Received: April 12, 2022. Revised: June 28, 2022.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multi-level Hyper-Heuristic for Combinatorial Optimization Problems

Ahmed Kafafy[1]* Asmaa Awaad[2] Nancy El-Hefnawy[3] Osama Abdel Raouf[2]

[1]*Operations Research & DSS Dept., Faculty of Computers and Information, Menoufia University, Menoufia, Egypt*
[2]*Faculty of Artificial intelligence, Menoufia University, Menoufia, Egypt*
[3]*Faculty of Computers and Information, Tanta University, Tanta, Egypt*
* Corresponding author's Email: ahmedkafafy80@gmail.com

Abstract: Hyper-heuristics are considered as one of the most popular search methods which can solve NP-hard problems. They aim to achieve level of generality of search techniques for solving a wide variety of problem domains. Hyper-heuristic framework involves two levels, high-level, and low-level heuristics. The high-level heuristic is responsible for selecting and applying an appropriate low-level heuristic to generate solutions and deciding whether to accept or reject the new solution. Low-level heuristics are a set of problem-specific heuristics. In this paper, we propose to improve the performance through adding a new level strategy to the hyper-heuristic framework. The highest-level strategy adopts the roulette wheel selection mechanism to select the appropriate hyper-heuristic according to its performance during the search process. The highest-level strategy selects the appropriate algorithm from a predefined set of hyper-heuristic algorithms to improve the generated solution. The performance of the proposed approach has been compared with one of the most recent methods as well as with other hyper-heuristics that used as components of the proposed approach. The results have been carried on six of commonly used benchmark datasets. The results show the effectiveness of the proposed framework. Hence it outperforms the other methods in five of the six benchmarks.

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

1. Introduction

Optimization problems is involved in many real-life situations, the goal of an optimization problem is to find the efficient solution from all feasible solutions. According to the decision variables domain, Optimization problems can be classified as discrete or continuous [1]. The combinatorial optimization problems (COPs) are a type of discrete optimization problems. Combinatorial Optimization Problems (COPs) arise in many real applications such as production planning, scheduling, routing, and resource allocation…etc. [2]. COPs have very large and often difficulty-constrained search space which makes the solution process more complicated, costly to solve, and their modelling becomes a very complex task [3]. So, Many COPs can be classified as NP-hard problems. Exact methods guarantee the optimality. But they are limited to small scale problems and fail to find the optimal solution for complex problems in a reasonable time [2]. Thus, adopting heuristics to solve these problems for good enough solutions within a realistic amount of time is a viable alternative. Meta-heuristics adopt some high-level control strategies which guarantee obtaining good enough solutions within an acceptable time. Many research works discuss this issue before [2, 4, 5]. Many meta-heuristics are adopted to tackle optimization problems. They include Genetic Algorithms (GA), Simulated Annealing (SA), Tabu Search (TS), Particle Swarm Optimization (PSO), differential evolution (DE), and GRASP…etc [6]. Selecting the most appropriate technique to the problem with determining its optimal structure and parameters are big challenges [7]. Trials and error arise as the most straightforward method to get the most suitable meta-heuristic and to identify its optimal structure and parameter values. The manually designed and tuned strategies are proposed for many meta-heuristics to obtain promising results in certain problem instances, in other cases, these

strategies typically fail to be improved to tackle other problems. There have been several attempts to build automatic search methodologies that can achieve good results through many problem domains and/or instances. One of these methodologies reflects in hyper-heuristics [8]. They are search techniques that can generate solutions for a wide range of problem domains instead of using specific technique to each problem instance. Instead of working in the solution space, hyper-heuristics work in the heuristic search space [3]. The essential role played by hyper-heuristics is increasing the level of generality in building frameworks based on the strengths and identifying the limitations of various heuristics. This role is often achieved during the search process through adopting an automatic manner to select and/or generate heuristics [8]. The most popular hyper-heuristic framework involves both high-level strategies and low-level heuristics (LLHs). The high-level strategy combines a selection mechanism and move acceptance methods. The selection mechanism elects a heuristic from a predefined set of LLHs. and applies it to a current solution. Then, a move acceptance is adopted to decide either accepting or rejecting the new solution. In case of accepting the new solution, the current solution is replaced by the new one, then the search proceeds iteratively. However, the selection mechanism is applied during the stages of the search process [9, 10]. The LLHs include a set of heuristics for a particular problem domain. They are also considered as constructive or perturbative. The effectiveness of a hyper-heuristic framework is typically related to how the high-level strategy is designed [11]. Moreover, the different combinations of selection mechanism and move acceptance affects the overall performance of a hyper-heuristic [11, 12]. Based on this concept, this research tends to develop a new multilevel hyper-heuristic framework that guarantees high level of generality to tackle many problem domains. In this work, the proposed multi-level hyper-heuristics framework consists of three levels. The highest layer is responsible for selecting among a set of algorithms to solve a wide range of problem domains instead of using specific algorithm to each problem. In this layer, a roulette wheel-based selection mechanism for selection among algorithms is adopted. The multi-level hyper-heuristics framework is proposed to support automatically designing and selecting a hyper-heuristic algorithm at different stages during the search process. This leads to increasing the generality and self-adaptation by adopting the appropriate algorithm in the suitable stage in the search process. Furthermore, a set of Benchmarks involves six problem domains from the Hyper-heuristic Flexible (HyFlex) [13] are used to evaluate the performance of the proposed approach. Experimental results indicate that the proposed multi-level hyper-heuristics are efficient. The rest of the paper is presented as follows: after introduction, Section 2 covers a review on hyper- heuristics, reviews the selection hyper-heuristics related to this work and introduces different move acceptance. The components of the proposed multi-level hyper-heuristic framework is described in Section 3. Section 4 presents the experimental results. Finally, Conclusions and points for father research are presented in Section 5.

2. Related work

Based on heuristic space, hyper-heuristics are categorized into two classes, Selection hyper-heuristics and generation hyper-heuristics [8, 14]. A generation hyper-heuristic combines the existing heuristic elements to produce new LLHs. Whereas a selection hyper-heuristic decides which LLH should be adopted at any search stage. Selection hyper-heuristics class comprises most of the current hyper-heuristic literature [15]. The main role of this class is to integrate the benefits of multiple LLHs through adopting the most convenient heuristic in the suitable stage of the search process. The LLH selection mechanism and the move acceptance are both components of a classical selection hyper-heuristic [11]. Simple Random, Greedy, Random Permutation, Tabu Search, Choice Function, Harmony Search, and Reinforcement Learning are some of the available LLH selection methods [16, 17]. While move acceptance strategies involves All Moves, Only Improvement, Simulated Annealing, and Late Acceptance [16]. The current move acceptance and LLH selection approaches are summarized in Table 1 and 2.

