Research Article

A Methodology to Determine the Subset of Heuristics for Hyperheuristics through Metalearning for Solving Graph Coloring and Capacitated Vehicle Routing Problems

Lucero Ortiz-Aguilar ¹, Martín Carpio ¹, Alfonso Rojas-Domínguez ¹,
Manuel Ornelas-Rodríguez ¹, H. J. Puga-Soberanes ¹ and Jorge A. Soria-Alcaraz ²

¹Tecnológico Nacional de México, Instituto Tecnológico de León, Guanajuato, Mexico

²Department of Organizational Studies, University of Guanajuato, Guanajuato, Mexico

Correspondence should be addressed to H. J. Puga-Soberanes; pugahector@yahoo.com

Received 6 October 2020; Revised 15 March 2021; Accepted 12 April 2021; Published 26 April 2021

Academic Editor: Abd E.I.-Baset Hassanien

Copyright © 2021 Lucero Ortiz-Aguilar et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this work, we focus on the problem of selecting low-level heuristics in a hyperheuristic approach with offline learning, for the solution of instances of different problem domains. The objective is to improve the performance of the offline hyperheuristic approach, identifying equivalence classes in a set of instances of different problems and selecting the best performing heuristics in each of them. A methodology is proposed as the first step of a set of instances of all problems, and the generic characteristics of each instance and the performance of the heuristics in each one of them are considered to define the vectors of characteristics and make a grouping of classes. Metalearning with statistical tests is used to select the heuristics for each class. Finally, we used the Naive Bayes to test the set instances with k-fold cross-validation, and we compared all results statistically with the best-known values. In this research, the methodology was tested by applying it to the problems of capacitated vehicle routing (CVRP) and graph coloring (GCP). The experimental results show that the proposed methodology can improve the performance of the offline hyperheuristic approach, correctly identifying the classes of instances and applying the appropriate heuristics in each case. This is based on the statistical comparison of the results obtained with those of the state of the art of each instance.

1. Introduction

In Computer Science, a heuristic is a technique designed to solve a problem when classical methods fail to find an exact solution or when they are too slow. Currently, there is great interest from the scientific community in offering ad hoc heuristic solutions for real-world optimization problems. To achieve this, it is necessary to have a priori knowledge of the problem and, often, computationally efficient solutions are produced in a reasonable time. However, the no free lunch theorem mentions [1] that no methodology or algorithm can solve all problems; that is, ad hoc heuristics are usually not generalizable, and they do not always work well when applied to other problems even when they share some similar characteristics. This fact has led research efforts towards the development of general-purpose search methodologies

known as hyperheuristics, whose main characteristic is that they are independent of the problem domain.

Hyperheuristics can be classified according to their learning methods, such as no learning, online learning [2], and offline learning [3]. In the context of combinatorial optimization, hyperheuristics are defined as “heuristics to choose heuristics” [4], or as “an automated methodology for selecting or generation heuristics to solve computational search problems,” [4]. According to Pillay in [2], the generality of a hyperheuristic can be seen from three levels: a generalization on instances of problems, generalization for a particular problem, and a generalization focused on different types of problems, the latter being high level. Some variations of hyperheuristics depend on the type of learning used (e.g., with online learning [2] and offline learning [3]) or the nature of the heuristics.

One of the main problems in hyperheuristics is to propose methodologies that allow generating and/or selecting the minimum set of heuristics that perform well for the problem at hand and this heuristic set is usually selected by expert researchers in the field [2]. In order to automatically select the best heuristic that performs best for the problem, an approach called metalearning was proposed [5, 6] and its use in hyperheuristics can be found in Amaya et al. [7]. Likewise, the different meanings used and taxonomy for each interaction between metalearning and optimization were studied by Song et al. [8].

As there is no methodology or algorithm that can solve all problems, our objective is to base ourselves on information about the problem and the performance of the algorithms to provide this knowledge to hyperheuristics. Metalearning generates metaknowledge, and we use this to select better heuristics for solving problems. With this approach, we pretend to propose methodologies that are like the intelligence of humans. Humans could learn from problems, their characteristics, their variables, and their restrictions, and after elaborating an analysis or discernment, the “human expert” proposes the best tool and solves the problem.

In this paper, we propose a methodology to determine a subset of heuristics for hyperheuristics through metalearning and partition for solving different problems (described below) without ad hoc adjustments by providing information about the problem and the performance of the heuristics to the hyperheuristic. It is well known that the correct characterization is the key to selecting the best heuristic [6]. Consequently, this affects the hyperheuristic design, and that is why in our approach we decided to use offline learning. Metalearning consists of two basic parts, the metafeatures and the metalearner; the first are generated from the information of the problem and the solution algorithms, while the second uses a grouping technique. Our methodology extends beyond the classic metalearner approach, and we apply nonparametric statistical tests to determine which heuristics will provide the same performance if the full set of heuristics is applied.

In order to test our proposal, we used two different well-known problem domains: capacitated vehicle routing problem and graph coloring problem. The capacitated vehicle routing problem (CVRP) has different restrictions such as minimizing distance, time, capacity, and delivery. This problem aims to find the subtour of n cities, without repeating two cities on the same tour or different tours. In the CVRP state of the art, there exist different variants that consider this basic definition and extra restrictions. On the other hand, the graph coloring problem consists of labeling each vertex of a given graph with k -colors and it is a well-known problem, which has been solved by exact methods, heuristics, metaheuristics, and hyperheuristics. Although each problem can be solved by ad hoc heuristics, to date, there is no general methodology capable of solving all variants of both problems. The use of partitions for constraint satisfaction problems, such as university timetabling and VRP, has had good results [9, 10]. Although there are taxonomies of the said problems, characterizing and

classifying the instances under a hyperheuristic context with metalearning with a statistical test is an approach that has not been explored in the literature.

Finally, it is worth mentioning that to do a better design or choice of hyperheuristics and predict which is the best algorithm according to the classification of the previous instance, we propose to use metalearning with statistical analysis of the heuristics, which will allow improving these points. Our proposal also provides information that allows us to understand the performance of heuristics and hyperheuristics in the problem of interest.

The remaining content of this article contains a description of related work in Section 2, which covers a review on heuristics, hyperheuristics, and metalearning. Problem definitions and theories related to heuristics are reported in Section 3. Sections 4 and 5 present the proposed methodology. The found results and findings including performance comparison are described in Section 6. Finally, concluding remarks are presented in Section 7.

2. Related Work

In this section, we will give a view of the heuristic and hyperheuristic algorithms. Moreover, we will define some basic concepts of metalearning. Finally, we will give a discussion of the pros and cons of the presented methods.

2.1. Heuristics. We made an extensive review of different heuristics applied to CVRP and GCP. We selected a total of 11 heuristics that, after a previous experimental analysis, those heuristics apply to both problems and we list them below.

K-flip or *K-opt* heuristic was proposed by Lin and Kernighan in [11] for the travel salesman problem (TSP). This heuristic was based on the general interchange transformation, i.e., a city must change its position with another city on the same tour. Besides, this heuristic is one of the most popular for TSP [12]; it has been applied in other problems such as planar graphs, unconstrained binary quadratic programming, and the study of its complexity in SAT and MAX-SAT. The two-point perturbation is a case of *k-flip*, and we give a detailed description and algorithm of these heuristics in the following sections.

The *k-swap* heuristic is similar and frequently confused with *k-flip*. The *K-swap* heuristic improves its performance as a perturbation move when it uses two or three movements [13].

The move to less conflict heuristic, also known as minimizing conflicts, was proposed by Minton et al. [14]. The minimizing conflict heuristic has been applied to different areas in Computer Science such as hyperheuristics, graph coloring problems, pickup-and-delivery problems, and scheduling problems. The move to less conflict heuristic is a variant of the first fit, and the only difference is that the first one takes a random variable and changes its value for another that generates the least cost.

The *first-fit* heuristic was proved by Baker [15] for the bin-packing problem. On the other hand, in recent decades,

this heuristic was applied to best-known problems such as bin packing, virtual machine relocation problem, and cutting stock. A remarkable variety of heuristics is *worst fit* which was studied by Baker [15] and Csirik [16], in particular, its application to bin-packing problem.

Soria-Alcaraz et al. [17] proposed three heuristics for university course timetabling best single perturbation (BSP), static-dynamic perturbation (SDP), and double dynamic perturbation as part of the pool low-level heuristics for hyperheuristic. Moreover, these heuristics were applied to the VRP in later research [18].

2.2. Hyperheuristics. We focused on offline learning hyperheuristics selection with perturbation heuristics, whose aim is to gather knowledge in the form of rules or programs, from training set instances. Usually, the offline selection hyperheuristics belong to machine learning methods, which are trained to create a tuned methodology for a problem domain [3]. Yates and Keedwell [19] demonstrated that subsequences of heuristics were found in the offline learning database that is effective for some problem domains. They used the Elman network to compute sequences of heuristics which were evaluated on unseen *HyFlex* example problems, and the results obtained are capable of intradomain learning and generalization with 99% confidence.

One of the crucial issues in hyperheuristics design is the quality and size of the heuristic pool [20]. Soria-Alcaraz et al. [20] proposed a methodology using nonparametric statistics and fitness landscape measurements for hyperheuristics design. This methodology was tested on course timetabling and vehicle routing problems; their hyperheuristic proposal had a compact heuristic pool and competed with some traditional methods in course timetabling. In the course timetabling problem, they obtained five best-known solutions of 24 PATAT instances [21]. Finally, a recent report by Amaya et al. [7] documented a model for creating selection hyperheuristics with constructive heuristics. The effectiveness of the model proposed by Amaya depends on the delta's values used, which is useful with higher deltas.

2.3. Metalearning. The importance of metalearning, machine learning, and optimization has been studied by Song et al. [8]. The metalearning aim may concern accumulating and adapting experiences on the performance of multiple applications of a learning system. The metalearning field is also known as “*learning to learn*” [22] and it brings systems that can help by searching patterns across different tasks to control the process of exploiting cumulative expertise. The metalearning concept has been present in the field of heuristics and metaheuristics for TSP [23], the quadratic assignment problem, and hyperheuristics.

On the other hand, Gutierrez-Rodríguez et al. [23] used VRP with time windows and proposed a methodology based on metalearning to select the best metaheuristic for each instance. Besides, their proposal shared and exploited an offline scheme for the instant solutions of academics and industry. Their main contributions were to propose a set of features for

characterizing VRPTW instances and design a classification process that predicts the most suitable metaheuristic for each instance. Nevertheless, they assumed that the solutions of the set instances could be stored, shared, and exploited in an offline scheme for predicting good solvers for new unseen instances.

The aim of this paper is not to present a survey on heuristics or hyperheuristics; our proposal is slightly different. Our proposal considers some vital aspects of the research, including the ones from Yates and Keedwell [19]. We took the offline hyperheuristic approach from Soria-Alcaraz et al. [20] and the statistical approach to selecting a pool heuristic from Kanda et al. [5]. The offline hyperheuristic approach is an effective and popular method in the machine learning area [8]. On the other hand, the statistical approach to selecting a pool heuristic is a useful and reliable method because it takes statistical information from the input data.

3. Combinatorial Problems

Our methodology is a general approach to competitive performance across several classes of problems. Thus, we used two problem domains: graph coloring and vehicle routing problems. In the following sections, we will review the formal definition of each of these problems as well as their benchmark instances.

3.1. Graph Coloring Problem. The graph coloring problem demonstration as an NP-hard problem was proposed by Karp [24]. According to [25], a formal vertex-coloring problem of a graph $G = (V, E)$ is a function $c: V \rightarrow \mathbb{N}$, in which any two incident vertices $u, v \in V$ are assigned different colors, that is, $\{u, v\} \in E \Rightarrow c(u) \neq c(v)$, and E is a finite set of unordered pairs of vertices named *edges*, where the function c is the coloring function and a graph G for which there exists a vertex-coloring which requires k colors is called *k-colorable*. The coloring function induces a partition of the graph G into independent subsets $V_1, V_2, \dots, V_k$, where $V_i \cap V_j = \emptyset$ and $V_1 \cup V_2 \cup \dots \cup V_k = V$. The benchmark instances can be found in <http://mat.gsia.cmu.edu/COLOR/instances.html>. The above lets the partitioning methodology work on the input design and it is possible to avoid the ad hoc modifications to the heuristics since it will only pass a different objective function that adequately evaluates the instances of this problem.

