





Research Article

A Methodology for Classifying Search Operators as Intensification or Diversification Heuristics

Jorge A. Soria-Alcaraz ¹, Gabriela Ochoa,² Andres Espinal ¹,
Marco A. Sotelo-Figueroa ¹, Manuel Ornelas-Rodriguez,³
and Horacio Rostro-Gonzalez ⁴

¹Department of Organizational Studies, University of Guanajuato, Guanajuato, Mexico

²University of Stirling, Stirling, UK

³Tecnológico Nacional De México, Instituto Tecnológico de León, Guanajuato, Mexico

⁴Department of Electronics Engineering, University de Guanajuato, Guanajuato 36885, Mexico

Correspondence should be addressed to Jorge A. Soria-Alcaraz; jorge.soria@ugto.mx

Received 11 June 2019; Revised 18 October 2019; Accepted 13 December 2019; Published 13 February 2020

Guest Editor: Francisco G. Montoya

Copyright © 2020 Jorge A. Soria-Alcaraz 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.

Selection hyper-heuristics are generic search tools that dynamically choose, from a given pool, the most promising operator (low-level heuristic) to apply at each iteration of the search process. The performance of these methods depends on the quality of the heuristic pool. Two types of heuristics can be part of the pool: diversification heuristics, which help to escape from local optima, and intensification heuristics, which effectively exploit promising regions in the vicinity of good solutions. An effective search strategy needs a balance between these two strategies. However, it is not straightforward to categorize an operator as intensification or diversification heuristic on complex domains. Therefore, we propose an automated methodology to do this classification. This brings methodological rigor to the configuration of an iterated local search hyper-heuristic featuring diversification and intensification stages. The methodology considers the empirical ranking of the heuristics based on an estimation of their capacity to either diversify or intensify the search. We incorporate the proposed approach into a state-of-the-art hyper-heuristic solving two domains: course timetabling and vehicle routing. Our results indicate improved performance, including new best-known solutions for the course timetabling problem.

1. Introduction

Hyper-heuristics are powerful tools for solving complex optimization problems [1–3]. The goal is to reduce the role of the human expert by means of more generally applicable search methodologies. Selection hyper-heuristics are high-level strategies that autonomously choose at run time the best-suited heuristic to apply at each step of the search process. A pool of heuristics to select from should be provided. There are, however, no clear guidelines in the literature about how to construct a potentially successful pool of heuristics [4]. It is well known that successful heuristic search methods should have a dynamic balance between *diversification* and *intensification* [5, 6]. Diversification refers to exploring new promising areas

of the search space, whereas intensification refers to focusing the search by exploiting the area nearby current good solutions. Move operators can thus be either predominantly diversifying or intensifying. However, for complex domains and solution representations, it is not straightforward to categorize a given operator as belonging to one of these two categories.

We consider an iterated local search hyper-heuristic for testing our proposed methodology. Iterated local search (Section 3.3) is a simple yet powerful search strategy [7]; it works by iteratively alternating between a diversification stage and an intensification stage. This algorithmic template makes it clear where to implement move operators of each category. Several variants of iterated local search hyper-heuristics have been proposed with good results [8, 9]. Soria-Alcaraz et al. [10]

proposed a methodology for determining the best subset of heuristics from a given pool, but they considered that all operators were intensification heuristics and the diversification stage contained a single fixed heuristic.

This article proposes an empirical methodology to classify a given operator as an intensification or diversification heuristic. The aim is to produce an effective operator pool in an iterated local search hyper-heuristic framework. The idea is to automatically divide the given complete set of operators into two groups: diversification and intensification. Then, these groups will be assigned to the respective stages of the hyper-heuristic framework. To the best of our knowledge, this is the first time such task is addressed in the literature. Our proposal is to have a preprocessing step where for a given problem domain, each heuristic in the pool is tested by means of a simple and fast probing technique, namely, a random walk with the given move operator. Based on this probing tool and statistical techniques, a set of measurements are collected indicating the effectiveness of each operator to either diversify or intensify the search.

The proposed methodology is tested within a state-of-the-art iterated local search hyper-heuristic [10] on two problem domains with different representations, namely, course timetabling and vehicle routing. Our results indicate improved performance, including new best-known solutions for the course timetabling problem.

The next section introduces relevant concepts and algorithms. The proposed methodology is described in Section 3, detailing the distance metrics, ranking criteria, and high-level search method used. Sections 4 and 5 report the empirical setup, results and analysis for two selected case studies, course timetabling, and vehicle routing, respectively. Finally, Section 6 summarizes our findings and gives suggestions for future work.

2. Background

2.1. Low-Level Heuristics. Heuristics are simple problem-solving techniques or “rules-of-thumb” that aim to produce good enough solutions in a reasonable time. This study considers perturbative heuristics also called move operators. Within a hyper-heuristic framework, these perturbative heuristics are called *low-level*, as they operate directly on the candidate solutions of the underlying optimization problem, leaving the higher-level decisions, such as which heuristic to apply next, to another mechanism. In this context, the low-level heuristics are below a *domain barrier* allowing a higher-level strategy to operate on a wide range of problems. Low-level heuristics are thus simpler procedures that perform changes in the incumbent solution and inform a higher-level mechanism about their performance. Within a hyper-heuristic framework, not all move operators have the same role, some operators are aimed at intensifying the search around the incumbent solution, while others at exploring new regions of the search space with potential better solutions.

