IJCI V11-1(2024) 17–29
International Journal of Computers and Information (IJCI)
Available online at *https://ijci.journals.ekb.eg/*

A Modified Multi-Level Hyper Heuristic for Tackling Combinatorial Optimization Problems

Asmaa Awad [a*],Osama Abdel Raouf [a], Nancy El-Hefnawy[a,b],Ahmed Kafafy [b]
[a]*Faculty of Artificial intelligence, Menoufia University, Menoufia, Egypt*
[b]*Operations Research & DSS Dept., Faculty of Computers and Information, Menoufia University, Menoufia, Egypt*
[a,b] *Faculty of Computers and Information, Tanta University, Tanta, Egypt*
asmaa@ai.menofia.edu.eg ,osama@ai.menofia.edu.eg, nancyabbas_1@ci.menofia.edu.eg, ahmedkafafy80@gmail.com

Abstract
Hyper-heuristics are search techniques that operate on heuristic spaces by selecting low-level heuristics appropriately. Hyper-heuristic aims to generate a generalized and automated approaches to solve various problem domains. However, the effectiveness of hyper-heuristics depends on the cooperation of several low-level heuristics to provide high-quality solutions. Low-level heuristics can be categorized as either constructive or perturbative. They aim to construct or improve existing solutions. This paper presents a proposed Modified Multi-Level Hyper-Heuristic (MMHH) applied on a multilevel framework with three layers. The first layer is highest-level heuristic which selects a suitable hyper-heuristic algorithm. While the second layer called high-level heuristic that chooses suitable low-level heuristic from a set of heuristics. Two reward techniques are provided, one based on the amount of improvement, and the other based on the number of improvements. Two different scenarios for combining two reward selection algorithms are investigated. The first scenario is based on adopting a set of different weights for each technique. The second one is based on adapting the balancing between the two reward techniques by utilizing the idea of simulated annealing. The performance of the proposed MMHH algorithm is assessed by comparing it to a set of state-of-the-art hyper-heuristics, which include multi-level hyper-heuristics amount of improvement (MHHA) and number of improvements (MHHN), Dynamic Multi-Armed Bandit, Fitness-Rate-Rank Multi-Armed Bandit, and Deep QNetwork, across six problem domains from the HyFlex Framework. Based on the experimental results, it is evident that the MMHH framework demonstrates high competitiveness and outperforms the other compared methods in five out of the six benchmarks

Keywords: Hyper-heuristic; Multilevel; Combinatorial Optimization problems.

1. Introduction

Hyper-heuristics (HHs) are a class of search algorithms that aims to automate the process of selecting and applying heuristics to solve combinatorial optimization problems [4][5]. Unlike to traditional optimization techniques that use a fixed set of heuristics to investigate the search space, hyper-heuristics can generate new heuristics or adapt existing ones to better suit a particular problem domain [3] [2]. The development of HHs has gained attention in the field of optimization due to their ability to provide automated and general-purpose solutions that can adapt to a wide range of problems [3]. The use of hyper-heuristics has been shown to be particularly effective for problems where the optimal solution is difficult to obtain or where traditional optimization algorithms are not well-suited. In general, HHs operates on the heuristics space rather than the search space of solutions [6]. HHs can be classified into two main categories: selection and generation [7]. Selection hyper-heuristics select the most promising heuristic at each step in the search process, whereas generation hyper-heuristics create new heuristics by combining or modifying existing heuristics [8]. Hyper-heuristics frameworks are typically composed of two layers: high-level heuristics (HLHs) and low-level heuristics (LLHs) [9]. HLHs select the most promising LLH to be applied on the current solution in order to generate a new one [10][15]. LLHs are responsible for making local changes to the current

solution to improve its quality. LLHs can be customized for a specific problem domain. They consist of a problem representation, evaluation function, and a set of LLHs [11]. Designing a robust hyper-heuristic model involves a non-trivial task of selecting appropriate LLH and the suitable move acceptance methods for a specific problem [12]. In recent years, several modifications and extensions have been proposed to improve the hyper-heuristics performance's, such as multi-level hyper-heuristics (MHHs) [1]. MHHs consist of three layers: the highest-level heuristic, an HLH, and a set of LLHs. The highest-level heuristic selects the appropriate hyper-heuristic algorithms that are capable of tackling a wide range of problem domains [1].

In this study, MMHH approach is suggested for solving combinatorial optimization problems. The proposed approach operates through a multilevel framework which involves three layers [1]. Two different scenarios are investigated to combine the two methods of reward selection algorithms, namely the amount of improvement and the number of improvements. In the first scenario, various weights are used to control the selecting method whereas the second scenario balances between the two selecting methods based on the idea of simulated annealing.

The structure of this article is organized as follows. Section 2 provides a comprehensive review of the related work on hyper-heuristics and its various components. Section 3 presents the Proposed MMHH, which addresses the limitations of the existing MHH frameworks. This section also provides a detailed description of the key components of the MMHH framework and how it works. Section 4 describes the experimental design and presents the results of the evaluation. Also, presents the performance of the proposed MMHH framework and the previous MHH frameworks. Finally, Section 5 presents the conclusions with different points for the farther research.

2. Related work

Recently, there is a great research work is developed for hyper-heuristics selection. The performance of hyper-heuristics is influenced by two essential elements, LLH selection mechanism and acceptance criteria. The related work for this research is presented in the following sentences.

The authors in [13] introduce a Monte Carlo tree search hyper-heuristic framework in which a tree-based search approach is utilized to find the optimal sequence of low-level heuristics for a given state. The framework performance is enhanced by incorporating a memory mechanism with a population of solutions and employs diverse population updating rules. The framework is evaluated across six domains of the hyper-heuristic (CHeSC) competition test suite to increase its generality. The experimental results demonstrate the framework's strong generalization abilities. They also show the ability to achieve competitive or even superior performance compared to the state-of-the-art results in the scientific literature.

In [14], a gene expression programming algorithm generates a HLH of a hyper-heuristic framework. The generated heuristic uses information from the current problem state to select low-level heuristics and evaluate the resulting solution. This framework enables the generation of different high-level heuristics for each instance, promotes solution diversity. Additionally, a memory mechanism maintains a population of high-quality and diverse solutions, continuously updated during the search process. The hyper-heuristic proposed in [14] is evaluated on six combinatorial optimization problems, achieve more generality and competitive performance compared to state-of-the-art methods across all domains.