Many studies on multi-stage selection hyper-heuristics are provided. The authors in [25] proposed an iterated multi-stage hyper-heuristic. They integrate a Dominance-based heuristic selection technique with a Random Descent hyper-heuristic and Naive Move Acceptance (DRD). In dominance-based selection, a greedy approach is adopted to find the suitable LLHs while considering the trade-off between the change in the objective value and the number of iterations required to obtain that result. In the second stage, a random descent hyper-heuristic is adopted to improve the quality of the solution on hand using the suitable LLHs from the previous stage. If the second stage becomes stagnant, the first stage is repeated with a certain probability to obtain a new subset of LLHs.

Table 1. Move acceptance methods

Move acceptance	**Related works**	**Description**
Only Improvement	[18]	accepts only improving solution
All Moves	[19]	accepts all solution
Metropolis acceptance	[20]	Accepted the improving solutions, and accepted worst one with probability
Simulated Annealing	[21]	Aanother type of Metropolis acceptance
Late Acceptance	[22], [23]	Better or equal quality solutions are accepted solutions as compared to previous iterations' solutions.
Naive Acceptance	[24]	Always accepted the improving solutions, and accepted worst solutions within 50% probability

Table 2. Low level heuristics selection methods

LLH selection methods	Related works
Simple random	[22], [17]
Random Gradient	[17]
Random Permutation	[17]
Greedy	[17]
Choice function	[19], [17]
Tabu search	[4], [26]
Harmony search (HM)	[27]
Ant-inspired algorithms	[28] [29]
Reinforcement learning	[30], [31]
Multi-Armed Bandit (MAB) selection	[32], [33]
Monte Carlo Tree Search (MCTS)	[34]
Hidden Markov Model selection	[35]

The authors in [36] introduce the Robinhood hyper-heuristic (RHH) which combines three hyper-heuristics. In RHH, a heuristic selection mechanism based on round-robin strategy as well as three acceptance criteria are employed. In the selection mechanism, the mutational and ruin and re-create heuristics are applied followed by crossover and hill climbing heuristics. Each LLH is assigned equal time. For move acceptance criteria, RHH adopts only improving, improving or equal, and adaptive acceptance methods in which the good move is always accepted, and bad moves are accepted with a modified probability. A fixed amount of time is allowed for each hyper-heuristic in RHH. RHH provides competitive results and outperforms the hyper-heuristics (HyFlex).

Kheiri *etal* [9] proposed a technique called a Hyper-heuristic Search Strategies and Timetabling (HySST) which involves two stages, diversification, and intensification stage. Diversification stage combines The Simple Random mutational operator with Adaptive Threshold move acceptance. Whereas intensifications stage combines Simple Random hill-climbing operator with Accept All Moves. HySST switches between the two stages form one stage to the other if no improvement is achieved in a given stage after a certain duration. A set of real-world instances are used to test HySST which competed at the three rounds of the (ITC 2011) competition. HySST achieve the best solutions for three instances in round 1 and took the second place in rounds 2 and 3.

In [37], the authors developed a hyper-heuristics framework called "An iterated multi-stage selection hyper-heuristic" (MSHH). During the search, MSHH adopts several hyper-heuristics at different stages. In MSHH, the authors provide a multi-stage level to control transition and manage information exchange among multiple hyper-heuristics. The hyper-heuristics selection mechanism involves two stages. In the first stage, a greedy strategy is applied to determine the scores of set of LLH. In the next stage, the roulette wheel is applied to select one of these heuristics according to the assigned scores at each step. An adaptive threshold move acceptance method is used in both stages. According to the experimentation, MSHH outperforms five other hyper-heuristics on HyFlex problem domains.

In [38], a dynamic heuristic set selection (DHSS) is introduced. In this approach, a dominant technique is utilized to pick the active heuristic set at several points in the hyper-heuristic lifecycle. The DHSS approach was evaluated on the benchmark set for the CHeSC cross-domain hyper-heuristic challenge. DHSS improves the performance of the best performing hyper-heuristic for this challenge.

From the previous discussion, its noted that the previous techniques suffer from a limited level of generality. They also lack the ability to effectively utilize the suitable heuristic and/or acceptance criteria in different search situations according to the problem domain. Thus, the need to develop a selection mechanism to control this process and to effectively utilize the suitable heuristic and/or the acceptance criteria is so critical.

In this work, a new multi-level hyper heuristic framework with three levels is developed. The new level consists of a selection mechanism for hyper-heuristics and acceptance criteria algorithm. The hyper-heuristic selection mechanism combines three hyper-heuristics algorithms. The advantage of the proposed algorithm is to combine different hyper-heuristics with different selection strategies and different acceptance criteria. This leads to high level of generality and reusability. The proposed framework is explained in detail in the next section.

3. The proposed framework

In this section, the frameworks of both traditional hyper-heuristic and the proposed multilevel hyper-heuristic are discussed in detail in the following subsections.

3.1 The traditional hyper-heuristic framework

In classical hyper-heuristics, the framework involves two layers, the first layer contains high-level heuristics, and the second layer contains Low-Level Heuristics (LLHs) as depicted in Fig. 1 (2nd Layer & 3rd Layer respectively). The high-level heuristic employs two components, heuristic selection strategy, and move acceptance methods. Traditional hyper-heuristic framework works as follows: The high-level heuristic selects one LLH from a set of predefined heuristics that are related to the problem. Then, the selected LLH is applied to the current solution to get a new one. After that, the acceptance criteria are applied to accept or reject the new solution. In case of acceptance, the new solution replaces the current one and the iteration is repeated until stopping criteria is met.

