A New Methodology for Solving the Vehicle Routing Problem with Time Windows Using Weighted Heuristics

Fernando Isunza Fonseca, Diego Villarreal Gutiérrez,
Santiago Enrique Conant Pablos

Instituto Tecnológico y de Estudios Superiores de Monterrey,
Departamento de Computación, Región Norte,
Mexico

{a01400624, a00820181}@itesm.mx, sconant@tec.mx

Abstract. VRPTW is an optimization problem that has been a focal point for many years by the scientific community, looking to create hyper-heuristics that are increasingly better to apply in real life. In this project, it is proposed a new hyper-heuristic based on probabilities and heuristics weights obtained through practice, after obtaining the ideal parameters in simulated annealing for each type of problem to attack. Starting by dividing and creating a categorization of VRPTW problems based on the number of clients and their distribution in space, and then dividing the problems into sections and giving weight to each heuristic in each section to improve the performance of these hyper-heuristics against other hyper-heuristics known as random, adaptive or probabilistic. It was observed that there is a behavior where this method improves compared to other hyper-heuristics when there are more clients. If it continues to develop, the optimal method for Big Data problems can be obtained.

Keywords: Hyper-heuristics, vehicle routing problem, time windows, heuristics, methodology.

1 Introduction

The Vehicle Routing Problem (VRP) is one of the most studied optimization problems. The generalization of the well-known Traveling Salesman Problem (TSP) and in this problem domain, an optimal route determination process is carried out to send products needed by a group of customers.

The most VRP researches are Vehicle Routing Problem with Time Windows (VRPTW) and Capacitated Vehicle Routing Problems (CVRP) [2]. The VRPTW involves determining an optimal set of routes on identical vehicle fleets with limited capacity to deliver customer demand items within a certain period, without violating the customer's time-window constraints and the vehicle-capacity constraints. This variant is chosen since the importance of VRPTW in many distribution systems has spurred intensive research efforts for both heuristic and exact optimization approaches.

In a real application, many shipping service companies use several similar vehicles, the shipping is divided into several shifts and there is a time limit for each shift. Modeling VRPTW as a distribution system consists of the main depot and some vehicles with the same capacity, to serve many scattered customers. Each customer has a certain time limit, their request is less than the capacity of the vehicle, and each customer is visited once by a single-vehicle. Simulated annealing as an extension of the Markov Chain Monte Carlo algorithm was first presented in 1953 [5]. Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a meta-heuristic to approximate global optimization in a large search space for an optimization problem [3]. It is often used when the search space is discrete (e.g., the VRP) optimization problems [4].

For many real-world problems, an exhaustive search for solutions is not a practical proposition. The search space may be far too big, or there may not even be a convenient way to enumerate the search space [2]. Hyper-heuristics may be seen as a generalization of meta-heuristics and an easier way for categorizing a large body of literature of heuristics and meta-heuristics that was rather difficult to classify previously [3]. A hyper-heuristic is a search methodology or learning mechanism to select or generate heuristics to solve a specific combinatorial problem [1].

The overall flow of algorithms is described in the following: first, an initial solution is constructed using a constructive heuristic. Then, through a learning process, the hyper-heuristic proposes a series of perturbation, constructive, and/or improvement heuristics to improve the solution. One of the main motivations for studying hyper-heuristic approaches is that they should be cheaper to implement and easier to use than problem-specific special-purpose methods and the goal is to produce good quality solutions in this more general framework. Of course, the overall aim of the a hyper-heuristic goal goes beyond meta-heuristic technology.

- Probabilistic hyper-heuristic,
- Random hyper-heuristic,
- Adaptive hyper-heuristic,
- Interroute relocate,
- Intraroute 2opt,
- Intraroute oropt,
- Intraroute relocate,
- Interroute exchange,
- Intraroute exchange,
- Interroute 2opt,
- Cross exchange,
- Geni Exchange.

2 Motivation

The search for optimizing the number of routes and the best distance has led to the discovery of new algorithms and methodologies. The motivation for finding a new methodology to solve this type of problem is the whole improvements that this will

mean in the delivery logistic area. Having the best route has ecological and financial impacts [1]. Using the best route minimizes distance and minimizes CO_2 emissions and, more importantly, minimizes the operational costs [10].