3.2. Capacitated Vehicle Routing Problem. The capacitated vehicle routing problem (CVRP) is a variant of VRP [26]. In this problem, we have an undirected graph G , m vehicles, Q capacity, and a set of cities $C = \{c_0, c_1, \dots, c_n\}$. Formally, the city c_0 is the *depot* and each vehicle must visit these cities starting from the *depot* and coming back to this. Alba and Dorronsoro [27] define a distance or travel time matrix $D = (d_{ij})$ between cities c_i and c_j . Each city c_i has a demand of things q_i . We denote it as a route $R = \{\vec{r}_0, \vec{r}_1, \dots, \vec{r}_m\}$, and $\vec{r}_i$ is a permutation of the cities, starting and finishing at the depot c_0 . For each route, $\vec{r}_i \cap \vec{r}_j = c_0$ with $i \neq j$. The cost of a problem solution is the sum of the costs of each route of R as

$$F(R) = \sum_{i=1}^m \text{Cost}(\vec{r}_i), \quad (1)$$

$$\text{Cost}(\vec{r}_i) = \sum_{j=0}^k d_{j,j+1}, \quad (2)$$

where k is the total of vehicles. This problem aims to determine for each vehicle the lowest cost (see equations (1) and (2)) tour or distance or travel time, considering the max capacity. Not a bene the hard constraints are the capacity of each vehicle and two vehicles cannot visit the same city. The CVRP has several constraints and a specific formal definition of this problem. These two characteristics let us apply our methodology with a design cities partition, where each vehicle is related to one part. As heuristics work with solutions that are already complete and respect important restrictions such as capacity, if any movement violates or exceeds this capacity, that solution is penalized in the objective function.

4. Methodology

For our methodology, it is important to know since the constraint modeling phase of the problem can be solved by partitions. The API-Carpio methodology and the methodology proposed by Soria-Alcaraz et al. [28] let us transform the instances of the problem with their restrictions, into inputs to apply the proposed methodology. The MMA matrix that is generated by applying the API-Carpio methodology lets us visualize the hard restrictions of the problem and evaluate the costs of visiting cities or nodes. For the soft constraints of the problem, the methodology proposed by Soria-Alcaraz et al. [28] is to be considered in the list of restrictions.

In this section, we describe the methodology to model the input data problem information used for the experimentation for two combinatorial problems based on the API-Carpio methodology. We integrated the API-Carpio methodology [29] and the methodology of the design proposed by Soria-Alcaraz et al. [28].

4.1. API-Carpio Methodology. This methodology is used to solve the university course timetabling problem and it considers three factors: students, teachers, and institutions (infrastructure). The methodology uses several structures for the equations previously described. One of the most important structures of this work is MMA. This matrix is constructed with information on the cities or nodes. For graph coloring, we use the information of the adjacency matrix, while for CVRP it is considered the cost matrix. Table 1 shows an example of an MMA matrix. The algorithm to construct this matrix is given in [29].

4.2. Methodology of Design. This methodology was extended from the proposal by Carpio [29] and their formal definition was proposed by Soria-Alcaraz et al. [28]. The methodology of design by Soria allows us to consider the objectives of course timetabling and to satisfy the different restrictions, by

TABLE 1: MMA matrix.

City/node	N_1	N_2	...	N_{n-1}	N_n
N_1	10	99	...	97	137
N_2	99	101	...	80	96
...	...	...	...	...	...
N_{n-1}	97	80	...	35	10
N_n	137	96	...	10	68

converting these to lists of time and space restrictions it is seeking to minimize student conflict.

To use this methodology, two structures are used to consider restrictions and variables: MMA matrix and LPH. The LPH has information about the possible restrictions that can be assigned to each node or city. An example of this list can be found in Table 2. The list shows in each row the number part, i.e., the node N_2 can be assigned in parts 1, 2, 4, 5, 7, or 8, but not in 6. The algorithm for generating artificial instances of LPH can be found in Ortiz-Aguilar [30].

5. Metalearning for Selecting a Subset of Heuristics for Hyperheuristics

According to Brazdil et al. [22], the approach of metalearning (ML) is to help the selection of an algorithm for a set of instances with metadata. According to Alpaydin [31], the metalearning aim is to find the best classifier for a set of data and to find the best classifier for the characterization when the data are considered. In our proposal, the data are associated with the problem instances of different problems, and the classifier is associated with the set of heuristics. Our objective is to be able to select the best set of heuristics for a hyperheuristic in a set of instances. In Figure 1, we show a diagram of the metalearning processes to obtain meta-knowledge for the selection of heuristics (diagram modified from Brazdil and Giraud-Carrier [32] to our methodology).

In this work, we use metalearning to select a set of heuristics for hyperheuristics in a dataset. We named the set of characterized instances of the two problem domains as the “metacharacteristics” and the model that maps each instance to the corresponding group of heuristics for hyperheuristics the “metalearner.” In this case, the metalearner selected is the *K-means* algorithm. The methodology proposed in this article consists of 5 steps in the metalearning stage:

- (i) Step 1: obtain the set of instances to be worked on. In this case, we have as a criterion to select those instances that are susceptible to being resolved by partitions.
- (ii) Step 2: evaluation and extraction of characteristics of the instances. In this step, the characteristics of the heuristics and the instances are generated. Heuristics that apply to both problems are selected, this task becomes simple with the use of the partitioning methodology, and this is because it allows working always with generic inputs where the variables and restrictions are modeled. Later, heuristics work with these generic inputs and solutions that only have a fitness function corresponding to their problem (where the objectives are evaluated).

TABLE 2: LPH list.

Node or city	Restrictions							
N_1	1	2	3	4	5	6	7	8
N_2	1	2	3	4	5		7	8
...	...	...	...	...	...	...	...	...
N_{n-1}	1	2	3	4	5	6	7	8
N_n	1	2	3	4	5	6	7	8

$N_1 \dots N_n$ represents the node or cities.

FIGURE 1: Metalearning to obtain metaknowledge for heuristic selection.

- (iii) Step 3: generation of metacharacteristics. Based on the characteristics of each problem and the performance of the heuristics applied to all instances, we generate vectors of characteristics that will be our metadata.
- (iv) Step 4: metalearning and the recommended model of heuristics. In the state of the art, research is limited to applying only a clustering technique for the recommendation of the algorithm model. We propose to incorporate a statistical analysis together with the clustering algorithm to improve the design of the basic subset of heuristics.

5.1. Problem Definition of Metalearning for Heuristic Selection.

Consider a problem P_i that belongs to the problem set (GCP). Let $H_i = \{h_{i,1}, h_{i,2}, \dots, h_{i,n_i}\}$ be a subset of low-level heuristics, which are used in the state of the art, to solve the problem P_i . We denote as RS_{H_i} a random selection of heuristics H_i , to be applied in the solution of P_i , where $|RS_{H_i}| = |H_i|$, with RS_{H_i} to a reduced set of heuristics, that is, $RS_{H_i} \subseteq H_i$.

Let $V_0 = \{\lambda\}$, where λ represents an empty string. We defined recursively $V_{n+1} = \{wh_{i,m} | w \in V_n; h_{i,m} \in H_i\}$ where $n \geq 0$ and $m \in \{1, 2, 3, \dots, n_i\}$. Then V_n represents the set of all strings of length $n \in \mathbb{Z}^+ = \{0, 1, 2, 3, \dots\}$, formed from the symbols in H_i . So Kleene's closure from H_i is [33]

$$H_i^* = \bigcup_{n \in \mathbb{Z}^+} V_n = V_0 \cup V_1 \cup V_2 \cup V_3 \cup \dots \quad (3)$$

When V_0 is omitted at the junction, we get the Kleene Plus closure H_i^+ :

$$H_i^+ = \bigcup_{n \in \mathbb{N}} V_n = V_1 \cup V_2 \cup V_3 \cup \dots \quad (4)$$

In other words, H_i^+ is the collection of all possible nonempty strings of finite length generated from the symbols in H_i .

Let HH be a heuristic selection hyperheuristic with offline training, where the training considers the set H_i . After training, the HH provides a methodology $M_{H_i} \in H_i^+$, with the best order of application of low-level heuristics, which we will denote by $BM_{H_i} \in H_i^+$, in the solution of the problem P_i , which improves the performance of the application of a RS_{H_i} , where $|BOS_{H_i}| \geq |H_i|$ [17].

We take two problems P_i and P_j that belong to the problem set (GCP) that comply with level 3 of generality proposed by Pillay [2] and are susceptible to being solved by partitions. Let $H_i = \{h_{i,1}, h_{i,2}, \dots, h_{i,n_i}\}$ and $H_j = \{h_{j,1}, h_{j,2}, \dots, h_{j,n_j}\}$ be subsets of low-level heuristics, which are used in the state of the art, to solve the problems P_i and P_j , and $UH = \{H_i \cup H_j\}$. We denote as RS_{UH} a random selection of the heuristics UH , to be applied in the solution of P_i and P_j , where $|RS_{UH}| = |UH|$, with RS_{UH} to a reduced set of heuristics, that is, $RS_{UH} \subseteq UH$.

Heuristic selection hyperheuristic is denoted as HH , with offline training, where the training considers the UH set. After training, the HH provides a methodology $M_{UH} \in UH^+$, with the best order of application of low-level heuristics, which we will denote by $BM_{UH} \in UH^+$ in solving the problems P_i and P_j which improve the performance of the application of a simple RS_{UH} , where $|BM_{UH}| \geq |UH^+|$.

Our objective is to propose a methodology that provides the HH , with a reduced subset $RS_{UH} \subseteq UH$ for its training, such that the HH provides a methodology $M_{RS_{UH}} \in RS_{UH}^+$, with the best order of application of the reduced set of heuristics RS_{UH} , which we will denote by $BM_{RS_{UH}} \in RS_{UH}^+$, in solving the problems P_i and P_j respectively, which equal the performance than with the application of the methodology $M_{UH} \in UH^+$ where $|BM_{RS_{UH}}| \geq |RS_{UH}|$.

To solve the problem, the independent application of each of the heuristics of H_i and H_j was proposed, measuring their performance in solving the problems P_i and P_j . Apply statistical tests to contrast the performance of the independent heuristics and thereby discriminate from each set H_i and H_j those heuristics that obtained the lowest performance.

Next, we will focus on describing the stage of extracting characteristics from heuristics and instances. Our methodology improves the metalearning stage (step 4) with the application of nonparametric statistical tests to determine which heuristics are the ones that will provide the same performance if the full set of heuristics will be applied. This means that, if there are heuristics that are redundant, it is possible to leave them out and consider only those that enhance the speed of the search for solutions to the problem. The metalearning process proposed in this work to select the pool of heuristics includes the following steps:

- (1) The problems will be the source of information for the basic features.
- (2) Given a set of instances denoted as I , for each of the instances I_i apply a number k of times the heuristic H_j . The results will be the inner features. For the CVRP, a greedy heuristic is used, which will allow us to build feasible solutions to the problem. For graph coloring, we will initialize with a random construction heuristic. The next step is to apply the heuristics. It is possible after this step that the instances can be solved to the best solution due to their complexity. This means that it is possible to avoid executing a complete and expensive computation process when solving problems with the application of a simple heuristic.
- (3) With the information obtained from points 1 and 2, feature vectors will be formed that will be our metadata.
- (4) For a better treatment of the metadata from the previous step, the following steps are carried out:
 - (a) The patterns that will be used in the k-means will be scaled with the following formula [34]:

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (5)$$

where $x \in \mathbb{R}$ are the values of the original variables (features).

- (b) Generate the pattern based on the *inner* and *basic features* per instance. The *basic features* will be the problem information and the *inner features* will be the fitness obtained by the heuristics applied k times to the problem.
- (5) Pass the feature vectors to the clustering algorithm to form classes. According to Brazdil and Giraud-Carrier [32], the k-means is a simple learning method, which we apply to carry out the grouping of instances in classes. To determine which are the number of classes to form, we use Sturges rule, since, with data that are the potency of 2, it is approximated in a good way. Determine the subclasses with Sturges rule [35] with

$$c = 1 + \frac{\log(T)}{\log(2)}, \quad (6)$$

where T is the total amount of data.

As the distance metric for the K-means, we use the Mahalanobis distance; this distance has properties such as being invariant to scale by nonsingular linear transformations. An in-depth study of different metrics [36, 37] will be a specific job to investigate whether it can improve the performance of the proposed methodology.

