A Fuzzy Hyper-Heuristic Approach for the 0-1 Knapsack Problem

Frumen Olivas, Ivan Amaya, José Carlos Ortiz-Bayliss, Santiago E. Conant-Pablos and Hugo Terashima-Marín
Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias
Ave. Eugenio Garza Sada 2501 Sur Col. Tecnológico C.P. 64849
Monterrey, Nuevo Leon, Mexico
{frumen, iamaya2, jcobayliss, sconant, terashima}@tec.mx

***Abstract*—Hyper-heuristics are potent techniques that represent the synergy of low-level heuristics when solving optimization problems. This synergy usually leads to better solutions. Similarly, fuzzy logic has been successfully applied to several domains, thanks to the expert knowledge it encompasses. Thus, combining the benefits of both approaches should lead to a more reliable and effective method. Hence, in this work, we propose a fuzzy-based selection hyper-heuristic model. We considered seven features and four low-level heuristics, which represent the inputs and output of the fuzzy inference system, respectively. Each input was defined with two membership functions. Since there is no expert knowledge available, we lay out all the rules (128) and use a genetic algorithm to find optimum values for the consequents of these rules. In other words, the genetic algorithm will evolve the rules of the fuzzy inference system until it become an expert, and will then save such knowledge as the set of fuzzy rules. The main concern of this paper is to find out if a fuzzy inference system can help to get better results in the inner working of a hyper-heuristic. To prove this, we make a comparison between a fuzzy hyper-heuristic model optimized by a genetic algorithm against three traditional selection hyper-heuristic models (with a different number of rules) optimized by a particle swarm optimization method. We applied all these methods using the same set of low-level heuristics to solve an 800 instance set of the 0-1 Knapsack problem as a testbed.**

***Index Terms*—Fuzzy logic, Hyper-Heuristic, Genetic algorithm**

I. Introduction

Fuzzy logic [1] represents a versatile set of tools. It can be implemented in many different problems due to its adaptability [2] and because it requires expert knowledge [3] instead of physical modeling. Such an approach requires the definition of membership functions and of fuzzy rules (for representing the expert knowledge) [4]. An advantage of using a fuzzy inference system is that we can "store" the expert knowledge of a problem in its fuzzy rules. Therefore optimizing a fuzzy inference system is similar to extracting knowledge from a problem because lately, we can read the fuzzy rules as a knowledge base from the problem. There are two main types of fuzzy inference systems, named after their creators: "Mamdani" and "Sugeno". Mamdani [5] proposed the first application of fuzzy logic into a controller problem, described a type of membership function, and a manner to approximate reasoning with fuzzy logic. Sugeno [6] proposed a faster way of computing the output of a fuzzy system.

Similarly, genetic algorithms (GAs) stands as a well-known optimization technique [7], free from the calculation of derivatives that traditional optimization techniques require. GAs seek to replicate the evolution of populations and has been widely used to tackle combinatorial problems [8], [9]. We use GAs in this work because they are easy to implement and it can handle integer values. However, we can apply any other optimization method that can handle integer values, since each gene of the chromosome is transformed into a consequent of a fuzzy rule of the fuzzy inference system.

Among the combinatorial problems of interest, a recurrent research subject is the knapsack problem. Here, the idea is to select a subset of items that maximizes the profit without breaking a weight constraint. Although solving the knapsack problem through low-level heuristics is a relatively old idea [10], it remains useful nowadays for many practical cases. For example, Morales et al. [11] proposed a divide-and-conquer heuristic to solve the knapsack problem. Another approach seeks to combine a set of low-level heuristics to generate a high-level solver known as a hyper-heuristic. So, we can describe hyper-heuristics as the process of creating "heuristics to choose heuristics". Even so, Burke et al. [12] defined hyper-heuristics as an automated methodology for selecting or generating low-level heuristics to solve hard computational problems, and they proposed a classification of such methods. Ross presented a survey of hyper-heuristics, where he explained basic concepts and typical applications [13].

To solve the knapsack problem, we use a fuzzy inference system as a selection hyper-heuristic, and the meta-heuristics are used only to find the best rules for our fuzzy proposal and traditional selection hyper-heuristics. Nowadays, there is a significant diversity of meta-heuristics applied to solve the 0-1 knapsack problem. A few of them are listed next: an improvement to the whale optimization algorithm (WOA) to handle binary values, was proposed by Abdel-Basset et al. [14], where they get better results than GA, harmony search (HS) and variants of PSO (Particle Swarm Optimization). The cohort intelligence (CI) with an educated approach was proposed by Sapre et al. [15], demonstrating that their approach is better than the original CI method. Abdel-Basset et al. [16]