3.2 The proposed multi-level hyper-heuristic framework

The proposed framework provides multi-level hyper-heuristic algorithms to automatically select hyper-heuristic algorithms and acceptance algorithm criteria. It adds a new upper level called the highest-level heuristic as shown in Fig. 1. This layer comprises two components, algorithm selection and acceptance criteria. The main idea is to divide the search process to a set of consecutive learning periods (LP), the performance of each algorithm in the current LP affects its probability of selection in the next LP, these probabilities are gradually adapted

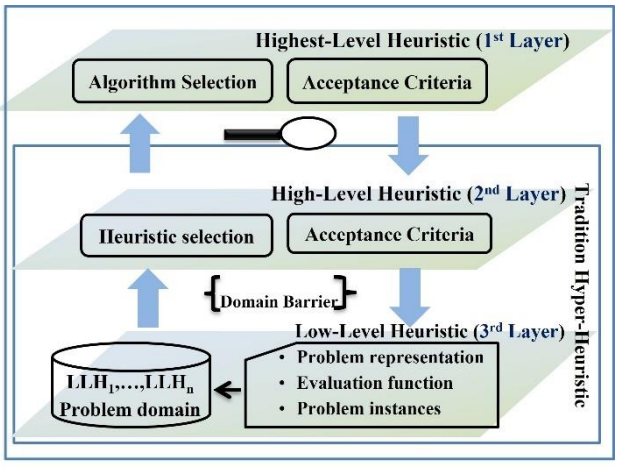

Figure. 1 Multilevel hyper heuristic framework

during evolution. In the initial LP, each algorithms have equal chance to be chosen i.e Alg_k has a probability $P_k=1/K$ where K is the number of algorithms in the pool. The proposed framework works as shown in Fig. 2. In each LP, it starts with selecting one algorithm from a pool of algorithms. The selected algorithm is applied to generate a new solution. Then, the new solution is evaluated. The algorithm selects a heuristic from LLHs and applies the selected heuristic then applies the acceptance criteria to the solution to accept or reject. The selected algorithm is rewarded based on the amount of improvement. Then, a new iteration starts. The reward of each algorithm is assigned as Eq. (1).

$$Reward_{k,l} = \begin{cases} Suc_{k,l} + \epsilon & if\ Suc_{k,l} > 0 \\ \epsilon & Otherwise \end{cases} \quad (1)$$

Where $Suc_{k,l}$ measures the success of algorithm k to generate accepted solutions at learning period l, and ϵ is a small value used to avoid null success and to keep a small chance to select the algorithms that achieves no improvement in the further iterations. We apply two methods for rewarding algorithm. The first method is to reward based on the amount of improvement achieved by the new solutions ($Suc_{k,l}$ registers the improvement amount) as in Eq. (2) below. The second one is based on if there is any improvement achieved by the solution or not ($Suc_{k,l}$ = acceptance rate) as in Eq. (3).

$$Suc_{k,l} = \sum_k Improves_{k,l} \ \forall\, k, l \quad (2)$$

$$Suc_{k,l} = \frac{Accpted_{k,l}}{Invokes_{k,l}} \ \forall\, k, l \quad (3)$$

Where $Improves_{k,l}$ is the amount of improvement achieved by Algorithm k, $Accpted_{k,l}$ is the number of accepted solutions generated by Algorithm k, *and* $Invokes_{k,l}$ represents the number of invokes of Algorithm k during the LP l. Based on the adopted rewarding method, there are two variants of the proposed approach, ML-HHA that uses improvement amount and ML-HHN which adopts the acceptance rate. At the end of each LP l, the probability for each algorithm k is adapted based on the value of reward to be used in the next LP $l+1$ as in Eq. (4).

$$P_{k,l+1} = Reward_{k,l} / \sum_{\forall j} Reward_{j,l} \quad (4)$$

The proposed framework allows using multiple hyper-heuristic algorithms interchangeably. Adopting many algorithms enables us to comprise

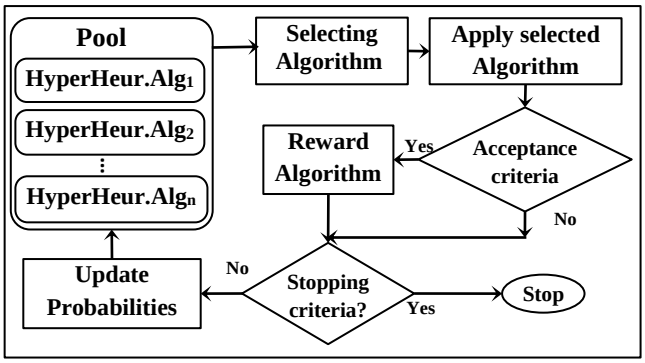

Figure. 2 Highest-level strategy

Algorithm 1:The proposed Multilevel-hyper-heuristic

Begin:

1. $Set_{HH} \leftarrow \{HH_1, \dots, HH_K\}$ //*define the hyper-heuristics*
2. $P_{HH_k} \leftarrow \frac{1}{K} \ \forall k \in \{1, \dots, K\}$ //*Assign equal Prob.*
3. $CurSol \leftarrow IntialzeSolution$ // *Initial solution*
4. **While** Stopping Criteria not met **do** //Main Loop
5. **For** each iteration *i* in LP l // Learning Period
6. $Alg_i \leftarrow$ RoulletWheelSelction(Set_{HH}, P_{HH_k})
7. NewSol $\leftarrow ApplyAlgoritm(Alg_i, CurSol)$
8. $CurSol \leftarrow AcceptancCreteria(NewSol)$
9. $Rewards \leftarrow AssignRewards(Alg_i, CurSol, NewSol)$
10. **EndFor**
11. **For** $\boldsymbol{each}\ k \in \{1, \dots, K\}$ **do**:
12. $Reward_k \leftarrow UpdateRewards(Rewards)$ // Eq.1
13. $P_{HH_k} \leftarrow UpdateProbablites(Reward_k)$ // Eq.4
14. **EndFor**
15. **EndWhile**

End.

advantages of each, such as to get powerful of different heuristic selection mechanism. Also, this enables us to control different sets of LLHs cooperatively and different acceptance criteria. The new Layer permits the switch among available hyper-heuristics to automatically control the selection of hyper-heuristics at different stages during the search process. This achieves high level of generality and reusability of the proposed framework. This work introduces a multi-level hyper-heuristic that utilizing three interacting hyper-heuristic algorithms interchanged and controlled by the highest-level as provided in Fig. 2. The details of the proposed approach are described in pseudocode presented in Alg. (1). The algorithm starts with a predefined set of hyper-heuristic algorithms with equal chance for selection (lines 1-2). The main loop of the algorithm is starts from line-4 to the end. At each LP, The Algorithms are randomly selected and rewarded according to its performance for some iteration (lines 5 to10). In lines 6 to 7, a hyper-heuristic Alg_i is randomly selected by roulette wheel and applied to generate a new solution $NewSol$. Then, the acceptance criteria is applied and in line 8, the selected algorithm Alg_ithat improves the solution or just remain same solution is rewarded by assigning a reward value (line 9) using one of the two methods mentioned above in Eq.2 and Eq.3. At the end of each learning period l, the total reward for each algorithm Alg_k is calculated for each $k \in \{1, \dots, K\}$ (lines 12). According to the total reward for each Alg_k in the pool, the probability of each algorithm Alg_k is recalculated based on Eq. (4) (line 13). The whole process is repeated, and the roulette wheel selection randomly selects a hyper-heuristic algorithm based on the new probability associated with each algorithm.