2.2. Fitness Landscape Probing. Fitness landscapes constitute a widely used metaphor to describe the dynamics of search and optimization algorithms. Formally, a fitness landscape

[11] is a triplet (S, N, f) , where S is a set of potential solutions, i.e., a search space, $N: S \rightarrow 2^S$, a neighborhood structure, is a function that assigns to every $s \in S$ a set of neighbors $N(s)$, and $f: S \rightarrow R$ is a fitness function that can be pictured as the *height* of the corresponding solutions. When several move operators are considered (as is the case of hyper-heuristics), each of them will induce a different fitness landscape. A common technique to gather fitness landscape’s data is to conduct *random walks*. Formally, a random walk is a sequence of solutions (x_1, x_2, \dots, x_m) where $x_{i+1} \in N(x_i)$ with equiprobability on $N(x_i)$.

Algorithm 1 outlines the random-walk probing technique we used to quantify the behavior of operators. The stopping condition is based on a fixed number of objective function calls. In line 3, a given low-level heuristic (move operator) is used to modify the incumbent solution, and after the stopping condition is reached, a function named *metric* is applied to compute a distance metric between the initial solution and the final solution. The random-walk algorithm (Algorithm 1) is used to estimate the extent to which an operator changes the state of a solution. Several distance metrics (described in Section 3.1) are computed between the initial and the final solution of the walk and aggregated across several walks. The intuition is that the larger these measurements are, the more an operator is diversifying. These metrics are used both to rank the heuristics from the less to more diversifying and to identify, by means of post hoc statistical tests, if it is possible to separate the heuristics into two sets.

2.3. Hyper-Heuristics. Hyper-heuristics, initially conceptualized as *heuristics to choose heuristics* in [1], can be seen as methodologies that reduce the need for a human expert in designing effective solution schemes and, consequently, raise the level of generality at which search methodologies can operate. A recent definition considers a hyper-heuristic as “*automated methodologies for selecting or generating heuristics to solve computational search problems*” [3]. Two types of hyper-heuristics have been studied in the literature:

- (i) Heuristic selection: methodologies for choosing or selecting existing heuristics
- (ii) Heuristic generation: methodologies for generating new heuristics from given components

This study focuses on the first type, i.e., selection hyper-heuristics. Figure 1 illustrates the traditional framework for selective hyper-heuristics, with the *domain barrier* insulating the high-level search strategy from the underlying problem domain. The framework requires a pool of low-level heuristics from which the high-level strategy selects and applies to the incumbent solution. It is worth mentioning, however, that low-level heuristics usually encapsulate domain-specific information.

When a hyper-heuristic uses feedback from the search process to rank the performance of the pool of heuristics, it can be considered as a learning algorithm. According to the source of the feedback during learning, we can distinguish between *online* and *offline* learning hyper-heuristics [2].

Require: *metric*: $Solution \times Solution \rightarrow \mathbb{R}$, *InitialSolution*: a given solution to work with, h_i : Heuristic or operator.

- (1) $s \leftarrow InitialSolution$
- (2) **while** !*StopCriteria*() **do**
- (3) $s \leftarrow apply(h_i, s)$
- (4) **end while**
- (5) **return** *metric*(*initialSolution*, s)

ALGORITHM 1: Random-walk algorithm.

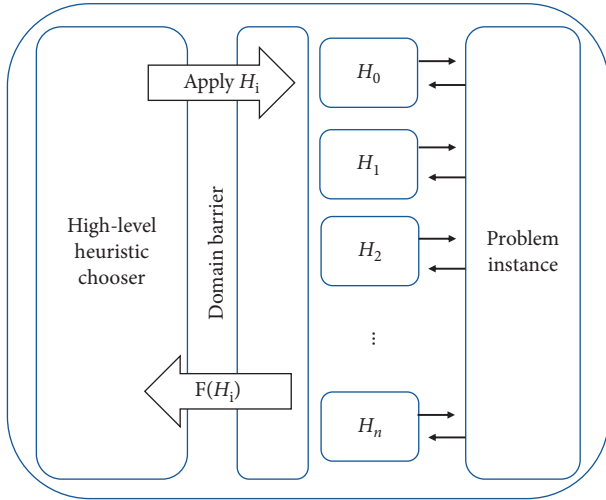


FIGURE 1: General framework of a selection hyper-heuristic based on [1].

Online learning or adaptation takes place on-the-fly when the algorithm is solving a problem instance, while offline learning requires a training process a priori of the execution of the hyper-heuristic. In this work, we use online learning through *adaptive operator selection*.

3. Methodology

Once a pool of low-level heuristics is selected for the problem domain under consideration, our approach studies the behavior of each heuristic separately. This is done by conducting several runs of the random-walk algorithm (Algorithm 1) with each operator from a fixed set of initial solutions generated uniformly at random. These runs produce a set of measurements that are used to rank operators from the less perturbative to most perturbative. The operators producing low distance measurements will be ranked top, while those producing high distance measurement will be ranked at the bottom. The top-ranked operators are categorized as intensification operators as they are the less perturbative.

Section 3.1 describes the distance metrics used to measure the behavior of heuristics. We considered two genotype-based metrics and one fitness-based metric. The genotype-based metrics measure the solution differences in representation space, while the fitness-based metric measures the solution difference in fitness value.

Once the measurements are taken, the operators are ranked using nonparametric ranking techniques. We use

three ranking statistical methods: Friedman, aligned Friedman, and Quade, detailed in Section 3.2, following the guidelines in [12]. Thereafter, we apply post hoc procedures in order to check for statistical significant differences between pairs of operator rankings. We identify those heuristics that do not show a statistical difference in their ranking. Thereafter, we separate heuristics into two groups: intensification heuristics and diversification heuristics. Finally, we execute the high-level iterated local search hyper-heuristic using the two groups of heuristics identified.

The sections below describe in more detail the components and steps of our proposed methodology.