The use of hyper-heuristics to solve optimization problems is very common today. However, it is a path that has not been completed, and finding the best hyper-heuristic for a particular problem is the ideal now. For the VRPTW problem, many hyper-heuristics, meta-heuristics, and heuristics have been studied and very good results have been obtained.

The motivation is to generate a new hyper-heuristic based on probabilities established based on the input variables of the problem and the behavior of various heuristics in the resolution of the optimal final distance in a specific number of iterations. The characteristics that will be taken into account will be the number of clients in the VRPTW and their distribution in the workspace, and the behavior of the individual heuristics Seeking to test the hypothesis that for each classification based on these parameters, a personalized hyper-heuristic would work better than the hyper-heuristics already known as probabilistic, adaptive or random. This could open the doors to new ways of finding better, faster, and more efficient results not only in the VRPTW but also in other optimization problems based on the initial conditions of the problem to be optimized.

3 Problem Description

VRP is a combinatorial optimization problem that responds to the question of the best routes to deliver packages to the company's clients. It is a generalization of the well-known Traveling Salesman Problem. VRP was introduced by George Dantzig and John Ramser in 1959 [7]. It's most general description is that "N" trucks leave a warehouse where the products bought by customers scattered throughout the city are stored. The objective is to minimize the cost of distribution, selecting the trucks that must go with each client on the shortest route [9].

It is complicated when the number of clients, warehouses, and trucks varies; when truck conditions vary, such as capacities; and when there are special orders or clients or time restrictions. In this research, the Solomon databases of 25, 50, 100, and 800 customers will be used. These datasets are well known in other VRPTW researches and many results are discovered using different hyper-heuristics. The datasets have the particularity of being separated according to the spatial distribution of the clients in three categories: grouped agglomerations of clients called *Cluster distribution*, randomly distribution of the clients in the space called *Random distribution*, and something random but conditioned, that is between the other two types called *Random clustered distribution*. There are 175 different datasets with the following variables each one:

1) Number of vehicles,
2) Customer ID,
3) Coordinate x of the customer,
4) Coordinate y of the customer,
5) Demand,

6) Ready time,
7) Due date,
8) Service time.

The principal component of the code used is the Simulated Annealing algorithm with a series of different heuristics. Another methods used are multi-class classifiers using decision trees to divide and categorize the distribution of the customers in the space (*x*,*y*) in three classes: cluster, that are grouped customers; random cluster, that are less grouped customers but not in random distribution; and random distribution. Polynomial regression is also used with *Scikit Learn* python's library to create the sections when working on the problems to construct this new hyper-heuristic.

4 Methods and Procedures

The goal is to create a hyper-heuristic that features weights for each heuristic, something similar to the probabilistic hyper-heuristic, but in this new method, there is the introduction of different sections along the run for different instances. In consequence, each heuristic will have different weights in every section. To achieve this, the first step of our method is to divide the instances based on the number of clients and their distribution in space. Then the sections were chosen by analyzing the behavior of the best distance metric along with the runs, and the calculation of the weights was by analyzing the performance of each heuristic used in each section. With the weights, it was designed as a probability rule to choose randomly a heuristic in every iteration. Each step is presented below.

4.1 Creation of Data-Frame of Independent Variables from Benchmark Problems

The database used includes the following manipulable variables: Number, demand, time windows, service time, and (x,y) coordinates for all the customers and number and capacity of the vehicles. The next calculated fields were created:Sum of all *x* coordinates:

- Sum of all *y* coordinates,
- Mean of all *x* coordinates,
- Mean of all *y* coordinates,
- Distribution of customers (Cluster, Random Cluster, or Random Distribution).

Solomon's datasets are divided into types of distribution. This research verifies and creates a new hyper-heuristic method, based on the distribution of the clients in every problem. In statistical hypothesis testing, a two-sample test is an analysis performed on the data of two random samples, each independently got from a different given population. The test aims to determine whether the difference between these two populations is statistically significant or not. The first hypothesis proposed is:

H0: The data is equal, regardless of distribution and number of clients.

Table 1. Comparison between two groups of values for the final length, when the first letter of every type is the type of distribution (cluster, random cluster, and random distribution) and the number is of the customers. α=0.05.