- (6) Label each pattern according to the group number in which each pattern (instance) was classified.
- (7) Apply again the three statistical tests to the results of heuristics per problem, according to the formulas in [38]. The test ranks 1 to the best performing heuristic, 2 to the second-best, and n to the worst-performing heuristic. From these tests, we will take the range of the heuristics and the range will now be considered as inner features. With this information and the class label, they will now form patterns.
- (8) Determine a cutoff point for each class based on the range, and in this case, it will be the average of the minimum and maximum range. Choose those heuristics that pass the cutoff criterion to be part of the minimum set.
- (9) The output will be the minimum set of heuristics per class.

This process is shown in a specific way in Figure 2. The two important aspects of metalearning in our work are heuristics and metacharacteristics, which are low-level heuristics and metafeatures.

5.2. Low-Level Heuristics. An important part of the hyperheuristic approach is the selection of the heuristic set. This article proposes to extract information from heuristics and problems to generate the metafeatures [32]. This lets us improve the design and testing of the hyperheuristic algorithm. The goal in this stage is to generate metafeatures in which the heuristics can have a better performance individually for all problem instances. This improves the next part in which the hyperheuristic must choose the sequence application for each heuristic and it uses a minimal pool of heuristics which is a fundamental part of it [2]. For all instances, it will be applied k times for each heuristic. The heuristics were applied to the two problems and their respective instances were as follows:

- (1) *K-Flip/Simple Random Perturbation (SRP) (H_1)*. The heuristic changes the value of one or more variables (in some cases k) to another feasible value. The GCP aim is to change the color of a certain node to another [39]. Finally, for CVRP, the movement implies changing a city to another specific vehicle [12].
- (2) *K-Swap/Kempe Chain Neighborhood/S-Chain Neighborhood (H_2)*. It must be selected two or more varieties and then interchange their values among them when possible; otherwise, the change is not made. We exchange the color between nodes previously selected by GCP. This heuristic is using in

FIGURE 2: Metalearning process for the heuristics set selection.

works related to TSP or CVRP, also called k-interchange [40, 41].

- (3) *Best Single Perturbation (BSP)* (H_3). This heuristic chooses a variable according to the list of hard restrictions (LPH) and changes its value. This exchange produces a better cost or in the worst case, the same cost [17]. Next time this heuristic is going to apply, the next variable will be chosen according to the next position of the last variable which was modified. The next node which must change color will be selected according to the last variable chosen for the graph coloring problem. The CVRP must be changing the city of the vehicle to another vehicle.
- (4) *Static-Dynamic Perturbation (SDP)* (H_4). It is also known as statically dynamic perturbation (SDP). It is based on the variable selection with a probability distribution of the frequency in the last k iterations. This heuristic chooses a variable and changes its value randomly [17]. The variables with fewer changes will have a higher probability to be selected. Applied to GCP, it would be a node with fewer color changes, and for the CVRP, the city has moved a few times to another vehicle.
- (5) *Two Points Perturbation (2pp)* (H_5). It is also known as k -opt, and it is a particular case of the K-swap with a value of $k = 2$.
- (6) *Double Dynamic Perturbation (DDP)* (H_6). This heuristic is based on the SDP, this receives a solution, and it modifies the value of a variable concerning a probability distribution. The difference is that a copy of the initial solution is kept and, in the end, the best of the two solutions is returned [17].
- (7) *Move to Less Conflict (MLC)* (H_7). This selects a random variable, and it assigns to a part of the value which generates the least cost [18]. In GCP, the color changes according to another which improves the fitness, and in CVRP, the city is moved to another vehicle where the total distance of the route is minimized.
- (8) *Min-Conflicts* (H_8). The heuristic selects a random variable, and it assigns to a part which generates the cheapest cost [18]. In GCP, the heuristic must change color from the selected node to another, which improves the result. For CVRP, the selected city with a lower cost in which it must minimize the total distance of the route.

- (9) *First-Fit* (H_9). It changes the value of a variable to another, which is the least repeated in other variables [18], i.e., in CVRP, the heuristic will take a city and it will change it to the vehicle that has fewer cities in its route. For the GCP, it will select a node and it will assign the color that is least repeated.
- (10) *Worse Fit* (H_{10}). It assigns the most repeated value if possible, without violating the hard constraints on a randomly selected variable [42]. For GCP and CVRP, we assign a node or city to the most repeated timeslot, color, or vehicle.
- (11) *Burke-Abdullah (BA)* (H_{11}). This heuristic was proposed by Abdullah et al. [43], in which it chooses a variable applying Fail-First or Brelaz Heuristic [44] and its value changes according to the one that has obtained better performance by applying the following algorithms: minimum conflict, random selection, sequential selection, and least constrained.

5.3. Metafeatures. The description and generation of characteristics permit differentiation into at least two groups of instances within the same problem class. We used the terms of basic feature and *inter feature* based on the proposal conducted by Gutierrez-Rodríguez et al. [23]. As basic features, these are given by the problem, e.g., the number of nodes, colors, vehicles, and so on, depending on each problem information. For both classes of problems, the number of different *basic features* is summarized in Table 3.

The fitness performance values of all heuristics are the inner feature key. Finally, the pattern per instance is *basic feature + inner feature*. The final pattern is shown in Table 4. For example, instance 1 has a pattern (3, 3, 8, 50, 3, 2, 1, 4), and the number of *inter feature* is according to the pool heuristics (eight features for the given example).

6. Methodology for Determining a Subset of Heuristics

In this section, we propose a new approach for selecting and determining a subset of heuristics to solve GCP and CVRP instances. We describe our methodology in the next steps and the graphical representation of our methodology is shown in Figure 3.

- (1) *Variables and Problem Restrictions Identification.* First, the variables and restrictions of the problem are identified according to the problem aims or

TABLE 3: Basic features for CVRP and graph coloring.

Problem	Min partition	Max partition	Variables	Edges
VRP	Vehicles	Vehicles	Cities	Cities connections
Graph coloring	Lower bound	Upper bound	Nodes	Edges

TABLE 4: Characteristics for instances.

Instance	Lower bound	Upper bound	Nodes	Edges	H_0	H_1	H_2	H_3
1	3	3	8	50	3	2	1	4
2	5	10	6	36	4	3	1.5	1.5
3	3	8	25	80	3	1	2	4
4	10	10	50	80	2	4	3	1

objectives. To model the GCP, the values in the MMA matrix represent the weights of the edges of nodes. If there is a zero in a certain position (x, y) in the matrix, this represents no connection between those nodes. For graph coloring, each node is colored considering that the adjacent nodes do not have the same color. CVRP is aligned with our methodology due to its aim seeking to get subroutes in which the tour cost (subgroup) must be the minimum or the cheapest.

- (2) *Problem's Restriction Modeling.* In both problems, we must design a partition of nodes or cities. First, it is necessary to model the restrictions for each variable in an LPH, e.g., a node cannot be colored by a specific color or a restricted city for a tour. Then, it must design the MMA which represents the edge or connection weight between nodes or cities. In GCP, the adjacency matrix corresponds to MMA, and in CVRP, the MMA matrix will be the matrix that has the distances of the node to node. For GC, our LPH is constructed based on the number of colors in which the nodes can be labeled. In case the problem restricts colors to five, the list will be like the one shown in Table 2. Similarly, this list will be built for the CVRP, where the number of vehicles is the number of *parts* that should be represented on the list (see Table 2). For the problems used in this work, it was not necessary to elaborate additional structures for soft restrictions. Besides, for an extensive review and how to model additional restrictions, the research proposed by Ortiz [10] details all possible cases and different features.
- (3) Apply the metalearning process described in Section 5.1
- (4) Separate the patterns (step 6) into training and test sets to proceed to the classification phase. It is important to consider at least one pattern of each class in the test set.
- (5) Use the classifier on the training set to make necessary adjustments to it. After describing and getting all pattern characteristics per instance, the next step is training and testing all instances by a classifier. For our approach, we prefer to use a simple classifier as

Bayesian because our objective was not to compare the performance between classification algorithms or to design ad hoc classifiers for our research. The NBC simplifies learning by assuming per class that all features are independent [45]. In our methodology, we assume that each heuristic performance is independent because we applied each heuristic to independent experiments. In the previous stage, each experiment must be run with only one heuristic, and thus, we did not apply two or more heuristics at a time. Finally, all features in the created dataset instances were normalized before applying the classifier.

- (6) Finally, the set of test instances will use the classifier to assign a “class” and solve it with its corresponding set of heuristics.

6.1. Designing and Testing the Hyperheuristic Offline Learning with K-Folds. To choose the minimal set of heuristics and design the hyperheuristic for each class in more detail, our methodology considered the hyperheuristics with offline training as it has demonstrated good results for constraint satisfaction problems in terms of generality solution [3]. A random constructive heuristic was used to generate solutions to our problem of GCP, and for CVRP, a greedy algorithm was used. A selection hyperheuristic algorithm has three components: the pool of operators (low-level heuristics), a high-level search strategy, and a control mechanism to select the operator, which will be applied at each search step.

6.1.1. High-Level Search Strategy. The iterated local search algorithm was used as a high-level search strategy. This metaheuristic was proposed by Lourenço et al. [46] and it is constructing a sequence of solutions generated by an embedded heuristic. The generated solutions could be better if they were only constructed randomly. The essence of this algorithm is to intensify an initial solution, exploring neighboring solutions to it. The algorithm is shown in Algorithm 1, which was taken from El-Ghazali [47]. In the field of hyperheuristics with offline learning, it refers to the fact that the high-level search strategy searches for a methodology (a sequence of heuristics) that solves a set of instances and then applies it to a given set of instances, in contrast to online learning, which refers to the construction of a given sequence of heuristics as the instances are presented.

6.1.2. Selection Operator. In the perturbation phase (step 4 in Algorithm 2), it is necessary to choose a variable following a probability distribution based on the frequency of variable selection in the last k iterations. This simple heuristic allows

FIGURE 3: Methodology to determine the subset of heuristics for hyperheuristics.

```

(1)  $s_0 = \text{GenerateInitialSolution}$ 
(2)  $s^* = \text{ImprovementStage}(s_0)$ 
(3) while ! $\text{StopCondition}()$  do
(4)    $s' = \text{SimpleRandomPerturbation}(s^*)$ 
(5)    $s^{**} = \text{ImprovementStage}(s')$ 
(6)   if  $f(s^{**}) < f(s^*)$  then
(7)      $s^* = s^{**}$ 
(8)   end if
(9) end while
(10) return  $s^*$ 

```

ALGORITHM 1: High-level iterated local search (ILS).

us to modify the methodology solution. We used the same hyperheuristic framework, and according to each class, we gave a different pool of low-level heuristics. For example, if for class 1 the best heuristics were H_1 , H_8 , H_{10} , and H_5 , these were our pool for the hyperheuristic. With a similar procedure, the design was done for each class. After this process, we trained each hyperheuristic on their respective classes for the next stage.

7. Experimental Results

This section describes our experiments in detail for graph coloring and CVRP benchmarks used in this paper. We give the configuration for the implementation of the iterated local search hyperheuristic. Finally, we described the statistical tests that we used to compare our results with the experimental methodology.

Our approach was implemented in JAVA language with JDK 1.8 using the IDE NetBeans IDE 8.2. The experiments were executed on a computer with processor Intel i7-7700U, 2.6 GHz, 16 GB DDR3 RAM, and operating system Windows 10 Home. The tests presented in this work were executed in a common notebook, with a single processor; it is showing the effectiveness of the exposed methodology.

For each heuristic, a limit of 100,000 function calls was given in each test run for all instances. We applied the

Shapiro–Wilks test to check if the data results were normal or not, hence choosing a better representative (average or median). If the data behavior is according to a normal distribution, the average was taken as representative and otherwise the median.

7.1. Heuristics Results for Graph Coloring and CVRP

7.1.1. Graph Coloring. We used the benchmark proposed for the second DIMACS challenge on graph coloring [48] and this is tested with 41 runs. In Tables 5 and 6, we show our results. We denote the best results with a bold face, and only the *myciel2* instance was solved with the application of the individual heuristics in their optimum.

