**Jenny Betsabé Vázquez-Aguirre *, Guadalupe Carmona-Arroyo, Marcela Quiroz-Castellanos and Nicandro Cruz-Ramírez**

Artificial Intelligence Research Institute, Universidad Veracruzana, Xalapa 91097, Veracruz, Mexico; gcarmonaarroyo@gmail.com (G.C.-A.); maquiroz@uv.mx (M.Q.-C.); ncruz@uv.mx (N.C.-R.)
* Correspondence: jenvazquez@uv.mx

**Abstract:** This work presents a knowledge discovery approach through Causal Bayesian Networks for understanding the conditions under which the performance of an optimization algorithm can be affected by the characteristics of the instances of a combinatorial optimization problem (COP). We introduce a case study for the causal analysis of the performance of two state-of-the-art algorithms for the one-dimensional Bin Packing Problem (BPP). We meticulously selected the set of features associated with the parameters that define the instances of the problem. Subsequently, we evaluated the algorithmic performance on instances with distinct features. Our analysis scrutinizes both instance features and algorithm performance, aiming to identify causes influencing the performance of the algorithms. The proposed study successfully identifies specific values affecting algorithmic effectiveness and efficiency, revealing shared causes within some value ranges across both algorithms. The knowledge generated establishes a robust foundation for future research, enabling predictions of algorithmic performance, as well as the selection and design of heuristic strategies for improving the performance in the most difficult instances. The causal analysis employed in this study did not require specific configurations, making it an invaluable tool for analyzing the performance of different algorithms in other COPs.

**Keywords:** causal analysis; intervention of variables; causal effect; bin packing problem

check for updates

**Citation:** Vázquez-Aguirre, J.B.; Carmona-Arroyo, G.; Quiroz-Castellanos, M.; Cruz-Ramírez, N. Causal Analysis to Explain the Performance of Algorithms: A Case Study for the Bin Packing Problem. *Math. Comput. Appl.* **2024**, *29*, 73. https://doi.org/10.3390/mca29050073

Received: 24 February 2024
Revised: 24 August 2024
Accepted: 26 August 2024
Published: 28 August 2024

**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

The off-line one-dimensional Bin Packing Problem (BPP) is a well-known combinatorial optimization problem with many practical real-world applications. For the solution of BPP, elaborate procedures have been designed, incorporating various techniques. While numerous research studies have focused on solving this problem, few works have focused on analyzing and characterizing the difficulty of the test instances.

In this paper, we have focused on performing a causal analysis of the results obtained by two state-of-the-art algorithms when solving instances of the BPP with different features and difficulty degrees. Based on the specialized literature and in previous exploratory analysis, we selected a set of features to characterize the instances and the algorithm performance, and we used them to carry out the causal analysis of the present work.

Most techniques for data analysis explain the association or correlation among variables. Furthermore, for prediction and classification, other techniques have been used, but their interpretation in causality terms is not recommended. In this work, we propose a causal analysis that focuses on the Intervention of Variables through Causal Bayesian Networks (CBNs). The Intervention of Variables in CBNs allows not only to clearly explain the variables that can be a potential cause of another effect variable but also to know which specific value of some variable causes a determined value of the effect. The latter does not only indicate the causal relationship but also the causal force with which this has been generated.

This paper continues as follows. Section 2 summarizes the most important approaches that have been used to study the difficulty of instances for different combinatorial optimization problems. Subsequently, the causality section (Section 3) focuses on showing the method by which we selected the possible causal variables; these variables explain the complexity of the algorithms to solve COPs; also in this section, we explain how to make the causal interpretation. Section 4 describes our case of study, BPP, explaining, in general terms, its purpose and complexity. Then, we present the instances used to analyze the performance of two state-of-the-art algorithms for solving this problem (Section 4.1). Next, we introduce the indexes for measuring the instances properties that we selected in a previous exploratory study (Section 4.2). In Section 4.3, we present the algorithms that we have selected (GGA-CGT and MBS), since, in previous analyses [1], they gave the best performance for solving BPP. Moreover, we also present the variables that we have used to analyze the performance of these algorithms. Section 5 presents the causal results obtained, showing the characteristics and the ranges of values that cause the good or bad performance of the algorithms. Finally, we provide a set of conclusions and recommendations for future work (Sections 6 and 7).

## 2. Experimental Study of Combinatorial Optimization Problems

Specialized literature includes several implementations of heuristic algorithms proposed to solve combinatorial optimization problems (COPs). In most of these implementations, the emphasis focuses on developing strategies to solve the problem, concentrating on the characterization of the specific COP. However, few research studies have focused on analyzing the performance of solution algorithms.

In most cases, research around these optimization problems focuses on comparing new algorithms against those that have obtained better results, often with the intention of having the best algorithm to answer a specific COP. Many authors have emphasized the importance of analyzing the characteristics of the problem instances to be solved, highlighting the need for a guide to describe and analyze the performance of the solution strategies [2–4].

Several works related to the characterization and analysis of the optimization process with different approaches exist; the main ones in this field are summarized in Smith-Miles and Lopes [5] and Quiroz-Castellanos et al. [6]. Additional work focuses on the selection and configuration of the algorithms, the analysis of the structure of the search space, and the analysis of the fitness landscape. Furthermore, to predict the performance of the algorithms that solve these COPs, significant efforts have been made, most of them making use of statistical methods, multivariate analysis, and machine learning techniques [7].

Alternative ways of characterizing the difficulty of NP-hard COP instances have focused on identifying phase transition properties for distinguishing test instances as easy or hard. Thus, they have been widely used, demonstrating their effect on both exact and heuristic algorithms [8,9]. Moreover, the difficulty of NP-hard COP instances can be measured by studying the structure of the optimal solutions [10].

All the proposed characterization approaches can be combined to generate the best subset of features that allow us to discriminate between instances of the problem with different levels of difficulty. This subset of features can be analyzed together with metrics of the performance of the algorithms (such as the final quality of the solutions, the execution time, the number of iterations, and robustness) to obtain relationships that explain the process behavior. To achieve this objective, cause–effect associations are explored among measures that characterize the features of the instances and the performance of the algorithm. Related studies presented in the state of the art perform statistical and multivariate analysis to obtain these associations. Few research studies have used causal analysis as a tool to explain the performance of algorithms. Quiroz-Castellanos et al. [11] present an experimental application of causal inference to the algorithmic optimization process, which aims to evaluate the impact of the parameters that control the behavior of a heuristic

algorithm. In another paper, Quiroz-Castellanos et al. [3] employ causal analysis for the refinement of features and the study of algorithmic performance relationships.

In this paper, we focus our efforts on performing a causal analysis to determine which characteristics of the instances cause the performance of the algorithms that solve one of the most popular COPs, the one-dimensional Bin Packing Problem. We propose a causal analysis through the Intervention of Variables in CBNs. This analysis will not only identify the potential variables that cause performance but, also, it will recognize the specific values of the variables that are having an effect on performance.

The construction of CBNs offers several advantages. Firstly, the judgments required to create the model are more meaningful and accessible, which enhances its reliability. Additionally, these models have the ability to adapt to external or spontaneous changes. For instance, in a conventional Bayesian network, the probability of an event can be determined from a joint distribution, and one can observe how this probability changes with new observations. However, causal models go further by also demonstrating how probabilities would be altered by external interventions. This feature makes them a valuable tool for causal analysis and decision-making, allowing for a better understanding of the impact of intentional changes on the system. Currently, there are few studies that apply causal analysis to explain the performance of algorithms. For this reason, we have incorporated this type of analysis in our proposal, aiming to determine whether the relationships between instance features and algorithm performance can better explain their behavior.

The next section presents a description of this causal analysis approach, which can be applied to understand the causal relationships between features of instances and algorithmic performance to explain the behavior of other algorithms that solve other COPs. The reader interested in reviewing characterization indexes that can be applied to measure the difficulty of different COPs and algorithmic performance can consult the works of Smith-Miles and Lopes [5] and McGeoch [12], where examples of a variety of COPs and algorithms as well as characterization measures are offered.

## 3. Causal Bayesian Networks

In this section, we introduce the causal models. First of all, we will talk about causal estimation through CBN. A CBN is a probabilistic model similar to a conventional Bayesian network (BN). Judea Pearl introduced BNs in the early 1980s to facilitate the prediction and abduction in Artificial Intelligence (AI) systems [13]. These models combine two theories: graph theory and Bayesian Probability Theory.