The study developed in [19] explores a Deep QNetwork (DQN) selection strategy within an online selection hyper-heuristic algorithm and compares its performance with two state-of-the-art Multi-Armed Bandit (MAB) approaches. Two Multi-Armed Bandit (MAB) approaches, namely Dynamic Multi-Armed Bandit (DMAB) [22] and Fitness-Rate-Rank Multi-Armed Bandit (FRRMAB) [23], are introduced. DMAB is an innovative method that combines an optimal MAB algorithm with the statistical Page-Hinkley test, enabling the effective identification of changes in time series. On the

other hand, FRRMAB utilizes a sliding window to track the recent fitness improvement rates of operators and employs a decaying mechanism to enhance the selection probability of the best operator. The experiments were carried out on all six problem domains of the HyFlex Framework [21]. By defining the state representation and reward scheme, the DQN rapidly identified effective and ineffective operators, which lead to improved performance compared to the MAB strategies in problem instances where exploitative behaviour was advantageous.

In a previous study [16], researchers have introduced a framework known as "MSHH" (Iterated Multi-Stage Selection Hyper-Heuristic), which utilizes a multi-stage selection hyper-heuristic approach. MSHH employs multiple hyper-heuristics at different stages, with a focus on controlling transitions and facilitating information exchange among them. In the MSHH framework, the selection mechanism of hyper-heuristics is comprised of two distinct stages. The first stage employs a greedy strategy to compute scores for a set of LLHs, while the second stage utilizes a roulette wheel selection approach to choose a heuristic based on the assigned scores. Both stages utilize an adaptive threshold move acceptance method. Experimental results demonstrate the superior performance of MSHH compared to five other hyper-heuristics across various problem domains in the HyFlex framework.

Different studies are provided on multi-level selection hyper-heuristics (MHH)such as MHHA and MHHN in [1]. The authors proposed a novel approach to enhance the performance of the hyper-heuristic framework. The MHH introduces a new level-based strategy that employs a roulette wheel selection mechanism to effectively choose the most suitable HH-Alg based on its performance through the search process. This layer chooses the most suitable HH-Alg from a predefined set of HH-Alg's to improve the solution obtained. The performance of the MHH algorithm is evaluated by comparing it with one of recent methods and other HHs that are employed as components of the MHH, across six commonly used benchmark datasets. The results indicate that the MHH framework is highly effective and outperforms other methods in five out of the six benchmarks.

As discussed earlier, it has been observed that the previous techniques have certain limitations in terms of their generality and their capability to effectively employ the appropriate heuristics and acceptance method for various search scenarios based on the problem domain. Therefore, it is crucial to develop a selection technique to effectively manage this process and provide optimal use of the relevant heuristics and acceptance criteria.

3. The multi-level hyper-heuristic framework

As mentioned of, the MHH framework consists of three layers: the highest-level heuristic, HLH and LLH, as illustrated in Fig.1. The HLH and LLH are similar to those used in the previous framework [2]. However, MHH introduces a new highest-level heuristic layer involving two modules: the hyper-heuristic algorithm (HH-Alg) selection and the HH-Alg acceptance criteria. The pool of high-level heuristic includes three HH-Alg's as follows: the MSHH [16], the "Robinhood hyper-heuristic" (RHH) [17], and the "Hyper-heuristic Search Strategies & Timetabling" (HYSST) [18]. The essential goal of this layer is to provide more generalized, reusable, and automated hyper-heuristic techniques to tackle various problems and generate high-quality solutions. The cooperation and competition among several LLH's are achieved through applying different method of LLH selection and using different acceptance criteria. The MHH framework performs as follows: in the first step, HH-Alg is chosen from a pool of algorithms using a roulette wheel selection mechanism. In the second step, the selected HH-Alg is applied to the current solution. In the third step, the solution obtained is evaluated using an acceptance method, and the selected HH-Alg is rewarded based on the obtained solution. In the previous work, two reward methods are applied. The first method rewards the selected HH-Alg based on amount of improvement (MHHA), while the second one rewards the selected HH-Alg based on number of improvement (MHHN) [1]. Finally, the previous steps are repeated until the stopping criterion is met.

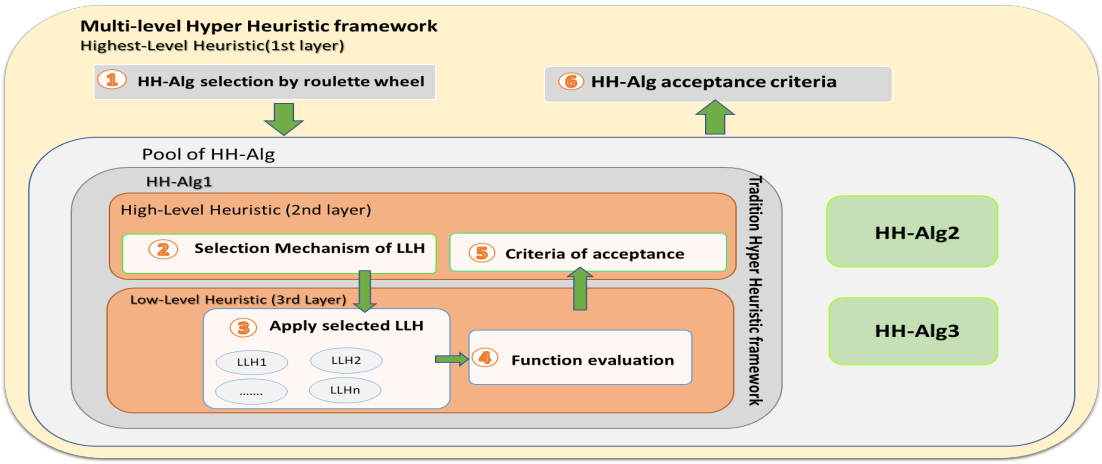

Fig.(1) MHH framework

4. The Proposed Modified Multi-Level Hyper-Heuristic Framework

The authors in [1] apply either the reward based on Amount of improvement (MHHA) or the reward based on number of improvements (MHHN). At the beginning of search processes, there is a significant improvement in the solution, making MHHA the preferred approach. However, at the end of search processes, the rate of improvement decreases, which means MHHN a more suitable choice. The basic idea of MMHH is to select the appropriate reward method to reward selected HH-Alg based on Amount of improvement or Number of improvements. The two rewarded approaches, MHHA and MHHN, are employed to improve the benefits. In this context, two different scenarios can be employed to select the appropriate reward method. These scenarios can be explained as follows:
The first scenario selects one of rewarded methods randomly with a constant probability. Scenario2 employed selecting one of the rewarded methods using a probability that is based on the concept of simulated annealing (SA). The fundamental concept behind SA is that it accepts the worst solution with a certain probability, which gradually decreases over the iteration process until it becomes zero. This means preventing the acceptance of the worst solution. In this case, the same concept is utilized by initially applying the same probability to accept the MHHA strategy at the start of the search, and then gradually decreasing this probability over time to discourage the utilization of MHHA and promote the adoption of the MHHN strategy towards the end of the search. In each scenario, MMHH calculates the reward of each HH-Alg after full sliding window(S). MMHH1 takes into consideration the history and update reward after half of sliding window. MMHH2 is online update the reward which calculates the reward after each iteration.