We applied a nonparametric test to verify that there are differences between the performance of each heuristic. Table 7 shows the ranges obtained in the three statistical tests for graph coloring instances. The three omnibus tests indicated there are significant differences between the heuristics. The heuristics which obtained the highest ranks in the three tests were H_6 and H_{11} , i.e., these have the worst performance. H_3 and H_8 were the best results for the problems, and this is because in all omnibus tests those heuristics obtained the lowest ranks (see Table 7 marked in bold). The time of each run is reported in Table 8 in.

```

(1)  $ls \leftarrow \text{IncumbentSolution}$ 
(2) while!  $\text{LocalStopCriteria}()$  do
(3)    $hi = \text{Perturbate}()$ 
(4)    $ls^* = \text{apply}(hi, ls)$ 
(5)   if  $f(ls^*) < f(ls)$  then
(6)      $ls = ls^*$ 
(7)   end if
(8) end while
(9) return  $ls$ 

```

ALGORITHM 2: Improvement stage.

TABLE 5: Heuristics results for graph coloring instances.

#	Name	Nodes	Colors	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
1	1-Insertions_4	67	3	12	18	11	15	10	66	61	13	25	28	75
2	2-Insertions_3	37	3	4	7	4	5	4	24	15	4	11	10	29
3	2-Insertions_4	149	3	33	41	16	32	32	148	158	29	45	47	155
4	3-Insertions_3	56	3	5	9	4	4	5	33	24	4	9	9	40
5	4-Insertions_3	79	2	14	19	16	13	11	61	66	13	25	22	71
6	anna	138	10	17	17	14	17	16	59	45	13	35	30	68
7	asH_3 31GPIA	662	3	99	76	85	96	82	1067	724	89	164	156	1082
8	asH_6 08GPIA	1216	3	1497	1448	729	1541	1370	2003	1120	1083	1513	1485	2066
9	asH_9 58GPIA	1916	3	2618	2553	1900	2621	2462	3161	1999	2039	2616	2624	3206
10	david	87	10	16	18	14	16	16	49	48	15	35	37	58
11	DSJC1000.1	1000	19	1847	1801	610	1837	1781	2565	1019	942	1789	1821	2634
12	DSJC1000.5	1000	82	2432	2513	838	2425	2379	3130	1126	1230	2327	2327	3234
13	DSJC1000.9	1000	214	1856	2073	828	1850	1824	2354	1204	1062	2375	1768	2414
14	DSJC125.1	125	4	48	59	36	48	54	161	97	37	56	66	173
15	DSJC125.5	125	15	113	156	72	114	124	275	133	68	250	112	298
16	DSJC125.9	125	42	125	173	77	124	131	210	132	72	208	112	221
17	DSJC250.5	250	25	350	415	146	349	368	658	278	157	339	326	681
18	DSJC250.9	250	70	311	388	152	315	319	477	234	153	487	294	485
19	DSJC500.1	500	11	629	646	197	634	598	1085	532	279	623	614	1114
20	DSJC500.5	500	47	927	1007	290	911	905	1381	471	368	859	853	1412
21	DSJC500.9	500	122	771	927	326	759	763	1044	487	382	1051	731	1129
22	DSJR500.1	500	11	124	130	33	125	119	334	128	46	129	127	343
23	DSJR500.1c	500	84	1257	1439	203	1262	1259	1566	1162	475	1539	1210	1589
24	DSJR500.5	500	121	376	425	201	372	369	619	299	211	627	352	662
25	fpsol2.i.1	496	64	141	183	82	139	122	263	169	91	255	262	267
26	fpsol2.i.2	451	29	52	81	59	51	52	337	281	52	204	215	348
27	fpsol2.i.3	425	29	51	82	55	51	54	364	272	54	211	232	368
28	games120	120	8	16	24	11	17	17	82	34	10	23	23	88
29	homer	561	12	23	22	22	24	23	165	139	24	53	44	176
30	huck	74	10	13	13	12	13	13	46	38	12	27	24	47
31	inithx.i.1	864	53	78	95	81	80	80	423	320	115	264	263	444
32	inithx.i.2	645	30	55	100	62	56	59	525	459	85	350	324	531
33	inithx.i.3	621	30	59	108	65	56	63	510	462	78	338	331	539
34	jean	80	9	13	12	13	14	12	47	31	13	27	26	46
35	le450_15a	450	14	292	293	103	291	291	580	410	137	292	289	597
36	le450_15b	450	14	289	297	106	290	285	589	427	125	300	288	624
37	le450_15c	450	14	716	762	358	713	713	1187	887	424	734	707	1179
38	le450_15d	450	14	723	751	371	718	733	1163	799	418	728	725	1197
39	le450_25a	450	24	160	162	50	158	153	372	229	63	164	162	395
40	le450_25b	450	24	156	161	51	164	160	377	188	58	156	156	404
41	le450_25c	450	24	407	417	150	400	393	736	510	180	396	394	780
42	le450_25d	450	24	400	422	150	400	403	736	461	179	397	394	791
43	le450_5a	450	4	780	772	484	755	738	1195	966	551	753	771	1209
44	le450_5b	450	4	769	768	498	759	738	1192	886	565	764	769	1280

TABLE 6: Heuristics results for graph coloring instances.

#	Name	Nodes	Colors	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
45	le450_5c	450	4	1459	1476	909	1422	1394	2024	1692	1083	1428	1464	2073
46	le450_5d	450	4	1464	1442	843	1449	1385	1981	1712	1060	1431	1436	2004
47	miles1000	128	41	52	58	49	51	49	133	75	49	129	70	137
48	miles1500	128	73	81	107	82	82	81	149	94	79	123	100	169
49	miles250	128	7	14	14	15	15	13	73	37	16	24	28	75
50	miles500	128	19	26	24	25	25	24	87	48	24	81	43	90
51	miles750	128	30	38	41	38	38	37	102	64	37	101	53	119
52	mugg100_1	100	3	5	12	4	6	5	55	19	5	12	12	57
53	mugg100_25	100	3	5	12	4	6	6	49	25	4	12	14	55
54	mugg88_1	88	3	5	10	4	5	5	43	19	5	9	10	50
55	mugg88_25	88	3	5	10	4	5	6	46	20	5	11	11	61
56	mulsol.i.1	197	48	60	68	60	60	57	142	76	59	131	80	153
57	mulsol.i.2	188	30	49	56	48	50	47	172	118	48	143	95	186
58	mulsol.i.3	184	30	50	56	48	49	47	177	117	48	160	99	176
59	mulsol.i.4	185	30	49	53	50	49	48	165	119	48	142	94	192
60	mulsol.i.5	186	30	50	62	48	50	50	171	114	48	107	100	191
61	myciel3	11	3	4	6	4	3	3	10	6	4	10	4	12
62	myciel4	23	4	5	9	6	5	5	24	12	5	15	9	27
63	myciel5	47	5	8	13	8	9	8	53	32	8	20	17	56
64	myciel6	95	6	18	24	12	19	19	135	105	13	37	51	143
65	myciel7	191	7	84	96	18	85	78	347	303	23	107	136	328
66	qg.order30	900	29	554	544	152	554	525	912	223	227	533	531	953
67	qg.order40	1600	39	1174	1105	627	1160	1078	1626	378	558	1131	1128	1671
68	qg.order60	3600	59	3076	2907	2425	3079	2897	3692	831	2234	3070	3008	3738
69	queen10_10	100	10	41	58	37	38	35	155	65	37	60	71	163
70	queen11_11	121	10	57	76	54	57	50	209	101	53	87	82	216
71	queen12_12	144	12	51	74	52	54	49	224	93	48	81	74	233
72	queen13_13	169	12	70	107	67	74	68	281	112	69	108	117	299
73	queen14_14	196	15	58	83	54	56	53	288	117	53	93	101	317
74	queen15_15	225	15	75	111	71	81	74	370	138	74	124	112	368
75	queen16_16	256	16	85	108	83	90	80	407	159	79	125	117	438
76	queen5_5	25	4	17	38	16	17	15	45	25	17	25	23	50
77	queen6_6	36	6	19	27	17	17	18	49	28	16	47	27	55
78	queen7_7	49	6	29	51	27	30	33	90	42	30	36	39	87
79	queen8_12	96	11	29	52	30	32	25	138	60	28	48	48	138
80	queen8_8	64	8	32	56	25	31	33	100	45	27	80	42	104
81	queen9_9	81	9	31	48	33	30	27	124	55	34	49	58	129
82	school1	385	13	997	1008	626	1009	987	1423	1330	691	1076	1037	1440
83	school1_nsh	352	13	723	742	445	735	720	1100	1048	490	734	781	1135
84	will199GPIA	701	3	438	418	438	453	430	1762	1542	442	547	555	1788
85	zeroin.i.1	211	48	62	72	62	61	60	147	80	60	126	89	159
86	zeroin.i.2	211	29	42	55	41	41	41	165	113	45	163	89	169
87	zeroin.i.3	206	29	40	55	43	40	39	179	111	40	162	91	164

TABLE 7: Statistic test results for graph coloring instances for Friedman (F), Friedman Aligned (FA), and Quade (Q).

	Statistics	p value	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
F	255.72	0	4.64	6.40	2.18	4.68	3.33	10.01	7.52	2.48	7.40	6.47	10.89
FA	77.86	0	374.88	461.35	209.30	370.94	340.22	820.66	575.50	212.84	568.01	505.86	829.45
Q	99.04	0	5.27	6.49	1.91	5.24	3.68	9.99	6.84	2.44	7.11	6.13	10.90

TABLE 8: Time for one run of each heuristic in seconds for graph coloring instances.

Instance	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
1-Insertions_4	15	14	18	15	14	11	13	6	15	14	13
2-Insertions_3	14	14	17	14	14	11	12	7	14	14	13
2-Insertions_4	16	16	17	17	16	12	14	7	16	16	14
3-Insertions_3	14	14	17	14	14	11	12	6	14	14	13
4-Insertions_3	15	14	18	15	14	11	15	7	15	15	13

TABLE 8: Continued.

Instance	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
anna	16	15	15	16	15	12	9	5	16	16	14
asH_3 31GPIA	58	44	31	46	44	43	61	27	46	47	45
asH_6 08GPIA	201	172	81	178	175	158	156	93	160	161	134
asH_9 58GPIA	488	321	140	370	354	371	317	203	371	371	275
david	15	14	14	15	14	11	9	5	15	15	14
DSJC1000.1	67	62	25	64	63	73	24	30	63	68	31
DSJC1000.5	51	53	21	55	59	48	17	23	59	59	21
DSJC1000.9	49	55	21	53	61	48	16	24	73	65	22
DSJC125.1	15	15	17	16	15	12	12	6	15	15	14
DSJC125.5	15	15	15	16	15	12	9	4	15	15	14
DSJC125.9	15	15	13	16	15	12	8	4	16	16	13
DSJC250.5	18	18	14	19	18	14	9	6	19	19	15
DSJC250.9	18	18	14	19	18	14	8	5	20	19	14
DSJC500.1	30	27	17	30	28	24	15	11	29	29	20
DSJC500.5	26	26	15	28	27	22	10	9	29	29	15
DSJC500.9	25	26	15	27	28	22	9	9	33	31	15
DSJR500.1	30	27	17	29	28	25	15	11	29	29	20
DSJR500.1c	26	26	15	28	28	22	9	9	31	30	15
DSJR500.5	25	26	15	27	28	22	9	9	33	31	15
fpsol2.i.1	27	26	15	29	27	24	13	10	30	33	15
fpsol2.i.2	25	25	15	26	25	21	10	9	26	27	16
fpsol2.i.3	24	25	15	25	24	20	10	8	25	26	15
games120	15	15	15	16	15	12	10	5	15	15	14
homer	32	30	18	33	34	27	17	12	32	34	24
huck	14	14	15	15	14	11	9	5	14	14	14
inithx.i.1	52	49	22	56	52	51	19	24	54	71	24
inithx.i.2	33	34	17	35	33	30	15	13	36	37	18
inithx.i.3	32	34	17	34	32	29	14	12	35	36	17
jean	14	15	15	15	14	11	9	5	15	14	14
le450_15a	25	26	16	27	25	22	12	9	27	28	17
le450_15b	25	27	16	27	25	22	12	9	27	28	17
le450_15c	25	27	16	27	25	22	12	9	26	28	17
le450_15d	25	27	16	27	25	22	12	9	27	28	17
le450_25a	24	25	15	26	25	21	10	9	27	27	16
le450_25b	24	25	15	26	25	21	10	9	27	27	15
le450_25c	25	26	15	26	25	21	10	9	27	27	16
le450_25d	25	25	15	27	25	21	10	9	27	27	16
le450_5a	26	31	20	28	26	23	25	12	27	32	22
le450_5b	26	31	20	28	26	23	24	12	27	32	22
le450_5c	25	29	20	27	25	22	24	12	27	28	22
le450_5d	25	29	20	27	25	22	24	12	26	28	22
miles1000	15	15	13	16	15	12	8	41	16	16	13
miles1500	15	16	15	16	15	12	8	41	15	16	13
miles250	15	16	15	16	15	12	10	5	15	16	13
miles500	15	15	14	16	15	12	8	41	16	15	14
miles750	15	15	13	16	15	12	8	4	16	16	13
mugg100_1	15	15	17	15	15	11	13	7	15	15	13
mugg100_25	15	15	17	15	15	11	13	7	15	15	13
mugg88_1	15	15	17	15	15	11	13	6	15	15	13
mugg88_25	15	15	17	15	15	11	13	7	15	15	13
multsol.i.1	17	18	14	18	17	13	8	5	18	18	14
multsol.i.2	16	18	14	17	16	13	8	5	17	17	14
multsol.i.3	16	18	14	17	16	13	8	5	17	17	14
multsol.i.4	16	18	14	17	16	13	8	5	17	17	14
multsol.i.5	16	18	14	17	16	13	8	5	17	17	14
myciel3	14	14	17	14	14	10	12	6	14	14	13
myciel4	14	14	16	14	14	10	11	6	14	14	13
myciel5	14	14	16	14	14	10	11	5	14	14	13
myciel6	14	15	15	15	15	11	11	5	15	15	14
myciel7	16	17	16	17	17	13	11	6	17	17	14

TABLE 8: Continued.