No. Test	Type 1	Type 2	p-value
1	C25	RC25	<0.001
2	C25	R25	<0.001
3	RC25	R25	<0.001
4	C50	RC50	<0.001
5	C50	R50	<0.001
6	RC50	RC50	<0.001
7	C100	RC100	<0.001
8	C100	R100	<0.001
9	RC100	R100	<0.001
10	C800	R800	<0.001

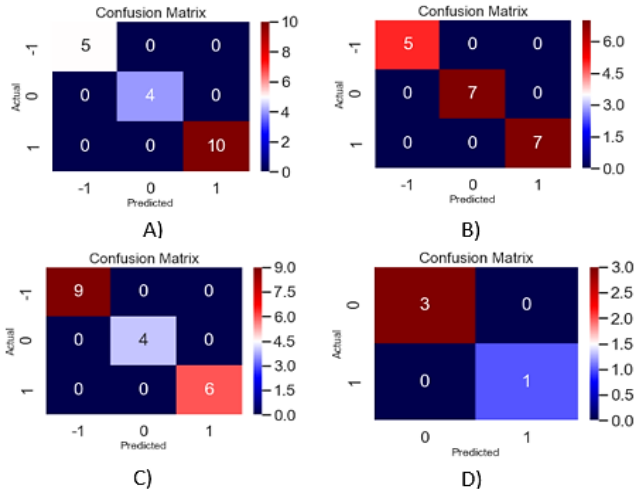

Fig. 1. Results of the test set for all types of customers. The matrix shows that the classification method succeeds in dividing for the type of distribution.

H1: A significant difference in the results exists when there are different distributions and numbers of customers.

In conclusion, the null hypothesis in every test was rejected. This means there is a statistically significant difference between the results of the problems according to their distribution and number of clients. Knowing this, it is valid to divide the problems by their distribution and number of clients. Therefore, it is possible to create the model to get a better hyper-heuristic based on the significant difference between instances. With these results, four different decision tree models were created for every class of a number of clients (25, 50, 100, and 800) to classify which type of distribution is the problem based on the spatial distribution of the clients. The results are shown figure 1.

Table 2. Sets used in the analysis. However, not all were used for every type of problem.

Set	Max Temp	Min Temp	Eq iter	Cool
C0	25	0.005	70	0.1
C1	25	0.005	90	0.3
C2	30	0.005	90	0.2
C3	40	0.005	70	0.1
C4	40	0.005	90	0.2
C5	40	0.005	110	0.3
C6	25	0.005	90	0.3
C7	30	0.005	70	0.1
C8	30	0.005	90	0.2
C9	40	0.005	70	0.1
C10	40	0.005	110	0.3
C11	50	0.005	90	0.2
C12	40	0.005	90	0.1
C13	50	0.005	90	0.1
C14	70	0.005	90	0.3
C15	70	0.005	90	0.4
C16	90	0.005	90	0.4
C17	90	0.005	90	0.5
C18	90	0.005	90	0.1
C19	100	0.005	90	0.1
C20	120	0.005	90	0.4
C21	130	0.005	90	0.3
C22	150	0.005	90	0.4

The selected models are *Decision Trees* created with *Scikit Learn Library* on Python with the default properties. With perfect precision, these models will be used to classify future datasets (different from Solomon's) to apply the resulting hyper-heuristic.

The method of Zulvia et. al. [10] was used, to select the best set of initial parameters to use in simulated annealing.

Where the Max/Min Temp is the maximum and minimum value of temperature, Eq iter is a parameter that limits the number of iterations with the same temperature value and Cool is the cooling rate of the simulated annealing.

An ANOVA one-way test was done to prove the hypothesis, the next hypothesis, and conclusion, the initial simulated annealing's parameters are statistically significant in most of the cases:

H0: The mean final length is equal regardless of the parameters used in simulated annealing.

Table 3. The best sets for each type of problem and their results.