The graphical part of Bayesian network models is always represented with Directed Acyclic Graphs (DAG). A relevant feature of these models is that the nodes represent the variables in the problem, and each one contains its associated probability tables. From the structure of a BN, we can obtain the probabilities of the variables in the data. Moreover, it is possible to get the probabilistic dependence of variables or the conditional independence of one variable given another one (or set of them). The influence force between the nodes of a network is in the conditional probabilities.

It is important to note that, in our case study, we have chosen to use this model specifically for categorical variables. Therefore, when dealing with continuous variables, a discretization process will be required.

Figure 1 shows a typical example of a BN for modeling the relationships between features of an instance of an optimization problem, which have an impact on the behavior of an algorithm and its final performance. In this model, $w$ is a measure of a COP's feature, $ac$ and $ad$ are algorithm behavior metrics, and $p$ is an algorithm performance measure. An arc between each pair of variables means that they are dependent on each other. The tables in the network represent the conditional probabilities of each variable given its parents. Only in the case when the variable does not possess parents is its value going to be the marginal probability, as represented by $w$ in the network in Figure 1.

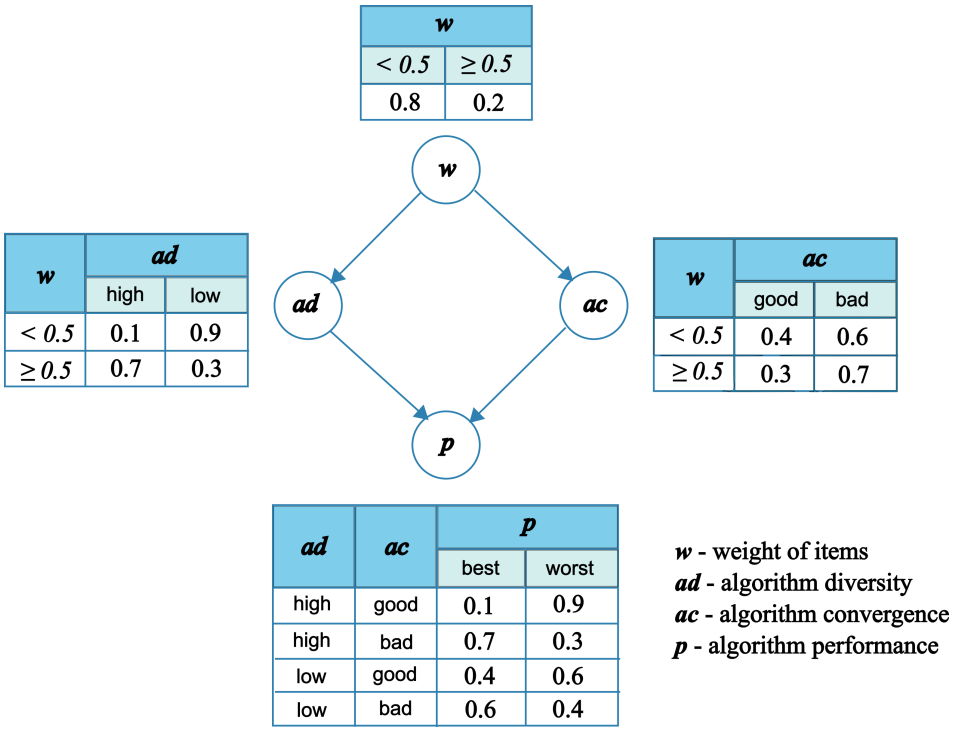



**Figure 1.** Representation of a conventional Bayesian network.

Formally, the joint probability of a BN can be obtained using Equation (1) [13]:

$$P(x_1, x_2, ..., x_n) = \prod_j P(x_j|pa_j) \tag{1}$$

where $x_i$ represents the nodes of the BN, and $pa_j$ the parents of each node. For example, we can express the complete model of Figure 1 as in Equation (2):

$$P(w, ad, ac, p) = P(w)P(ad|w)P(ac|w)P(p|ad, ac) \tag{2}$$

From Equation (1), it is possible to get probabilities, whether for a query or a belief update. For example, we could estimate the probability of each variable or the conditional probability of a specific variable given another that we know. Returning to Figure 1, we can obtain $P(ac = \text{good}|ad = \text{low}) = 0.39$; this means that the probability of *ac* taking the value "good", given that *ad* is observed to have the value "low", is 0.39. A recommendation about a BN is that the conclusions obtained from them should never be used for causal purposes. These probabilities are equivalent to making an "observation" about the dataset. In addition, an observation only tells us that one variable can occur given another or others.

Judea Pearl proposed CBNs as an alternative for obtaining causal probabilities [14]. These probabilities are estimated after the Intervention of Variables. An Intervention of Variables consists of simulating a physical intervention for the variables in the network. If the resulting probabilities after observation and intervention are not equal, the probabilities obtained by intervention can be considered causals. The fundamental difference between both models (BN and CBN) is the Intervention of Variables. In a CBN, each intervened variable could be called a potential cause. The intervention is done through the method known as do-operator, which is represented by $do(x)$.

It is important to note that, currently, traditional AI algorithms used to build BNs can find cause–effect relations among variables. However, they have no way to assess whether those relations are indeed causal. Therefore, we have proposed two methods for their construction. The first consists of building the network with a traditional AI algorithm, then validating the causal relationships found. For this, we lean on the expert who has provided us with the dataset. This process implies that any relations can be eliminated, changed,

or added to the network, but this implicates minimal changes in the network. In other words, the expert could identify whether the direction of the arrows between variables maintains a causal relationship (the arrowhead always indicates which is the effect). The second method only considers expert knowledge, which requires drawing the graphical structure manually. In this case, the expert should draw the network, ensuring that the graph is acyclic and its directions represent causes and effects. For the case study presented in this paper, we only utilized the last method because our expert knew the nature of the data perfectly. So, all our CBNs have been made up manually.

After obtaining the graphical model, we can perform an intervention by following the steps outlined below:

1. Mutilate the network: delete all parent arrows of the causal variable. This step eliminates the influence produced by other variables over the intervening variable.
2. Obtain the new causality expression: once the arrows of the intervening variables are deleted, we must obtain the new causality expression. To do so, it is necessary to follow some rules of inference. For details on these rules, refer to Pearl [14].

We consider Figure 1; we suppose that we want to do an intervention on variable *ad* and observe its effect over $p$. The mutilated CBN would look like Figure 2.

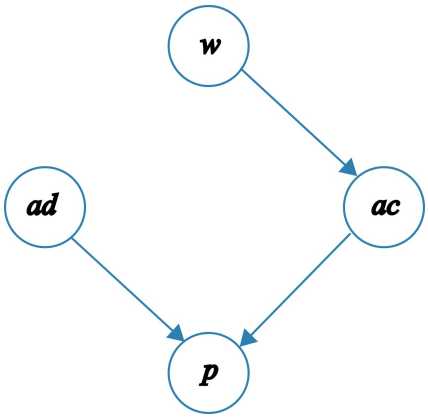



**Figure 2.** A mutilated Causal Bayesian Network taken from the BN shown in Figure 1.

In a conventional BN, the probabilities are obtained for each node, given the set of its parents. In the case of a causal model where we deleted all influences that arrived at the causal node, the new causal expression of probability also needs modification, as in the graphical model. The above is expressed formally with Equation (3) [14].

$$P_x(v) = \prod_{j|V_j \notin X} P(v_j|pa_j) \tag{3}$$

where the following definitions apply:

- $x$ represents a specific value of the intervened variable $X$, this value is known as $do(x)$.
- $V_j$ represents the sets of variables in the network, except the intervened variable.
- $v_j$ represents each variable in the set of variables $V_j$.
- $pa_j$ are the parents of all variables $v_j$, except the intervened variable.