Instance	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
qg.order30	50	53	21	54	51	49	20	23	54	57	26
qg.order40	149	151	46	157	152	187	44	71	159	167	53
qg.order60	1016	1016	184	983	1087	983	191	532	1099	1113	226
queen10_10	15	15	15	15	15	11	9	5	15	15	14
queen11_11	15	15	15	15	15	12	9	5	16	16	14
queen12_12	16	15	14	16	15	12	9	5	16	16	14
queen13_13	16	16	15	17	16	12	9	5	16	16	14
queen14_14	17	17	15	18	17	13	9	5	17	17	15
queen15_15	18	18	15	19	18	14	9	6	18	18	15
queen16_16	18	19	14	19	19	15	9	6	19	19	15
queen5_5	14	14	18	14	14	10	12	6	14	14	14
queen6_6	14	14	16	14	14	10	10	5	14	14	14
queen7_7	14	14	16	14	14	11	11	5	14	14	14
queen8_12	15	15	14	15	14	11	9	5	15	15	14
queen8_8	14	14	16	15	14	11	10	5	14	14	14
queen9_9	14	14	14	15	14	11	10	5	15	15	14
school1	23	24	16	24	23	19	12	8	24	25	17
school1_nsh	21	22	16	22	21	18	11	7	22	23	16
will199GPIA	47	51	35	59	67	51	68	32	54	52	46
zero.in.i.1	17	18	14	18	17	14	8	5	18	18	14
zero.in.i.2	17	18	15	18	17	14	9	5	18	18	14
zero.in.i.3	17	18	15	18	17	13	9	5	18	18	14

7.1.2. Capacitated Vehicle Routing Problem (CVRP). Three sets of the state of the art were used and tested on 41 runs:

- (1) Augerat et al. (SET A), 9 instances, proposed in [49]
- (2) Christofides, Mingozzi, and Toth (CMT), 14 instances, proposed in [50]
- (3) Golden, Wasil, Kelly, and Chao (GWKC), 20 instances, proposed in [51]
- (4) Uchoa et al., 9 instances, proposed in [52]

In Table 9, we show the fitness values for the instances and the lowest city cost tour is indicated in bold, where n is the number of nodes, Q is the capacity of each vehicle, and k is the number of vehicles (colors in the case of graph coloring). The time of each run is reported in Table 10.

We applied the same procedure to the statistical tests of Friedman (FT), Alienated Friedman (AFT), and Quade (Qt) to distinguish the behavior of the heuristics set. We established $\alpha = 0.05$ and h_0 as there are no differences between the performance of the heuristics and established h_a as there are differences between the performance of the heuristics. Table 11 shows the ranks obtained in the three statistical tests.

In this case, the heuristic H_5 has the lowest rank for the tests and H_6 has the second-lowest rank for QT and FT.

7.2. Selection of Features and Classes by Statistical Tests. According to the steps mentioned in Section 5.1, we must determine first the number of clusters or classes to split all our test instances. In this case, $T = 139$ and $c = 8$.

We considered 8 classes and used k -means clusters and we expected uniformly distributed instances in the clusters. The k -means algorithm was applied with a maximum number of Iterations = 500, initially random starting points.

To consider the uniform distribution of classes into clusters, we used the Manhattan distance obtained after the experimental work, with the best results.

Table 12 contains the class details, number of instances per cluster/class, number of GCP (3rd column), or CVRP (4th column) per class, min and max nodes, and min and max number of colors nodes. In this experimentation, clusters 1, 5, 6, and 7 have only GCP instances, clusters 3, 4, and 8 have CVRP instances, and only cluster 2 has both problem domains.

7.3. Training and Test Classifiers for the Instance's Classes. After the heuristic pool design phase for the hyperheuristics, we split our dataset into training and test. The training dataset was created by 125 instances with 15 features (basic + inner) and the unseen instances were made by 15 instances. The results of the classification with Naive Bayes are reported in Table 13.

Table 14 contains the confusion matrix of the process classification. We observed that, for some classes like 3, 4, 7, and 8, the patterns were classified correctly. The rest of the classes have some patterns classified incorrectly, but, e.g., for the 3 patterns of class 1 classified into 5 and 6, we used the same pool of low-level heuristics and this does not represent an issue for the next step.

7.4. Designing and Testing the Hyperheuristic Offline Learning with K-Folds. In the next step, the statistical tests were applied to heuristics and will form the characteristics of our instances, graph coloring, and CVRP per class. In this phase, we choose according to the rankings the heuristics which have $x < (\text{minrank} + \text{maxrank})/2$. If, for example, the Aligned Friedman has a min rank = 776.5 and max rank = 3604.5, the

TABLE 9: Results for CVRP-capacitated instances.

	Instance	Cities	Vehicles	OP	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
88	CMT1	50	5	524.61	1583	1161	1569	1568	1117	1141	1664	1576	1664	1664	1664
89	CMT2	75	10	835.26	2353	1591	2351	2349	1728	1898	2577	2547	2577	2577	2577
90	CMT3	100	8	826.14	1904	1821	1939	2017	1699	1743	3293	2249	3293	3293	3293
91	CMT4	150	12	1028.42	2578	2674	2520	2519	2347	2462	5096	3115	5096	5096	5096
92	CMT5	199	17	1291.29	3159	4469	6490	3144	3008	3155	6742	6707	6742	6692	6742
93	CMT6	50	6	555.43	1027	1160	1569	1028	941	933	1664	1573	1664	1571	1664
94	CMT7	75	11	909.68	1492	1616	2351	1465	1357	1407	2577	2517	2577	2531	2577
95	CMT8	100	9	865.95	1743	1890	1799	1906	1598	1636	3293	2203	3293	3195	3293
96	CMT9	150	14	1162.55	2308	2768	2489	2325	2185	2256	5096	2971	5096	4918	5096
97	CMT10	199	18	1395.85	4607	4607	2852	2852	2791	2870	6742	6707	6742	6513	6742
98	CMT11	120	7	1042.12	2077	2077	2372	2372	2046	2110	2608	2449	2608	2608	2608
99	CMT12	100	10	819.56	1692	1692	1733	1733	1472	1528	2306	2144	2287	2300	2306
100	CMT13	120	11	1541.14	2040	2040	2041	2041	1827	1906	2608	2476	2608	2577	2608
101	CMT14	100	11	866.37	1717	1717	1609	1609	1444	1517	2306	2171	2306	2270	2306
102	GWKC1	240	9	5623.47	9497	9155	8607	9261	7350	8565	13312	10546	13312	13312	13312
103	GWKC2	320	10	8404.61	14571	16631	15161	13399	12320	10903	20961	16801	20787	20735	21080
104	GWKC3	400	9	11036.2	19197	21734	24161	20070	17167	18793	26117	22304	26117	26117	26117
105	GWKC4	480	10	13590	27046	30204	32382	26643	27133	26504	38732	31793	38892	38892	38892
106	GWKC5	200	5	6460.98	14224	15677	13009	14207	12800	13435	19558	15763	21378	21378	21378
107	GWKC6	280	7	8412.9	15407	21043	14925	15701	14022	14539	26699	17688	26699	26699	26699
108	GWKC7	360	8	10102.7	25909	23841	25909	25909	24611	25718	25909	25909	25909	25909	25909
109	GWKC8	440	10	11635.3	21500	23969	20817	20483	18560	18677	31506	26204	31506	31506	31506
110	GWKC9	255	14	579.71	1102	989	2323	1264	954	916	2581	2319	2581	2581	2581
111	GWKC10	323	16	735.66	1344	1312	3476	1277	1222	1229	3589	3451	3589	3589	3589
112	GWKC11	399	17	912.03	1751	1919	4758	1640	1572	1485	5040	4478	5039	5039	5040
113	GWKC12	483	19	1101.5	2295	1979	6738	2462	2192	2363	5416	6520	6750	6750	6750
114	GWKC13	252	26	857.19	2010	1301	2062	2031	1475	1615	2117	2078	2158	2151	2158
115	GWKC14	320	29	1080.55	2351	1637	2637	2322	2221	2389	2748	2665	2745	2744	2748
116	GWKC15	396	33	1337.87	3262	2062	3322	3207	2793	3025	3366	3308	3366	3364	3366
117	GWKC16	480	36	1611.56	3978	2596	4107	3820	3492	3783	4198	4092	4217	4212	4217
118	GWKC17	240	22	707.76	1754	1518	2163	1729	1611	1669	2376	2165	2376	2376	2376
119	GWKC18	300	27	995.13	2130	2165	3443	2150	2073	2155	3443	3443	3443	3425	3443
120	GWKC19	360	33	1365.6	3238	2808	4540	3024	3226	3575	4713	4466	4713	4713	4713
121	GWKC20	420	38	1817.59	3519	3746	6190	3511	3189	3327	6186	5992	6191	6084	6191
122	A-n32-k5	32	5	784	1538	1476	1431	1474	1265	1255	2230	1662	2229	2229	2230
123	A-n45-k6	45	6	1733.36	1733	1733	1733	1733	1273	1156	1733	1733	1733	1733	1733
124	A-n55-k9	55	9	1073	1660	1413	1453	1642	1439	1328	2376	1901	2376	2376	2376
125	A-n61-k9	61	9	1034	2500	1814	2338	2464	1731	1838	2592	2517	2592	2592	2592
126	A-n62-k8	62	8	1288	1772	1724	2039	1792	1551	1615	3194	2219	3194	3194	3194
127	A-n63-k9	63	9	1314	2480	1879	2452	2396	1813	1835	2677	2564	2677	2677	2677
128	A-n64-k9	65	9	1401	1935	1893	2169	1948	1744	1715	2613	2203	2613	2613	2613
129	A-n65-k9	65	9	1174	2609	1940	2600	2546	1820	1892	2949	2808	2949	2949	2949
130	A-n69-k9	69	9	1159	2149	1803	2080	1979	1626	1687	3220	2477	3220	3220	3220
131	X-n148-k46	148	46	43448	71826	54940	75062	71665	68924	71179	77623	78715	79116	79116	79116
132	X-n153-k22	153	22	21220	36700	39843	52741	36155	34116	34337	71801	56820	71799	71793	71801
133	X-n157-k13	157	13	16876	36364	24507	36364	36364	27917	29210	36364	36364	36364	36364	36364
134	X-n162-k11	162	11	14138	39232	37225	36381	38162	32918	34059	79973	60657	79969	79963	79973
135	X-n367-k17	367	17	22814	71209	85326	97502	72482	68648	70027	154244	112919	154244	154244	154244
136	X-n393-k38	393	39	38260	141432	74629	152513	143398	130939	148728	158058	155203	158058	158058	158058
137	X-n401-k29	401	29	66187	140100	106590	163158	141582	141056	145043	197323	163230	197323	197323	197323
138	X-n411-k19	411	19	19718	67963	79642	81034	67090	63038	66173	173269	88152	173269	173269	173269
139	X-n420-k130	420	130	10798	182376	132745	207213	183053	203757	200726	217483	216871	217483	217483	217483