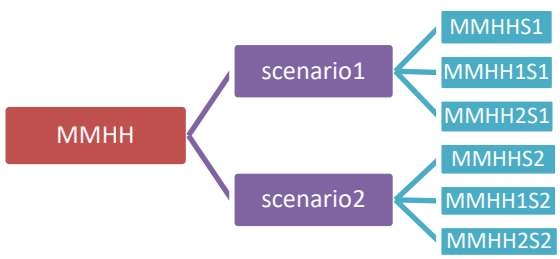

Fig.(2): Two scenarios of MMHH Framework

4.1 The Proposed Algorithm

The MMHH Framework adopts a multi-level approach consisting of two components at the highest-level. The first component is responsible for HH_Alg selection, which selects an HH-Alg from a pool

of n algorithms using a roulette wheel policy, and credit assignment, which rewards each HH-Alg based on its performance. This reward affects the selection of HH-Alg in the next sliding window (S). The second component is the acceptance criteria, which evaluates a new solution make a decision whether to accept or reject it.

The proposed algorithm starts with all HH-Algs such that have the same probability of selection, which is equal to $1/n$. MMHH chooses an HH-Alg randomly based on the roulette wheel policy with equal chance for all HH-Algs. The selected HH-Alg applying the selection method to choose a LLH and acceptance criteria. Then a new solution is generated and evaluated. The new solution is compared with the current solution. If the generated solution is better than or equal to the current solution, it is accepted. Then, the reward selection method is determined randomly by a constant probability as scenario 1 or by the idea of simulated annealing (SA) in scenario 2. Scenario2 applies Eq.1 to calculate the probability Ps_i at ith iteration in S to selects one of two reward methods.

$$Ps_i = e^{(-Improves/(100/(i+1)))} \tag{1}$$

If MHHA is selected , the amount of improvement achieved by the selected HH-Alg in the new solution through S is calculated as " $Improves$ " . if MHHN is selected, add one to $Accpted$ if the generated solution is better than or equal to the current solution and add one to $Invokes$ either solution is better or equal or not. After S, the accumulative reward $Suc_{n,S}$ for each HH-Alg is calculated using Eq.2 the total amount of improvement and the number of improvement $Accpted$ from all number HH-Alg $Invokes$ according to Eq.3.

$$Suc_{n,S} = \sum_n Improves_{n,S} \ \forall\, n, S \tag{2}$$

$$Suc_{n,S} = \frac{Accpted_{n,S}}{Invokes_{n,S}} \qquad \forall\, n, S \tag{3}$$

For an HH-Alg that did not achieve any improvement, a small value ϵ is assigned to give it a small chance in the next iterations as mentioned by Eq.4. The probability for n HH-Algs is then calculated by dividing the $Reward$ of each algorithm by the total improvement achieved during S according to Eq.5. This cycle is repeated with the new probabilities calculated for S, until the stopping criteria are met.

$$Reward_{n,l} = \begin{cases} Suc_{n,l} + \epsilon & if\ Suc_{n,l} > 0 \\ \epsilon & Otherwise \end{cases} \tag{4}$$

$$P_{n,S+1} = Reward_{n,S} / \sum_{\forall j} Reward_{j,S} \tag{5}$$

The MMHH algorithm automatically selects among different HH-Algs at different points during the sliding window (S) and evaluates each HH-Alg based on its performance. This allows the appropriate HH-Alg to participate more than the others. This leads to improve the performance, level of generality, and reusability of MMHH. Each HH-Alg uses a different selection heuristic and acceptance criteria. The algorithm also provides an automatic mechanism to use both HH-Alg reward methods by one of the two scenarios mentioned above. This improves the quality of the solution.

The pseudocode of MMHH is described in Alg (1). Lines 1 and 2 define the pool of HH-Algs with an equal chance for n HH-Algs. Line 4 shows the main loop of MMHH, which selects an HH-Alg using a roulette wheel and applies it to the current solution, then checks if the quality of the generated solution improves or remains the same. If the new solution is accepted, the selected HH-Alg is rewarded, as shown from line 6 to 8.Lines 9-15 apply one scenario from two scenarios explained above to determine the method of reward. A random value r is generated. In case scenario 1 the probability Ps_i is constant. In scenario 2 the probability Ps_i is calculated based on Eq.1. If r is less than Ps_i, the amount of improvement is applied, otherwise, the number of improvements is used to

reward the selected HH-Alg.Lines 17-20 calculate the accumulative reward and the probability of the amount of improving method for each HH-Alg using Eq.2 and Eq.5. Lines 21-24 calculate the accumulative reward and the probability of the number of improved method for each HH-Alg using Eq.3 and Eq.6. The previous steps are then repeated until the stopping criteria are met. From line 17-24 can be calculated the reward of each HH-Alg after full Sliding window(S) (MMHH) or after half of S (MMHH1) or after each iteration (MMHH2).