As mentioned before, the proposed framework involves three layers, the highest-level strategy, high-level heuristic, and the Low-level heuristics. These layers are addressed as follows:

3.2.1. Highest level strategy

Highest level strategy is the upper hand of the proposed framework to control high-level heuristic and low-level heuristic. This layer consists of two component, algorithm selection and acceptance criteria as shown in Fig. 1 (1st Layer). This level enables to achieve better performance than the previous hyper-heuristics framework and improve the level of generality of the proposed framework.

3.2.1.1. Algorithm selection

In this phase, the proposed approach begins with selecting one from a pool contains three algorithms. These algorithms are "An Iterated Multi-stage Selection Hyper-heuristic" (MSHH) [37], "Robinhood hyper-heuristic" (RHH) [36] , and "Hyper-heuristic Search Strategies & Timetabling" (HYSST) [9]. The roulette-wheel based selection is adopted as an algorithm selection mechanism. Initially all algorithms have the same probabilities. These probabilities decide which algorithm is selected at each decision point. After that, the selected algorithm is applied to find a new solution. Then, the new solution is evaluated, and the algorithm is rewarded based on the solution obtained. After each LP, the probabilities are adapted according to the reward for each algorithm during the learning period as shown in Fig. 2.

3.2.1.2. Acceptance criteria

The role of the acceptance criterion is to decides accepting or rejecting the solution generated by the selected algorithm [11]. Acceptance criterion accepts only the same or better solution. Here, two ways to reward the algorithm are proposed when a new solution is accepted. First way is to get amount of improvement in the solution and reward algorithm based on this amount. The other way is to get number of times that algorithm improves the solution or keep solution same. These values are used when credit assignment is applied to calculate the probability of each algorithm.

3.2.2. High level heuristic

The High-Level heuristic comprises two components, the first component selects a LLH, and a second one for move acceptance to decide whether the solution obtained by the selected LLH should be accepted or not. At each stage in the search process, heuristic selection module selects LLH to be applied. The proposed approach is capable of handling new problem domains without any modifications.

3.2.3. Low level heuristic

Low-level heuristics are used to create new solutions. LLHs are usually specified for a certain problem domain and therefore they are problem specific [39].

4. Experimental results

Here, the experimental design and results are presented in the following subsections.

4.1 Experimental design & parameters

The performance of the proposed multilevel hyper-heuristic is verified against some of previously proposed hyper-heuristics such as An Iterated Multi-stage Selection Hyper-heuristic(MSHH) [37], Robinhood hyper-heuristic(RHH) [36] and Hyper-heuristic Search Strategies & Timetabling (HYSST) [9] which used as a pool components in the proposed framework. Besides, the recently developed in [38], A dynamic heuristic set selection (DHSS) is also used in verification. The performance of the compared hyper-heuristics is evaluated across six HyFlex problem domains. These Problems are MAX-SAT, Bin Packing (BP), Flow-shop (FSH), Personnel Scheduling (PS), VRP, and TSP. Each of these problem domains has five instances which are labelled as inst1 to inst5. The proposed approach and its competitors are run on the five instances for each problem domain about 31 different independent runs. The number of iterations in each learning Period (l) is assigned to 10.

4.2 Computational results

As discussed above, the proposed approach has two different variants according to the adopted rewarding strategy. The first variant named MHHA adopts the reward strategy based on the amount of improvement achieved by the algorithm during the learning period. Whereas the second variant named MHHN adopts the reward strategy based on the number of accepted solutions generated by the algorithm at the learning period. These two proposed variants have experimented with the recently developed DHSS [38] and the hyper-heuristics components included in the pool at the highest-level, denoted as RHH, MSHH, and HYSST. Table 3 (SAT row) and Fig. 3 show the average results of SAT domain. On average, the proposed two variants MHHA and MHHN outperform the others RHH, MSHH, HYSST and DHSS and this performance is statistically significant for all instances in SAT domain.

For Bin Packing (BP) problem domain, the average results are shown in Table 3 and Fig. 4. It is so clear the superiority of our proposals MHHA against their competitors DHSS, RHH, MSHH and HYSST for all problem instants. The other variant MHHN achieves the second best performance in these instances. Except in instance 5, the competitor MSHH achieves the second best performance.

In the Flow-shop (FSH) problem domain, Table 3 (FSH row) and Fig. 5 show the average results. Although DHSS outperforms the others in all test instances, the proposed variant MHHA still have the second-best performance in inst3 and inst5 and the 3rd best performance in the other instance. Also, it is noted that MSHH achieves the second-best performance in ints1, inst2 and inst4. In this case, it