From Equation (3), we can obtain the new probability expression that will allow us to calculate the causal probabilities. We must calculate this new probability by modifying Equation (2), where we will delete probabilities corresponding to the intervening variable. Following our example, we will obtain Equation (4):

$$P(w, ad, ac, p) = P(w)P(ac|w)P(p|ad, ac) \tag{4}$$

For example, if we want to intervene on *ad* when its value is equal to "high", to see its effect on $p$, the new expression of probability could look like Equation (5):

$$P_{ad=\text{high}}(w, ac, p) = P(w)P(ac|w)P(p|ad = \text{high}, ac) \tag{5}$$

This last equation allows us to extract the actual estimation of the Intervention of Variables in the network. According to our example, their causal probabilities are as follows:

$P(p = \text{best}|ad = \text{high}) = 0.48$
$P(p = \text{worst}|ad = \text{high}) = 0.52$
$P(p = \text{best}|ad = \text{low}) = 0.54$
$P(p = \text{worst}|ad = \text{low}) = 0.46$

The probabilities obtained by observation in the BN are as follows:

$P(p = \text{best}|ad = \text{high}) = 0.50$
$P(p = \text{worst}|ad = \text{high}) = 0.50$
$P(p = \text{best}|ad = \text{low}) = 0.53$
$P(p = \text{worst}|ad = \text{low}) = 0.47$

We can see the difference with the simple observation obtained from the BN. Because the difference between both estimations exists, the probabilities obtained by intervention can already be considered causal, and we will use them to calculate each causal effect (CE).

Once the causal probabilities have been calculated, we can estimate the CE or the average causal effect that one variable has on the other, for which we are going to use Equation (6) [14]. The CE tells us whether the variation in one variable can cause a change in another variable.

$$CE = P(Y = y|do(\hat{x})) - P(Y = y|do(\hat{x}')) \tag{6}$$

where the following definitions apply:

- $Y$ represents the variable effect that we want to estimate.
- $Y = y$ is a specific value of the effect.
- $\hat{x}$ is a specific value of the intervened variable.
- $\hat{x}'$ is another value of the intervened variable for the same value of $y$.

For example, if we continue with our example, obtaining a causal effect from Equation (6) is equivalent to

$$CE = P(p = \text{best}|do(ad = \text{high})) - P(p = \text{best}|do(ad = \text{low})) = -0.06 \tag{7}$$

Finally, we can interpret the causal effects as well. If the causal effect takes a positive value, we can say that a generative cause exists. If the value is negative, we can say that the variable is a preventive cause and, if the value is equal to zero, the variable is not causal [15]. There is no threshold for determining the causal strength of CEs. We recognize that, as the values of CE approach to $|1|$, the causal strength increases. Therefore, while we primarily focus on interpreting values greater than $|0.3|$, it is important to note that there could be cases where some values fall below this threshold but are still considered for analysis. In our example, we have obtained a CE equal to $-0.06$ (close to zero), so we can consider that $ac$ is not a cause of $p$. In other words, we can consider that the "algorithm diversity" is not a causal factor in the algorithm "performance".

As we could observe in the example presented in this section, the causal analysis can be applied to identify the set of critical features of an optimization problem that have a causal effect on the performance of an algorithm, explaining why the algorithm under study follows a specific behavior when solving an instance of an optimization problem. In this work, we select the optimization process of the one-dimensional BPP as our case study. The aim is to find which features of the instances are causing the performance of two algorithms. In Section 5, we present the results obtained with this analysis. In the following sections, we describe our case study.

## 4. The Bin Packing Problem Optimization Process

A well-known grouping COP is the one-dimensional BPP. Its goal is to pack a set of $n$ items, each with a specific weight $w_i$ in a minimum number of bins with fixed capacity $c$, with the restriction of not exceeding the capacity of any bin.

BPP belongs to the NP-hard class [16,17], and its complexity makes it practically impossible to use exact algorithms to solve it. To find an optimal solution, all possible partitions of the set of $n$ items in $n$ or fewer subsets must be considered. Inconveniently, the complexity of BPP grows dramatically with the problem size since the number of possible partitions can be longer than $(n/2)^{n/2}$.

Formally, given the same fixed capacity $c$ for all bins, and given a set $N = 1, \ldots, n$ of items to be packed, with respective weights $w_i$, where $i \in N$ and $0 < w_i \leq c$, BPP comprises packing all of the items into the minimum number of bins $m$ such that the sum of the items in each bin does not exceed $c$. That is, we seek to find the smallest amount of subsets $B_j$, for $1 \leq j \leq m$, of a partition of the set $N$.

$$\bigcup_{j=1}^{m} B_j = N \tag{8}$$

Such that

$$\sum_{i \in B_j} w_i \leq c, \quad 1 \leq j \leq m, \quad i \in N \tag{9}$$

Several algorithms have been proposed to solve BPP instances. Some of them have obtained interesting results. There are algorithms based on heuristics and metaheuristics; some are easy to implement and others can be more complicated. However, few studies have focused on the analysis of the BPP instances and their difficulty.

In this paper, we focus on the performance analysis for two algorithms implemented to solve BPP. This analysis has centered on finding out what features of the instances could cause the performance of these algorithms.

Certain efforts have been made to investigate the impact of the items weights, the bin capacity $c$, and the number of items $n$ on the difficulty of instances [1,6,18]. The range of the weights of the items can impact the difficulty of the instances, which has been proven especially for small ranges between 0.15 and 0.5 of the bin capacity. Another feature that showed an important impact on the difficulty is when the average weight is within 0.3 and 0.5 of the bin capacity. In addition, average weights greater than 0.5 of the bin capacity seem to favor the solution of instances for most of the algorithms used to solve BPP, especially for the state-of-the-art instances. The average weights of less than 0.3 of the bin capacity have been shown to be beneficial for algorithms that include randomized strategies. Moreover, the bin capacity has been an important feature to consider; it has been shown that the higher the bin capacity, the more difficult the instances become to solve. Finally, it has also been proven that instances where the optimal solution leaves little free space in the bins are harder to solve.

In this paper, we perform a causal analysis of two instance sets; the aim is to determine which characteristics cause the performance of two well-known algorithms for solving BPP. The following sections detail each instance set and their features. We used two groups of instances and their features to carry out each causal analysis. We have used the features as causal variables and the result of the solution algorithm as effect variables.

### *4.1. BPP Instances*

To evaluate the algorithm's performance, we used two groups of instances. The first one was generated with specific features; all of them have perfect packing; that is, in the optimal solution, the items must be packed such that the bins do not hold available space.

This set of instances was proposed by Carmona-Arroyo et al. [1]; it includes 2800 instances, referred to as $BPPv_u_c$, and they are available at https://github.com/gcarmonaar/

CO/tree/master/bpp_vu_c-instances (accessed on 23 February 2024). The instances are grouped into four classes: $BPP.25$, $BPP.5$, $BPP.75$, and $BPP1$. The weights of the items in the classes of instances oscillate between $(0, 0.25c]$, $(0, 0.50c]$, $(0, 0.75c]$, and $(0, c]$, where $c$ is the capacity of each bin. There are seven sets in each class, each one consists of 100 instances with bin capacities $c = 10^2$, $c = 10^3$, $c = 10^4$, $c = 10^5$, $c = 10^6$, $c = 10^7$, and $c = 10^8$, respectively. Table 1 shows the features of the instances in each group: the number of instances (#), the distribution of the the weights of the items ($(0, v_u c]$), the bin capacities ($c$), and the number of items ($n$).

**Table 1.** Group of instances $BPPv_u_c$.

| Class | # | $(0, v_u c]$ | $c$ | $n$ |
|---|---|---|---|---|
| BPP.25 | 700 | $(0, 0.25c]$ | | $[110, 154]$ |
| BPP.50 | 700 | $(0, 0.50c]$ | $10^2, 10^3, 10^4,$ | $[124, 167]$ |
| BPP.75 | 700 | $(0, 0.75c]$ | $10^5, 10^6, 10^7, 10^8$ | $[132, 165]$ |
| BPP1 | 700 | $(0, c]$ | | $[148, 188]$ |

Moreover, to accomplish a deeper study of the features that define the difficulty of a BPP instance, we proposed and used new instances, referred to as $fv_l_v_u_c$, and they are available at https://github.com/gcarmonaar/CO/tree/master/f_vl_vu_c-instances. The weights were generated in the intervals $(v_l c, v_u c]$, so that $v_l$ and $v_u$ correspond to the lower and upper bounds on the weights, respectively, relative to the bin capacity $c$. The instances are grouped into five classes with $v_l = 0$, $v_l = 0.1$, $v_l = 0.2$, $v_l = 0.3$, and $v_l = 0.4$, respectively. Table 2 shows the features of the instances in each class: the number of instances (#), the intervals of the the weights of the items $(v_l c, v_u c]$, the bin capacities ($c$), and the number of items ($n$).