This work was supported in part by Consejo Nacional de Ciencia y Tecnología (CONACyT) Basic Science Project [Grant number 287479], CONACyT Posdoctoral Grant (Frumencio Olivas) and by ITESM Research Group with Strategic Focus in Intelligent Systems.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

proposed a binary flower pollination algorithm (FPA) which obtained better results when compared against a GA and a binary version of PSO. Ezugwu et al. [17] studied a set of meta-heuristics such as GAs, simulated annealing (SA), as well as the exact methods branch and bound, and dynamic programming. In their study, exact methods obtained not only the best results but also the lowest computational times. Ye et al. [18] proposed a tissue P system that demonstrates that can solve the knapsack problem in linear time. Zhang et al. [19] described an improvement to the bee colony algorithm (BCA). They compared it against some variants of differential evolution (DE) and PSO, improving the performance and convergence against the original method. Huang et al. [20] introduced a binary modification of a quantum harmonic oscillator algorithm (QHOA). This algorithm was compared against binary versions of bat algorithm (BA), PSO, dragonfly algorithm (DA), and a hybrid PSO with gravitational search algorithm (GSA), demonstrating that their approach is superior in accuracy, convergence capability, and stability. Xue et al. [21] proposed a binary version of the fireworks algorithm (FWA) and compared it with the quantum genetic algorithm (QGA), binary PSO, and binary cuckoo search (CS). Their results show that their proposal is competitive against the other meta-heuristics. Zhan et al. [22] proposed a hybrid greedy repair operator to noising methods, compared to several meta-heuristics revealing that their approach is competitive. All the previously mentioned meta-heuristics differ from our approach to the fact that they lack saving any knowledge extracted from the problem, while our approach using a GA can save the extracted knowledge as fuzzy rules.

The main contribution of this work is the proposal of a new fuzzy selection hyper-heuristic approach, with the potential to be applied to a wide variety of problems. Still, in this case, we use the knapsack problem as a benchmark scenario to see the performance when compared against a traditional selection hyper-heuristic. Although several solution approaches exist for solving the knapsack problem, there is a knowledge gap regarding the feasibility of using fuzzy logic combined with selection hyper-heuristics. Then, we consider that it is feasible to use a GA to extract knowledge from the knapsack problem and convert it into a set of fuzzy rules. That can be later compared against selection hyper-heuristics optimized by PSO to produce robust hyper-heuristics for solving this problem. The results show that the combination of fuzzy logic and hyper-heuristics provides solutions with a reduced standard deviation. This means that the proposed fuzzy approach helps the hyper-heuristics to get better quality results with a small variation, making our proposal more robust and precise.

This manuscript is organized as follows. Section II provides information about the knapsack problem and other concepts related to this investigation. The solution approach proposed in this investigation is depicted in Section III. The experiments and results are presented in Section IV. Finally, Section V presents the conclusion as well as some directions for future work for this investigation.

II. Background and Related Work

A. The knapsack problem

The knapsack problem considered for this work has the restriction that the items can either be packed within the knapsack or not. This version of the knapsack problem is usually referred to as the 0-1 knapsack problem. A formal definition of the 0-1 knapsack problem is given by Equations 1 and 2, where n is the number of items, x_i is the number of copies of an item (0 or 1), i is the ID of such an item, and w_i and p_i represent the weight and profit of item i, respectively. Moreover, the knapsack has a maximum weight capacity given by W.

$$\max \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{subject to } \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\} \tag{2}$$

Regarding the instances used for this work, we consider a balanced set of instances of the 0-1 knapsack problem, which were tailored for each heuristic [23]. This way, every heuristic has the same number of instances where it excels –a desirable scenario for testing a hyper-heuristic approach. A total of 800 knapsack problem instances are used in this investigation, so each heuristic represents the best choice for 200 instances. Each instance contains 40 items with a weight ranging from 1 to 32 and a profit ranging from 1 to 128. The maximum capacity of the knapsack for all instances is 25.

B. Hyper-heuristics with fuzzy logic

The current literature contains some examples of hyper-heuristics that, at least at some point, incorporate fuzzy logic to make their decisions. For example, the work conducted by Asmuni et al. [24] proposed a fuzzy inference system as a metric to order the exams in a scholar calendar. The ordering was made through low-level heuristics, but a fuzzy inference system was used to improve their approach. In a further study, Asmuni et al. [25] extended their fuzzy inference system and applied it to order courses instead of exams. Chaudhuri et al. [26] proposed a modified genetic algorithm through hill-climbing methods applied to the timetabling of resources (teachers, classrooms, and students) from a university. They used a fuzzy inference system to improve the objective function for the soft constraints, aside from the objective function from the genetic algorithm to satisfy the hard constraints. A more recent work conducted by Jackson et al. [27] proposed using a fuzzy inference system to control the late acceptance parameter in a hyper-heuristic, which produced a performance rank of the applied low-level heuristics. In their work, new evaluations were compared against previous ones to see if there was an improvement. This was done not only with the last solution but with a given number of the previous ones. Based on this, a decision was made about which low-level heuristic yielded the best performance. Zamli et al. [28] applied a fuzzy inference system as a selection hyper-heuristic.