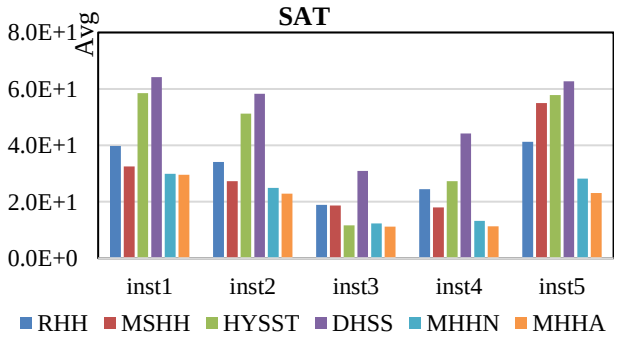

Figure. 3 Average results of SAT problem for 5 instances

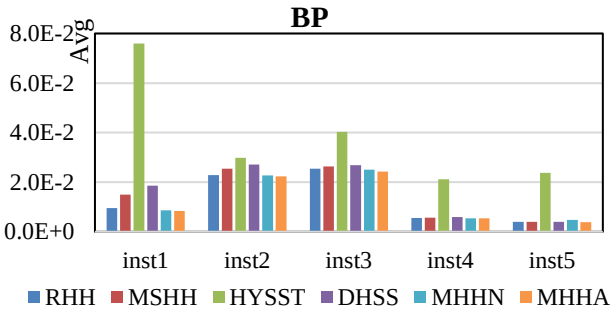

Figure. 4 Average results of Bin-Packing for 5 instances

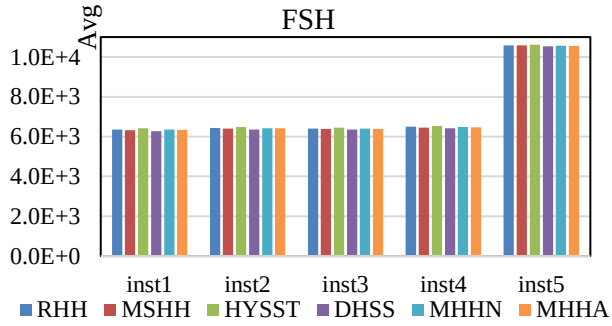

Figure. 5 Average results of Flow Shop for 5 instances

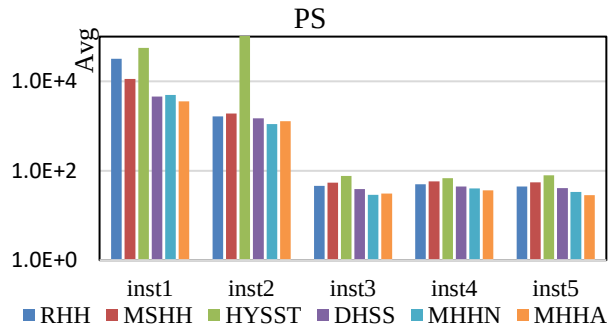

Figure. 6 Results of personnel schedule for 5 instances

seems MSHH more appropriate for FSH domain. Adopting the proposed MLHH gives the chance to other algorithms to be applied in some stages of the search process. Thus, it may deteriorate the performance in this problem compared to the pure MSHH. This meaning can be observed from Fig. 10 which indicates the performance of each component in each learning period during the search.

On the Personnel Scheduling (PS) problem domains, Table 3 (PS row) and Fig. 6 show the average results obtained over 31 independent runs. From the average results, it is so clear the superiority of the proposed MHHA variant in all test instances over the other competitors except in inst4 in which the proposed MHHN has the best performance. Thus, our proposals outperform all other competitors in this

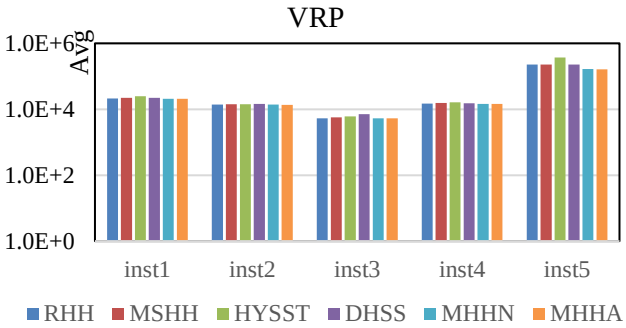

Figure. 7 Average results of Flow Shop for 5 instances

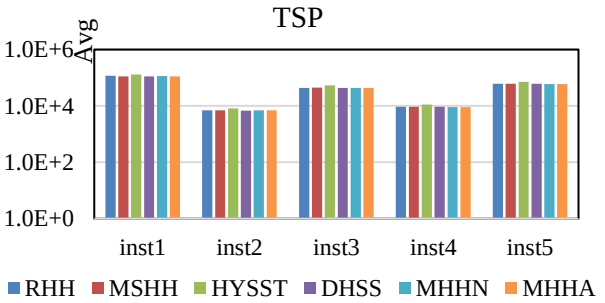

Figure. 8 Average results of Flow Shop for 5 instances

problem domain. For VRP problem domain, Table 3 (VRP row) and Fig. 7 depicts the average results. Here, the best performance is achieved by the proposed MHHA variant on all test instances. the other proposed MHHN provide the second-best performance in all test instances except instance 3 in which RHH achieves the second-best performance.

On TSP problem domain, the proposed two variants MHHA and MHHN manage to provide the best average results in three instances from inst3 to inst5 and achieves the second-best performance in the first two instances inst1 and inst2. However, DHSS shows the best performance on average in Inst1 and inst2 as viewed in Table 3 (TSP row) and Fig. 8.

Fig. 9 and 10 show the competition among the three hyper-heuristics components RHH, MSHH and HYSST included in the pool for SAT and FSH domains respectively. They show the probability of selection for each hyper-heuristics in deferent learning periods. In Fig. 9, the cooperation among three hyper-heuristics algorithms involved in algorithm pool in highest level is shown. This allows the proposed MHHA & MHHN to achieve the best performance in SAT five instances. In some cases, after a few trials one algorithm win the competition with some difference in some other cases there is a variety of success in competition.

Finally, Fig. 11 shows the box plots of the

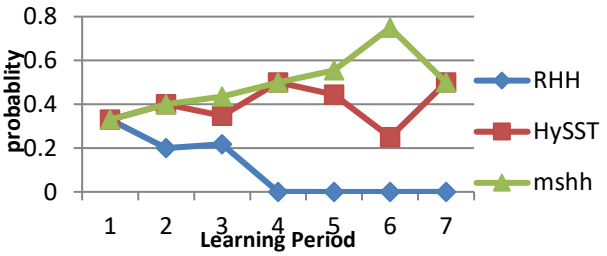

Figure. 9 Competition among three algorithms in SAT

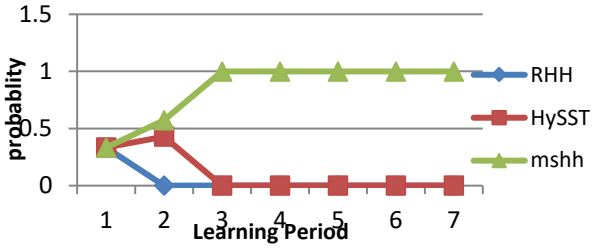

Figure. 10 Competition among three algorithms in FSH

Table 3. Average and standard deviations Result of All test problems over 31 independent runs