Type	Best set	Mean length	Std Length	Min length
C25	C5	393.03	72.68	223.31
RC25	C5	729.36	145.48	315.27
R25	C5	693.76	127.27	340.18
C50	C10	997.45	258.09	410.5
RC50	C10	1489.97	347.51	762.89
R50	C10	1229.94	260.53	686.52
C100	C17	2731.44	775.15	948.18
RC100	C18	2975.81	551.34	1679.86
R100	C18	2346.67	431.73	1328.03
C800	C22	119161.85	3135.64	127208.45
R800	C22	109454.49	2864.01	102110.65

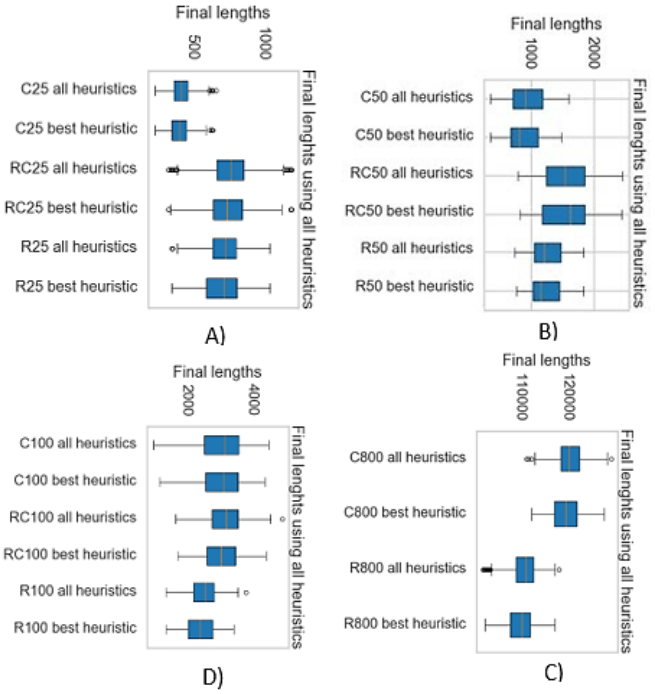

Fig. 2. Graphic comparison between one heuristic used with random parameters and with the best parameters (simulated annealing method of Zulvia [10]) for all the datasets.

H1: A significant difference in the results exists when there are different parameters in simulated annealing.

Table 4. One-way ANOVA to verify that the method improves the final length.

Problem	p-value
C25	<0.001
RC25	<0.001
R25	<0.001
C50	0.06
RC50	0.67
R50	0.15
C100	0.39
RC100	<0.001
R100	<0.001
C800	<0.001
R800	<0.001

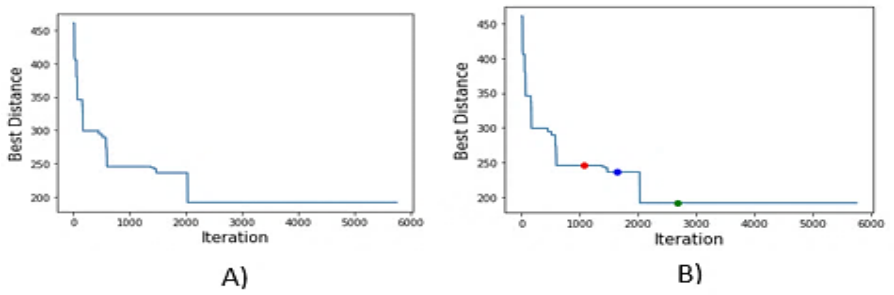

Fig. 3. The best distance as a function of iterations.

4.2 Dividing by Sections

The calculation of the length of each section was by analyzing the evolution of the best distance in every instance. Each instance was solved 30 times with the Random Saah hyper-heuristic, saving in all of them the change of the best distance. The mean evolution of the data of the best distance was calculated for each instance and then used to get the length of the sections. To achieve what has previously mentioned the following pre-programmed tools of *Sklearn Python Library* were used: *Linear Regression* and *PolynomialFeatures*. To make a polynomial regression of degree 4 [6]. This degree was selected because it was the one that obtained the best results.

Figure 9 A) shows the behavior of the best distance as a function of the iterations. To divide this function, a polynomial regression was used in different segments. The first one was the first 20 iterations and then adding intervals of 20 iterations. For each segment, the R2 was saved to know what segment fits better the regression. In other words, to know which group of iterations have similar behavioral changes at the best

Table 7. Weights for one section on instances S25.