**Table 2.** Group of instances $fv_l_v_u_c$.

| Class | # | $(v_l c, v_u c]$ | $c$ | $n$ |
|---|---|---|---|---|
| $f0$ | 3500 | $(0c, 0.6c](0c, 0.7c](0c, 0.8c]$ $(0c, 0.9c](0c, 1c]$ | | $[246, 438]$ |
| $f0.1$ | 2800 | $(0.1c, 0.6c](0.1c, 0.7c]$ $(0.1c, 0.8c](0.1c, 0.9c]$ | $10^3, 10^4, 10^5, 10^6$ | $[236, 353]$ |
| $f0.2$ | 2100 | $(0.2c, 0.6c](0.2c, 0.7c]$ $(0.2c, 0.8c]$ | $10^7, 10^8, 10^9$ | $[223, 295]$ |
| $f0.3$ | 1400 | $(0.3c, 0.6c](0.3c, 0.7c]$ | | $[207, 245]$ |
| $f0.4$ | 700 | $(0.4c, 0.6c]$ | | 200 |

Figure 3 contains an example of the initial part of a BPP instance with $n = 393$ items and bin capacity $c = 1000$, where it is known that the optimal solution packs the items in 100 bins; finally, the instance includes the list of weights of the 393 items. For each instance of the classes described in Tables 1 and 2, there is a text file similar to the one presented in Figure 3, which includes the parameters that define the instance and that will be used to calculate the features that could have an impact on its difficulty in terms of algorithmic performance.

The characterization indexes that will be used to measure these properties will be described in the following section. It is important to mention that the indexes were discretized, as the applied causal analysis operates solely with categorical variables. The discretization process is detailed in Section 5.1.

```
//Number of items
393
//Bin Capacity
1000
//Optimal Solution
100
//Weight of Item
1
79
454
276
190
132
29
.
.
.
```

**Figure 3.** Example of a BPP instance file.

### *4.2. BPP Features*

Different works have proposed characterization indexes by studying the difficulty of the BPP instances. Cruz [19] formulated a set of indexes to measure the characteristics of BPP, and Quiroz-Castellanos [6] presented a set of indexes to measure the difficulty of BPP.

In Carmona-Arroyo et al. [1], we presented a comprehensive study of the performance of four well-known BPP algorithms, identifying and quantifying the features of the instances, and the performance of the algorithms, to recognize the characteristics that define the hardness of the BPP instances. Table 3 describes the characterization indexes selected to measure the difficulty of the BPP instances.

**Table 3.** Indexes presented by Cruz [19] and Quiroz-Castellanos [6].

| Equation | Description |
|---|---|
| $t = \frac{\sum_{i=1}^{n} \frac{w_i}{n}}{c}$ | Capacity occupied by an average item. |
| $d = \sqrt{\frac{\sum_{i=1}^{n} [t - (\frac{w_i}{c})]^2}{n}}$ | Degree of dispersion of the quotient of the weight of the items and the bin capacity. |
| $Range = \max(w_i) - \min(w_i)$ | Difference between the largest and the smallest weight. |
| $RC = \frac{Range}{c}$ | Range of the weights relative to the bin capacity $c$. |
| $Major_Multiplicity = \max(M)$ | Maximum frequency of weight recurrence for the $n$ items, where $M$ saves the frequency for each different weight. |
| $Minor_Multiplicity = \min(M)$ | Minimum frequency of weight recurrence for the $n$ items, where $M$ saves the frequency for each different weight. |
| $Pct_\tilde{n} = \frac{\tilde{n}}{n}$ | Proportion of large items, where $\tilde{n}$ is the number of items with weight larger than fifty percent of the bin capacity, i.e., $w_i > c/2$. |

### *4.3. BPP Algorithms*

As we mentioned in previous sections, several algorithms have been proposed to solve BPP. In Carmona-Arroyo et al. [1], we presented a complete exploratory analysis of features of BPP instances. Furthermore, we show results obtained from four different algorithms. In this work, we used the following two algorithms that gave better results in our preliminary analysis.

#### 4.3.1. MBS

Minimum Bin Slack (MBS) was proposed by Gupta and Ho [20] and is known as a bin-focused heuristic. The packing of each bin is determined in a search procedure known as MBS One-Packing Search, which tests all possible subsets of items on the list that fit the bin capacity, keeping the subset that leaves the least free space in the bin; if the algorithm finds a subset that fills the bin, the search stops since there is no other best possible subset.

Algorithm 1 shows the recursive procedure to fill a bin presented by Fleszar and Hindi [21]; it begins with $q = 1$ and $A = A^* = \emptyset$, where $A$ is the subset currently assigned to the packing, and $A^*$ is the subset with the smaller slack $(s(A))$. Given a list $Z$ of $n'$ items not assigned to bins so far, sorted in the non-increasing order of sizes, the algorithm tests subsets of items recursively, where $q$ is the index of the item in $Z$ from which the processing is started. The procedure is executed by removing the chosen items in a bin until the list Z is empty. The complexity of this procedure is $O(2^n)$.

**Algorithm 1:** MBS One-Packing Search ($q$).

```
Input:
Output: A*
for r = q to n' do
    Let i be the r−th element ∈ Z
    if w_i ≤ s(A) then
        A = A ∪ {i}
        MBSOnePackingSearch(r + 1)
        A = A \ {i}
        if s(A*) = 0 then
            Exit
if s(A) < s(A*) then
    A* = A
```

#### 4.3.2. GGA-CGT

The Grouping Genetic Algorithm with Controlled Gene Transmission (GGA-CGT) was proposed by Quiroz-Castellanos et al. [22]. This algorithm favors the generation and evolution of high-quality solutions. It transmits the best genes in the population and maintains a balance between selective pressure and diversity.

To carry out the transmission of the best genes, it uses a set of grouping genetic operators. The evolutionary process balances through controlled reproduction for exploitation, avoiding premature convergence. Furthermore, this algorithm has been considered one of the best in the state of the art to solve BPP. The procedure is shown in Algorithm 2.

**Algorithm 2:** GGA-CGT.

```
Generate an initial population P with FF−ñ
while generation < max_gen and Size(best_solution) > L_2 do
    Select n_c individuals to cross by means of Controlled_Selection
    Apply Gene_Level_Crossover+FFD to the n_c selected individuals
    Apply Controlled_Replacement to introduce progeny
    Select n_m individuals and clone elite solutions by means of
      Controlled_Selection
    Apply Adaptive_Mutation+RP to the best n_m individuals
    Apply Controlled_Replacement to introduce clones
    Update the global_best_solution
```

4.3.3. Algorithmic Performance Indexes

To measure the algorithm's performance, we evaluated two characteristics. One measure is related to the algorithm's effectiveness and gives information about the the quality of the solution. The other measure is related to efficiency, providing information about the number of iterations/generations performed by the algorithms. To measure the solution quality, we used the $F_{BPP}$ function proposed by Falkenauer and Delchambre [23]. The $F_{BPP}$ measure provides the mean of the fill of bins found by the solution algorithm. It realizes the division of the sum of the item weights in each $bin_j$ between its capacity $(l(j)/c)$. After that, all quotients are squared and summed. Lastly, they are divided between the number of bins $m$ of the solution. Equation (10) shows this calculation.

$$F_{BPP} = \frac{\sum_{j=1}^{m}(l(j)/c)^2}{m} \tag{10}$$

## 5. Experiments and Results

From the previous analysis presented by Carmona-Arroyo et al. [1], we identified which indexes had a direct impact on the algorithm's performance. In its majority, the exploratory analysis was conducted through descriptive statistics and correlation analysis. We selected only indexes with the higher association and, with this set, we did the causal analysis through CBN.

In this section, we present the results of performing a causal analysis to learn the relationship between the features of the problem and the performance metrics considering two groups of instances (described in Section 4.1), all of them with the same indexes. The data preprocessing procedure was similar for both sets, with the causal variables being the $F_{BPP}$ measure and the number of generations for the GGA-CGT algorithm. On the other hand, for the MBS algorithm, the causal variables were the $F_{BPP}$ measure and the number of iterations.

### *5.1. Data Preprocessing*

As described in Section 3, CBN models are designed to operate with categorical variables. Because of that, it was necessary to preprocess the BPP indexes and the algorithmic performance indexes before starting the causal analysis.

The range of discretized values for the features is presented in the causal results tables, under the column labeled "Cause Value". The discretized values corresponding to the performance of the algorithms are found in the "Effect Value" column, which is displayed in all causal results tables. The following section provides a detailed description of the discretization processes undertaken.