The inputs they considered represented various metrics about the applied low-level heuristics. With this, the system decided to change, stay, or maybe change the last low-level heuristic.

The main difference between all these works (where they combine hyper-heuristics with fuzzy logic) is that, in our proposal, the fuzzy inference system takes the inputs and gives us the low-level heuristic to use in the next iteration of the solution process. All other methods use a fuzzy inference system but as a tool to ordering some items, to improve their objective function, to control a parameter, or to decide if a low-level heuristic must be changed or not.

C. Heuristics and features

Heuristics are practical methods that, unlike exact methods, frequently trade optimality for speed in solving complex problems. In this work we use heuristics that every time we apply them to the same problem, they always get the same result and need the same amount of time.

We have included four commonly used low-level heuristics for solving the knapsack problem [29] [30]. Default (Def) selects the next unpacked item that can fit into the knapsack by preserving the original ordering of the items. Maximum profit (MaxP) selects the unpacked item with the maximum profit (as long as it can be packed). Minimum weight (MinW) selects the unpacked item with the minimum weight (as long as it can be packed). Finally, maximum profit per weight unit (MaxPW) selects, among the unpacked items, the one with the maximum value resulting from dividing its profit over its weight.

In this work, we characterize the instances of the 0-1 knapsack problem by using seven features. These features are computed based on the unpacked items in the instance being solved: the normalized mean weight (MeanW), the normalized median weight (MedianW), the normalized standard deviation of the weight (StdW), the normalized mean profit (MeanP), the normalized median profit (MedianP), the normalized standard deviation of the profit (StdP) and the correlation between weight and profit (Corr).

The hyper-heuristics use these features to decide which heuristic to apply given the current state of the solving process of an instance. Both the proposed fuzzy approach and the traditional hyper-heuristics use the same features of the problem and the same set of heuristics. The only difference is the rules used to decide which heuristic to select in the next step.

III. Solution Approach

The problem we want to overcome is to obtain the best results in each instance of the knapsack problem by using a fuzzy inference system as a selection hyper-heuristic. In our model, a GA is responsible for finding the fuzzy rule set that maximizes the performance of the fuzzy inference system.

The GA used in this work is a custom discrete version of the original GA, where the genes can only have integer values. In our case, 1, 2, 3, or 4, that represents the heuristic selected by the fuzzy rule. The selection is made via tournament, where two chromosomes are randomly selected and the one with the best fitness is selected for the crossover. The process is repeated until the desired percentage of the population is selected. For the crossover operation, two genes are randomly selected to split the chromosome into three parts. The offspring is created by switching the middle part between parent one and parent two. The mutation is performed by randomly selecting a percentage of genes from the offspring, where their values are randomly changed. The next generation is created via a tournament between the populations of parents and offspring.

The PSO algorithm used to optimize the rules of the selection hyper-heuristics is the original version, which uses Equations 3 and 4 to update the position and velocity of the particles, respectively. The position of the particle i is represented by x_i and its velocity by v_i in time t, while the velocity is computed using its previous velocity plus a cognitive and a social component, given by Equations 5 and 6, respectively.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3}$$

$$v_{ij}(t+1) = v_{ij}(t) + Cognitive + Social \tag{4}$$

$$Cognitive = c_1 r_1(t)[y_{ij}(t) - x_{ij}(t)] \tag{5}$$

$$Social = c_2 r_2(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{6}$$

Internally, the fuzzy hyper-heuristic works as follows. Given an instance of the 0-1 knapsack problem, the seven features described in Section II are computed and processed as inputs for the fuzzy inference system. By using the current set of fuzzy rules, the system computes the output —which corresponds to one of the four available low-level heuristics in the system. Once the low-level heuristic is selected, it is used to choose the item to be added to the knapsack. This process is repeated on the unpacked items until no more items can be packed within the knapsack.

Figure 1 illustrates how each chromosome in the GA represents a candidate set of fuzzy rules to be used within the fuzzy inference system. In each iteration, the fuzzy rules represented by each chromosome are extracted and replace the existing rules in the fuzzy inference system. The objective function, in this case, is using the fuzzy inference system with its new set of fuzzy rules to compute a solution for all the instances in the training set.

The solution approach illustrated in Figure 2 works as a step-by-step process described next:

1) From a set instances (training when GA is optimizing the fuzzy rules and testing when we are using the best fuzzy rule set found by GA), one instance is selected to be processed.
2) All the features are computed from the unpacked items in the instance.