Problem	Heuristic	Weight	Weight normalized	Cumulative Sum	Segments
C25	Intraroute_oropt	0.021467578	0.02854978	0.02854978	0 - 0.0285
C25	Intraroute_2-opt	0.066586257	0.088553211	0.117102991	0.02854 - 0.1171
C25	Intraroute_relocate	0.186704936	0.248299308	0.365402299	0.1171 -0 .3654
C25	Interroute_2-opt	0.065070949	0.086538	0.451940299	0.3654 -0 .4519
C25	Interroute_relocate2	0.191707086	0.254951678	0.706891977	0.45194 - .07068
C25	Interroute_exchange	0.179467234	0.238673872	0.945565849	0.706 -0 .9455
C25	Cross-exchange	0.03497635	0.046515126	0.992080974	0.9455 - 0.992
C25	GENI-exchange	0.005954592	0.007919026	1	0.992 - 1

distance. The first group went from 0-20, the second one from 0-40, and so on. This happened until half of the total iterations were reached (to ensure three sections minimum). In all the segments, the iteration where the section with the best R2 ends was saved.

With it, the search is started using this as the initial point, and the rest of the search range continues through the following iterations. For the previous action to stop, one of the following two conditions had to be fulfilled: the code reached the maximum capacity of sections (6) wanted or the search range remaining represented less than 10% of the total iterations.

Figure 9 B) shows an example of how the divided sections look. The first section goes from iteration 1 to the redpoint, the second from the red to the blue point, the third between points blue and green, and finally, the fourth from the green to the end of the iterations. These steps were repeated for each instance and then, their mean sections were calculated. The final results are presented below.

4.3 Weights

To calculate the weights, each problem was solved 30 times using only one heuristic, while analyzing their performance individually. In each section, the number of changes that were feasible, improvements, or best was saved for each heuristic. Then, using these numbers it was calculated the percentage of the total iterations that they represented in each section. To formulate the next equation, to assign the weights: P=X1 pf + X2 pi +X3 PB [8].

Where pf, pi, and PB represented the percentage of the feasible, the improvement, and the best changes, and Xi represented an "importance" for each percentage.

This importance was introduced to assure more heaviness to the improvement changes than the other two. One reason for this is that all the improved changes are feasible but not all the feasible ones improve the distance. The final coefficients Xi chosen were: 0.6 to the improvement changes, 0.15 to the feasible changes, and 0.25 to the best. Making the summary X1+X2+X3=1. As it can be seen in the formulas, if all the changes of a heuristic were feasible, improved, and best it corresponding weight would be 1 in that section. Making all weights to be between 0 and 1 [10]. Table 6 shows the mean weights for the instances with 25 customers.

Table 5. Sections for each type of problem.

Problem	Sections	# Sections
C25	[228,647,853]	4
R25	[101, 262]	3
RC25	[242, 656, 863]	4
C50	[297, 682, 859]	4
R50	[302, 851]	3
RC50	[348, 701, 844]	4
C100	[533, 1319]	3
R100	[400, 949]	3
RC100	[518, 1320]	3

Table 6. Weights for a section on instances S25.