TABLE 10: Time for one run of each heuristic in seconds for CVRP-capacitated instances.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
A-n32-k5	7	7	12	7	7	5	7	4	7	7	9
A-n45-k6	7	7	12	7	7	5	6	3	7	7	9
A-n55-k9	7	7	11	7	7	5	5	3	7	7	9
A-n61-k9	7	7	11	7	7	5	5	3	7	7	9
A-n62-k8	7	7	11	8	7	5	6	3	7	7	9
A-n63-k9	7	7	11	7	7	5	5	3	7	7	9
A-n63-k10	7	7	11	7	7	5	5	3	7	7	9
A-n64-k9	7	7	11	7	7	5	5	3	7	7	9
A-n65-k9	7	7	11	7	7	5	5	3	7	7	9
A-n69-k9	7	7	11	8	7	5	5	3	8	7	9
CMT1	7	7	12	7	7	5	7	4	7	7	9
CMT2	7	7	11	7	7	5	5	3	8	8	9
CMT3	7	7	11	8	7	5	6	3	8	8	9
CMT4	8	8	11	8	8	6	6	3	9	9	9
CMT5	8	8	10	9	8	6	5	3	10	10	9
CMT6	7	7	12	7	7	5	6	3	7	7	9
CMT7	7	7	11	8	7	5	5	2	8	7	9
CMT8	7	7	11	8	7	5	6	3	8	8	9
CMT9	8	8	10	8	8	6	5	2	9	9	9
CMT10	8	8	10	9	8	6	5	2	10	10	9
CMT11	7	7	11	8	7	6	6	3	8	8	9
CMT12	7	7	12	7	7	5	7	3	7	7	9
CMT13	7	7	12	7	7	5	6	3	7	7	9
CMT14	7	7	12	7	7	5	7	4	7	7	9
X-n420-k130	11	11	16	11	11	9	11	8	11	11	13
X-n411-k19	11	11	16	11	11	9	11	8	11	11	13
X-n401-k29	11	11	16	11	11	9	10	7	11	11	13
GWKC1	8	8	11	9	8	6	6	3	9	9	9
GWKC2	8	8	11	10	8	6	6	3	10	10	10
GWKC3	9	9	11	11	9	7	8	4	12	12	13
GWKC4	10	10	13	12	10	8	8	3	12	12	11
GWKC5	8	8	13	9	8	6	9	4	9	8	9
GWKC6	8	8	12	9	8	6	7	3	9	9	10
GWKC7	9	9	11	9	9	7	7	3	10	10	10
GWKC8	9	9	11	11	9	7	7	3	12	12	10
GWKC9	8	8	11	9	8	6	5	3	10	10	10
GWKC10	9	9	11	10	9	7	6	3	12	11	10
GWKC11	9	9	11	11	9	7	6	3	13	13	10
GWKC12	10	10	11	12	10	8	6	3	15	15	11
GWKC13	8	8	11	10	9	7	5	2	12	12	10
GWKC14	9	9	11	10	9	7	5	3	14	14	10
GWKC15	10	10	11	11	9	7	5	3	16	16	10
GWKC16	10	10	11	12	10	8	5	3	19	18	10
GWKC17	8	8	10	9	8	6	5	2	11	11	10
GWKC18	9	9	10	10	9	7	5	2	13	13	10
GWKC19	9	9	11	11	9	7	5	3	16	15	10
GWKC20	10	10	11	12	10	8	5	3	18	18	11
X-n148-k46	11	11	16	11	11	9	11	8	11	11	13
X-n153-k22	11	11	16	11	11	9	11	7	11	11	13
X-n157-k13	11	11	16	11	11	9	10	7	11	11	13
X-n162-k11	11	11	15	11	11	9	10	7	11	11	13
X-n367-k17	11	11	15	11	11	9	10	7	11	11	13
X-n393-k38	11	11	15	11	11	9	10	7	11	11	13

TABLE 11: Statistic test results for capacitated vehicle routing problem instances for Friedman (F), Friedman Aligned (FA), and Quade (Q).

	Statistics	p value	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
F	420.84	0	4.49	8.79	9.69	3.53	5.59	4.16	1.60	2.46	9.32	6.92	9.45
FA	47.3	0	170.82	441.25	443.96	143.53	280.22	169.32	127.19	147.94	441.70	342.00	443.57
Q	79.97	0	4.29	8.87	9.72	3.68	5.59	3.96	1.67	2.57	9.26	6.91	9.47

TABLE 12: Summary cluster/classes with k -means algorithm.

Cluster	# of instances	%	GC	CVRP	Min nodes	Max nodes	Min colors/vehicles	Max colors/vehicles
1	18	12.9	18	0	11	149	3	6
2	10	7.2	8	2	420	3600	3	214
3	11	7.9	0	11	148	480	5	46
4	5	3.6	0	5	367	420	17	130
5	19	13.7	19	0	64	256	7	30
6	31	22.3	31	0	125	900	11	122
7	11	7.9	11	0	352	701	3	14
8	34	24.5	0	34	32	483	5	33

TABLE 13: Classification value results.

Correctly classified	114	91.20%
Incorrectly classified	11	8.80%
TP rate	0.912	
FP rate	0.022	
Precision	0.918	
Recall	0.912	

TABLE 14: Confusion matrix, TP rate (TPR), FP rate (FPR), and precision (P) for each class.

Class	1	2	3	4	5	6	7	8	TPR	FPR	P
1	13	0	0	0	2	1	0	0	0.813	0	1
2	0	7	0	0	0	0	0	2	0.778	0	1
3	0	0	10	0	0	0	0	0	1	0	1
4	0	0	0	4	0	0	0	0	1	0	1
5	0	0	0	0	12	5	0	0	0.706	0.019	0.857
6	0	0	0	0	0	27	1	0	0.964	0.062	0.818
7	0	0	0	0	0	0	10	0	1	0.009	0.909
8	0	0	0	0	0	0	0	31	1	0.021	0.939

limit value for considering a heuristic must be 2190.5. Because heuristic 11 has the worst performance for both datasets, we did not consider this heuristic for this experimentation phase.

Table 15 shows the ranks and the heuristic fitness for each class, e.g., class 1 for QT, and AFT has the same heuristics H_5 , H_6 , H_4 , H_1 , H_2 , and H_3 , while FT does not consider H_3 . We only consider H_5 , H_6 , H_4 , H_1 , H_2 , and H_3 as a minimum set.

The selected heuristics for each class can be summarized into five groups:

- (1) H_3 , H_8 , H_5 , H_4 , and H_1
- (2) H_3 , H_8 , H_7 , H_5 , and H_2
- (3) H_5 , H_6 , H_2 , H_4 , H_1 , and H_3
- (4) H_2 , H_1 , H_5 , H_4 , and H_6
- (5) H_3 , H_8 , H_5 , and H_4

Therefore, we design 5 algorithms for these 8 classes. This means the hyperheuristic with the high-level search strategy was the same, but the heuristics pool set was different according to these 5 subset heuristics. We trained and tested the hyperheuristic on short-length pools. We left 10% of instances as unseen for the hyperheuristics and the results are shown in Table 16.

The hyperheuristic configuration was 10 iterations for local search and 100,000 function calls. For some GCP instances, we got the optimum number of colors (denoted in bold in Table 16). Besides, for the instances A-n45-k6,

A-n55-k9, A-n62-k8, A-n64-k9, CMT13, GWKC1, GWKC2, GWKC3, and X-n148-k46, we get values near to the optimal with a maximum of 20% of the distance.

7.5. Classification of the Test Instances and Application of the Hyperheuristic to the Corresponding Instance. Finally, for the 14 unseen instances, we used the Naive Bayes classifier, and it determines the class for these instances. Later, we applied the hyperheuristic with the corresponding pool heuristics according to the previous design and we obtained the results shown in Tables 17 and 18.

Table 17 shows the confusion matrix, TP rate, FP rate, and precision of the classification test. In these results, two patterns that belong to class five were classified incorrectly, but this does not affect the hyperheuristic solution because this class shares the same heuristics with class 6.

7.6. Statistical Comparison of Results. Finally, to compare if there are differences between the results applying the methodology and without applying the methodology, an experiment was carried out where the hyperheuristics were executed 33 times with the entire set of heuristics and 100,000 function calls. The results are shown in Table 19.

First, the statistical distributions followed by each set of results by class were analyzed, that is, the Shapiro-Wilks test [53] was applied to determine if the results of the methodology, hyperheuristic without methodology (HHPC), and the optimal state of the art followed a normal distribution.

TABLE 15: Heuristics ranks per classes, Friedman (Fr), Aligned Friedman (A-Fr), and Quade (Qu).

Class 1						Class 2						
*Fr	H_3 1.50	H_8 1.69	H_5 3.78	H_1 4.03	H_4 4.08	H_2 6.69	H_3 2.8	H_8 3	H_7 3.4	H_5 4	H_2 5.7	H_{10} 6.25
*A-Fr	H_3 627	H_8 638	H_1 1089	H_5 1091	H_4 1102	H_2 1760.5	H_3 293	H_8 320	H_7 335	H_5 428	H_2 508	H_4 540
*Qu	H_3 1.55	H_8 1.59	H_5 3.77	H_4 4.04	H_1 4.15	H_2 6.81	H_3 2.24	H_8 2.73	H_7 2.98	H_5 4.18	H_2 5.51	H_{10} 6.39
Class 3						Class 4						
*Fr	H_5 1.64	H_6 2.18	H_2 4.27	H_4 4.27	H_1 4.82	H_3 5.18	H_5 2.4	H_2 2.6	H_1 2.8	H_4 3.6	H_6 3.6	H_3 6
*A-Fr	H_5 242	H_6 283	H_4 403	H_2 418	H_1 419	H_3 568	H_2 33	H_1 74	H_5 78	H_4 79	H_6 91	H_3 128
*Qu	H_5 1.71	H_6 2.26	H_2 4.20	H_4 4.52	H_1 4.88	H_3 5.38	H_2 2.07	H_1 2.67	H_5 2.67	H_4 3.67	H_6 3.93	H_3 6
Class 5						Class 6						
*Fr	H_8 1.45	H_3 1.55	H_5 3.53	H_1 4.37	H_4 4.68	H_2 6.03	H_3 1.32	H_8 1.69	H_5 4.35	H_4 4.92	H_1 5.05	H_{10} 5.92
*A-Fr	H_8 646	H_3 651	H_5 1144.5	H_1 1197	H_4 1225.5	H_2 2010.5	H_3 1306	H_8 1345	H_5 4112	H_4 4211	H_1 4227.50	H_2 5058
*Qu	H_8 1.47	H_3 1.53	H_5 3.67	H_1 4.44	H_4 4.69	H_2 5.78	H_3 1.48	H_8 1.53	H_5 3.90	H_4 4.55	H_1 4.60	H_2 6.47
Class 7						Class 8						
*Fr	H_3 1.32	H_8 2.45	H_5 3.45	H_4 5.32	H_1 5.86	H_9 5.91	H_5 1.47	H_6 2.38	H_2 3.44	H_4 4.21	H_1 4.59	H_3 5.46
*A-Fr	H_3 108.50	H_8 171	H_5 477.50	H_4 578.50	H_1 637.50	H_9 642	H_5 2328	H_6 2691	H_2 2989	H_4 3847	H_1 4070	H_3 6169.50
*Qu	H_3 1.12	H_8 2.17	H_5 3.65	H_4 5.09	H_9 5.74	H_1 5.82	H_5 1.50	H_6 2.28	H_2 3.27	H_4 4.46	H_1 4.70	H_3 5.43

TABLE 16: Hyperheuristics results for graph coloring and CVRP.