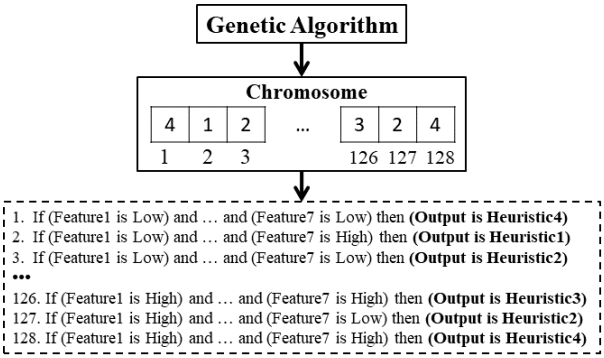

Fig. 1. Transformation of a chromosome from GA to a fuzzy rule set.

3) The fuzzy hyper-heuristic takes the features as inputs, and using the firing strength of the fuzzy rules, it computes an output through the defuzzification process.
4) Using the output of the fuzzy hyper-heuristic a low-level heuristic is selected to be applied to the instance.
5) The chosen heuristic selects the next item to pack.
6) If there are no more unpacked items or the knapsack is full, the system selects another instance, otherwise continue from step (2).

The fuzzy inference system (Sugeno) used within the hyper-heuristic has seven input variables (with two triangular membership functions each), one output (with four constant membership functions, one per low-level heuristic) and 128 fuzzy rules. The value of each input ranges from 0 to 1. Moreover, two triangular membership functions (equally distributed) are considered for mapping the values of such features. We adopted this type of membership function in this work because of its simplicity, which leads to faster computing. Such membership functions (Equation 7) have the following parameters:

- **Low**: $a = -1$, $b = 1$, $m = 0$.
- **High**: $a = 0$, $b = 2$, $m = 1$.

$$\mu_A(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{m-a}, & a < x \leq m \\ \frac{b-x}{b-m}, & m < x < b \\ 0, & x \geq b \end{cases} \quad (7)$$

The seven inputs are directly related to the seven features described in Section II. The number of membership functions was selected based on the number of fuzzy rules that can be generated. Two membership functions per input generate a total of 128 rules. Although this is a large number of rules to handle, we consider that reducing the number of membership functions per input would decrease the performance of the system since the problem state would not be properly captured. We consider that optimizing the number of rules would represent an important and interesting future research path.

We lack expert knowledge about when to select a low-level heuristic based on a certain level of each feature. So, a first approach considered randomly assigning the consequent of the fuzzy rules i.e., the low-level heuristic. Hence, we generated 3000 random configurations. This value was selected because it represents the same number of configurations that GA will test for finding appropriate rules. Alas, there is a humongous number of possible fuzzy rules, since there are 4^{128} possible combinations.

TABLE I
PARAMETERS USED FOR THE GENETIC ALGORITHM AND FOR PARTICLE SWARM OPTIMIZATION.

Parameter	GA	PSO
Population	30	30
Iterations	100	100
Dimensions	128	32, 48 and 64
Crossover	0.80	N/A
Mutation	0.10	N/A
C1=C2	N/A	2
Inertia Weight	N/A	Linear decreasing

The GA was used as a more complex (and intelligent) way of finding the best possible set of fuzzy rules, with the parameters shown in Table I, and using the aforementioned type of chromosome. Table I also presents the parameters used for PSO for the optimization of the selection hyper-heuristics. These parameters are taken from [31]. Note that these methods use a different number of dimensions since we apply them to different optimization processes. While we use GA to find 128 consequents for the fuzzy rules, PSO needs to find all rules for the selection hyper-heuristics. Here we make each rule up of eight dimensions, one for each feature and the heuristic to select. So to generate four rules we need 32 dimensions in each particle, 48 for six rules and 64 for 8 rules.

In order to perform a comparison between our proposed fuzzy hyper-heuristic approach and a traditional one, we use the hyper-heuristic model illustrated in Figure 3. This model uses an optimizer for finding a set of rules that determines which low-level heuristic to use next, depending on the results of the calculated features. The model depicts that the hyper-heuristic is problem-independent and only can compute the features from a problem state. In this example, the hyper-heuristic has three rules with three features each, plus an action (or the low-level heuristic to select). A distance metric between the features from the problem and each rule allows determining the low-level heuristic to be selected.

The set of instances used in this work is balanced in the sense that each low-level heuristic excels over the same number of instances (200). So, there is a total of 800 instances. To create the training test, we randomly select 30 instances favoring each solver (i.e. the 15%). The remaining 170 instances per solver (i.e. 85%) are left as the testing set.