Problems	Heuristic	Weights
C25	Intraroute_oropt	[0.0214, 0.0133, 0.0109, 0.0111]
C25	Intraroute_2-opt	[0.0665, 0.0399, 0.0294, 0.0286]
C25	Intraroute_relocate	[0.186, 0.098, 0.0751, 0.0747]
C25	Interroute_2-opt	[0.065, 0.043, 0.0322, 0.0319]
C25	Interroute_relocate2	[0.1917, 0.0848, 0.065, 0.0638]
C25	Interroute_exchange	[0.17946, 0.10069, 0.0903, 0.089]
C25	Cross-exchange	[0.0349, 0.02611, 0.01996 0.019766]
C25	GENI-exchange	[0.0059, 0.0014769 0.001057, 0.00102]
R25	Intraroute_oropt	[0.0439, 0.0268, 0.0234]
R25	Intraroute_2-opt	[0.0951, 0.0583, 0.0502]
R25	Intraroute_relocate	[0.212, 0.1147, 0.0962]
R25	Interroute_2-opt	[0.0941, 0.0726, 0.0641]
R25	Interroute_relocate2	[0.2039, 0.12, 0.1004]
R25	Interroute_exchange	[0.185, 0.115, 0.1047]
R25	Cross-exchange	[0.05778, 0.04375, 0.0406]
R25	GENI-exchange	[0.0209, 0.01133, 0.00985]
RC25	Intraroute_oropt	[0.035, 0.02188, 0.01754, 0.018]
RC25	Intraroute_2-opt	[0.0964, 0.05779, 0.0433, 0.0422]
RC25	Intraroute_relocate	[0.2348, 0.12873, 0.09639, 0.09412]
RC25	Interroute_2-opt	[0.0826, 0.0635, 0.055, 0.055]
RC25	Interroute_relocate2	[0.17152, 0.082, 0.06675, 0.0644]
RC25	Interroute_exchange	[0.1588, 0.098762, 0.09, 0.09]
RC25	Cross-exchange	[0.041, 0.0319, 0.02819, 0.0269]
RC25	GENI-exchange	[0.0139, 0.00776, 0.0064, 0.00608]

Table 8. Probabilities for each heuristic on *Probabilistic hyper-heuristic.*

Heuristic	Probability
Intraroute_oropt	0.04
Intraroute_2-opt	0.11
Intraroute_relocate	0.3
Interroute_2-opt	0.06
Interroute_relocate2	0.23
Interroute_exchange	0.2
Cross-exchange	0.03
GENI-exchange	0.03

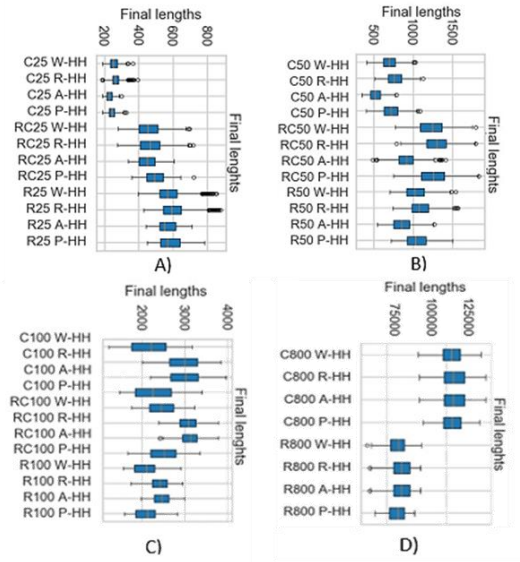

Fig. 4. Graphic comparison between the four hyper-heuristics, weighted, random choice of heuristics, adaptive, and probabilistic HH.

4.4 Probability Rule

For the selection of one heuristic in each iteration using the weights, the next step was the design of a probability rule. To achieve that the first step was to normalize the weights of every heuristics in each section, making the summary equal to 1. With the values normalized, it was used an accumulative sum to assign segments from 0 to 1, for each heuristics [13].

The first heuristic "Intra route_oropt" had the segment from 0 to the value of its weight normalized, the second one "intraroute_2-opt" had the segment between the values of the past normalized weight to this value, plus its correspondent normalized weight.

Table 9. One way ANOVA to verify that the method of the new hyper-heuristic is different from the other results in the mean of the final length and improves the final results.

Problem	p-value
C25	<0.001
RC25	<0.001
R25	<0.001
C50	<0.001
RC50	<0.001
R50	<0.001
C100	<0.001
RC100	<0.001
R100	<0.001
C800	0.0379
R800	0.0379

Table 10. Comparison of means of final lengths between methods for 25 customers. In bold the best performance of an hyper-heuristic for every type of problem.