3.1. Distance Metrics. We consider three distance metrics; the first two operate at the solution (genotype) level, while the last metric simply calculates the fitness (objective function) difference.

Hamming distance: it measures the number of substitutions required to change one string into the other [13]. Formally, it is a metric on \mathfrak{R}^n on two strings x_1, x_2, \dots, x_n and y_1, y_2, \dots, y_n of length n over a q -ary alphabet $0, 1, \dots, q-1$ described in the following equation:

$$|\{i: 1 \leq i \leq n, x_i \neq y_i\}|. \quad (1)$$

Lee distance: given two strings x_1, x_2, \dots, x_n and y_1, y_2, \dots, y_n of length n over a q -ary alphabet $0, 1, \dots, q-1$ of size $q \geq 2$ [14, 15]; this distance is defined as

$$\sum_{i=1}^n \min(|x_i - y_i|, q - |x_i - y_i|). \quad (2)$$

Fitness distance: it measures the difference in the fitness value between two given solutions using the absolute value operator. In our study, this metric evaluates how much the fitness of an initial solution is affected by a given heuristic. Equation (3) defines this metric. Given two strings $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ of length n over a q -ary alphabet $0, 1, \dots, q-1$,

$$|\text{Fit}(x) - \text{Fit}(y)|, \quad (3)$$

where $\text{Fit: String} \rightarrow \mathfrak{R}$. This function is domain-dependent and corresponds to a quality measurement of solutions, the objective of the optimization process.

We selected the two distance metrics at the solution level as they are well known in the study of meta-heuristics and offer a simple yet descriptive approach to measure the variation induced by the different operators. The Hamming distance measures the number of raw differences between two given solutions. For example, let us consider two integer-based strings, “5623” and “5827”; the Hamming distance between these strings is 2 since they differ in two locations. The Lee distance is more sophisticated since it reports not only how many differences are there between two given strings but also how large these differences are. Consider the same two strings, “5623” and “5827”; the Lee distance between them is 6 because $|6 - 8| + |3 - 7| = 2 + 4 = 6$. For binary strings, the Hamming and Lee distances produce the same values. In our study, solutions are represented as integer-based strings. The third metric gauges the solution differences in terms of quality or fitness, this is complementary to two genotype-based metrics.

The stopping condition in this phase is fixed to 200 function calls. On each test instance, the random-walk algorithm is run 500 times from different initial randomly generated solutions. This produces 500 distance values for each metric, and the average distance values are calculated. Finally, these average values are used to rank the heuristics from less to more perturbative in the next phase.

3.2. Ranking Heuristics with Nonparametric Statistics.

Once the distance metrics are gathered, we apply statistical tests in order to rank the heuristics. Parametric tests are sometimes used when contrasting the performance of stochastic algorithms. However, they assume independence, normality, and homoscedasticity of the data, which are not guaranteed in the case of low-level heuristics. In such cases, nonparametric statistics overcome this limitation. We used CONTROLTEST and MULTITEST [12] which are specially designed for nonparametric comparison among heuristic algorithms and considered the three tests proposed: Friedman, aligned Friedman, and Quade. The Friedman test uses mainly arithmetic mean, alignment Friedman uses a value of location computed as the average performance achieved by all heuristics in each problem, and Quade considers that some instances might be more difficult than others. All these tests consider ranks; therefore, the lower distance value will produce a higher rank. In our context, this means that heuristics with less perturbative behavior (i.e., intensification heuristics) will be ranking top.

We applied post hoc tests in order to determine if two operators are similar in terms of their perturbative behavior. A p value under 0.05 (0.95 confidence level) rejects this null hypothesis, indicating that the given pair of operators are significantly different in their perturbative behavior. Therefore, those operators are not grouped together. Operator pairs for which the p value is greater than 0.05 are classified in the same group. We applied the Holm procedure to adjust the p values. In our experiments, only two

well-defined groups were detected by the post hoc tests. We defined the group including the top-ranking heuristics as the intensification group and the other group as the diversification group.

3.3. Iterated Local Search Hyper-Heuristic. In order to test the performance of the previously defined intensification and diversification groups of heuristics, we follow the experimental setup in [10] with some adjustments detailed in this section. We also provide a description of the high-level strategy and adaptive operator selection mechanism used. The pool of heuristics is problem specific, and they will be described in the respective case study section.

3.3.1. High-Level Strategy. We utilize iterated local search (ILS) as the high-level strategy [8, 10, 16]. Iterated local search is a simple yet effective strategy, which works by iteratively alternating between an exploration move (diversification) and an exploitation move (intensification) from the perturbed solution [7]. With this search strategy, it is straightforward to identify where to apply the intensification and diversification heuristics groups.

Our implementation is outlined in Algorithm 2. Two independent adaptive operator selection steps are used: one in the local search phase (lines 2 and 5) using the intensification group of heuristics and another in the perturbation phase (line 4) using the diversification group. This implementation differs from our previous ILS hyper-heuristic [9, 10] in which adaptive operator selection is used on the two algorithm stages instead of only on the intensification stage. Line 6 must be set for maximization ($>$) or minimization ($<$) problems.