A discretization of the values of features was carried out considering two strategies. The measures of the BPP features were discretized with the WEKA software, while the algorithmic performance measures were discretized taking into account what was considered to be a good, regular, or poor performance of the algorithms.

For the values of the BPP features, we used the unsupervised discretize algorithm included in WEKA; for this process, we configured the algorithm to obtain three groups or categories per variable and a maximum of three decimals for all.

For the values of the algorithmic performance $F_{BPP}$ measure, in the discretization, we considered three groups of values: *best*, *regular*, and *worst*. The values where $F_{BPP} = 1$ were the *best* group; a value of $F_{BPP}$ equal to one means that the algorithm found the optimal solution. Concerning values less than one, $F_{BPP} < 1$, we split them into two groups; those less than the median ($Md$), $F_{BPP} < Md$, were the *worst* group, and those equal to or greater than the median, $F_{BPP} >= Md$ were the *regular* group.

In this study, we also examined other effect variables, such as the number of generations performed by the genetic algorithm and the number of iterations executed by the MBS algorithm. To discretize the number of generations, we used the following process: instances with exactly 500 generations were classified into the *worst* group, while the re-

maining values were divided into two groups based on the median. Specifically, values ranging from the median up to 499 generations were assigned to the *regular* group, whereas values below the median, down to the minimum, were categorized as the *best* group.

The discretization process for the number of iterations required special treatment to achieve balanced categories. Due to our objective of having an equal number of groups for all variables, we decided to categorize this variable into the same three groups: *worst*, *regular*, and *best*. The *best* group consisted of values ranging from the minimum value up to the value located at the 33*rd* percentile. The *regular* group encompassed values from the 33*rd* percentile up to the 66*th* percentile. Finally, all values falling between the 66*th* percentile and the maximum value belong to the *worst* group.

### 5.2. Exploratory Analysis of the Effects

Before the causal analysis, we present the exploratory analysis to describe the effects. Table 4 shows the descriptive statistics for the $F_{BPP}$ values. First, it presents the values for the complete variable (Complete Data section in the table). The instances of $fv_l_v_u_c$ and $BPPv_u_c$ obtained similar values, especially on average with values of 0.99 and 0.98. In addition, the values for the median were equal in both sets (one for both), and the Standard Deviation (SD) is equal to 0.01 for set $fv_l_v_u_c$ and 0.02 for set $BPPv_u_c$.

The For Discretization section of Table 4 shows the statistics of the $F_{BPP}$ values different from one. We have used the values corresponding to the median to discretize this variable. However, on this occasion, the values less than the medians are within the *worst* group, and the values between the medians and one are in the *regular* group.

**Table 4.** Descriptive statistics for $F_{BPP}$ values obtained through the GGA-CGT algorithm.

| Instances | Min | Max | Mean | Median | SD |
|---|---|---|---|---|---|
| | | | **Complete Data** | | |
| $fv_l_v_u_c$ | 0.986 | 1.000 | 0.994 | 1.000 | 0.010 |
| $BPPv_u_c$ | 0.936 | 1.000 | 0.984 | 1.000 | 0.020 |
| | | | **For Discretization** | | |
| $fv_l_v_u_c$ | 0.986 | 0.990 | 0.989 | 0.990 | 0.001 |
| $BPPv_u_c$ | 0.936 | 0.984 | 0.967 | 0.977 | 0.016 |

Table 5 shows the descriptive statistics for the number of generations obtained through the GGA-CGT. In the first part (Complete Data), the table shows the statistics for the whole variable, and the second part, defined as For Discretization, shows the descriptive statistics for the variable without values equal to 500. The decision to separate the datasets was because, at 500 generations, the GGA-CGT executed the maximum number of generations without finding the optimal solution.

**Table 5.** Descriptive statistics for the number of generations executed by the GGA-CGT algorithm.

| Instances | Min | Max | Mean | Median | SD |
|---|---|---|---|---|---|
| | | | **Complete Data** | | |
| $fv_l_v_u_c$ | 0.00 | 500.00 | 266.58 | 243.00 | 227.55 |
| $BPPv_u_c$ | 0.00 | 500.00 | 277.96 | 392.00 | 228.82 |
| | | | **For Discretization** | | |
| $fv_l_v_u_c$ | 0.00 | 498.00 | 63.89 | 31.00 | 91.34 |
| $BPPv_u_c$ | 0.00 | 499.00 | 68.85 | 12.00 | 107.16 |

In the first part, we can observe that the maximum value is 500, i.e., the maximal number of generations with which we configured the algorithm. The values corresponding to mean and Standard Deviation are similar in the sets $fv_l_v_u_c$ and $BPPv_u_c$, having an average of 266.58 and 277.96 bins with a Standard Deviation of 227.55 and 228.82, respectively.

In the For Discretization part, we can observe that the average drops considerably in the two sets. Furthermore, the decrease in the number of generations used to solve the

instances is notorious. In this part, we can observe the medians from which we discretized the variable, that is, values less than or equal to 31 and 12 within the *best* group ($fv_l_v_u_c$ and $BPPv_u_c$, respectively), and the rest are between the median and less than 500 (in the *regular* group).

Another descriptive analysis was carried out for the variable $F_{BPP}$ obtained by the MBS algorithm. Table 6 shows the results for the complete variable and the discretized variable. Like the GGA-CGT, the MBS algorithm also obtained high values for $F_{BPP}$. However, the variation for the $fv_l_v_u_c$ and $BPPv_u_c$ sets is wider. Furthermore, we can observe that the mean values are high, and the medians for these two sets are not equal to one.

**Table 6.** Descriptive statistics for $F_{BPP}$ values obtained through the MBS algorithm.

| Instances | Min | Max | Mean | Median | SD |
|---|---|---|---|---|---|
| | | | **Complete Data** | | |
| $fv_l_v_u_c$ | 0.916 | 1.000 | 0.989 | 0.988 | 0.014 |
| $BPPv_u_c$ | 0.895 | 1.000 | 0.978 | 0.979 | 0.024 |
| | | | **For Discretization** | | |
| $fv_l_v_u_c$ | 0.916 | 0.989 | 0.979 | 0.986 | 0.013 |
| $BPPv_u_c$ | 0.895 | 0.982 | 0.958 | 0.961 | 0.016 |

Finally, we analyzed the number of iterations performed by the MBS algorithm. However, in contrast to the previous analyses, we diverged from using the median for discretization. This deviation arose from the need for a balanced representation of the variable, as explained in Section 5.1. In this particular case, we divided the variable into three equally sized groups, ordered from smallest to largest, to ensure a more equitable distribution (see Table 7).

**Table 7.** Descriptive statistics for the number of iterations executed by the MBS algorithm.

| | | | Complete Data | | | |
|---|---|---|---|---|---|---|
| **Instances** | **Min** | **Max** | **30%** | **60%** | **Mean** | **SD** |
| $fv_l_v_u_c$ | 200 | 404,567,999 | 4719 | 49,237.766 | 3,262,179.100 | 20,082,895.129 |
| $BPPv_u_c$ | 110 | 59,262,713 | 2690 | 69,412.687 | 1,503,355.209 | 5,719,468.579 |

In the following sections, we present the causal results for the characteristics of the instances that we selected in the previous analysis [1] and described in Section 4.2.

### *5.3. Causal Analysis of $fv_l_v_u_c$ Set Instances*

In this section, we present the results of the causal analysis of the $fv_l_v_u_c$ set instances. We start by showing the outcomes of the GGA-CGT, with the $F_{BPP}$ measure and the number of generations; both variables were considered as the effect variables. Subsequently, we showed the results of the MBS algorithm, where the $F_{BPP}$ measure and the number of iterations were employed as the effect variables.

#### 5.3.1. GGA-CGT—$F_{BPP}$

It is worth highlighting that the discretization of the $F_{BPP}$ solution quality measure led to the construction of three distinct groups. The *best* group consists of values equal to 1, while the *regular* group contains values less than 1 up to the median. The remaining values constitute the *worst* group.

Table 8 presents the featured causal results between the characteristics that define the hardness of the problem and the performance metric $F_{BPP}$ obtained by the GGA-CGT for the $fv_l_v_u_c$ instances. It is important to note that, although the causal effects for all variables were calculated, only the cause variables with CE values greater than $|0.3|$ or equal to zero are included in the table.

For the *Major_Multiplicity* feature, values between 4.66 and 8.33 are directly associated with achieving the *best* performance. Despite the low causal effect ($CE$ = 0.44), this specific range of values tends to generate this level of performance.