		RHH		MSHH		HYSST		DHSS		MHHN		MHHA	
	Dom	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**
SAT	inst1	3.98E+01	4.76E+00	3.25E+01	2.05E+00	5.85E+01	1.00E+01	6.41E+01	7.15E+00	2.99E+01	4.42E+00	**2.95E+01**	7.40E+00
	inst2	3.42E+01	4.25E+00	2.73E+01	2.19E+00	5.13E+01	8.93E+00	5.83E+01	7.48E+00	2.49E+01	3.37E+00	**2.29E+01**	2.96E+00
	inst3	1.89E+01	3.08E+00	1.87E+01	2.23E+00	1.16E+01	4.18E+00	3.09E+01	5.06E+00	1.24E+01	3.10E+00	**1.12E+01**	3.35E+00
	inst4	2.45E+01	9.73E+00	1.80E+01	5.34E+00	2.74E+01	1.34E+01	4.42E+01	5.92E+00	1.33E+01	7.24E+00	**1.13E+01**	4.65E+00
	inst5	4.12E+01	1.03E+01	5.49E+01	4.04E+00	5.78E+01	1.24E+01	6.27E+01	8.08E+00	2.82E+01	1.18E+01	**2.31E+01**	1.12E+01
BP	inst1	9.58E-03	2.06E-03	1.50E-02	2.81E-03	7.59E-02	7.02E-03	1.86E-02	3.82E-03	8.71E-03	1.66E-03	**8.35E-03**	1.28E-03
	inst2	2.29E-02	4.73E-04	2.55E-02	1.57E-03	2.99E-02	2.02E-03	2.71E-02	2.44E-03	2.27E-02	6.53E-04	**2.24E-02**	5.51E-04
	inst3	2.55E-02	7.68E-04	2.64E-02	1.31E-03	4.03E-02	1.08E-02	2.68E-02	1.29E-03	2.51E-02	9.64E-04	**2.43E-02**	9.30E-04
	inst4	5.61E-03	4.95E-04	5.65E-03	5.51E-04	2.12E-02	1.56E-03	5.91E-03	1.36E-03	5.48E-03	5.70E-04	**5.45E-03**	2.00E-04
	inst5	4.00E-03	2.65E-18	4.00E-03	2.65E-18	2.38E-02	1.09E-03	4.03E-03	8.28E-04	4.77E-03	1.80E-03	**3.94E-03**	1.08E-03
FSH	inst1	6.36E+03	1.58E+01	6.32E+03	1.89E+01	6.41E+03	1.92E+01	**6.28E+03**	1.27E+01	6.35E+03	1.58E+01	6.34E+03	1.19E+01
	inst2	6.43E+03	1.50E+01	6.40E+03	1.76E+01	6.48E+03	2.02E+01	**6.36E+03**	1.02E+01	6.42E+03	1.23E+01	6.41E+03	1.43E+01
	inst3	6.41E+03	1.66E+01	6.39E+03	2.10E+01	6.45E+03	1.86E+01	**6.36E+03**	1.16E+01	6.40E+03	1.29E+01	6.38E+03	7.93E+00
	inst4	6.49E+03	1.26E+01	6.45E+03	2.80E+01	6.54E+03	1.95E+01	**6.41E+03**	1.07E+01	6.48E+03	1.61E+01	6.46E+03	9.58E+00
	inst5	1.06E+04	1.46E+01	1.06E+04	1.03E+01	1.06E+04	6.33E+00	**1.05E+04**	1.82E+01	1.06E+04	1.29E+01	1.05E+04	8.21E+00
PS	inst1	3.21E+04	2.81E+04	1.14E+04	1.57E+04	5.55E+04	0.00E+00	4.54E+03	5.02E+03	4.96E+03	9.42E+03	**3.54E+03**	1.09E+03
	inst2	1.65E+03	1.30E+03	1.90E+03	1.21E+03	2.20E+05	0.00E+00	1.48E+03	8.12E+02	1.10E+03	8.97E+02	**1.28E+03**	1.64E+03
	inst3	4.62E+01	4.94E+00	5.45E+01	7.56E+00	7.66E+01	5.06E+00	3.89E+01	6.47E+00	**2.89E+01**	8.40E+00	3.10E+01	8.76E+00
	inst4	5.02E+01	7.43E+00	5.74E+01	4.06E+00	6.82E+01	6.07E+00	4.40E+01	8.52E+00	4.03E+01	2.38E+01	**3.67E+01**	8.77E+00
	inst5	4.47E+01	4.07E+00	5.48E+01	9.98E+00	7.91E+01	4.62E+00	4.10E+01	5.50E+00	3.37E+01	6.97E+00	**2.85E+01**	4.98E+00
VRP	inst1	2.12E+04	4.97E+02	2.21E+04	4.99E+02	2.49E+04	5.69E+02	2.20E+04	8.46E+02	2.09E+04	3.99E+02	**2.07E+04**	4.42E+02
	inst2	1.40E+04	5.11E+02	1.43E+04	4.87E+02	1.43E+04	5.34E+02	1.46E+04	5.87E+02	1.37E+04	4.92E+02	**1.36E+04**	4.22E+02
	inst3	5.34E+03	3.57E+01	5.66E+03	4.04E+02	6.10E+03	4.90E+02	7.08E+03	6.51E+02	5.35E+03	2.19E+01	**5.34E+03**	1.64E+01
	inst4	1.48E+04	5.01E+02	1.55E+04	5.35E+02	1.61E+04	6.64E+02	1.52E+04	4.65E+02	1.46E+04	4.76E+02	**1.45E+04**	3.72E+02
	inst5	2.28E+05	4.50E+03	2.29E+05	2.23E+04	3.71E+05	7.57E+03	2.27E+05	3.28E+04	1.67E+05	2.38E+04	**1.61E+05**	3.48E+04
TSP	inst1	1.18E+05	1.72E+03	1.13E+05	8.78E+02	1.32E+05	0.00E+00	**1.10E+05**	1.15E+03	1.13E+05	1.00E+03	1.13E+05	1.15E+03
	inst2	7.07E+03	2.50E+01	7.07E+03	1.29E+01	8.11E+03	0.00E+00	**6.85E+03**	2.09E+01	7.02E+03	2.78E+01	6.99E+03	1.82E+01
	inst3	4.40E+04	1.47E+02	4.42E+04	2.23E+02	5.39E+04	2.22E-11	4.40E+04	1.47E+02	4.34E+04	1.74E+02	**4.31E+04**	1.21E+02
	inst4	9.26E+03	2.27E+01	9.23E+03	2.43E+01	1.12E+04	1.85E-12	9.26E+03	2.27E+01	9.17E+03	3.32E+01	**9.17E+03**	4.02E+01
	inst5	6.11E+04	2.37E+02	6.08E+04	1.26E+02	7.24E+04	1.48E-11	6.11E+04	2.37E+02	5.98E+04	3.60E+02	**5.98E+04**	4.25E+02