#	C	Median	DE	Average	Opt	#	C	Median	DE	Average	Opt	#	C	Median	DE	Average	Opt
4	1	3	2	4	3	30	5	10	4	12	10	7	7	22	284	162	3
1	1	4	39	30	3	34	5	10	5	12	9	19	7	160	212	490	11
3	1	4	12	9	3	49	5	8	10	15	7	37	7	326	185	623	14
4	1	3	5	5	3	50	5	20	11	26	19	38	7	321	183	620	14
5	1	3	10	7	3	51	5	32	14	40	30	43	7	407	160	674	4
14	1	18	28	40	4	65	5	7	73	72	7	44	7	402	168	672	4
53	1	3	8	6	3	69	5	22	25	36	10	45	7	294	398	1,147	4
54	1	3	5	5	3	70	5	37	33	56	10	46	7	293	390	1,156	4
55	1	3	6	5	3	71	5	32	42	55	12	83	7	340	171	658	13
61	1	3	0	3	3	72	5	52	55	79	12	84	7	320	413	548	3
62	1	4	2	5	4	73	5	35	58	64	15	123	8	1,821	261	1,524	1,733
63	1	5	6	7	5	75	5	60	80	101	16	124	8	1,147	517	1,598	1,073
64	1	6	24	19	6	80	5	15	15	25	8	125	8	1,366	468	2,082	1,034
76	1	4	7	8	4	81	5	18	21	29	9	126	8	1,410	767	2,052	1,288
77	1	8	8	12	6	16	6	60	33	99	42	127	8	1,582	437	2,145	1,314
78	1	16	11	23	6	18	6	127	82	255	70	128	8	1,510	455	1,975	1,401
8	2	573	365	1,176	3	20	6	247	285	715	47	129	8	1,433	583	2,271	1,174
9	2	1,011	510	2,190	3	21	6	285	206	628	122	130	8	1,420	747	2,120	1,159
11	2	541	535	1,435	19	22	6	21	59	96	11	88	8	909	291	1,339	525
13	2	771	478	1,542	214	24	6	181	97	318	121	97	8	2,469	1,897	4,622	1,396
23	2	153	441	962	84	25	6	75	43	133	64	98	8	1,707	342	2,174	1,042
117	2	2,179	776	3,343	1,612	26	6	42	77	98	29	99	8	1,116	426	1,605	820
121	2	2,858	1,520	4,528	1,818	27	6	40	80	100	29	100	8	1,548	418	2,051	1,541
67	2	318	330	897	39	29	6	14	35	35	12	101	8	1,117	420	1,564	866
68	2	753	689	2,522	59	31	6	68	86	131	53	90	8	1,441	787	2,105	826
103	3	9,429	4,587	13,967	8,405	32	6	45	130	142	30	91	8	2,049	1,313	3,099	1,028
104	3	13,211	5,617	19,240	11,036	33	6	46	136	145	30	92	8	2,619	1,842	4,702	1,291
106	3	11,376	4,025	14,762	6,461	35	6	85	104	239	14	93	8	815	342	1,165	555
107	3	12,566	6,118	17,696	8,413	36	6	85	108	238	14	94	8	1,188	572	1,800	910

TABLE 16: Continued.

#	C	Median	DE	Average	Opt	#	C	Median	DE	Average	Opt	#	C	Median	DE	Average	Opt
108	3	13,221	4,905	23,107	10,103	39	6	37	68	126	24	95	8	1,436	727	2,024	866
109	3	14,390	7,055	21,411	11,635	40	6	35	69	124	24	102	8	6,023	3,021	8,874	5,624
131	3	50,634	9,126	68,656	43,448	42	6	127	137	332	24	111	8	1,056	1,172	2,201	736
132	3	30,323	17,173	47,025	21,220	47	6	43	15	53	41	112	8	1,340	1,671	2,996	912
133	3	22,175	5,420	32,883	16,876	48	6	74	15	86	73	113	8	1,744	2,287	4,038	1,102
134	3	28,769	21,922	45,678	14,138	56	6	50	17	62	48	114	8	1,138	422	1,658	857
135	4	65,098	59,475	160,617	22,814	57	6	40	28	59	30	115	8	1,447	522	2,154	1,081
136	4	67,184	29,544	127,827	38,260	58	6	37	26	57	30	116	8	1,811	574	2,737	1,338
137	4	97,074	34,915	149,750	66,187	60	6	40	27	58	30	118	8	1,300	462	1,810	708
139	4	125,875	27,756	192,093	10,798	66	6	131	193	417	29	119	8	1,623	837	2,587	995
10	5	10	6	15	10	85	6	53	17	65	48	120	8	2,370	1,032	3,519	1,366
15	5	45	48	95	15	86	6	34	28	55	29	110	8	823	803	1,603	580
28	5	8	14	14	8	87	6	33	26	53	29						

The best results are highlighted in bold.

TABLE 17: Confusion matrix, TP rate (TPR), FP rate (FPR), and precision (P) for each test instance.

Class	1	2	3	4	5	6	7	8	TPR	FPR	P
1	2	0	0	0	0	0	0	0	1	0	1
2	0	1	0	0	0	0	0	0	1	0	1
3	0	0	1	0	0	0	0	0	1	0	1
4	0	0	0	1	0	0	0	0	1	0	1
5	0	0	0	0	0	2	0	0	0	0	—
6	0	0	0	0	0	3	0	0	1	0.182	0.6
7	0	0	0	0	0	0	1	0	1	0	1
8	0	0	0	0	0	0	0	3	1	0	1

TABLE 18: Fitness results of test instances (hyperheuristic).

Instance	Class	Fitness	Instance	Class	Fitness
anna	1	3	DSJC250.5	6	27
mugg100_1	1	3	le450_25c	6	26
DSJC1000.5	2	83	multsol.i.4	6	30
GWKC4	3	14255.68	school1	7	13
X-n411-k19	4	20005.37	A-n32-k5	8	876.19
queen15_15	5	29	CMT2	8	911.37
queen8_12	5	21	CMT9	8	1258.57

The test was applied with a $\alpha = 0.05$. The results of the test are shown in Table 20, and the data shown in Tables 16, 18, and 19 were taken for the tests. It should be noted that the results of the methodology only for clusters 4 and 5 were normal, the results of the optimal state of the art only for only cluster 4 were normal, and the results of HHPC for clusters 1, 4, 5, and 7 were normal.

Student' t -tests for methodology and state of the art were applied. For the state of the art, methodology, and HHPC, we established $\alpha = 0.05$ as a level of significance. The null and alternative hypotheses for methodology and HHPC are as follows:

- (i) h_0 : there are no differences between the performance of hyperheuristic with the methodology and without the methodology
- (ii) h_a : there are differences between the performance of hyperheuristics with the methodology and without the methodology

For methodology and state of the art,

- (i) h_0 : there are no differences between the performance of hyperheuristics and the optimal state of the art
- (ii) h_a : there are differences between the performance of hyperheuristics

The statistical results of the tests are shown in Table 21. With these values, we can observe the following:

- (i) *Methodology and HHPC*. It can be inferred that the results of the methodology are significantly different from those of hyperheuristics with the whole set of heuristics. This means that the methodology improved performance and allowed limiting the set of heuristics for each of the clusters.
- (ii) *Methodology and State of the Art*. It can be inferred that no statistical evidence was found that the results of the methodology differ from the optimal ones of the state of the art, except in clusters 5 and 7. This is because it is where there are more atypical data or that they were badly classified which opens an area of opportunity for the refinement of the methodology.

TABLE 19: Results of hyperheuristic with the complete set heuristics and without methodology (HHPC), cluster (C), and T which are the training set instances.

Instance	c	HHPC	Instance	c	HHPC	Instance	c	HHPC	Instance	c	HHPC
1-Insertions_4	1	34	X-n367-k17	4	155287	le450_15b	6	242	CMT12	8	2345
2-Insertions_3	1	11	X-n393-k38	4	158696	le450_25a	6	137	CMT13	8	2769
2-Insertions_4	1	60	X-n401-k29	4	198461	le450_25b	6	140	CMT14	8	2361
3-Insertions_3	1	14	X-n420-k130	4	218854	le450_25d	6	341	CMT3	8	3335
4-Insertions_3	1	28	david	5	27	miles1000	6	72	CMT4	8	5185
DSJC125.1	1	70	DSJC125.5	5	138	miles1500	6	97	CMT5	8	6785
mugg100_25	1	15	games120	5	31	mulsol.i.1	6	74	CMT6	8	1671
mugg88_1	1	20	huck	5	22	mulsol.i.2	6	87	CMT7	8	2582
mugg88_25	1	14	jean	5	20	mulsol.i.3	6	74	CMT8	8	3355
myciel3	1	6	miles250	5	24	mulsol.i.5	6	70	GWKC1	8	13602
myciel4	1	9	miles500	5	41	qg.order30	6	351	GWKC10	8	3630
myciel5	1	26	miles750	5	51	zeroin.i.1	6	78	GWKC11	8	5057
myciel6	1	48	myciel7	5	117	zeroin.i.2	6	70	GWKC12	8	6775
queen5_5	1	26	queen10_10	5	82	zeroin.i.3	6	78	GWKC13	8	2191
queen6_6	1	29	queen11_11	5	108	asH_3 31GPIA	7	433	GWKC14	8	2793
queen7_7	1	42	queen12_12	5	105	DSJC500.1	7	477	GWKC15	8	3413
asH_6 08GPIA	2	746	queen13_13	5	141	le450_15c	7	618	GWKC17	8	2398
asH_9 58GPIA	2	1157	queen14_14	5	143	le450_15d	7	643	GWKC18	8	3467
DSJC1000.1	2	1090	queen16_16	5	184	le450_5a	7	632	GWKC19	8	4756
DSJC1000.9	2	1625	queen8_8	5	50	le450_5b	7	699	GWKC9	8	2604
DSJR500.1c	2	930	queen9_9	5	69	le450_5c	7	861	A-n32-k5	T	2235
GWKC16	2	4264	DSJC125.9	6	119	le450_5d	7	792	anna	T	27
GWKC20	2	6248	DSJC250.9	6	255	school1_nsh	7	671	CMT2	T	2611
qg.order40	2	623	DSJC500.5	6	656	will199GPIA	7	898	CMT9	T	5145
qg.order60	2	2098	DSJC500.9	6	611	A-n45-k6	8	1856	DSJC1000.5	T	1765
GWKC2	3	21504	DSJR500.1	6	100	A-n55-k9	8	2474	DSJC250.5	T	336
GWKC3	3	26716	DSJR500.5	6	314	A-n61-k9	8	2681	GWKC4	T	39592
GWKC5	3	21807	fpsol2.i.1	6	137	A-n62-k8	8	3200	le450_25c	T	321
GWKC6	3	27213	fpsol2.i.2	6	136	A-n63-k9	8	2774	mugg100_1	T	17
GWKC7	3	26506	fpsol2.i.3	6	131	A-n64-k9	8	2709	mulsol.i.4	T	104
GWKC8	3	32073	homer	6	50	A-n65-k9	8	2977	queen15_15	T	164
X-n148-k46	3	80822	inithx.i.1	6	190	A-n69-k9	8	3258	queen8_12	T	52
X-n153-k22	3	72255	inithx.i.2	6	188	CMT1	8	1709	school1	T	850
X-n157-k13	3	37747	inithx.i.3	6	178	CMT10	8	6763	X-n411-k19	T	174414
X-n162-k11	3	80314	le450_15a	6	251	CMT11	8	2624			

TABLE 20: Normality test results of the methodology, state of the art, and hyperheuristics with complete pool.

Cluster	Methodology	State of the art	HHPC
1	$2.26E-05$	$2.69E-04$	$1.13E-01$
2	0.0309	$1.57E-04$	$7.52E-03$
3	0.01438	$1.80E-03$	$4.28E-03$
4	0.3551	$7.91E-01$	$3.29E-01$
5	0.05466	$1.05E-03$	$9.61E-02$
6	$1.20E-05$	$6.53E-05$	$2.25E-05$
7	0.0255	$3.32E-03$	$7.31E-01$
8	$4.95E-08$	$2.40E-09$	$4.59E-07$

TABLE 21: Results of the Student t -tests for methodology and Edo. Art and methodology and HHPC.

Cluster	Methodology and Edo. Art P_{value}	Methodology and HHPC P_{value}
1	0.1641	5.12E-05
2	0.1489	0.1496
3	0.3211	0.02586
4	0.02814	0.004389
5	0.00601	0.0002388
6	0.01144	0.0009709
7	4.43E-07	6.65E-06
8	0.06235	2.52E-05

8. Conclusion

In this work, a methodology is proposed to select low-level heuristics for a hyperheuristic approach to offline learning oriented to the solution of instances of different constraint satisfaction problems. The proposal was applied to two different problems well known and studied in the state of the art, which were the coloring of graphs and the vehicle routing problem with a specific capacity, GCP and CVRP, respectively.

The methodology is focused on optimizing the number of heuristics that can be applied to different constraint satisfaction problems in a hyperheuristic approach. Information on the performance of an original set of heuristics for the instances of the problem is obtained from the different problems. The performance information is used to generate characteristic vectors for each instance, which is used to generate equivalence classes of instances of the problem. The grouping in classes allows to identify the heuristics that apply to each class and from that information, a reduction of the number of heuristics necessary to obtain good solutions in the instance of each class is made and to reduce the total number of heuristics that can be applied in the hyperheuristic approach to solving the problems involved.