IV. RESULTS AND DISCUSSION

As mentioned above, the training set used in the experiments represents the 15% of the total instances (i.e. $30 \times 4 = 120$) while the testing set is comprised of 680 instances (i.e. 85%). Table II presents the profit achieved (in both sets of instances), after using the GA for training the fuzzy model (using the

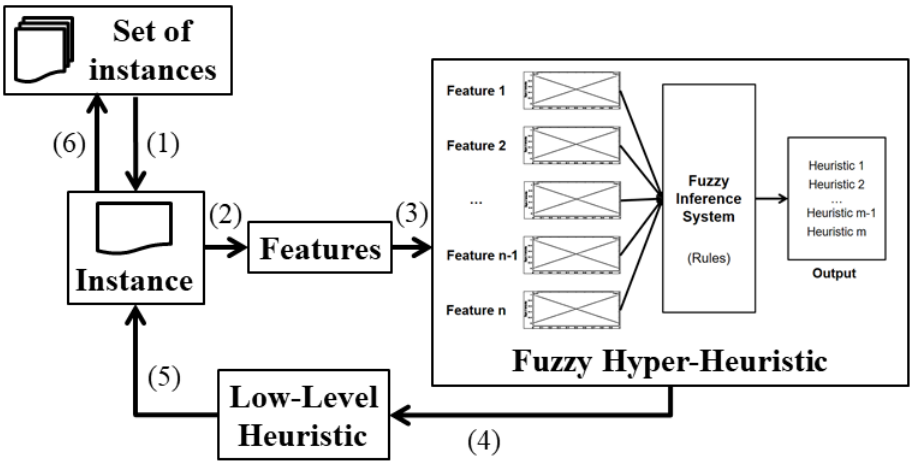

Fig. 2. Evaluation of a set of instances using our proposed fuzzy hyper-heuristic approach optimized by GA.

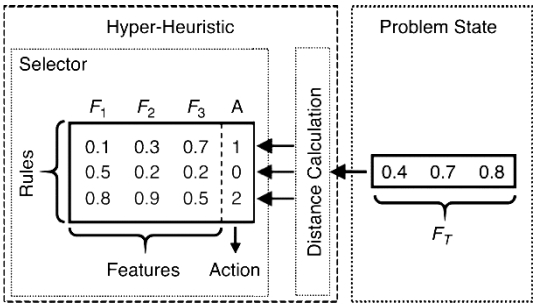

Fig. 3. A traditional selection hyper-heuristic model, as depicted in [32]

parameters shown in Table I). Moreover, data for traditional selection hyper-heuristics with four, six, and eight rules are given for comparison purposes. These selectors were trained with PSO using the parameters from Table I. Please note that each result from Table II is the sum of the profit achieved on each instance, so larger values mean better ones (highlighted in bold).

Table III summarizes data from Table II by providing the average profit values of each approach on both the training and testing sets. It is worth mentioning that the random approach corresponds to the best values after 3000 tests. We select this amount of tests because it corresponds to the number of chromosomes that the GA tests in one experiment over all the generations.

As shown in Table III, our proposed fuzzy selection hyper-heuristic model obtains better results (on average) than low-level heuristics. As expected, it also outperforms a purely random approach with the same number of candidate solutions, and traditional selection hyper-heuristics with four, six, and eight rules. Hence the fuzzy inference system helps in obtaining a better model of the problem. This assumption is strengthened by the fact that the fuzzy model was the one with the lowest standard deviation in both training (Figure 4) and testing (Figure 5).

Data from Tables II and III show that the advantage of

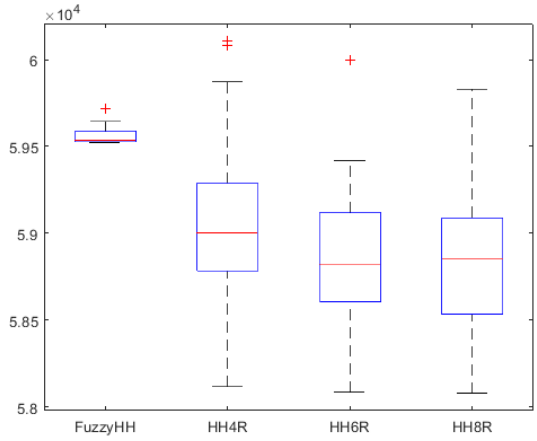

Fig. 4. Box plot of the results achieved during training (30 runs). FuzzyHH: Fuzzy-based selection hyper-heuristic. HHXR: Traditional selection hyper-heuristic with 4, 6, or 8 rules.

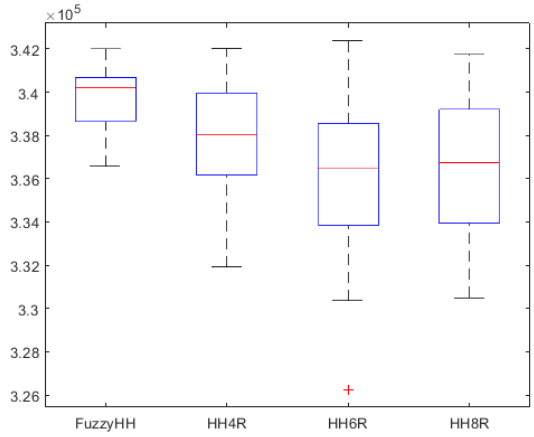

Fig. 5. Box plot of the results achieved during testing (30 runs). FuzzyHH: Fuzzy-based selection hyper-heuristic. HHXR: Traditional selection hyper-heuristic with 4, 6, or 8 rules.