Algorithm 1: The proposed Multilevel-hyper-heuristic

Inputs:

Set_{HH}: *hyper-heuristics Algorithms*

P: problem to apply

Outputs:

the best solution found

Begin:

1. $Set_{HH} \leftarrow \{HH - Alg_1, \dots, HH - Alg_n\}$ *//define HH-Alg's*
2. $P_{HH-Alg_n} \leftarrow \frac{1}{n} \ \forall n \in \{1, \dots, n\}$ *//Assign each HH-Alg equal Probability.*
3. $Cur\_Sol \leftarrow IntialzeSolution$ // *Initial Solution*
4. **While** Stopping condition not met **do**
5. **For** each $i \in \{1, \dots, S\}$ **do**: // Sliding Window
6. $HH - Alg_i \leftarrow$ RoulleteWheelSelction$(Set_{HH-Alg}, P_{HH-Alg_k})$
7. $\text{New}_{\text{Sol}} \leftarrow ApplyHH - Alg(HH - \text{Alg}_i, Cur\_Sol)$
8. $Cur\_Sol \leftarrow AcceptancCriteria(New\_Sol)$
9. $r \leftarrow Generate\ random\ value\ of\ r\ \epsilon$ {0,1}
10. $Ps_i \leftarrow Calculate\ probability\ based\ on\ scenario$
11. **If** r < Ps_i **do**
12. $RewardA \leftarrow AssignAmountOfRewards(HH - \text{Alg}_i, Cur\_Sol, New\_Sol)$
13. **Else do**
14. $RewardN \leftarrow AssignNumOfRewards(HH - \text{Alg}_i, Cur\_Sol, New\_Sol)$
15. **EndIF**
16. **EndFor**
17. **For *each*** $n \in \{1, \dots, n\}$ **do**:
18. $Reward_n \leftarrow UpdateRewardofAlg(RewardA)$ // as Eq.2
19. $P_{HH-Alg_n} \leftarrow UpdateProbablities(RewardA_n)$ // as Eq.5
20. **EndFor**
21. **For *each n*** $\in \{1, \dots, n\}$ **do**:
22. $Reward_n \leftarrow UpdateRewardofAlg(RewardN)$ // Eq.3
23. $P_{HH-Alg_n} \leftarrow UpdateProbablites(RewardN_n)$ // Eq.6
24. **EndFor**
25. **EndWhile**
26. **End.**

The computational complexity analysis of the provided algorithm can be summarized as follows: The initialization process, encompassing Lines 1 to 3, boasts a constant complexity denoted as O(1). The central main loop, executed in Line 4, exhibits a complexity of O(m), where m signifies the number of iterations. The sliding window operation, taking place in Line 5, carries a complexity of

O(s), with s indicating the number of sliding window iterations. The selection and application of algorithms, as depicted in Lines 6 to 16, maintains a constant complexity denoted as O(1). The updating of rewards and probabilities, executed in Lines 17 to 20 and 21 to 24, respectively, each presents a complexity of O(n), where n represents the number of algorithms considered. Total complexity according to Eq.6:

$$\begin{aligned} Total &\approx O(1) + O(m)\big(O(s) * O(Alg) + O(1) + O(n) + O(n)\big) + O(mn) + O(mn) \\ &\approx O(1) + O(ms) * O(Alg) + O(m) + O(mn) + O(mn) \\ &\approx O(ms) * O(Alg) \end{aligned} \quad (6)$$

In summary, the algorithm's overall computational complexity is determined by the product of the number of iterations (m) and the number of sliding window iterations (s), which is further multiplied by the complexity of the heuristic algorithms (O(Alg)). This complexity analysis provides insight into the algorithm's resource requirements and scalability for different problem sizes.

5.Experimental Results

In this study, the effectiveness of the proposed MMHH is evaluated using six problem domains of HyFlex that represent various difficulties in most COPs according to literature review. These domains included SAT, BP, FSH, PS, VRP, and TSP. To assess the performance of MMHH, it is compared with some of the state of the art algorithms by conducting 31 different independent runs for each problem domain.

5.1 Performance Analysis of scenario1 and scenario2

Table 1 shows the result of two different scenarios to select MHHA or MHHN. In scenario 1, MHHA or MHHN are selected based on a specified probability(p), which means to select MHHA with probability p and MHHN with probability (1-p). In our experiment, two values are assigned to p (p =0.7, p=0.3). The proposed MMHH has three variants (MMHH, MMHH1 and MMHH2) according to reward calculation as mentioned in section 4. The reward calculation is applied according to the state of the sliding window (the proposed MMHH calculates the reward after full sliding window (S=20) where MMHH1 calculates the reward after half of S. Finally, MMHH2 calculates the reward after each iteration). The performance of all alternative MMHH, MMHH1, and MMHH2 are evaluated. In scenario 2, The principle of SA is applied through calculating a certain probability using Eq.1. The reward calculation is applied with the same manner as in scenario1 and then the same experiment is performed.

The results, presented in Table 1, indicate the average objective function value over 31 independent runs. Its clear that MMHH in scenario 2 achieves the best performance over 21 instances out of 30 instances of the six problem domains, specifically all instances in BP, four instances in SAT as Fig.3 FSH, and PS. In VRP, it performs significantly better in three instances. Unfortunately, there is no clear differences between scenario1 and scenario2 regarding the average result for TSP instances.

General speaking, scenario2 improves the overall performance in most test instance. So, it will be generalized and used to increase the performance of the MHH (MHHA and MHHN) to obtain the modified version.