**Table 8.** Causal results for the $F_{BPP}$ measure of $fv_l_v_u_c$ instances with GGA-CGT.

| Cause | Cause Value | Effect Value $F_{BPP}$ | CE | Type of Cause |
|---|---|---|---|---|
| *Major_Multiplicity* | (4.66, 8.33) | *best* | 0.44 | Generative |
| *t* | (<0.409) | *best* | −0.73 | Preventive |
| *d* | (>0.133) | *best* | −0.39 | Preventive |
| *RC* | (>0.466) | *best* | −0.30 | Preventive |

The characteristic *t*, with values less than 0.409, is a preventive cause that hinders the GGA-CGT from achieving the *best* performance ($CE = -0.73$). Although the causal effects may not be high, they indicate that *d* values above 0.133 and *RC* values above 0.466 prevent the GGA-CGT from reaching the *best* performance.

#### 5.3.2. GGA-CGT—*Generations*

To recapitulate, as described in Section 5.1, the discretization of the number of generations involved splitting them into three groups: *best*, *regular*, and *worst*. Instances with exactly 500 generations were allocated to the *worst* group, while the remaining values were categorized into two groups. Values ranging from the median up to 499 generations were assigned to the *regular* group, and values below the median, extending to the minimum value, were assigned to the *best* group.

Table 9 presents the outstanding causal results between the features of the problem and the number of generations obtained by the GGA-CGT on the $fv_l_v_u_c$ instances; only variables with CE values greater than $|0.3|$ or equal to zero are included.

**Table 9.** Causal results for the efficiency measure *Generations* from $fv_l_v_u_c$ instances with the GGA-CGT.

| Cause | Cause Value | Effect Value *Generations* | CE | Type of Cause |
|---|---|---|---|---|
| *Major_Multiplicity* | (<4.66) and (>8.33) | *worst* | 0 | Not cause |
| *t* | (<0.321) | *worst* | 0.73 | Generative |
| *Pct_ñ* | (<0.355) | *best* | −0.66 | Preventive |

We identify that values of *Major_Multiplicity* less than 4.66 and greater than 8.33 do not cause the algorithm's *worst* performance. On the other hand, *t* values lower than 0.321 result in the algorithm's *worst* performance ($CE = 0.73$).

Finally, in the case of the *Pct_ñ* feature, values less than 0.355 impede the attainment of the *best* performance by the GGA-CGT ($CE = -0.66$).

#### 5.3.3. MBS Algorithm— $F_{BPP}$

As previously stated, the discretization of the $F_{BPP}$ solution quality measure involved three distinct groups: *best*, *regular*, and *worst*. The *best* group consisted of $F_{BPP}$ values equal to 1, signifying the algorithm's successful achievement of the *best* solution. Values different from 1 ($F_{BPP} < 1$) were further divided into two subgroups: the *worst* group contained values less than the median (Md), and the *regular* group included values equal to or greater than this ($F_{BPP} \geq Md$).

Concerning to the values of $F_{BPP}$ obtained by the MBS algorithm on the set of $fv_l_v_u_c$ instances, we present the causal variables with CE values greater than $|0.3|$ or equal to zero in Table 10.

**Table 10.** Causal results for $F_{BPP}$ measure of $fv_l_v_u_c$ instances with the MBS algorithm.

| Cause | Cause Value | Effect Value $F_{BPP}$ | CE | Type of Cause |
|---|---|---|---|---|
| *Major_Multiplicity* | (>8.333) | *worst* | 0 | Not cause |
| *Range* | (<333,298,245.333) | *best* | −0.54 | Preventive |
| *Pct_ñ* | (<0.352) | *best* | −0.48 | Preventive |

We identified that *Major_Multiplicity* values greater than 8.33 do not act as a cause for the *worst* performance of the algorithm, and all the causal effects associated with these values are very close to zero.

Furthermore, based on the analysis of the *Range* feature, it was observed that values less than 333,298,245.333 serve as a preventive cause, hindering the algorithm from achieving its *best* performance; the maximum causal effect observed in this case is −0.54.

Lastly, for the *Pct_ñ* feature, values less than 0.352 are identified as a preventive cause for the *best* performance, with a maximum causal effect value of −0.48.

#### 5.3.4. MBS Algorithm—*Iterations*

Let us remember the discretization of the efficiency measure *Iterations*. We identified the *best* group, which included values ranging from the minimum to the 33rd percentile. The *regular* group consisted of values ranging from the 33rd percentile to the 66th percentile. Lastly, the *worst* group contained all values falling between the 66th percentile and the maximum. Table 11 shows the featured causal results for the number of *Iterations* obtained by the MBS with $fv_l_v_u_c$ instances.

**Table 11.** Causal results for the efficiency measure *Iterations* of $fv_l_v_u_c$ instances with the MBS algorithm.

| Cause | Cause Value | Effect Value *Iterations* | CE | Type of Cause |
|---|---|---|---|---|
| *Major_Multiplicity* | (>8.33) | *best* | 0.75 | Generative |
| *t* | (<0.409) | *best* | −0.83 | Preventive |

Instances with *Major_Multiplicity* values greater than 8.33 act as a generative cause in optimizing the algorithm's performance ($CE = 0.75$), provided that *Major_Multiplicity* values below 8.33 are absent.

The causal results for the average weight of the items *t* indicates that values less than 0.409 serve as a preventive cause, impeding the algorithm from attaining its best performance and resulting in a negative *CE* of −0.83. This underscores that values of *t* less than 0.409 generate a challenging performance for MBS.

### *5.4. Causal Analysis of $BPPv_u_c$ Instances*

Concerning the $BPPv_u_c$ instances, in this section, we present the results of the causal analysis. In the first part, we show the results of the GGA-CGT with $F_{BPP}$ and the measures *Generations* as the effect variables. Secondly, we present the results of the MBS algorithm, using the measures $F_{BPP}$ and *Iterations* as the effect variables.

#### 5.4.1. GGA-CGT—$F_{BPP}$

It is important to note that the discretization of this variable involved dividing it into three groups. The *best* group comprises values equal to 1, while values different from 1 are split into two additional groups. The *regular* group includes values higher than the median, and the *worst* group contains values lower than the median, as described in Section 5.1.

The causal effect results (greater than $|0.3|$ or equal to zero) of the $F_{BPP}$ values obtained with the GGA-CGT algorithm on the set $BPPv_u_c$ of instances are shown in Table 12.

The *Major_Multiplicity* feature suggests that values surpassing 5.667 act as a generative cause for achieving the *best* performance ($CE = 0.52$).

For the *RC* feature, we observe that values higher than 0.743 act as a generative cause for the algorithm to achieve a *regular* performance for $F_{BPP}$ $(CE = 0.63)$. In addition, *RC* values less than 0.743 do not generate the *best* performance in the outcome.

The causal effect results obtained for the *Range* feature establish that values within the range of (33,324,133.333, 66,648,243.667) directly influence the GGA-CGT's worst performance, under the condition that values outside of this range are absent $(CE = 0.96)$. On the other hand, the *Range* values less than 66,648,243.667 effectively prevent the causes leading to *regular* performance $(CE = -0.81)$.

**Table 12.** Causal results for $F_{BPP}$ measure of $BPPv_u_c$ instances with the GGA-CGT algorithm.

| Cause | Cause Value | Effect Value $F_{BPP}$ | CE | Type of Cause |
|---|---|---|---|---|
| *Major_Multiplicity* | (>5.667) | *best* | 0.52 | Generative |
| *RC* | (>0.743) | *regular* | 0.63 | Generative |
| | (<0.743) | *best* | 0 | Not cause |
| *Range* | (33,324,133.333, 66,648,243.667) | *worst* | 0.96 | Generative |
| | (<66,648,243.667) | *regular* | −0.81 | Preventive |
| *Pct_ñ* | (<0.128) | *regular* | −0 .67 | Preventive |
| | (<0.128) | *best* | 0.31 | Generative |

Finally, results show that *Pct_ñ* values below 0.128 are a preventive cause of *regular* performance $(CE = -0.67)$. However, this range of values is a generative cause of the *best* performance, although the maximum causal force obtained is only 0.31.