In the application to the GCP and CVRP, the information on the performance of the heuristics was obtained through a metalearning process, and this information was used to obtain the basic and internal characteristics of the instances. The instances were grouped into 5 classes using the k-means algorithm with the Mahalanobis metric. For each class, the sets of heuristics that could be applied to all their instances were identified, and through a process of hierarchization and cutoff criteria, the number of heuristics per class was reduced.

For training and testing, the Naive Bayes classifier and information on the characteristics of the instances were used. The experimental results show that the hyperheuristic in each class could efficiently solve each instance, and the classifier was able to predict the class for each problem instance.

The identification and reduction of heuristics to find the solution of complex problems is an optimization strategy that can do the search for solutions to problems of satisfaction of restrictions efficiently. The methodology presented allows generating a framework with a level of generality that can be trained to solve different problems of satisfaction of constraints simultaneously under the hyperheuristic approach. Once trained, it can allow finding good solutions to different problems with a common base of heuristics for instances of problems grouped by the efficiency of the solution heuristics.

Finally, the methodology makes it possible to improve the search for solutions to sets of problems by exploring the diversification of some of its components such as classification algorithms, metrics, heuristics, and selection criteria, which may be different for sets of different problems. A study of these possibilities is proposed as future work.

Data Availability

The instances data used to support the findings of this study have been deposited in the graph coloring repository <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/>, <http://archive.dim.acs.rutgers.edu/pub/challenge/graph/benchmarks/color/>, <https://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/>. They are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank Tecnológico Nacional de México/I. T. León and Universidad de Guanajuato. This work was supported by the National Council of Science and Technology of Mexico (CONACYT) via the Scholarship for Postgraduate Study 446106 (L. Ortiz) and Research Grant: CÁTEDRAS-2598 (A. Rojas). The authors thank the participation of Valentin Calzada-Ledesma from Instituto Tecnológico Superior de Purísima for the revision and correction of this article.

References

- [1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.
- [2] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Annals of Operations Research*, vol. 239, no. 1, pp. 3–38, 2016.
- [3] W. B. Yates and E. C. Keedwell, "An analysis of heuristic subsequences for offline hyper-heuristic learning," *Journal of Heuristics*, vol. 25, no. 3, pp. 399–430, 2019.
- [4] E. K. Burke, M. R. Hyde, G. Kendall et al., "A classification of hyper-heuristic approaches: revisited," in *Handbook of Metaheuristics*, pp. 453–477, Springer International Publishing, Berlin, Germany, 2018.
- [5] J. Kanda, A. D. Carvalho, E. Hruschka, C. Soares, and P. Brazdil, "Meta-learning to select the best meta-heuristic for the traveling salesman problem: a comparison of meta-features," *Neurocomputing*, vol. 205, pp. 393–406, 2016.
- [6] J. R. Rice, "The algorithm selection problem," in *Advances in Computers*, pp. 65–118, Elsevier, Amsterdam, Netherlands, 1976.
- [7] I. Amaya, J. C. Ortiz-Bayliss, S. Conant-Pablos, and H. Terashima-Marin, "Hyper-heuristics reversed: learning to combine solvers by evolving instances," in *Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC)*, December 2019.
- [8] H. Song, I. Triguero, and E. Özcan, "A review on the self and dual interactions between machine learning and optimisation," *Progress in Artificial Intelligence*, vol. 8, no. 2, pp. 143–165, 2019.
- [9] J. A. Montesino-Guerra, H. Puga, J. M. Carpio, M. Ornelas-Rodríguez, A. Rojas-Domínguez, and L. Ortiz-Aguilar, "Combinatorial designs on constraint satisfaction problem (VRP)," in *Intuitionistic and Type-2 Fuzzy Logic*

- Enhancements in Neural and Optimization Algorithms: Theory and Applications*, pp. 509–526, Springer, Cham, Switzerland, 2020.
- [10] L. D. M. Ortiz-Aguilar, M. Carpio, H. Puga et al., “Increase methodology of design of course timetabling problem for students, classrooms, and teachers,” in *Nature-Inspired Design of Hybrid Intelligent Systems*, pp. 713–728, Springer, Berlin, Germany, 2017.
 - [11] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.
 - [12] K. Helsgaun, “General k-opt submoves for the Lin–Kernighan TSP heuristic,” *Mathematical Programming Computation*, vol. 1, no. 2-3, pp. 119–163, 2009.
 - [13] A. Blazinskas and A. Misevicius, “Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem,” in *Proceedings of the 17th International Conference on Information and Software Technologies*, pp. 27–29, Kaunas, Lithuania, April 2011.
 - [14] S. Minton, M. D. Johnston, A. B. Philips et al., “Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems,” *Artificial Intelligence*, vol. 58, no. 1–3, pp. 161–205, 1992.
 - [15] B. S. Baker, “A new proof for the first-fit decreasing bin-packing algorithm,” *Journal of Algorithms*, vol. 6, no. 1, pp. 49–70, 1985.
 - [16] J. Csirik, “The parametric behavior of the first-fit decreasing bin packing algorithm,” *Journal of Algorithms*, vol. 15, no. 1, pp. 1–28, 1993.
 - [17] J. A. Soria-Alcaraz, G. Ochoa, M. Carpio et al., “Effective learning hyper-heuristics for the course timetabling problem,” *European Journal of Operational Research*, vol. 238, no. 1, pp. 77–86, 2014.
 - [18] J. A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall, and M. Carpio, “Iterated local search using an add and delete hyper-heuristic for university course timetabling,” *Applied Soft Computing*, vol. 40, pp. 581–593, 2016.
 - [19] W. B. Yates and E. C. Keedwell, “Offline learning for selection hyper-heuristics with Elman networks,” in *Lecture Notes in Computer Science*, pp. 217–230, Springer International Publishing, Berlin, Germany, 2018.
 - [20] J. A. Soria-Alcaraz, G. Ochoa, M. A. Sotelo-Figeroa, and E. K. Burke, “A methodology for determining an effective subset of heuristics in selection hyper-heuristics,” *European Journal of Operational Research*, vol. 260, no. 3, pp. 972–983, 2017.
 - [21] C. Martin, P. Héctor, T. M. Hugo et al., “Methodology of design: a novel generic approach applied to the course timetabling problem,” in *Soft Computing Applications in Optimization, Control, and Recognition*, pp. 287–319, Springer, Berlin, Germany, 2013.
 - [22] P. Brazdil, C. Giraud-Carrier, C. Soares et al., *Metalearning*, Springer, Berlin, Germany, 2009.
 - [23] A. E. Gutierrez-Rodríguez, S. E. Conant-Pablos, J. C. Ortiz-Bayliss, and H. Terashima-Marin, “Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning,” *Expert Systems with Applications*, vol. 118, pp. 470–481, 2019.
 - [24] R. Karp, *Complexity of Computer Computations*, Springer US, Boston, MA, USA, 1972.
 - [25] A. Kosowski and K. Manuszewski, “Classical coloring of graphs,” *Graph Colorings*, vol. 352, pp. 1–19, 2004.
 - [26] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.
 - [27] E. Alba and B. Dorronsoro, “Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm,” *Information Processing Letters*, vol. 98, no. 6, pp. 225–230, 2006.
 - [28] J. A. Soria-Alcaraz, M. Carpio, P. Hector et al., “Comparison of metaheuristic algorithms with a methodology of design for the evaluation of hard constraints over the course timetabling problem,” *Studies in Computational Intelligence*, Vol. 451, Springer, Berlin, Germany, 2013.
 - [29] M. Carpio, “Modelo integral de asignación optima de carga academica usando un algoritmo heurístico,” in *Actas de la Encuentro de Investigación en Ingeniería Eléctrica*, Zacatecas, Mexico, Febrero 2006.
 - [30] L. D. M. Ortiz Aguilar, “Diseño de horarios de alumnos y maestros mediante técnicas de soft computing, para una institución educativa,” Master’s thesis, Instituto Tecnológico de León, León, Mexico, 2016.
 - [31] E. Alpaydin, *Introduction to Machine Learning*, The MIT Press, Cambridge, MA, USA, 3rd edition, 2014.
 - [32] P. Brazdil and C. Giraud-Carrier, “Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue,” *Machine Learning*, vol. 107, no. 1, pp. 1–14, 2018.
 - [33] J. Sakarovitch, “Kleene’s theorem revisited,” in *International Meeting of Young Computer Scientists*, pp. 39–50, Springer, Berlin, Germany, 1986.
 - [34] R. J. Campello, D. Moulavi, and J. Sander, “Density-based clustering based on hierarchical density estimates,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 160–172, Springer, Berlin, Germany, 2013.
 - [35] D. W. Scott, “Sturges’ rule,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 3, pp. 303–306, 2009.
 - [36] J. Jokinen, T. Raty, and T. Lintonen, “Clustering structure analysis in time-series data with density-based clusterability measure,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1332–1343, 2019.
 - [37] X. Xu, J. Li, M. Zhou, J. Xu, and J. Cao, “Accelerated two-stage particle swarm optimization for clustering not-well-separated data,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4212–4223, 2020.
 - [38] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
 - [39] D. de Werra, “Heuristics for graph coloring,” in *Computational Graph Theory*, pp. 191–208, Springer, Berlin, Germany, 1990.
 - [40] H. Derbel, B. Jarboui, and R. Bhiri, “A skewed general variable neighbourhood search algorithm with fixed threshold for the heterogeneous fleet vehicle routing problem,” *Annals of Operations Research*, vol. 272, no. 1-2, pp. 243–272, 2019.
 - [41] M. M. Solomon, “On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints,” *Networks*, vol. 16, no. 2, pp. 161–174, 1986.
 - [42] S. S. Habashi, C. Salama, A. H. Yousef et al., “Adaptive diversifying hyper-heuristic based approach for timetabling problems,” in *Proceedings of the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 259–266, IEEE, Vancouver, Canada, November 2018.
 - [43] S. Abdullah, E. K. Burke, and B. McCollum, “Using a randomised iterative improvement algorithm with composite

- neighbourhood structures for the university course timetabling problem,” in *Metaheuristics, Operations Research Computer Science Interfaces Series*, vol. 39, pp. 153–169, Springer US, Boston, MA, USA, 2007.
- [44] I. P. Gent, M. Ewan, P. Patrick et al., “An empirical study of dynamic variable ordering heuristics for the CSP,” in *Principles and Practice of Constraint Programming CP96, Lecture Notes in Computer Science*, E. Freuder, Ed., vol. 1118, pp. 179–193, Springer, Berlin, Germany, 1996.
 - [45] D. D. Lewis, “Naive (bayes) at forty: the independence assumption in information retrieval,” in *Machine Learning: ECML-98*, pp. 4–15, Springer, Berlin, Germany, 1998.
 - [46] H. Lourenço, O. Martin, and T. Stützle, “Iterated local search,” in *Handbook of Metaheuristics, International Series in Operations Research & Management Science*, F. Glover, G. Kochenberger, and F. S. Hillier, Eds., vol. 57, pp. 320–353, Springer, New York, NY, USA, 2003.
 - [47] T. El-Ghazali, *Metaheuristics: From Design to Implementation*, Wiley Publishing, Hoboken, NJ, USA, 2009.
 - [48] F. T. Leighton, “A graph coloring algorithm for large scheduling problems,” *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–506, 1979.
 - [49] P. Augerat, J. M. Belenguer, E. Benavent et al., “Computational results with a branch and cut code for the capacitated vehicle routing problem,” 1995.
 - [50] N. Christofides, “The vehicle routing problem,” in *Combined Optimization*, N. Christofides, A. Mingozzi, P. Toth et al., Eds., Wiley, New York, NY, USA, 1979.
 - [51] B. L. Golden, E. A. Wasil, J. P. Kelly et al., “The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results,” in *Fleet Management and Logistics*, pp. 33–56, Springer, Berlin, Germany, 1998.
 - [52] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, “New benchmark instances for the capacitated vehicle routing problem,” *European Journal of Operational Research*, vol. 257, no. 3, pp. 845–858, 2017.
 - [53] Z. Hanusz, J. Tarasinska, and W. Zielinski, “Shapiro-Wilk test with known mean,” *REVSTAT-Statistical Journal*, vol. 14, no. 1, pp. 89–100, 2016.