Table 1: Average of two scenarios of MMHH Algorithms

		Scenario1(P=0.7)			Scenario1(P=0.3)			Scenario2		
Dom	inst	MMHH	MMHH1	MMHH2	MMHH	MMHH1	MMHH2	MMHH	MMHH1	MMHH2
SAT	inst1	2.66E+01	2.67E+01	2.63E+01	2.80E+01	2.76E+01	2.82E+01	2.61E+01	**2.59E+01**	2.70E+01
	inst2	2.21E+01	2.21E+01	2.18E+01	2.31E+01	2.21E+01	2.23E+01	**2.11E+01**	2.15E+01	2.19E+01
	inst3	1.12E+01	1.10E+01	1.17E+01	1.10E+01	1.16E+01	1.13E+01	**9.45E+00**	1.11E+01	1.06E+01
	inst4	8.65E+00	9.97E+00	1.05E+01	9.39E+00	1.03E+01	9.42E+00	**7.10E+00**	9.48E+00	9.84E+00
	inst5	2.05E+01	1.98E+01	1.87E+01	1.80E+01	1.88E+01	1.91E+01	**1.64E+01**	1.82E+01	2.00E+01
BP	inst1	7.71E-03	7.62E-03	7.66E-03	7.72E-03	7.90E-03	7.76E-03	**7.43E-03**	7.52E-03	7.56E-03
	inst2	2.25E-02	2.24E-02	2.23E-02	2.23E-02	2.24E-02	2.24E-02	**2.22E-02**	2.23E-02	2.23E-02
	inst3	2.44E-02	2.43E-02	2.46E-02	2.42E-02	2.42E-02	2.41E-02	**2.40E-02**	2.41E-02	2.41E-02
	inst4	5.56E-03	5.63E-03	5.63E-03	5.73E-03	5.75E-03	5.64E-03	**5.51E-03**	5.55E-03	5.61E-03
	inst5	4.16E-03	4.02E-03	4.02E-03	4.16E-03	4.04E-03	4.03E-03	**3.95E-03**	4.02E-03	3.99E-03
FSH	inst1	6.34E+03	6.34E+03	6.34E+03	6.34E+03	6.34E+03	6.34E+03	**6.33E+03**	6.34E+03	6.33E+03
	inst2	6.42E+03	6.41E+03	6.41E+03	6.41E+03	6.42E+03	6.42E+03	**6.41E+03**	6.41E+03	6.41E+03
	inst3	6.39E+03	6.40E+03	6.39E+03	6.39E+03	6.39E+03	6.39E+03	**6.39E+03**	6.39E+03	6.39E+03
	inst4	6.47E+03	6.47E+03	6.47E+03	6.47E+03	6.47E+03	6.47E+03	**6.46E+03**	6.47E+03	6.47E+03
	inst5	1.06E+04	1.05E+04	1.06E+04	1.05E+04	**1.05E+04**	1.05E+04	1.05E+04	1.05E+04	1.05E+04
PS	inst1	2.34E+03	**2.27E+03**	2.33E+03	2.30E+03	2.32E+03	4.07E+03	2.40E+03	2.43E+03	3.96E+03
	inst2	5.97E+02	7.49E+02	7.99E+03	1.52E+04	2.18E+04	5.87E+02	**3.96E+02**	4.53E+02	5.02E+02
	inst3	2.35E+01	2.65E+01	2.19E+01	2.46E+01	2.33E+01	2.32E+01	**1.93E+01**	2.11E+01	2.22E+01
	inst4	2.66E+01	2.80E+01	2.87E+01	2.83E+01	2.88E+01	2.85E+01	**2.39E+01**	2.73E+01	2.63E+01
	inst5	2.74E+01	2.81E+01	2.80E+01	3.28E+01	2.86E+01	2.66E+01	**2.19E+01**	2.55E+01	2.45E+01
VRP	inst1	2.07E+04	2.08E+04	2.07E+04	2.07E+04	2.07E+04	2.07E+04	**2.07E+04**	2.07E+04	2.07E+04
	inst2	1.34E+04	1.35E+04	1.34E+04	**1.34E+04**	1.34E+04	1.34E+04	1.34E+04	1.34E+04	1.34E+04
	inst3	5.35E+03	5.36E+03	5.35E+03	5.35E+03	5.35E+03	5.36E+03	**5.34E+03**	5.35E+03	5.35E+03
	inst4	1.44E+04	1.45E+04	1.43E+04	1.43E+04	1.43E+04	1.43E+04	**1.43E+04**	1.43E+04	1.43E+04
	inst5	1.50E+05	1.54E+05	1.49E+05	1.51E+05	1.51E+05	**1.48E+05**	1.49E+05	1.49E+05	1.49E+05
TSP	inst1	1.13E+05	1.13E+05	1.13E+05	1.13E+05	**1.13E+05**	1.13E+05	1.13E+05	1.14E+05	1.13E+05
	inst2	7.00E+03	7.00E+03	7.01E+03	7.00E+03	7.01E+03	7.01E+03	**6.99E+03**	7.02E+03	7.02E+03
	inst3	4.32E+04	4.32E+04	4.33E+04	**4.32E+04**	4.33E+04	4.34E+04	4.32E+04	4.34E+04	4.33E+04
	inst4	9.16E+03	9.15E+03	9.16E+03	9.14E+03	**9.14E+03**	9.16E+03	9.15E+03	9.17E+03	9.17E+03
	inst5	**5.95E+04**	5.95E+04	5.96E+04	5.97E+04	5.96E+04	5.97E+04	5.97E+04	5.97E+04	5.97E+04

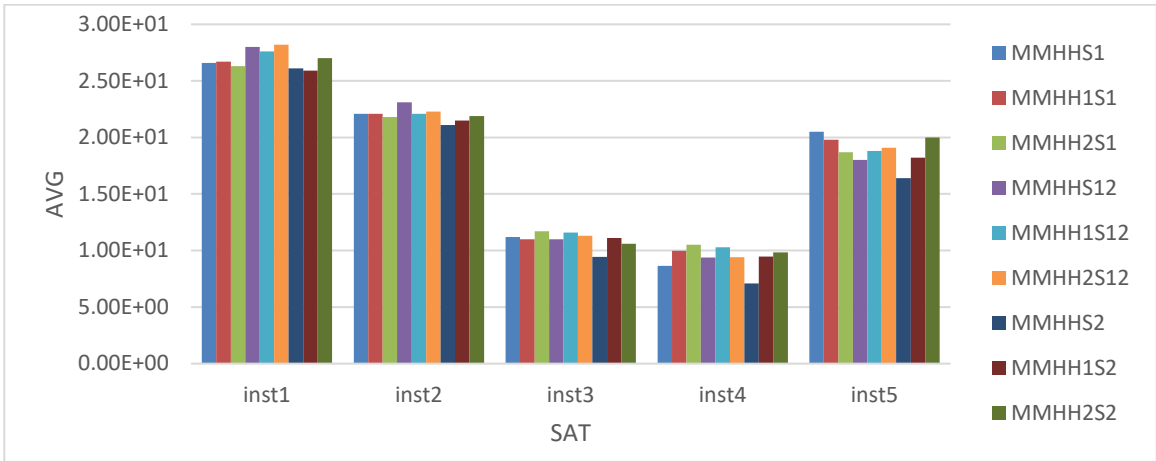

Fig.(3) Average of SAT of two scenarios of MMHH Algorithms

5.2 Performance Comparison to the state-of-the-art Hyper-heuristics

As mentioned above, MMHH with scenario 2 achieves the best performance. So, MMHH has been evaluated against several state-of-the-art hyper-heuristics, including DMAB, FRRMAB, and DQN, as well as the recent MHHA and MHHN. The comparison was conducted on six different problem domains of HyFlex, including SAT, BP, FSH, PS, VRP, and TSP, with 31 independent runs for each problem.