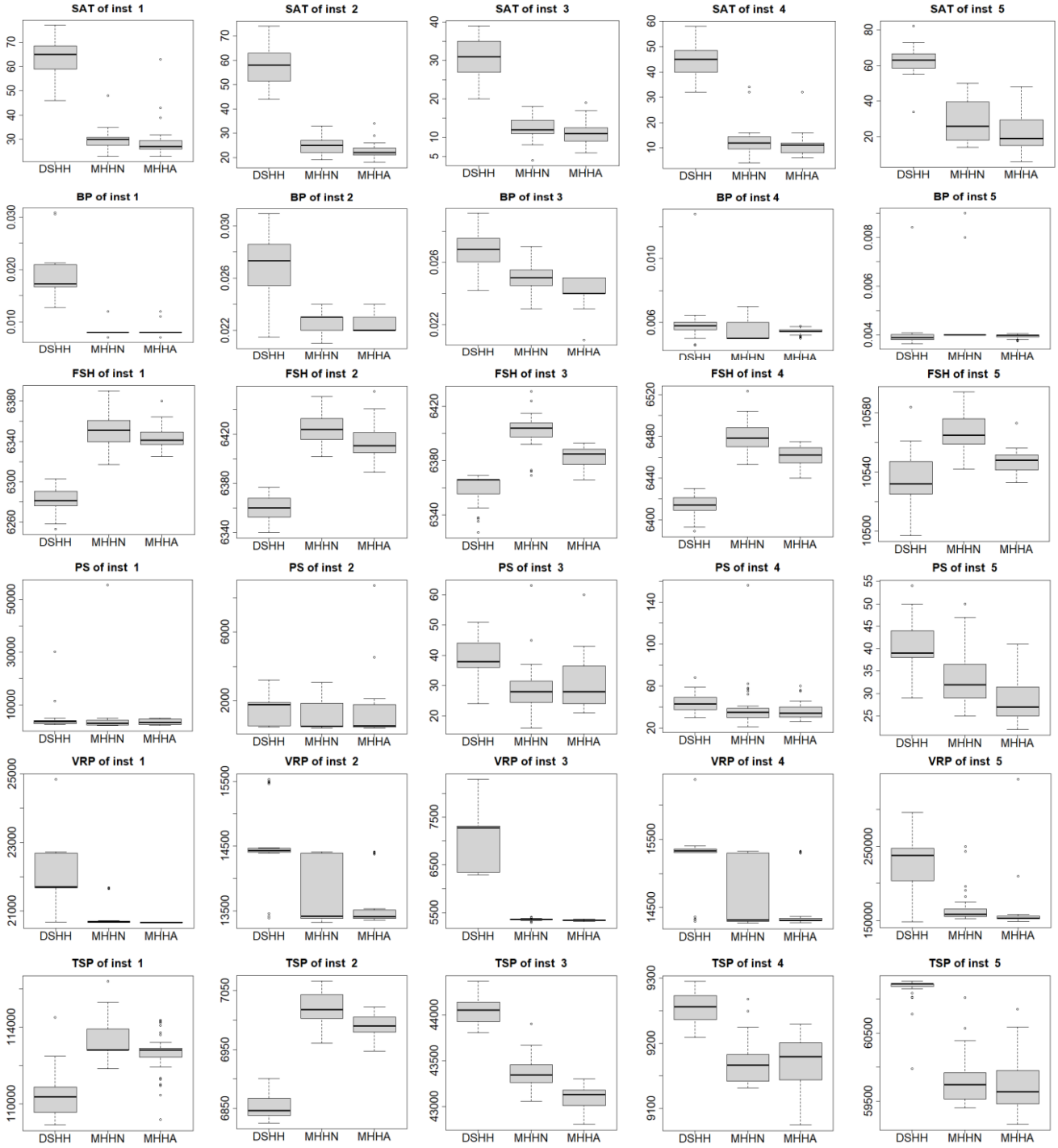

Figure. 11 DSHH, MHHN, and MHHA boxplots for all test problems for 5 instances

normalized median of the objective values for the proposed variants MHHN, MHHA, and the recently developed DHSS competitor for each instance in each problem domain. Each of these fingers provides the shape of the distribution of the results of each algorithm on each problem instances. It is observed that MHHA outperforms DHSS overall and in SAT, BP, PS, VRP, and last 3 instances in TSP problem domains. However, it achieves the second-best performance in FSH domain. In most HyFlex problem domains, MHHA turns out to be a viable general methodology which outperforms the other hyper-heuristic approaches. This reflects the success of the proposed framework in exploiting the suitable heuristic and/or acceptance criteria in different search points according to the problem domain. The poor performance in FSH problem can be explained by the bad performance of hyper-heuristic components used

compared with DHSS. Thus, involving other components to improve the performance in this problem domain is considered as future research.

5. Conclusion

Hyper-heuristic is a type of search technique that automates to combine, and control set of heuristics in order to solve a number of computationally hard problems. A traditional hyper- heuristic framework accomplishes this through two level. First, a high-level heuristic consists of two main components, that is (a heuristic selection mechanism and a move acceptance strategy). Then, low level heuristics are a set of problem specific heuristics. The design of the high-level heuristic is crucial. Since, each problem instance has a different landscape structure. This paper presents a new hyper-heuristic framework which adopts an additional level on top of the tradition hyper-heuristic framework. The proposed framework is a way to automatically design hyper-heuristic models through intelligently selecting suitable combinations of the highest-level heuristic components (i.e. Algorithm selection and move acceptance strategies) during the different stages of the optimization process. It is more generalized, reusable and helpful in releasing the complexity of choosing a hyper-heuristic method for solving a problem. Also, it is more independent to problem through new level added. The proposed MHH framework integrates a heuristic selection mechanism with a set of hyper-heuristics as well as a variety of acceptance criteria. At the top level, the selection-based roulette wheel is adopted to select one algorithm to be applied from three hyper-heuristic algorithms. Six challenging problems from the hyper-heuristic competition (CHeSC) test suite are used to illustrate the generality of the proposed framework. The experimental results show that the proposed framework produces highly competitive results against the competitors. This reflects the role of the adopted reward strategies and the selection mechanism. In future work, the combination of different selection mechanisms is studied as well as combinations of different reward methods to improve the performance. Also, combining different heuristics and different acceptance criteria provide more generality.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Ahmed kafafy, Asmaa Awaad; methodology, Ahmed Kafafy; software, Asmaa Awaad; validation, Ahmed kafafy, Asmaa Awaad; formal analysis, Ahmed Kafafy; investigation, Ahmed Kafafy; resources, Ahmed kafafy, Asmaa Awaad,Osama Abdelraouf; data curation, Ahmed Kafafy; writing—original draft preparation, Asmaa Awaad, Ahmed Kafafy; writing—review and editing, Ahmed Kafafy; visualization, Ahmed Kafafy, Asmaa Awaad; supervision, Ahmed Kafafy, Osama Abdelraouf, Nancy Elhefnawy.