TABLE II
RESULTS ACHIEVED WITH THE GA-TRAINED FUZZY SELECTION HYPER-HEURISTIC AND WITH TRADITIONAL HYPER-HEURISTICS.

Experiments	Proposal		Hyper-heuristics 4 Rules		6 Rules		8 Rules	
Set	Train	Test	Train	Test	Train	Test	Train	Test
1	59526	**341235**	58933	337920	58694	336556	58692	333842
2	59589	336988	59086	339967	59057	337315	58964	338763
3	59586	338773	59872	**342036**	58687	332939	58803	336557
4	59642	340809	58974	339905	58486	335705	58932	339207
5	59565	336077	58613	337251	58924	336586	58830	333445
6	59527	340774	60079	340868	58963	339089	58881	337612
7	59527	340858	58784	337067	**59996**	339793	59688	337903
8	59643	339134	59009	337723	58149	330367	58791	337286
9	59576	339994	59237	336150	58820	333839	59006	333959
10	59589	338846	58997	335964	58087	326238	59096	336340
11	59586	340210	59495	340880	58606	333607	59656	336563
12	59589	336873	59167	333256	59055	339428	58194	333118
13	59535	341180	**60107**	338194	58815	337397	58250	333988
14	59527	340729	59734	339352	59295	336074	59450	340233
15	59604	336964	59045	339454	58519	335488	58535	330495
16	**59658**	339695	58779	336772	59258	338708	58283	332022
17	59527	340431	58937	341720	58828	334294	58823	336507
18	59548	340284	58481	334449	58498	331030	58628	337797
19	59586	340297	58973	336653	58769	335686	58383	334232
20	59527	339804	58980	338186	58409	336360	59180	340653
21	59536	338484	58732	336182	58622	332845	58080	332225
22	59586	340900	59142	339468	58759	338563	58419	335350
23	59586	339715	58588	334204	59323	**342381**	**59831**	340420
24	59586	340453	58909	339396	58935	338298	59089	339531
25	59544	340604	59387	340609	59198	335589	58491	333937
26	59527	338858	58769	335983	58752	337117	58974	337958
27	59586	338416	59261	337062	59118	338358	58938	336934
28	59527	340798	59287	340705	59205	340038	59127	339345
29	59586	337493	58116	331936	58388	332839	58893	339265
30	59526	340474	59454	340069	59420	339339	58798	**341739**

TABLE III
SUMMARY OF ALL METHODS ON ALL INSTANCES (SPLIT INTO TRAINING AND TESTING). DATA CORRESPONDS TO AVERAGE AND STANDARD DEVIATION VALUES ACROSS 30 REPETITIONS.

Method	Training	Testing
Default	32275	191080
MaxP	41864	234917
MinW	49329	282972
MaxPW	57585	330949
Random (best of 3000)	50238±3263.09	282189±20683.35
Fuzzy Hyper-Heuristic	**59572±39.86**	**339538±1454.34**
Hyper-Heuristic (4 Rules)	59098±450.20	337979±2543.48
Hyper-Heuristic (6 Rules)	58854±402.23	336062±3369.03
Hyper-Heuristic (8 Rules)	58857±434.16	336574±2918.92

using the proposed fuzzy approach mainly stems from stability (lower standard deviation). The best results found in Table II (60107 in training and 342381 in testing), are given by a traditional selection hyper-heuristics. Moreover, the traditional hyper-heuristic methods also get the worst results, when compared to the fuzzy approach.

On the other hand, the results on average from our proposed fuzzy approach are better than the other methods, which are depicted in Figures 4 and 5. Although, our proposed approach and the traditional hyper-heuristic with four rules yield similar average values. In terms of standard deviation, the hyper-heuristic produced with our approach is about a tenth (training) and about a half (testing) of the values given by the traditional model.

It is important to mention that the pure random approach was only executed once, since it was used as a starting point. This approach is not reliable due to its nature, and it is not an intelligent way of finding a good set of fuzzy rules. During our tests, the random approach obtained better results than three low-level heuristics but worst than the hyper-heuristics. We believe that the fuzzy inference system helps to get these results, but also it shows that our proposed fuzzy selection hyper-heuristic requires an optimization process. A drawback that traditional selection hyper-heuristics also have.

Table IV summarizes the time, in seconds, needed for each method to process all the instances from both the training and testing sets. Data show that the random and the optimized fuzzy hyper-heuristic require similar computational time. This is due to the fact that both of them use 128 fuzzy rules. The traditional selection hyper-heuristics need roughly a third of the time compared with the fuzzy approach. Hence, it is computationally cheaper but also more unstable.