The results, as shown in Table 2 and Figures 4 to 9, indicate that the proposed MMHH approach outperforms all the other algorithms in several instances of different problem domains. In the SAT problem domain, MMHH achieves the best average outcomes compared to DMAB, FRRMAB, DQN, MHHA, and MHHN in all instances, as shown in Figure 2 and Table 2. Similarly, in the BP problem domain, MMHH consistently performs the best average compared to all other algorithms, and this performance is statistically significant for all instances as demonstrated in Figure 3 and Table 2. Regarding the FSH problem domain, MMHH provides the best average outcomes for instances inst1, inst2, and inst4. Also, MMHH has the best performance for inst2 with FRRMAB algorithm. However, MHHA outperforms in inst3, and FRRMAB is the best in inst5, as indicated in Figure 4 and Table 2. In the PS problem domain, MMHH provides the best average outcomes in four instances (inst1, inst3, inst4, and inst5), and provides the second-best performance in inst2 with FRRMAB algorithm. However, DMAB shows the best performance on average in Inst2, as shown in Figure 5. and Table 2. For the VRP problem domain, the superiority of the proposed MMHH approach over all the compared algorithms is clearly demonstrated in all instances, as depicted in Figure 6 and Table 2. In the TSP problem domain, MMHH outperforms all the compared algorithms in several instances (inst2, inst4, and inst5), but its performance is slightly lower than DQN in inst1 and MHHA in inst3, as shown in Figure 7 and Table 2.

In scenario2, SA calculate the probability which selects MHHA in the beginning of the search in state of MHHN, and then gradually decreasing this probability over time to discourage the utilization of MHHA and promote the adoption of the MHHN strategy towards the end of the search. This means enhancing the overall results. In general, the results of this study confirm that the proposed MMHH approach can potentially lead to improved overall performance compared to several state-of-the-art hyper-heuristics.

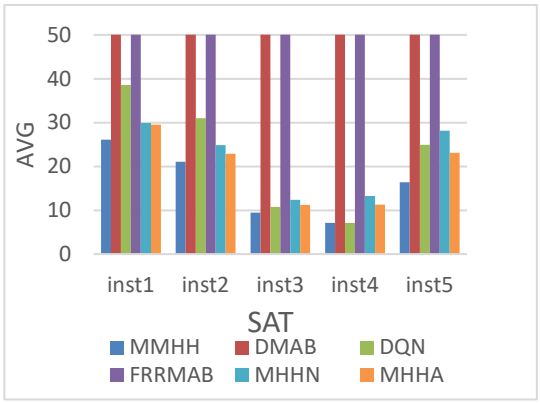

Fig.(4) average performance results for the SAT problem across five instances.

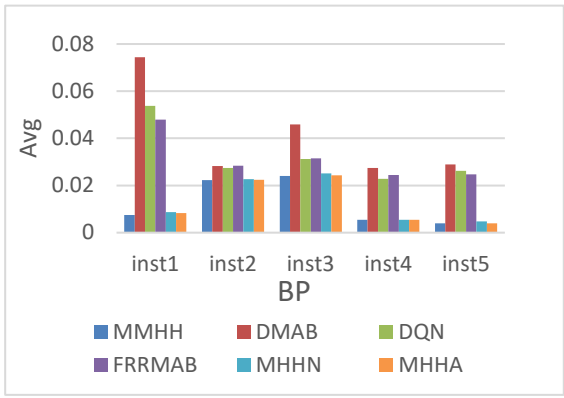

Fig.(5) Average performance results of BP problem across five instances.

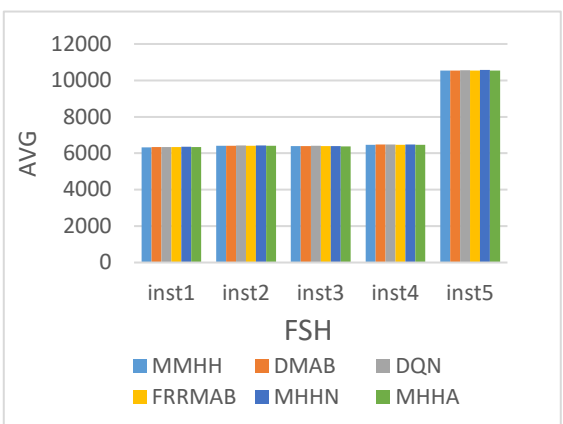

Fig.(6) Average performance Results of FSH problem across five instances

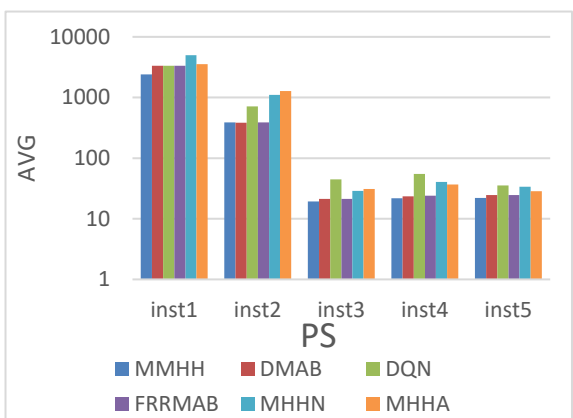

Fig.(7) Average performance Results of PS problem across five instances

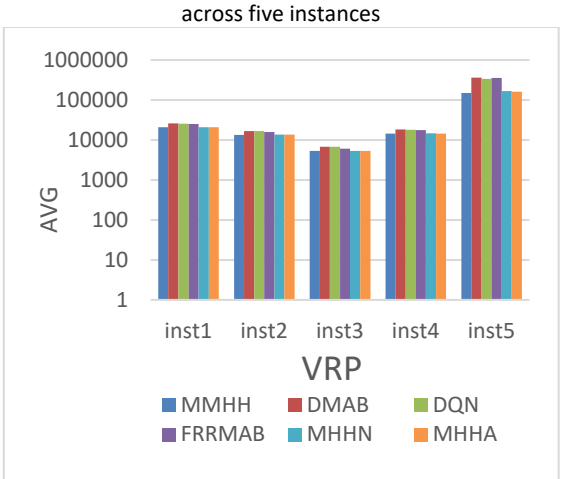

Fig.(8): Average performance Results of VRP problem across five instances

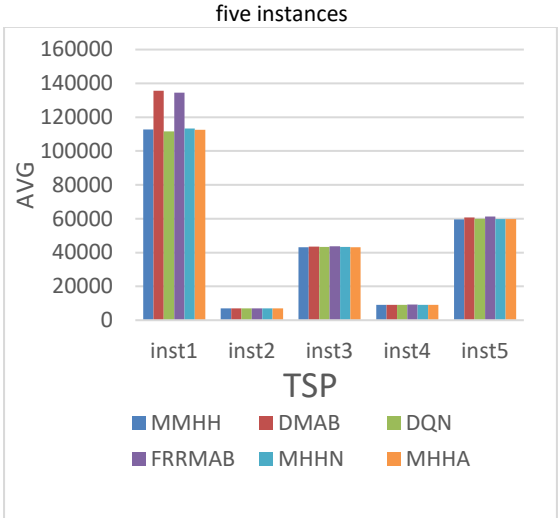

Fig.(9) Average performance Results of TSP problem across five instances

Table 2:Average Obtained by MMHH against state-of-the-art Hyper-Heuristics Algorithms.