Problem	Method	Mean Length
C25	Weighted hyper-heuristic	257.94
C25	Random hyper-heuristic	265.23
C25	Adaptive hyper-heuristic	**237.86**
C25	Probabilistic hyper-heuristic	261.78
RC25	Weighted hyper-heuristic	455.92
RC25	Random hyper-heuristic	469.60
RC25	Adaptive hyper-heuristic	**400.31**
RC25	Probabilistic hyper-heuristic	463.85
R25	Weighted hyper-heuristic	578.66
R25	Random hyper-heuristic	513.89
R25	Adaptive hyper-heuristic	**472.58**
R25	Probabilistic hyper-heuristic	582.99

Therefore, the third one was between the values of the cumulative sum and the cumulative sum, plus its correspondent normalized weight and so on. It will continue until the sum is equal to one and all the heuristics have a segment assigned. As we can infer from the procedure, if the normalized weight of a heuristics is near or equal to 0, then the probability for it to be chosen is also close to 0.

With the assigned segment from 0 and 1 to each heuristic, a random number between 0 and 1 was generated in each iteration. The random number was created using the pre-programmed python library random. Then, using this number a search was made for the segment where it falls into, thus a selection of that specific heuristic in that iteration.

The weights were actualized when the number of iterations entered the next section. An example of how does the segments look can be seen in the table 7.

Table 11. Comparison of means of final lengths between methods for 50 customers. In bold the best performance of an hyper-heuristic for every type of problem.

Problem	Method	Mean Length
C50	Weighted hyper-heuristic	703.71
C50	Random hyper-heuristic	774.52
C50	Adaptive hyper-heuristic	**528.51**
C50	Probabilistic hyper-heuristic	728.79
RC50	Weighted hyper-heuristic	1239.94
RC50	Random hyper-heuristic	1298.94
RC50	Adaptive hyper-heuristic	**917.74**
RC50	Probabilistic hyper-heuristic	1257.34
R50	Weighted hyper-heuristic	1046.7
R50	Random hyper-heuristic	1096.32
R50	Adaptive hyper-heuristic	**860.68**
R50	Probabilistic hyper-heuristic	1055.24

Table 12. Comparison of means of final lengths between methods for 100 customers. In bold the best performance of an hyper-heuristic for every type of problem.

Problem	Method	Mean Length
C100	Weighted hyper-heuristic	**2173.30**
C100	Random hyper-heuristic	2989.73
C100	Adaptive hyper-heuristic	3005.65
C100	Probabilistic hyper-heuristic	2282.08
RC100	Weighted hyper-heuristic	**2462.53**
RC100	Random hyper-heuristic	3071.33
RC100	Adaptive hyper-heuristic	3124.85
RC100	Probabilistic hyper-heuristic	2506.17
R100	Weighted hyper-heuristic	**2099.30**
R100	Random hyper-heuristic	2425.99
R100	Adaptive hyper-heuristic	2470.17
R100	Probabilistic hyper-heuristic	2115.83

5 Results and Discussion

Four hyper-heuristics are being compared: Probabilistic hyper-heuristic, Random hyper-heuristic, and Adaptive hyper-heuristic, and the other is the one proposed in this paper, the Weighted hyper-heuristic.

All of them work with the meta-heuristic algorithm: Simulated Annealing. The *Random hyper-heuristic* selects a random heuristic in each iteration, all the heuristics have the same probability of being chosen along the run. The *Probabilistic hyper-heuristic* has probabilities for each heuristics previously assigned by the programmer

Table 13. Comparison of means of final lengths between methods for 800 customers. In bold the best performance of an hyper-heuristic for every type of problem.

Problem	Method	Mean Length
C800	Weighted hyper-heuristic	**121946.45**
C800	Random hyper-heuristic	126614.63
C800	Adaptive hyper-heuristic	122837.43
C800	Probabilistic hyper-heuristic	122160.83
R800	Weighted hyper-heuristic	103416.00
R800	Random hyper-heuristic	104625.47
R800	Adaptive hyper-heuristic	104625.47
R800	Probabilistic hyper-heuristic	**103388.03**

uniform along the run. The ones assigned were the following. The last one is an *Adaptive hyper-heuristic*. It gives the same probability of being chosen to all the heuristics. As the iterations pass these probabilities are actualized.

Giving more probability of being chosen to the ones that had better performance in a particular run. The results for 100 runs per instance, with the initial simulated annealing parameters given in table 3, with the same limit of iterations (1080) per problem, are presented below in Figure 4 and tables 10, 11, 12, and 13.