The results in time are as expected: including the fuzzy system increases computational requirements. However, we believe that this can be improved by optimizing the number of rules. Profits achieved show that the proposed fuzzy approach

TABLE IV
TIME IN SECONDS NEEDED TO PROCESS EACH SET OF INSTANCES BY EACH METHOD.

Method	Training set	Testing set
Default	0.0458	0.1367
MaxP	0.0333	0.1567
MinW	0.0412	0.2011
MaxPW	0.0415	0.2162
Random	2.5361±0.4606	13.1353±0.8654
Fuzzy Hyper-Heuristic	2.1085±0.1606	12.0324±0.8842
Hyper-Heuristic (4 Rules)	0.6817±0.0408	3.7993±0.0564
Hyper-Heuristic (6 Rules)	0.6987±0.0432	3.9365±0.1782
Hyper-Heuristic (8 Rules)	0.7107±0.0262	4.0176±0.2330

can yield better and more stable results than a traditional approach. Nonetheless, they differ from the optimal value, e.g. achieved with a dynamic programming approach. Even so, it is an infeasible approach in terms of scaling. For example, even though it achieved profits of 65810 and 377570 in the training and testing set, respectively, it required 8.1292 and 45.1803 seconds in each case. Hence, it required over thrice the computing time than the most expensive approach. It is important to highlight that we are aware that we are not proposing the best method for the knapsack problem. Instead, the main objective of this work was to show the feasibility of a fuzzy-based hyper-heuristic model, by comparing it against a traditional selection hyper-heuristic.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a fuzzy-based selection hyper-heuristic. Our data revealed that this model can obtain better results (on average) than low-level heuristics and traditional selection hyper-heuristics for the knapsack problem. Our model was also more stable than the other approaches in both the training and testing sets. It is important to remark that our proposal was unable to achieve the same result as dynamic programming. However, and as it was mentioned, our goal for this work was not to achieve the optimal solution, but to show that a fuzzy-based hyper-heuristic model is feasible and may yield good results. We plan on further improving our approach in future works.

MaxPW was the best low-level heuristic. Even so, our proposed approach achieved an average result with a profit of about 2000 units higher. Moreover, this represents an increase of about 500 units with respect to the best traditional hyper-heuristic considered for this work (the one using four rules). Nonetheless, our approach has some drawbacks. For example, we are currently using the combination of all rules within the fuzzy model. Nevertheless, not all of them may contribute to performance in a significant way. Instead, they provide a computational burden. Besides, using all combinations limits the scalability of the approach, since the number of rules would escalate too quickly. Thus, our proposal needs to include a refinement stage, where a smaller set of fuzzy rules can be identified. Therefore, a future research path that stems from this work is to analyze the contribution of each rule to the overall performance, so that a reduced number of rules can be obtained. This will lower the computational effort required to select a low-level heuristic and, in turn, would make our proposal more attractive.

The fuzzy approach does not always improve the result, and we believe this is due to the large number of fuzzy rules (128) used that causes an "interference" in the reasoning (defuzzification) of the proposed fuzzy approach. The firing strength of the fuzzy rules is being decimated by other rules that have opposite consequent. We are working on an improvement of that issue for future works by reducing the number of fuzzy rules.

Throughout this work, we used a GA for finding the outputs of each rule in the fuzzy system. Although this approach worked adequately, it does not inhibit the idea of exploring and comparing the performance of other metaheuristics for such a task. Moreover, using a GA for identifying the aforementioned set of rules seems like a feasible approach.

Another path for future work rests on the problem domain. We believe that our proposed approach can be applied to several types of problems, aside from the knapsack problem. The modular nature of our proposed hyper-heuristic model eases its application to other domains, as it is only required to define the fuzzy set in terms of the already existing features. Hence, this application can be viewed as a proof of concept. We shall delve deeper into this idea in future studies. Finally, it is also important to analyze the behavior of our proposed approach when solving harder instances. Thus, the behavior under different instance sets should be explored in the future.

REFERENCES

[1] L. A. Zadeh *et al.*, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[2] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," in *Proceedings of the sixth international symposium on Multiple-valued logic*, pp. 196–202, IEEE Computer Society Press, 1976.

[3] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy sets and systems*, vol. 90, no. 2, pp. 111–127, 1997.

[4] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[5] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," in *Proceedings of the institution of electrical engineers*, vol. 121, pp. 1585–1588, IET, 1974.

[6] M. Sugeno, *Industrial applications of fuzzy control*. Elsevier Science Inc., 1985.

[7] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[8] C. Martinez, O. Castillo, and O. Montiel, "Comparison between ant colony and genetic algorithms for fuzzy system optimization," in *Soft computing for hybrid intelligent systems*, pp. 71–86, Springer, 2008.

[9] J. F. Gonçalves and M. G. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 17, no. 5, pp. 487–525, 2011.