Domai		MMHH		DMAB		DON		FRRMAB		MHHN		MHHA	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD	AVG	SD	AVG	SD
SAT	inst1	**2.61E+01**	2.35E+00	2.78E+02	5.52E+00	3.86E+01	4.81E+00	2.78E+02	6.23E+00	2.99E+01	4.42E+00	2.95E+01	7.40E+00
	inst2	**2.11E+01**	1.39E+00	2.74E+02	5.22E+00	3.10E+01	4.20E+00	2.74E+02	6.41E+00	2.49E+01	3.37E+00	2.29E+01	2.96E+00
	inst3	**9.45E+00**	1.55E+00	1.56E+02	4.41E+00	1.08E+01	2.54E+00	1.60E+02	4.42E+00	1.24E+01	3.10E+00	1.12E+01	3.35E+00
	inst4	**7.09E+00**	2.64E+00	1.65E+02	3.46E+00	7.10E+00	1.87E+00	1.69E+02	4.76E+00	1.33E+01	7.24E+00	1.13E+01	4.65E+00
	inst5	**1.64E+01**	4.99E+00	2.57E+02	5.20E+00	2.49E+01	8.69E+00	2.58E+02	6.10E+00	2.82E+01	1.18E+01	2.31E+01	1.12E+01
BP	inst1	**7.43E-03**	2.60E-04	7.44E-02	2.00E-03	5.38E-02	4.90E-03	4.79E-02	2.70E-03	8.71E-03	1.66E-03	8.35E-03	1.28E-03
	inst2	**2.22E-02**	2.80E-04	2.83E-02	6.00E-04	2.75E-02	5.00E-04	2.84E-02	5.00E-04	2.27E-02	6.50E-04	2.24E-02	5.50E-04
	inst3	**2.40E-02**	6.70E-04	4.59E-02	8.30E-03	3.12E-02	1.10E-03	3.15E-02	5.00E-04	2.51E-02	9.60E-04	2.43E-02	9.30E-04
	inst4	**5.51E-03**	2.50E-04	2.75E-02	1.70E-03	2.28E-02	6.30E-03	2.45E-02	2.20E-03	5.48E-03	5.70E-04	**5.45E-03**	5.60E-04
	inst5	**3.95E-03**	8.00E-05	2.89E-02	2.00E-03	2.62E-02	8.10E-03	2.47E-02	7.00E-04	4.77E-03	1.80E-03	**3.94E-03**	1.08E-03
FSH	inst1	**6.33E+03**	1.00E+01	6.34E+03	8.80E+00	6.35E+03	1.40E+01	6.33E+03	1.10E+01	6.35E+03	1.58E+01	6.34E+03	1.19E+01
	inst2	**6.41E+03**	8.80E+00	6.41E+03	8.20E+00	6.42E+03	7.90E+00	**6.41E+03**	7.30E+00	6.42E+03	1.23E+01	6.41E+03	1.43E+01
	inst3	6.39E+03	8.10E+00	6.39E+03	6.60E+00	6.41E+03	9.10E+00	6.39E+03	7.70E+00	6.40E+03	1.29E+01	**6.38E+03**	1.28E+01
	inst4	**6.46E+03**	1.00E+01	6.47E+03	7.80E+00	6.49E+03	1.09E+01	6.47E+03	1.09E+01	6.48E+03	1.61E+01	6.46E+03	1.10E+01
	inst5	1.05E+04	9.10E+00	1.05E+04	8.60E+00	1.06E+04	8.20E+00	**1.05E+04**	9.50E+00	1.06E+04	1.29E+01	1.05E+04	1.75E+01
PS	inst1	**2.40E+03**	4.14E+02	3.34E+03	1.19E+01	3.35E+03	2.69E+01	3.35E+03	1.37E+01	4.96E+03	9.42E+03	3.54E+03	1.57E+04
	inst2	**3.90E+02**	2.55E+01	3.85E+02	2.05E+01	7.15E+02	6.24E+02	3.90E+02	1.94E+01	1.10E+03	8.97E+02	1.28E+03	1.64E+03
	inst3	**1.93E+01**	2.50E+00	2.13E+01	3.00E+00	4.45E+01	7.07E+01	2.12E+01	3.10E+00	2.89E+01	8.40E+00	3.10E+01	1.09E+01
	inst4	**2.18E+01**	2.55E+00	2.35E+01	2.80E+00	5.46E+01	6.31E+01	2.40E+01	2.60E+00	4.03E+01	2.38E+01	3.67E+01	8.80E+00
	inst5	**2.19E+01**	2.00E+00	2.47E+01	3.00E+00	3.53E+01	7.70E+00	2.46E+01	3.00E+00	3.37E+01	7.00E+00	2.85E+01	1.05E+01
VRP	inst1	**2.07E+04**	3.00E+00	2.61E+04	5.35E+02	2.54E+04	5.05E+02	2.51E+04	4.45E+02	2.09E+04	3.99E+02	2.07E+04	4.42E+02
	inst2	**1.34E+04**	1.91E+01	1.67E+04	4.28E+02	1.67E+04	4.14E+02	1.59E+04	6.08E+01	1.37E+04	4.92E+02	1.36E+04	4.22E+02
	inst3	**5.34E+03**	1.13E+01	6.73E+03	2.67E+02	6.77E+03	1.55E+02	6.10E+03	3.30E+02	5.35E+03	2.19E+01	5.34E+03	4.64E+01
	inst4	**1.43E+04**	1.42E+01	1.83E+04	5.33E+02	1.79E+04	3.38E+02	1.76E+04	4.04E+02	1.46E+04	4.76E+02	1.45E+04	3.72E+02
	inst5	**1.49E+05**	2.09E+03	3.61E+05	4.11E+03	3.38E+05	1.71E+04	3.58E+05	5.40E+03	1.67E+05	2.38E+04	1.61E+05	3.48E+04
TSP	inst1	1.13E+05	1.02E+03	1.36E+05	3.76E+03	**1.12E+05**	8.41E+02	1.34E+05	3.31E+03	1.13E+05	1.00E+03	1.13E+05	1.78E+03
	inst2	**6.99E+03**	1.66E+01	7.01E+03	1.62E+01	7.00E+03	1.28E+01	7.04E+03	1.27E+01	7.02E+03	2.78E+01	**6.99E+03**	2.48E+01
	inst3	4.32E+04	1.24E+02	4.35E+04	9.46E+01	4.33E+04	8.65E+01	4.37E+04	1.32E+02	4.34E+04	1.74E+02	**4.31E+04**	2.07E+02
	inst4	**9.15E+03**	2.66E+01	9.16E+03	2.05E+01	9.15E+03	1.65E+01	9.19E+03	1.98E+01	9.17E+03	3.32E+01	9.17E+03	4.02E+01
	inst5	**5.97E+04**	2.55E+02	6.07E+04	2.22E+02	5.99E+04	2.77E+02	6.13E+04	3.21E+02	5.98E+04	3.61E+02	5.98E+04	4.25E+02