Here is another hypothesis where we assume that the mean of the final distance in each hyper-heuristic differs from the others:

> **H0:** The mean of the final length is equal regardless of the hyper-heuristic used.
>
> **H1:** A significant difference in the results exists when different hyper-heuristics are used.

5.1 Results for all Type of Instances

In tables 10-13 it is shown that the *Weighted hyper-heuristic* proposed in this research, scored second place when comparing the mean final distance for cases C25 and RC25 and third place in random distribution type behind *Random Hyper-heuristic* and *Adaptive hyper-heuristic*. It is only surpassed by the *Adaptive hyper-heuristic* in all the 50 clients' classes and outperformed the rest.

However, when it surpassed 100 clients, the *Weighted Hyper-heuristic* is superior in efficiency compared to the other hyper-heuristics in this report, except for the case of R800 where it ranks second. Weighted hyper-heuristics performed well and was consistent, as it always came in first or second place in every type of problem except for one occasion. Its performance increases with a higher number of clients.

6 Conclusion

Analyzing the obtained results, it is concluded that using the *Adaptive hyper-heuristic* is better in an instance, with a small number of customers. As the cities increase, the

random and adaptive methods are less efficient. In this case, having a rule of probability or a weighted method is better. The advantage of the weighted method is that, despite the number of customers, its efficiency is constant. The *weighted hyper-heuristic* improves when the number of customers grows, as it is observed in the results, and when using statistical tests, the majority of cases are seen as better than the adaptive, random, or probabilistic hyper-heuristics.

It is important to analyze what makes each method strong and in what conditions for future approaches in order to take advantage of the potential of the weighted hyper-heuristic and improve the results obtained. The hypothesis of the hyper-heuristic has to be proved through the improvement of growth in its number of customers, the continuation of experimentation, and being able to obtain a non-supervised way to divide the problems into sections. Further implementation of the hyper-heuristic will result in better and more accurate through the law of large numbers (LLN).

References

1. Ahmed, L., Mumford, C., Kheiri, A.: Solving urban transit route design problem using selection hyper-heuristics. European Journal of Operational Research, 274(2), pp. 545−559 (2018)
2. FoodData: Agricultural Research Service. U.S.D.A. National Nutrient Data base for Standard Reference. http://ndb.nal.usda.gov/ndb/search/list (2019)
3. Caric, T., Gold, H.: Vehicle routing problem (2008)
4. Doerr, B., Lissovoi, A., Oliveto, P.S., Warwicker, J.A.: On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1015−1022 (2008)
5. Garza-Santisteban, F., Sánchez-Pámanes, R., Puente-Rodríguez, L.A., Amaya, I., Ortiz-Bayliss, J.C., Conant-Pablos, S.E., Terashima-Marín, H.: A simulated annealing hyper-heuristic for job shop scheduling problems. In: IEEE Congress on Evolutionary Com-putation (CEC), pp. 57−64 (2019)
6. Gutiérrez-Rodríguez, A.E., Conant-Pablos, S.E., Ortiz-Bayliss, J.C., Terashima-Marín, H.: Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning. Expert Systems with Applications, 118, pp. 470−481 (2019)
7. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science, 220(4598), pp. 671−680 (1983)
8. Masand, B.M., Smith, S.J.: U.S. Patent No. 5, 251,131. DC: U.S. Patent and Trademark Office (2019)
9. Max, T.A., Burkhart, H.E.: Segmented polynomial regression applied to taper equations. Forest Science, 22(3), pp. 283−289 (1976)
10. Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. Annals of Operations Research, 239(1), pp. 3−38 (2014)
11. Rembold, C.M., Watson, D.: Posttest probability calculation by weights: A simple form of Bayes' theorem. Annals of internal medicine, 108(1), pp. 115−120 (1988)
12. Tan, C.C.R., Beasley, J.E.: A heuristic algorithm for the period vehicle routing problem. Omega, 12(5), pp. 497−504 (1984)

13. van Laarhoven, P.J., Aarts, E.H.: Simulated annealing: Theory and applications. pp. 7−15 (1987)
14. Zulvia, F.E., Kuo, R.J., Nugroho, D.Y.: A many-objective gradient evolution algorithm for solving a green vehicle routing problem with time windows and time dependency for perishable products. Journal of Cleaner Production, 242 (2019)