[10] R. Loulou and E. Michaelides, "New greedy-like heuristics for the multidimensional 0-1 knapsack problem," *Operations Research*, vol. 27, no. 6, pp. 1101–1114, 1979.

[11] F. A. Morales and J. A. Martínez, "On the implementation and assessment of several divide & conquer matheuristic strategies for the solution of the knapsack problem," *arXiv preprint arXiv:1901.01215*, 2019.

[12] K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward, "Handbook of metaheuristics. chap. a classification of hyper-heuristic approaches. international series in operations research & management science," 2009.

[13] P. Ross, "Hyper-heuristics," in *Search methodologies*, pp. 611–638, Springer, 2005.

[14] M. Abdel-Basset, D. El-Shahat, and A. K. Sangaiah, "A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 3, pp. 495–514, 2019.

[15] M. S. Sapre, H. Patel, K. Vaishnani, R. Thaker, and A. S. Shastri, "Solution to small size 0–1 knapsack problem using cohort intelligence with educated approach," in *Socio-cultural Inspired Metaheuristics*, pp. 137–149, Springer, 2019.

[16] M. Abdel-Basset, D. El-Shahat, and I. El-Henawy, "Solving 0–1 knapsack problem by binary flower pollination algorithm," *Neural Computing and Applications*, vol. 31, no. 9, pp. 5477–5495, 2019.

[17] A. E. Ezugwu, V. Pillay, D. Hirasen, K. Sivanarain, and M. Govender, "A comparative study of meta-heuristic optimization algorithms for 0–1 knapsack problem: Some initial results," *IEEE Access*, vol. 7, pp. 43979–44001, 2019.

[18] L. Ye, J. Zheng, P. Guo, and M. J. Pérez-Jiménez, "Solving the 0-1 knapsack problem by using tissue p system with cell division," *IEEE Access*, vol. 7, pp. 66055–66067, 2019.

[19] S. Zhang and S. Liu, "A discrete improved artificial bee colony algorithm for 0–1 knapsack problem," *IEEE Access*, vol. 7, pp. 104982–104991, 2019.

[20] Y. Huang, P. Wang, J. Li, X. Chen, and T. Li, "A binary multi-scale quantum harmonic oscillator algorithm for 0–1 knapsack problem with genetic operator," *IEEE Access*, vol. 7, pp. 137251–137265, 2019.

[21] J. Xue, J. Xiao, and J. Zhu, "Binary fireworks algorithm for 0-1 knapsack problem," in *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pp. 218–222, IEEE, 2019.

[22] S. Zhan, L. Wang, Z. Zhang, and Y. Zhong, "Noising methods with hybrid greedy repair operator for 0–1 knapsack problem," *Memetic Computing*, pp. 1–14, 2019.

[23] L. F. Plata-González, I. Amaya, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, H. Terashima-Marín, and C. A. Coello Coello, "Evolutionary-based tailoring of synthetic instances for the Knapsack problem," *Soft Computing*, vol. 23, pp. 12711–12728, dec 2019.

[24] H. Asmuni, E. K. Burke, J. M. Garibaldi, and B. McCollum, "Fuzzy multiple heuristic orderings for examination timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*, pp. 334–353, Springer, 2004.

[25] H. Asmuni, E. K. Burke, and J. M. Garibaldi, "Fuzzy multiple heuristic ordering for course timetabling," in *Proceedings of the 5th United Kingdom workshop on computational intelligence (UKCI 2005)*, pp. 302–309, Citeseer, 2005.

[26] A. Chaudhuri and K. De, "Fuzzy genetic heuristic for university course timetable problem," *Int. J. Advance. Soft Comput. Appl*, vol. 2, no. 1, pp. 100–121, 2010.

[27] W. G. Jackson, E. Özcan, and R. I. John, "Fuzzy adaptive parameter control of a late acceptance hyper-heuristic," in *Computational Intelligence (UKCI), 2014 14th UK Workshop on*, pp. 1–8, IEEE, 2014.

[28] K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, "An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation," *Information Sciences*, vol. 399, pp. 121–153, 2017.

[29] H. Kellerer, U. Pferschy, and D. Pisinger, "Knapsack problems". SpringerVerlag, Berlin Heidelberg, 2004.

[30] X. F. C. S. Dıaz, "Analysis of a Feature-independent Hyper-heuristic Model for Constraint Satisfaction and Binary Knapsack Problems" PhD diss., School of Engineering and Sciences. Instituto Tecnológico y de Estudios Superiores de Monterrey, 2017.

[31] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[32] I. Amaya, J. C. Ortiz-Bayliss, A. Rosales-Perez, A. E. Gutierrez-Rodriguez, S. E. Conant-Pablos, H. Terashima-Marin, and C. A. C. Coello, "Enhancing selection hyper-heuristics via feature transformations," *IEEE Computational Intelligence Magazine*, vol. 13, no. 2, pp. 30–41, 2018.