#### 5.4.2. GGA-CGT—*Generations*

Let us recall the discretization process, where instances with exactly 500 generations were placed in the group denoting the *worst* performance. In contrast, we divided the remaining values into two groups. Values ranging from the median up to 499 generations were assigned to the group representing a *regular* performance, while values below the median, extending to the minimum value, were allocated to the group associated with the *best* performance.

Table 13 shows the most relevant causal results for the number of *Generations* performed by the GGA-CGT with the set of $BPPv_u_c$ instances.

**Table 13.** Causal results for the number of *Generations* of $BPPv_u_c$ instances with the GGA-CGT.

| Cause | Cause Value | Effect Value *Generations* | CE | Type of Cause |
|---|---|---|---|---|
| *Major_Multiplicity* | (>5.667) | *best* | 0.77 | Generative |
| *Range* | (<33,324,133.333) | *best* | 0 | Not cause |
| *RC* | (<0.743) | *worst* | 0 | Not cause |
| *Minor_Multiplicity* | (<1.33) | *best* | −0.69 | Preventive |

According Table 13, the causal effect $(0.77)$ reveals that *Major_Multiplicity* values exceeding 5.667 cause the algorithm to solve instances easily, thereby leading to the achievement of the *best* performance.

The *Range* and *RC* features do not yield significant results in terms of causal probability or causal effects. However, their analysis reveals an interesting finding: values below 33,324,133.333 for *Range* do not exert any influence on the algorithm's *best* performance $(CE = 0)$. Furthermore, values below 0.743 for *RC* do not lead to the *worst* performance of the algorithm; in fact, they even tend to prevent such a performance from occurring.

Ultimately, *Minor_Multiplicity* values below 1.33 prevent the algorithm from achieving its *best* performance $(CE = -0.69)$. Conversely, values exceeding this threshold act as a generative cause for the *best* performance.

5.4.3. MBS Algorithm—$F_{BPP}$

As mentioned in Section 5.1, we categorized the values into three groups: *best*, *regular*, and *worst*. The *best* group consisted of $F_{BPP}$ values equal to 1, indicating that the algorithm successfully found the best solution. For values less than 1 ($F_{BPP} < 1$), we divided them into two groups: the *worst* group comprised values below the median ($Md$), as long as the *regular* group included values equal to or greater than the median ($F_{BPP} \geq Md$).

The causal analysis for the $F_{BPP}$ value with this set of instances is shown in Table 14.

**Table 14.** Causal results for $F_{BPP}$ of $BPPv_u_c$ instances with the MBS algorithm.

| Cause | Cause Value | Effect Value $F_{BPP}$ | CE | Type of Cause |
|---|---|---|---|---|
| *d* | (>0.145) | *best* | 0 | Not cause |
| | (<0.145) | *best* | 0.32 | Generative |
| *Range* | (>66,648,243.667) | *best* | 0.74 | Generative |
| | (33,324,133.333, 66,648,243.667) | *worst* | 0.93 | Generative |
| *Minor_Multiplicity* | (>0.007) | *best* | 0.47 | Generative |

Values of $d$ greater than 0.145 do not lead to the accomplishment of the *best* performance ($CE = 0$), whereas values of $d$ less than 0.145 serve as a generative cause for achieving the *best* performance.

The *Range* feature reveals that values greater than 66,648,243.667 prevent the algorithm from attaining the *best* performance ($CE = 0.74$). Additionally, the presence of the range of values between 33,324,133.333 and 66,648,243.667, associated with the absence of values exceeding 66,648,243.667, generates the *worst* $F_{BPP}$ values ($CE = 0.93$).

Finally, although the causal effect obtained for the *Minor_Multiplicity* feature is not notably high, it is worth mentioning that *Minor_Multiplicity* values above 0.007 serve as a generative cause for the *best* performance.

5.4.4. MBS Algorithm—*Iterations*

For the discretization of this variable, we identified the *best* group values, which spanned from the minimum value to the value positioned at the 33rd percentile. The *regular* group contained values ranging from the 33rd percentile to the 66th percentile. Lastly, all values falling between the 66th percentile and the maximum value were categorized as part of the *worst* group.

In terms of the number of iterations performed by with the MBS algorithm (see Table 15), we found that values of the range of the weights less than 33,324,133.333 prevent the *worst* performance with a causal effect of $-0.67$; in addition, ranges between 33,324,133.333 and 66,648,243.667 also prevent this performance ($CE = -0.54$).

Lastly, the *Minor_Multiplicity* feature indicates that values exceeding 0.007 are considered to be a generative cause in attaining the *best* performance, with a $CE$ value of 0.71.

**Table 15.** Causal results for *Iterations* of $BPPv_u_c$ instances with the MBS algorithm.

| Cause | Cause Value | Effect Value *Iterations* | CE | Type of Cause |
|---|---|---|---|---|
| *Range* | (<33,324,133.333) | *worst* | −0.67 | Preventive |
| | (33,324,133.333, 66,648,243.667) | *worst* | −0.54 | Preventive |
| *Minor_Multiplicity* | (>0.007) | *best* | 0.71 | Generative |

## 6. Discussion

The previous exploratory data analysis [1] provided us with sufficient tools for the selection of the indexes that appeared to influence the performance of two high-performance

BPP algorithms. The causal analysis further substantiated this evidence by identifying that the value ranges within each algorithm exhibit higher complexity. Additionally, the study shed light on the potential causes contributing to these results based on the evaluation of the two sets of instances analyzed in this work.

The causal analysis allowed us to determine the values that facilitate optimal algorithmic performance for the two sets of instances. One of the critical BPP features that impacts the performance of both algorithms is the multiplicity in the weights of the items; the cause values for the characterization indexes related to this feature, (*Major_Multiplicity* and *Minor_Multiplicity*), revealed that instances with higher values for this property are easier to solve for the algorithms under study. On the other hand, the causal effects revealed that instances with small- and medium-sized items and with large bin capacities are especially difficult, causing the worst results in terms of the quality of the solutions for both algorithms. For each algorithm, specific values were identified for the average weight (*t*), the range of the weights of the items (*Range*), and the percentage of large items (*Ptc_ñ*) that generate de worst performance or impede the appearance of the best results. Other common hardness-revealing causes were the features related with the dispersion of the weights of the items (*d* and *RC*); however, the values of the causal effects did not show that these indexes had a significant impact on the best or worst performance of the algorithms.

Furthermore, our findings indicate that the *Major_Multiplicity* and *RC* features exhibit an increasing causal probability as their values increase, consistently demonstrating causality in the majority of the tests conducted. Additionally, we have identified the value ranges that have the potential to enhance performance.

Other features, such as *d*, *t*, *Minor_Multiplicity*, and *Ptc_ñ*, did not stand out in all tests, but they did stand out in the number of generations performed with the GGA-CGT. It has been intriguing to analyze the value ranges of these instances, always with the hope of identifying common groups that, as potential generative causes, can be determined and highlighted even when using other algorithms.

We can infer that the GGA-CGT algorithm is more stable in solving BPP when evaluating its performance on the new sets of instances, being much better than MBS in finding more optimal solutions for the test instances in the two groups, 1549 vs. 1337 of the $BPPv_u_c$ set and 5620 vs. 4985 of the $fv_l_v_u_c$ set, without additional parameter configuration and calibration.

Given the above, the task remains to analyze the causes of the difficulty in other proposed algorithms for solving BPP, for both the new difficult instances and the state-of-the-art instances, to determine if the causal values found in this work are consistent with other algorithms. It is important to note that, in the context of the BPP, we have used specific characterization indexes that have proven to be effective in measuring instance difficulty. However, it is crucial to acknowledge that these indexes are specific to the BPP instances and may not be directly applicable to other COPs. To measure difficulty in other COPs, it will be necessary to define characterization indexes that are suitable for describing the specific features of their instances. To begin the process of characterizing a new COP, we suggest reviewing previous works related to the study of this problem or problems exhibiting similar structures, as well as analyzing how the difficulty measures developed for similar problems may be transferred to study the problem of interest [5].

## 7. Conclusions

In this work, we introduced a knowledge discovery approach through the intervention of variables in Causal Bayesian Networks that assist in the analysis of the conditions under which the performance of an optimization algorithm can be caused by the features of the instances of a combinatorial optimization problem (COP). A crucial point in our proposal is the identification of possible indexes to measure the intrinsic properties that determine the difficulty of a set of problem instances, as well as the definition of metrics to evaluate the algorithmic performance. After introducing the proposed approach, we presented a case study for the causal analysis of the performance of two state-of-the-art algorithms for the one-dimensional Bin Packing Problem.