6.Conclusion

Hyper-heuristic algorithms are effective approaches for solving various complex problems by searching through heuristics space to find the best possible solution. LLH is a set of heuristics that are applied to generate new solutions. The effectiveness of the hyper-heuristic framework depends on the selection method used for LLH and the acceptance criterion used to evaluate the new solution. The proposed algorithm is applied in multi-level hyper-heuristic (MHH) framework. The proposed MMHH enhances the level of generality by adopting different methods of reward-selected HH-Alg, and using various acceptance criteria mechanisms. The proposed algorithm is evaluated using six problem domains of HyFlex Framework. The experimental results demonstrate that the MMHH algorithm outperforms other state-of-the-art hyper-heuristics in most test problem such as MHHA, MHHN, DMAB, FRRMAB, and DQN. Future work involves tuning the parameter in proposed MMHH, apply MMHH to extension problems and changing the pool of HH-Alg in the highest-level heuristic. These improvements will further enhance the efficiency and applicability of the proposed algorithm in solving various complex problems.

References

[1] A. Kafafy, A. Awaad, N. El-Hefnawy and O. A. Raouf, "Multi-level Hyper-Heuristic for Combinatorial Optimization Problems," *International Journal of Intelligent Engineering and Systems,* vol. 15, 2022.

[2] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan and J. R. Woodward, "A classification of hyper-heuristic approaches," Handbook of metaheuristics, p. 449–468, 2010.

[3] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan and R. Qu, "Hyper-heuristics: A survey of the state of the art," Journal of the Operational Research Society, vol. 64, p. 1695–1724, 2013.

[4] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan and J. R. Woodward, "Exploring Hyper-heuristic Methodologies with Genetic Programming," in *Computational Intelligence: Collaboration, Fusion and Emergence*, C. L. Mumford and L. C. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, p. 177–201.

[5] G. Mweshi and N. Pillay, "An improved grammatical evolution approach for generating perturbative heuristics to solve combinatorial optimization problems," *Expert Systems with Applications,* vol. 165, p. 113853, 2021.

[6] P. Ross, "Hyper-heuristics," Search methodologies: introductory tutorials in optimization and decision support techniques, p. 529–556, 2005.

[7] X. Hao, R. Qu and J. Liu, "A unified framework of graph-based evolutionary multitasking hyper-heuristic," IEEE Transactions on Evolutionary Computation, vol. 25, p. 35–47, 2020.

[8] J. H. Drake, A. Kheiri, E. Özcan and E. K. Burke, "Recent advances in selection hyper-heuristics," European Journal of Operational Research, vol. 285, p. 405–428, 2020.

[9] F. Zhao, S. Di, J. Cao, J. Tang and others, "A novel cooperative multi-stage hyper-heuristic for combination optimization problems," Complex System Modeling and Simulation, vol. 1, p. 91–108, 2021.

[10] D. Oliva, M. S. R. Martins, S. Hinojosa, M. A. Elaziz, P. V. dos Santos, G. da Cruz and S. J. Mousavirad, "A hyper-heuristic guided by a probabilistic graphical model for single-objective real-parameter optimization," International Journal of Machine Learning and Cybernetics, vol. 13, p. 3743–3772, 2022.

[11] N. R. Sabar, M. Ayob, G. Kendall and R. Qu, "A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," IEEE transactions on cybernetics, vol. 45, p. 217–228, 2014.

[12] S. S. Choong, L.-P. Wong and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," Information Sciences, vol. 436, p. 89–107, 2018.

[13] N. R. Sabar and G. Kendall, Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems, Inf. Sci. (Ny)., vol. 314, pp. 225–239, 2015.

[14] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems, IEEE Trans. Evol. Comput., vol. 19, no. 3, pp. 309–325, 2015.

[15] E. Özcan , B. Bilgin and E.E. Korkmaz , "A comprehensive analysis of hyper-heuristics", Intell. Data Anal. 12 (1) (2008) 3–23 .

[16] A. Kheiri and E. Özcan, "An iterated multi-stage selection hyper-heuristic," European Journal of Operational Research, vol. 250, p. 77–90, 2016.

[17] A. Kheiri and E. Özcan, "A hyper-heuristic with a round robin neighbourhood selection," in Evolutionary Computation in Combinatorial Optimization: 13th European Conference, EvoCOP 2013, Vienna, Austria, April 3-5, 2013. Proceedings 13, 2013.

[18] A. Kheiri, E. Özcan and A. J. Parkes, "A stochastic local search algorithm with adaptive acceptance for high-school timetabling," Annals of Operations Research, vol. 239, p. 135–151, 2016.

[19] A. Dantas and A. Pozo, "Online Selection of Heuristic Operators with Deep Q-Network: A Study on the HyFlex Framework," in Intelligent Systems: 10th Brazilian Conference, BRACIS 2021, Virtual Event, November 29–December 3, 2021, Proceedings, Part I 10, 2021.

[20] N. Pillay and R. Qu, Hyper-heuristics: theory and applications, Springer, 2018.

[21] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic and others, "Hyflex: A benchmark framework for cross-domain heuristic search," in European Conference on Evolutionary Computation in Combinatorial Optimization, 2012.

[22] DaCosta, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 913–920. GECCO '08, Association for Computing Machinery, New York, NY, USA (2008)

[23] Li, K., Fialho, A., Kwong, S., Zhang, Q.: Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. 18(1), 114–130 (2014)