3.3.2. Adaptive Operator Selection. Adaptive operator selection [17, 18] allows high-level algorithms to autonomously select the next heuristic to apply to the incumbent solution. Two cooperating mechanisms are required: *selection rule*, which defines how to select the next operator or low-level heuristic from the pool according to their estimated qualities, and *credit assignment*, which defines how to estimate the operators’ quality based on the impact brought by their recent application. The mechanisms we implemented are described in detail:

Selection rule: we use dynamic multiarmed bandits (DMAB) [19] as the selection rule, where each operator is viewed as an arm. Let $l_{i,t}$ denote the number of times the i^{th} arm (heuristic) has been played and $\hat{r}_{i,t}$ the average reward it has received up to time t . At each time step t , from K alternative arms (heuristics), the algorithm selects the arm maximizing the quantity computed by the following expression:

$$\hat{r}_{j,t} + C \sqrt{\frac{2 * \log_{10}(\sum_{i=1}^K l_{i,t})}{l_{j,t}}}. \quad (4)$$


```

(1)  $s_0 = \text{GenerateInitialSolution}$ 
(2)  $s^* = \text{LocalSearch}(s_0)$ 
(3) while ! $\text{StopCriteria}()$  do
(4)    $s' = \text{perturbation}(s^*)$ 
(5)    $s^{*'} = \text{LocalSearch}(s')$ 
(6)   if  $f(s^{*'})$  better than  $f(s^*)$  then
(7)      $s^* = s^{*'}$ 
(8)   end if
(9) end while
(10) return  $s^*$ 

```

ALGORITHM 2: High-level strategy: iterated local search.

Factor C is used to balance the inner exploration and exploitation phases. The *DMAB* algorithm uses a *Page-Hinkley* statistical test, where two parameters are introduced: γ_{ph} , which controls the trade-off between false alarms and unnoticed changes, and δ , which enforces the robustness when dealing with slowly varying environments. Parameters C and γ_{ph} need to be tuned for every problem. We found in preliminary experiments that the values $C=10$ and $\gamma_{ph}=100$ obtain encouraging results consistently. For the parameter δ , we used the value suggested in [18] ($\delta=0.15$) for all our experiments.

Credit assignment: we use *extreme value* criteria for determining the operator's credit [17, 18]. When a heuristic op is selected and applied to the current solution by the selection rule, it is necessary to calculate an update in the reward value with the most recent behavior information of the last applied heuristic. Rewards are updated as follows depending on the ILS phase (diversification and intensification). For the *intensification phase*, the fitness of the new solution is computed and the change in fitness Δ_f is added to a FIFO list of size W . For the *diversification phase*, the Hamming distance between the new solution and the initial one is calculated and added to a FIFO list of size W . A separate list is kept for each operator for both phases. FIFO data structure is used to guarantee that only the latest observations in fitness improvement or Hamming distance are considered in operator selection computations for the last W iterations before being erased of credit assignment memory. We kept a list for each operator in order to identify which operator has achieved the best performance in the last W iterations.

Thereafter, the specific operator reward is updated to the maximal value in the list. Formally, let t be the current step and $\text{metric}(t)$ be the metric value (Δ_f or Hamming) estimated at time t for a given heuristic op and the expected reward \hat{r}_t for heuristic op is computed using the following equation:

$$\hat{r}_t = \text{argmax}[\text{metric}(t_i), i = 1, \dots, W]. \quad (5)$$

4. Course Timetabling Case Study

We first apply the proposed methodology to the course timetabling problem. This problem requires the assignment

of a fixed number of subjects into a number of time slots. The objective is to obtain a timetable minimizing the number of conflicts. Our formulation uses a generic modeling approach where solutions are represented as vectors of integer numbers of length equal to the number of events (courses) [9]. As test instances, we use the 24 instances from the *International timetabling competition (ITC) 2007 track 2* (postenrollment course timetabling) [20]. Many meta-heuristic [21, 22] and hyper-heuristic [9, 10] approaches have been proposed for solving variants of educational timetabling. Recent surveys have also been published [4, 23]. Here, we improve the performance of hyper-heuristics by incorporating our automated approach for categorizing low-level heuristics into intensification and diversification groups.

4.1. Low-Level Heuristics. Our implementation considers the following set of five low-level heuristics [10]. They range from a simple randomized exchange or swap neighborhoods to greedy and more informed procedures:

MLC (move to less conflict): it locates the variable producing the most conflicts and changes its value to that causing the minimum possible conflict

BSP (best single perturbation): it chooses a variable following a sequential order and changes its value to that producing the minimum conflict

WMLC (worst move to less conflict): it locates the variable and value that once modified cause the less conflict

MLS (move to less size): it changes the value of a given variable to that causing the event to move to the less occupied time slot

Two points: it selects uniformly at random two indexes in the integer string representation and modifies all variables between the indexes randomly

4.2. Ranking and Grouping of the Low-Level Heuristics. Table 1 shows the nonparametric ranking of the heuristics according to the genotypic and fitness distance metrics. The ranking indicates that closeness in genotypic space correlates with closeness in fitness space, a desirable property for heuristic search methods. Table 2 shows the adjusted p values of each pair of heuristics. The null hypothesis in this test establishes that between two heuristics (row and column), there is no significant difference in terms of perturbative behavior. A p value less than 0.05 (0.95 confidence level) means the null hypothesis is rejected. According to this criterion groups, {WMLC, MLC, BSP} and {MLS, TwoPoints} have a statistical difference. The top-ranking three heuristics {WMLC, MLC, BSP} are assigned to the intensification group, while the bottom two heuristics {MLS, TwoPoints} to the diversification group.

4.3. Hyper-Heuristic Performance Comparison. The selected groups of heuristics are then deployed within the iterated local search hyper-heuristic framework described in Section

TABLE 1: Ranking of the course timetabling heuristics.