The causal analysis applied to the study of the difficulty of BPP instances provided interesting results. We were able to detect cause-and-effect values critical for understanding the effectiveness and efficiency of the algorithms. It is worth mentioning that the results obtained from the analysis may vary depending on the set of instances and the discretization methods employed. However, these insights can help identify specific values influencing the difficulty of the instances and facilitate decision-making in subsequent analysis stages associated with problem-solving.

The proposed causal approach did not require additional configurations to adapt to the characteristics of our case study. For the construction of the Bayesian network, we used the simplest model, in which all the cause variables (features) arrive at the effect (algorithm performance) without additional connections between them. However, it is possible to change the process for the construction of the network through another method of AI, integrating more connections between the variables. In the construction of the Bayesian network structure, specific modifications may be necessary and should be considered by the domain expert. This could involve the manual network design based on the expert knowledge or the use of graphical model construction algorithms to create the initial network. Once the network structure is obtained, the expert may adjust the direction of the arrows as deemed appropriate. Another important point about the set of variables (cause and effect) is that these do not require specifications for the causal analysis application. On the other hand, the discretization processes could be performed manually (from expert knowledge) or with a known discretization algorithm. The selection of the variables and values to intervene could be left to the experts too. In this work, we analyzed all the causal variables (features) to find the highest number of ranges of values that cause the performance of the algorithms.

Because the requirements for performing causal analysis through the Intervention of Variables in Causal Bayesian Networks are generalizable, we consider that this knowledge discovery approach could be applied to study the algorithmic performance of other optimization algorithms for other COPs. In addition to providing us with an interpretation of the causal effects, the proposed approach also gives us a causal probability, which we can freely interpret in causal terms, allowing us to acquire detailed knowledge about the conditions under which an instance of a COP can be easy or difficult to solve for a specific algorithm. The causal values obtained can be applied to classify instances by their degree of difficulty, distinguishing between easy and difficult instances according to the algorithms that solve them. On the other hand, with the knowledge of the cause-and-effect values, it might be possible, when a new instance of the problem occurs, to predict the behavior of a set of algorithms and choose the best one. This would make it possible to save time in solving problems, obtaining good results. Furthermore, the results shown by the algorithms could be improved by identifying the strategies that make one algorithm superior to another under certain conditions and integrating these strategies in a complementary way, for example, through hybridization.

Finally, exploring the obtained causal model within a metaheuristic framework or algorithm selection portfolio presents an engaging avenue for future research. By integrating this approach, we could potentially enhance the efficiency and effectiveness of our methodology. Additionally, for those interested in reproducing the experiments presented in this study, both the code and the characterization indexes used for the case study are available upon request by contacting the corresponding author.

**Author Contributions:** Conceptualization, J.B.V.-A., G.C.-A., M.Q.-C. and N.C.-R.; methodology, M.Q.-C.; software, J.B.V.-A. and G.C.-A.; validation, M.Q.-C. and N.C.-R.; formal analysis, M.Q.-C.; investigation, J.B.V.-A., G.C.-A., M.Q.-C. and N.C.-R.; resources, J.B.V.-A.; writing—original draft preparation, J.B.V.-A.; writing—review and editing, J.B.V.-A., G.C.-A., M.Q.-C. and N.C.-R.; visualization, J.B.V.-A. and M.Q.-C.; supervision, M.Q.-C.; project administration, M.Q.-C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The raw data supporting the conclusions of this article will be made available by the authors on request.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References


1. Carmona-Arroyo G.; Vázquez-Aguirre J.B.; Quiroz-Castellanos M. One-Dimensional Bin Packing Problem: An Experimental Study of Instances Difficulty and Algorithms Performance. In *Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 171–201.
2. Chiarandini, M.; Paquete, L.; Preuss, M.; Ridge, E. *Experiments on Metaheuristics: Methodological Overview and Open Issues*; Technical Report DMF-2007-03-003; The Danish Mathematical Society: Aarhus, Denmark, 2007.
3. Quiroz-Castellanos, M.; Cruz-Reyes, L.; Torres-Jimenez, J.; Santillán, C.G.; Huacuja, H.J.F.; Melin, P. Characterization of the optimization process. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*; Springer: Cham, Switzerland, 2014; pp. 493–507.
4. Sörensen, K.; Sevaux, M.; Glover, F. A history of metaheuristics. *arXiv* **2017**, arXiv:1704.00853.
5. Smith-Miles, K.; Lopes, L. Measuring instance difficulty for combinatorial optimization problems. *Comput. Oper. Res.* **2012**, *39*, 875–889. [CrossRef]
6. Quiroz-Castellanos, M. *Caracterización del Proceso de Optimización de Algoritmos Heurísticos Aplicados al Problema de Empacado de Objetos en Contenedores*; Instituto Tecnológico de Tijuana: Tijuana, Mexico, 2014.
7. Kotthoff, L. Algorithm Selection for Combinatorial Search Problems: A Survey. In *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach*; Springer: Cham, Switzerland, 2016; pp. 149–190.
8. Hartmann, A.K.; Weigt, M. *Phase Transitions in Combinatorial Optimization Problems (Vol. 67)*; Wiley-VCH: Weinheim, Germany, 2005.
9. Wang, Z.; O'Gorman, B.; Tran, T.; Rieffel, E.; Frank, J.; Do, M. An investigation of phase transitions in single-machine scheduling problems. In *Proceedings of the International Conference on Automated Planning and Scheduling*; ICAPS: Palo Alto, CA, USA, 2017; Volume 27, pp. 325–329.
10. Slaney, J.; Walsh, T. Backbones in optimization and approximation. In Proceedings of the IJCAI, Seattle, WA USA, 4–10 August 2001; pp. 254–259.
11. Quiroz-Castellanos, M.; Reyes, L.C.; Torres-Jiménez, J.; Santillán, C.G.; Locés, M.C.L.; Ibarra, J.E.C.; Valdez, G.C. Improving the performance of heuristic algorithms based on causal inference. In *Advances in Artificial Intelligence, Proceedings of the 10th Mexican International Conference on Artificial Intelligence, MICAI 2011, Puebla, Mexico, 26 November–4 December 2011*; Proceedings, Part I 10; Springer: Berlin/Heidelberg, Germany, 2011; pp. 137–148.
12. McGeoch, C.C. *A Guide to Experimental Algorithmics*; Cambridge University Press: Cambridge, UK, 2012.
13. Sucar, L.E. Redes Bayesianas. In *Aprendizaje Automático: Conceptos Básicos y Avanzados*; Araujo, B.S., Ed.; Pearson Prentice Hall: Madrid, Spain, 2006; pp. 77–100.
14. Pearl, J. *Causality: Models, Reasoning, and Inference*; Reino Unido, Cambridge University Press: Cambridge, UK, 2000.
15. Cruz, R.N. Building Bayesian Networks from Data: A Constraint-Based Approach. Ph.D. Thesis, Department of Psychology, University of Sheffield, Sheffield, UK, 2004.
16. Basse, S. *Computer Algorithms, Introduction to Design and Analysis*; Addison-Wesley Publishing Company: London, UK, 1998.
17. Garey, M.R.; Johnson, D.S. *Computers and Intractability (Vol. 174)*; W. H. Freeman and Company: Brisbane, CA, USA, 1979.
18. Schwerin, P.; Wäscher, G. The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *Int. Trans. Oper. Res.* **1997**, *4*, 377–389.
19. Cruz, L. Caracterización de Algoritmos Heurísticos Aplicados al Diseño de Bases de Datos Distribuidas. Ph.D. Thesis, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Mexico, 2004.
20. Gupta, J.N.; Ho, J.C. A new heuristic algorithm for the one-dimensional bin-packing problem. *Prod. Plan. Control.* **1999**, *10*, 598–603. [CrossRef]
21. Fleszar, K.; Hindi, K.S. New heuristics for one-dimensional bin-packing. *Comput. Oper. Res.* **2002**, *29*, 821–839. [CrossRef]
22. Quiroz-Castellanos, M.; Cruz-Reyes, L.; Torres-Jimenez, J.; Gómez, C.; Huacuja, H.J.F.; Alvim, A.C. A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Comput. Oper. Res.* **2015**, *55*, 52–64. [CrossRef]
23. Falkenauer, E.; Delchambre, A. A genetic algorithm for bin packing and line balancing. In Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, 12–14 May 1992; pp. 1186–1192.


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.