References

[1] J. K. Mandal, “Handbook of research on natural computing for optimization problems”, *IGI Global*, 2016.

[2] E. G. Talbi, “Metaheuristics: from design to implementation”, *John Wiley & Sons*, Vol. 74, 2009.

[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, pp. 1695-1724, 2013.

[4] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, “A tabu search hyper-heuristic strategy for t-way test suite generation”, *Applied Soft Computing*, Vol. 44, pp. 57-74, 2016.

[5] X. Cai, H. Qiu, L. Gao, C. Jiang, and X. Shao, “An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems”, *Knowledge-Based Systems*, Vol. 184, p. 104901, 2019.

[6] X. S. Yang, “Nature-inspired metaheuristic algorithms”, *Luniver Press*, 2010.

[7] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, “Grammatical eVolution hyper-heuristic for combinatorial optimization problems”, *IEEE Transactions on EVolutionary Computation*, Vol. 17, pp. 840-861, 2013.

[8] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, “A classification of hyper-heuristic approaches”, *Handbook of Metaheuristics*, pp. 449-468, 2010.

[9] 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, pp. 135-151, 2016.

[10] S. Asta, D. Karapetyan, A. Kheiri, E. Özcan, and A. J. Parkes, “Combining Monte-Carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling

problem", *Information Sciences*, Vol. 373, pp. 476-498, 2016.

[11] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics", *Intelligent Data Analysis*, Vol. 12, pp. 3-23, 2008.

[12] B. Bilgin, E. Ozcan, and E. E. Korkmaz, "An experimental study on hyper-heuristics and exam scheduling", *Burke EK, Rudová H, eds., Practice and Theory of Automated Timetabling VI*, Vol. 3867 of Lecture Notes in Computer Science, 394-412, 2007.

[13] G. Ochoa, M. Hyde, T. Curtois, J. A. V. 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: *Proc. of European Conference on EVolutionary Computation in Combinatorial Optimization*, 2012.

[14] R. Qu and E. K. Burke, "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems", *Journal of the Operational Research Society*, Vol. 60, pp. 1273-1285, 2009.

[15] 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 Transactions on EVolutionary Computation*, Vol. 19, pp. 309-325, 2014.

[16] S. S. Choong, L. P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning", *Information Sciences*, Vol. 436, pp. 89-107, 2018.

[17] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit", In: *Proc. of the Practice and Theory of Automated Timetabling*, 2000.

[18] K. Chakhlevitch and P. Cowling, "Hyperheuristics: recent developments", *Adaptive and Multilevel Metaheuristics*, pp. 3-29, 2008.

[19] J. H. Drake, E. Özcan, and E. K. Burke, "A modified choice function hyper-heuristic controlling unary and binary operators", *2015 IEEE Congress on EVolutionary Computation*, 2015.

[20] S. Adriaensen, T. Brys, and A. Nowé, "Fair-share ILS: a simple state-of-the-art iterated local search hyperheuristic", In: *Proc. of the 2014 Annual Conference on Genetic and eVolutionary Computation*, 2014.

[21] M. Kalender, A. Kheiri, E. Özcan, and E. K. Burke, "A greedy gradient-simulated annealing selection hyper-heuristic", *Soft Computing*, Vol. 17, pp. 2279-2292, 2013.

[22] W. G. Jackson, E. Özcan, and J. H. Drake, "Late acceptance-based selection hyper-heuristics for cross-domain heuristic search", *2013 13th UK Workshop on Computational Intelligence*, 2013.

[23] W. G. Jackson, E. Özcan, and R. I. John, "Fuzzy adaptive parameter control of a late acceptance hyper-heuristic", *2014 14th UK Workshop on Computational Intelligence*, 2014.

[24] E. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J. A. V. Rodríguez, and M. Gendreau, "Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms", *IEEE Congress on eVolutionary Computation*, 2010.

[25] S. S. Choong, L. P. Wong, and C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the traveling salesman problem", *Swarm and eVolutionary Computation*, Vol. 44, pp. 622-635, 2019.

[26] P. Dempster and J. H. Drake, "Two frameworks for cross-domain heuristic and parameter selection using harmony search", *Harmony Search Algorithm*, pp. 83-94, 2016.

[27] L. Chen, H. Zheng, D. Zheng, and D. Li, "An ant colony optimization-based hyper-heuristic with genetic programming approach for a hybrid flow shop scheduling problem", *2015 IEEE Congress on EVolutionary Computation*, 2015.

[28] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A reinforcement learning: great-deluge hyper-heuristic for examination timetabling", *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends, IGI Global*, pp. 34-55, 2012.

[29] I. Khamassi, "Ant-Q Hyper Heuristic Approach applied to the Cross-domain Heuristic Search Challenge problems", 2011.

[30] L. D. Gaspero and T. Urli, "Evaluation of a family of reinforcement learning cross-domain optimization heuristics", In: *Proc. of International Conference on Learning and Intelligent Optimization*, 2012.

[31] 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, pp. 217-228, 2014.

[32] A. S. Ferreira, "A cross-domain multi-armed bandit hyper-heuristic", 2016.

[33] A. S. Ferreira, R. A. Goncalves, and A. T. R. Pozo, "A multi-armed bandit hyper-heuristic", *2015 Brazilian Conference on Intelligent Systems*, 2015.

[34] A. Kheiri and E. Keedwell, "A sequence-based selection hyper-heuristic utilising a hidden Markov model", In: *Proc. of the 2015 Annual Conference on Genetic and eVolutionary Computation*, 2015.
[35] E. Özcan and A. Kheiri, "A hyper-heuristic based on random gradient, greedy and dominance", *Computer and Information Sciences II*, pp. 557-563, 2011.
[36] A. Kheiri and E. Özcan, "A hyper-heuristic with a round robin neighbourhood selection", In: *Proc. of European Conference on eVolutionary Computation in Combinatorial Optimization*, 2013.
[37] A. Kheiri and E. Özcan, "An iterated multi-stage selection hyper-heuristic", *European Journal of Operational Research*, Vol. 250, pp. 77-90, 2016.
[38] A. Hassan and N. Pillay, "Dynamic Heuristic Set Selection for Cross-Domain Selection Hyper-heuristics", *Theory and Practice of Natural Computing*, 2021.
[39] R. Q. N. Pillay, "Hyper Heuristics:Theory and Applications, 1 ed.", *Springer Cham*, 2018.