Heuristic	Friedman	Aligned Friedman	Quade
Hamming distance			
WMLC	3.18	14.76	1.77
MLC	1.83	12.34	2.30
BSP	2.14	25.76	2.77
MLS	5.9	42.43	3.60
TwoPoints	7.95	51.9	4.2
Lee distance			
WMLC	2.15	11.38	1.47
BSP	1.25	12.724	2.17
MLC	3.37	28.12	2.47
MLS	6.31	47.21	3.12
TwoPoints	7.47	41.6	4.45
Fitness distance			
WMLC	1.56	11.76	1.12
MLC	2.95	15.33	1.77
BSP	3.66	27.92	2.14
MLS	6.72	36.42	2.77
TwoPoints	8.24	46.9	4.10

TABLE 2: Adjusted P values for every pair of heuristics in course timetabling domain.

	TwoPoints	MLS	BSP	MLC
Hamming distance				
WMLC	0.005	0.007	0.052	0.055
MLC	0.005	0.008	0.05	—
BSP	0.006	0.01	—	—
MLS	0.051	—	—	—
Lee distance				
WMLC	0.003	0.032	0.046	0.051
MLC	0.002	0.041	0.003	—
BSP	0.016	0.014	—	—
MLS	0.051	—	—	—
Fitness distance				
WMLC	0.001	0.017	0.048	0.045
MLC	0.002	0.035	0.05	—
BSP	0.003	0.035	—	—
MLS	0.051	—	—	—

3.3. Following the *ITC-2007* competition rules, each hyper-heuristic variant is run 10 times and the stopping condition corresponds to a time limit of about 10 minutes following the benchmark algorithm provided in the competition website (<http://www.cs.qub.ac.uk/itc2007/>). We compare our approach against the following methods:

Cambazard: the winner of the *ICT-2007* competition [24], a multistage local search algorithm considering several neighborhoods

Ceschia: a single-step meta-heuristic approach based on simulated annealing, with a neighborhood composed of moves that reschedule one event or swap two events [21]

AdapExAP: an adaptive iterated local search hyper-heuristics coupled with an adaptive mechanism based on the *adaptive pursuit* selection rule [9]

Goh: Iterative two-stage algorithm that uses tabu search and simulated annealing [25]

Nagata: a local search-based algorithm with a mechanism for adapting the size of search neighborhood [26]

HHADL: an iterated local search hyper-heuristic with Add-Delete list, which generates heuristics based on a fixed number of add and delete operations [27]

HHDMAb: an iterated local search with dynamic multiarm bandits, which selects from a pool of heuristics using an autonomous strategy [10]

The main difference between our proposal and other recent methodologies is the application of a categorization process to a predefined set of low-level heuristics. This categorization, detailed in Section 3, leads us to empirical construction of a reasonable good group of heuristics to use as intensification and diversification operators in a selection hyper-heuristic approach. Our selection hyper-heuristic has an empirically effective group of operators determined a priori of any exhaustive experimentation; this characteristic enhances the performance of our approach against other methodologies whose setting was made by the mere human expertise.

Table 3 shows comparative results of our proposed approach HH2DMAb against state-of-the-art solvers. The evaluation of the best-found solutions is shown, with the average and standard deviation results reported in brackets in the form (\bar{s}_σ) , when available. The best solutions are given in bold font.

Configurations designed by human experts are represented by the other entries in Table 3. Our approach offers an automated operator grouping and selection of the heuristics with no expert knowledge required. Results for instances 3 and 23 are new best-known solutions found by our approach. Consistently, our iterated local search hyper-heuristic with automatic selection of heuristic groups, HH2DMAb, presents lower average and standard deviation values than previous approaches. We argue that this improved performance is because of having additional heuristics at the diversification stage, which gives the algorithm more alternatives to escape from local optima. Figures 2 and 3 show the dynamic of selection probabilities for the intensification group (a) and diversification group (b) of heuristics during a HH2DMAb run on a selected instance (instance 2).

Figures 2 and 3 shows in X-axis the iterations of each run (1000x) and Y-axis shows the probability of selection of each heuristic (color line); the sum of selection probabilities for all heuristics is 1 at each iteration. In the intensification group, Figure 2, the heuristics WMLC and MLC are most frequently selected across the run than the third BSP heuristic and WMLC and MLC take dominance at different stages of the run. The diversification group dynamic, Figure 3, shows that the two heuristics are useful during the search, with heuristic MLS having prominence.

5. Vehicle Routing Case Study

In order to illustrate the generality of the proposed methodology, we considered a second case study, the vehicle

TABLE 3: Course timetabling problem. Comparison of our proposed approach HH2DMAB against state-of-the-art solvers. The evaluation of the best-found solutions are shown, with the average and standard deviation results reported in brackets in the form (\bar{s}_σ) , when available. The best solutions are given in bold font.

Instance	Cambazard	Ceschia	AdapExAP	HHADL	HHDMAAB	Goh	Nagata	HH2DMAB
1	571	59	650 (780.45 _{148.5})	630	630 (860.12 _{110.7})	307.6	81.7	630 (850.06 _{98.45})
2	993	0	470 (960.7 _{270.4})	450	380 (530.15 _{130.6})	63.4	48.0	365 (505.74 _{98.64})
3	164	148	290 (337 _{88.7})	300	137 (315.12 _{57.15})	199.4	155	135 (304.10 _{51.27})
4	310	25	600 (815 _{42.6})	602	475 (559.12 _{38.62})	328.8	254	452 (480.61 _{37.11})
5	5	0	35 (39.16 _{9.3})	6	0 (0 ₀)	2.7	0	0 (0 ₀)
6	0	0	20 (29.4 _{7.3})	0	0 (4.6 _{1.4})	33.2	0	0 (0 ₀)
7	6	0	30 (33.74 _{2.1})	0	0 (5.2 _{1.1})	18.0	3.6	0 (0 ₀)
8	0	0	0(0 ₀)	0	0 (0 ₀)	0.0	0.0	0 (0 ₀)
9	1560	0	630 (861.1 _{127.4})	640	602 (854.3 ₁₅₈)	100.7	58.9	595 (630.26 _{75.2})
10	2163	3	2349 (2458.2 _{185.2})	663	482 (615.12 _{72.14})	65.3	6.4	469 (630.57 _{70.46})
11	178	142	350 (405.7 _{57.3})	344	159 (355.12 _{49.12})	244.3	140.4	159 (303.82 _{34.55})
12	146	267	480 (506.4 _{27.4})	198	140 (278.43 _{25.9})	318.2	33	135 (230.63 _{22.46})
13	0	1	46 (77.37 _{49.2})	0	0 (15.2 _{12.7})	99.5	0	0 (3.27 _{1.2})
14	1	0	80 (108.3 _{33.5})	35	20 (25.12 _{17.4})	0.2	0	16 (23.58 _{12.55})
15	0	0	0 (5.75 _{9.4})	0	12 (44.16 _{16.0})	192.0	0	0 (2.04 _{1.2})
16	2	0	0 (2.22 _{4.1})	140	0 (73.76 _{33.4})	105.8	1.5	0 (50.14 _{18.66})
17	0	0	0(0 ₀)	0	0 (3 _{1.5})	0.8	0	0 (0 ₀)
18	0	0	20 (25.16 ₆)	0	0 (5.3 _{2.1})	12.5	0	0 (0 ₀)
19	1824	0	360 (404.51 _{39.1})	400	133 (214.6 _{33.6})	516.7	0	104 (188.2 _{25.44})
20	445	543	150 (177.12 _{37.1})	150	106 (311.07 _{31.5})	650.7	438	106 (230.11 _{15.41})
21	0	5	0 (3.78 _{5.7})	0	0 (2.5 _{2.3})	12.5	0	0 (1.87 _{1.11})
22	29	5	33 (45.71 _{12.7})	32	25 (54 _{18.5})	136.0	0	21 (37.22 _{9.55})
23	238	1292	1007 (1378.45 _{319.4})	238	267.4 (515.23 _{89.4})	504.4	777	233 (404.57 _{85.66})
24	21	0	0 (45.88 _{60.0})	640	76 (337.34 _{51.9})	192.6	0	77 (199.62 _{40.11})

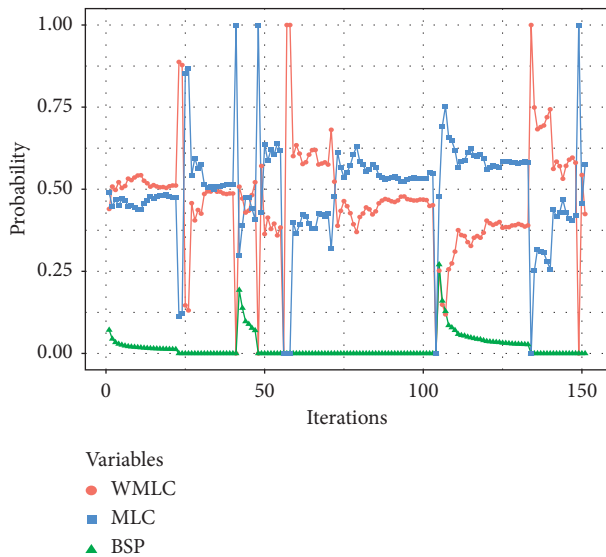


FIGURE 2: Intensification heuristics.

routing problem with time windows (VRPTW). In this problem, a set of customer demands must be addressed using as few vehicles as possible. The time window constraint indicates that customer demands can only be served in a time window. The formulation and experimental setting follows the rules of the *Cross-Domain Heuristic Search Competition (CHeSC) 2011* [28]. CHeSC instances were taken from [29] and include 5 from the Solomon data set and 5 from the Gehring and Homberger data set.

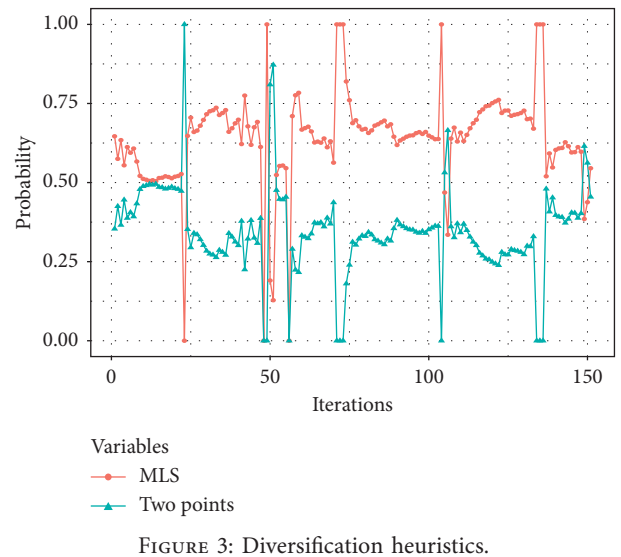


FIGURE 3: Diversification heuristics.

5.1. *Low-Level Heuristics.* Our implementation uses the following four heuristics:

TimeRR: it removes a number of locations based on time window proximity, reinserting into the best route possible

TwoOptStar: it takes the end sections of two routes and swaps them to create two new routes

locRR: it removes a number of locations based on location proximity, reinserting into the best route possible

ShiftMutate: it moves a single location from one route to another

5.2. Ranking and Grouping of the Low-Level Heuristics.

The experimental design resembles that of the previous case study. Each sampling procedure is executed 500 times, using 200 function calls for every instance-heuristic pair. The time expended in this phase was about 37 minutes using an i7 Intel Core, 8 Gb in Ram, Linux Operating System and JAVA Language. Table 4 shows the collected metric rankings.

As Table 4 indicates, heuristics locRR and TimeRR are the top two according to all metrics. This suggests they are more suitable as intensification heuristics. Heuristics TwoOpt* and Shift are the bottom two in the rank for all metrics. Table 5 shows the post hoc tests for heuristic pairs as the null hypothesis establishes that between two heuristics (row and column), there is no significant difference in terms of perturbative behavior. A p value less than 0.05 means the null hypothesis is rejected. According to this evidence, the groups {locRR, TimeRR} (intensification) and {TwoOpt*, Shift} (diversification) are defined.

5.3. Hyper-Heuristic Performance Comparison.

The selected groups of heuristics are deployed within the iterated local search hyper-heuristic framework. In order to compare its performance, we consider the contestants in the *Cross-Domain Heuristic Search Competition (CHeSC) 2011* (<http://www.asap.cs.nott.ac.uk/external/chesc2011/results.html>). Following the competition rules, 31 runs are conducted, each lasting 600 CPU seconds according to the benchmark tool provided by the competition webpage. The algorithms are ranked according to their median performance and receive points according to a system inspired by formula (1) in [28]. The top eight performing algorithms receive 10, 8, 6, 5, 4, 3, 2, and 1 point, respectively. In case of ties, the points of the concerned positions are summed and equally shared.

According to this scoring system, our method HH2DMAB achieves 28 points (Table 6), which represents a tie in the 2nd position when compared against CHeSC contestants [30]. Moreover, HH2DMAB achieves better performance when compared with or previous state-of-the-art HHDMA B [10] hyper-heuristic, which obtained the 3rd position. Again, the evidence suggests that our current approach with more heuristics at the diversification stage allows the hyper-heuristic to achieve better results.

Our selection hyper-heuristic was enhanced by the application of *a priori* phase were the diversification and intensification operators were selected using the methodology detailed in Section 3. This allowed our approach to work with an empirically good-selected group of operators. Table 7 contrasts the best results obtained by HH2DMAB against the previous HHDMA B hyper-heuristic [10] and the best-known results for each instance. Consistently, HH2DMAB outperforms the previous HHDMA B and produces results that are close to the best-known solutions. This is a good result since the best-known solutions are achieved by problem-specific algorithms. Our approach, instead, is a more general methodology that usually requires only changing the set of low-level heuristics to address other problem domains. This

TABLE 4: Ranking of the vehicle routing heuristics.

Heuristic	Friedman	Aligned Friedman	Quade
Hamming distance			
locRR	2.33	13.12	2.10
TimeRR	1.62	16.24	2.40
TwoOpt*	2.74	22.14	3.66
Shift	3.14	29.89	4.50
Lee distance			
locRR	2.37	12.33	1.91
TimeRR	1.61	15.44	2.10
TwoOpt*	3.02	11.16	2.60
Shift	5.01	27.94	3.47
Fitness distance			
TimeRR	1.36	13.19	1.01
locRR	2.65	11.95	2.37
Shift	2.42	20.42	3.15
TwoOpt*	5.22	27.14	4.18

TABLE 5: Adjusted P values for every pair of heuristics in vehicle routing domain.

	Shift	TwoOpt*	TimeRR
Hamming distance			
locRR	0.0027	0.022	0.052
TimeRR	0.028	0.025	—
TwoOpt*	0.63	—	—
Lee distance			
locRR	0.017	0.020	0.061
TimeRR	0.021	0.007	—
TwoOpt*	0.077	—	—
Fitness distance			
locRR	0.031	0.016	0.057
TimeRR	0.024	0.020	—
TwoOpt*	0.075	—	—

TABLE 6: Comparison with CHeSC 2011 contestants.

Rank	Algorithm	Score
1	PHUNTER	33
2	HAEA	28
2	HH2DMAB	28
3	HHDMA B	25
4	KSATS-HH	23
5	ML	22
6	AdapHH	16
6	HAHA	16
8	EPH	12
9	AVEG-Nep	10
10	GISS	6
10	GenHive	6
10	VNS-TW	6
13	ISEA	5
13	XCJ	5
15	SA-ILS	5
16	ACO-HH	2
17	DynILS	1
18	NA-SLS	0
18	SelfSearch	0
18	Ant-Q	0
18	MCHH-S	0

TABLE 7: Comparison with best-known solutions as reported in SINTEF website.

Instance	HHDMAB		HH2DMAB		Best known	
RC207	4	1094.14	3	1075.67	3	1061.14
R101	19	1655.98	19	1650.95	19	1650.80
RC103	12	1395.01	11	1305.48	11	1261.67
R201	4	1275.30	4	1255.46	4	1252.37
R106	13	1296.86	12	1467.28	11	1424.73
C1-10-1	107	43956.7	102	426843.4	100	424478.9
RC2-10-1	22	31163.6	21	30291.5	20	30278.5
R1-10-1	101	55345.7	100	54075.4	100	53501.3
C1-10-8	113	49366.4	103	46133.9	92	44092.7
RC1-10-5	98	49154.5	94	46327.2	90	45564.8

evidence suggests that our algorithm setting was more efficient, producing competitive results against other methodologies configured only by human expertise.

6. Conclusions

We have proposed an empirical methodology to classify a set of operators into intensification and diversification heuristics, to be used within hyper-heuristic methods. Starting from a suitable set of low-level heuristics or search operators for a given domain, the proposed methodology probes their performance using fitness landscape and distance metrics and ranks the heuristics through non-parametric statistics instead of human expertise. This contributes to increasing the methodological rigor and automation for deploying hyper-heuristic approaches. Our methodology was tested within a state-of-the-art hyper-heuristic framework over two complex combinatorial optimization problems, namely, course timetabling and vehicle routing problem with time windows achieving new best-known solutions for ITC 2007 track 2 course timetabling instances and better results against previous studies in the case of CHESC vehicle routing instances. Our results indicate improved hyper-heuristic performance on both domains when our methodology is used to empirically identify a group of intensification and diversification operators. This suggests that well-designed hyper-heuristic methods are not only more general but can also be more effective than problem-specific meta-heuristics. In the future, we will investigate additional ranking methods and methodologies to automatically tune hyper-heuristic parameters. Finally, it is necessary to test this approach to other problem domains and heuristic pools.

Data Availability

The result data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors wish to thank the Consejo Nacional de Ciencia y Tecnología (CONACyT) for the support in the project Neurociencia Computacional: de la teoría al desarrollo de sistemas neuromórficos, N. 1961 and the University of Guanajuato. Special thanks to University of Stirling.

References

- [1] P. Cowling, G. Kendall, and E. Soubeiga, *A Hyperheuristic Approach to Scheduling a Sales Summit*, Springer, Berlin, Germany, 2001.
- [2] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, *A Classification of Hyper-Heuristic Approaches*, Springer, Boston, MA, USA, 2010.
- [3] E. K. Burke, M. Gendreau, M. Hyde et al., “Hyper-heuristics: a survey of the state of the art,” *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.
- [4] N. Pillay, “A review of hyper-heuristics for educational timetabling,” *Annals of Operations Research*, vol. 239, no. 1, pp. 3–38, 2016.
- [5] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization,” *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.
- [6] X. S. Yang, S. Deb, and S. Fong, “Metaheuristic algorithms: optimal balance of intensification and diversification,” *Applied Mathematics & Information Sciences*, vol. 8, no. 3, pp. 977–983, 2014.
- [7] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search*, Springer, Boston, MA, USA, 2003.
- [8] G. Ochoa, J. Walker, M. Hyde, and T. Curtois, *Adaptive Evolutionary Algorithms and Extensions to the HyFlex Hyper-Heuristic Framework*, Springer, Berlin, Germany, 2012.
- [9] J. A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E. K. Burke, “Effective learning hyper-heuristics for the course timetabling problem,” *European Journal of Operational Research*, vol. 238, no. 1, pp. 77–86, 2014.
- [10] 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.
- [11] P. F. Stadler, *Fitness Landscapes*, Springer, Berlin, Germany, 2002.
- [12] 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.
- [13] R. W. Hamming, “Error detecting and error correcting codes,” *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.
- [14] C. Lee, “Some properties of nonbinary error-correcting codes,” *IEEE Transactions on Information Theory*, vol. 4, no. 2, pp. 77–82, 1958.
- [15] M. M. Deza and E. Deza, *Encyclopedia of Distances*, Springer, Berlin, Germany, 2009.
- [16] J. A. Soria Alcaraz, G. Ochoa, M. Carpio, and H. Puga, “Evolvability metrics in adaptive operator selection,” in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 1327–1334, ACM, New York, NY, USA, 2014.

- [17] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, “Analyzing bandit-based adaptive operator selection mechanisms,” *Annals of Mathematics and Artificial Intelligence*, vol. 60, no. 1-2, pp. 25–64, 2010.
- [18] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, “Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms,” *Lecture Notes in Computer Science*, vol. 3, pp. 176–190, 2009.
- [19] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, “Adaptive operator selection with dynamic multi-armed bandits,” in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pp. 913–920, ACM, New York, NY, USA, 2008.
- [20] R. Lewis, B. Paechter, and B. McCollum, *Post Enrolment Based Course Timetabling: A Description of the Problem Model Used for Track Two of the Second International Timetabling Competition*, Cardiff Business School, Cardiff, Wales, 2007.
- [21] S. Ceschia, L. Di Gaspero, and A. Schaerf, “Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem,” *Computers & Operations Research*, vol. 39, no. 7, pp. 1615–1624, 2012.
- [22] R. Lewis and J. Thompson, “Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem,” *European Journal of Operational Research*, vol. 240, no. 3, pp. 637–648, 2015.
- [23] H. Babaei, J. Karimpour, and A. Hadidi, “A survey of approaches for university course timetabling problem,” *Computers & Industrial Engineering*, vol. 86, pp. 43–59, 2015.
- [24] H. Cambazard, E. Hebrard, B. O’Sullivan, and A. Papadopoulos, “Local search and constraint programming for the post enrolment-based course timetabling problem,” *Annals of Operations Research*, vol. 194, no. 1, pp. 111–135, 2012.
- [25] S. L. Goh, G. Kendall, and N. R. Sabar, “Improved local search approaches to solve the post enrolment course timetabling problem,” *European Journal of Operational Research*, vol. 261, no. 1, pp. 17–29, 2017.
- [26] Y. Nagata, “Random partial neighborhood search for the post-enrollment course timetabling problem,” *Computers & Operations Research*, vol. 90, pp. 84–96, 2018.
- [27] 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.
- [28] G. Ochoa, M. Hyde, T. Curtois et al., *HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search*, Springer, Berlin, Germany, 2012.
- [29] SINTEF, *Transportation Optimization Portal—Vehicle Routing Problem with Time Windows*, SINTEF, Trondheim, Norway, 2008.
- [30] F. Mascia and T. Stützle, *A Non-adaptive Stochastic Local Search Algorithm for the CHeSC 2011 Competition*, Springer, Berlin, Germany, 2012.