ANKARA YILDIRIM BEYAZIT UNIVERSITY
AYBU

AYBU Business Journal, 4(1), 1-23
https://dergipark.org.tr/tr/pub/abj
https://aybu.edu.tr/isletme/tr/sayfa/5218/AYBU-BUSINESS-JOURNAL

AYBU Business Journal

RESEARCH ARTICLE

Submitted: 11 June 2024
Accepted: 29 June 2024
Published Online: 30 June 2024
DOI: 10.61725/abj.1499654

A Comprehensive Analysis of Multi-Strategy Memetic Algorithms Incorporating Low-Level Heuristics and Acceptance Mechanisms

Mazlum Özçağdavul[a*]

[a]*Management Information Systems, Ankara Yildirim Beyazit University, Ankara, Turkey*

Abstract

Hyper-heuristics are designed to be reusable, domain-independent methods for addressing complex computational issues. While there are specialized approaches that work well for particular problems, they often require parameter tuning and cannot be transferred to other problems. Memetic Algorithms combine genetic algorithms and local search techniques. The evolutionary interaction of memes allows for the creation of intelligent complexes capable of solving computational problems. Hyper-heuristics are a high-level search technique that operates on a set of low-level heuristics that directly address the solution. They have two main components: heuristic selection and move acceptance mechanisms. The heuristic selection method determines which low-level heuristic to use, while the move acceptance mechanism decides whether to accept or reject the resulting solution. In this study, we explore a multi-meme memetic algorithm as a hyper-heuristic that integrates and manages multiple hyper-heuristics (Modified Choice Function All Moves, Reinforcement Learning with Great Deluge, and Simple Random Only Improvement) and parameters of heuristics (such as mutation rates and search depth). We conducted an empirical study testing two different variations of the proposed hyper-heuristic. The first algorithm uses the Only Improvement acceptance technique for both Reinforcement Learning and Simple Random, and All Moves for Modified Choice Function. In the second version, the Great Deluge method replaces Only Improvement for Reinforcement Learning. The second algorithm's results were the best of all competitors from the CHeSC2011 competition, achieving the fourth-best hyper-heuristic performance.

Keyword

Hyper-Heuristic, Cross-domain Heuristic Search Challenge (CHeSC 2011), Multi-meme memetic algorithm, parameter tuning.

1. INTRODUCTION

The effectiveness of heuristic approaches in addressing practical computer optimisation problems has been demonstrated. Applying them to both new problem domains and new instances of the same problem domain has many disadvantages, though. It is challenging to apply such search strategies to a wide range of

*Corresponding author.

Contact: Mazlum Özçağdavul ✉ mozcagdavul@aybu.edu.tr

To cite this article: Özçağdavul, M. (2024). An emprical study on multi-meme memetic algorithms which includes Choice Function, Reinforcement Learning and Simple Random Hyper-heuristics. *AYBU Business Journal,* 4(1), 1-23.

computational search problems since there is a lack of guidance about the selection of algorithms and their parameters during the search process (Özcan et al., 2010; Burke et al., 2003). Furthermore, the fact that state-of-the-art heuristics are custom problem-specific approaches that take a long time and extensive domain expertise to design adds to the difficulty of actual procedures (Burke et al., 2009; Ross, 2005)

Burke et al.'s description of metaheuristics as a framework for structuring a search algorithm to locate an ideal or nearly ideal solution for challenging combinatorial optimization problems is found in (Burke and Kendall, 2005). Despite being categorized as universal algorithms, metaheuristics require modification for every issue domain. In addition to single-point based metaheuristics, which employ a single candidate solution, multi-point based metaheuristics employ several candidate solutions to carry out the search process. Among the population-based metaheuristics are genetic programming and genetic algorithms (Burke and Kendall, 2005).

Using a variety of genetic operators, including crossover and mutation to produce new people, genetic algorithms aim to raise the objective value of candidate solutions within a population (Moscato, 1989). Afterwards, created individuals are swapped out for other population members using replacement procedures such as the Steady State Genetic Algorithm (Gendreau and Potvin, 2005). The selection of these operators may have a significant impact on the balance between intensification and diversification.

By adding an extra intensification level to the evolutionary iterations, multipoint-based evolutionary algorithms combined with single-point-based local search heuristics get better results. This is Memetic Algorithms' primary reasoning. It was Muscato who initially proposed memetic algorithms (Moscato, 1989). That being said, Dawkins was the first scientist to coin the term "meme." According to him, a meme is a bit of information that spreads and becomes managed, accepted, acclimated to, and passed to the affected individual (Dawkins, 1976). Furthermore, a genetic algorithm (GA) can be employed to identify the optimal feature selection for learning-based issues by applying a metaheuristic approach (Yılmaz et al., 2020)

There are similarities between this acclimatization process and local search. Thus, hill climbing is often used in memetic algorithms. Stated differently, memetic algorithms combine local search (hill climbers) and genetic algorithms. In order to enhance the population's objective quality, hill climbers are used in a specific lap of each course throughout the development phase. Numerous memetic algorithms have been proposed. Two primary examples of these algorithms are transgenerational memetic algorithms and steady state memetic algorithms. Memetic algorithms are typically customized for a particular problem domain (Neri and Cotta, 2012)

Aims and Objectives

It is challenging to create a custom method for every problem domain since computational problems have a large number of distinct domains. Furthermore, in order to construct a solver, each instance of a domain could call for specialized domain knowledge. The goal is to produce high-level algorithms that are independent of issue domains because of this. Numerous studies have been conducted in the literature, and it has been demonstrated that a number of algorithms including Reinforcement Learning, Choice Function, Simple Random and acceptance mechanisms including Great Deluge, Simulated Annealing, and Only Improvement can yield encouraging outcomes.

This work examines a multi-meme memetic algorithm as a hyper-heuristic that combines and regulates many heuristic selection techniques (Modified Choice Function, Reinforcement Learning,

and Simple Random), as well as low-level heuristic parameters (operators), in light of these studies. The primary goal is to create an intelligent algorithm that can select the appropriate heuristic selection technique on its own and adjust the low-level heuristic's parameters for each kind of heuristic selection method.

2. Hyper-heuristics

2.1. Hyper-Heuristics overview and History

A unique, problem-independent method for resolving and optimizing computational issues is the use of hyper-heuristics. In a work published in 2001, Cowling et al. (2001), used the term "hyper-heuristic" for the first time. Hyper-heuristics are defined as general techniques that can be applied to a wide range of problems. Özcan et al. highlight in (Kendall and Mohamad, 2004) that the development of meta-heuristics stems from their application to a variety of problem domains. The neighborhood operators that transform meta-heuristics into problem-specific structures rather than broad frameworks are what make them effective. For this reason, it is inappropriate to use meta-heuristics as a framework independent of problems.

Since hyper-heuristics operate on the space of heuristics rather than the space of solutions, they are often applicable to a wide range of problem domains (Cowling et al., 2001). From a given set of heuristics, a hyper-heuristic technique can intelligently select the most appropriate low-level heuristic to utilize at any given time. Thus, rather than providing a solution for a particular issue instance at hand, we in hyper-heuristics concentrate on adaptively developing organization approaches (Özcan et al., 2010). Hyper-heuristics can be employed in place of meta-heuristics since they don't require as much knowledge and experience with the problem domain. Thus, even a programmer without any prior knowledge of the issue domain can use them (Kaelbling et al., 1996).

This research began with a study published at the start of the 1960s that expressed a concept comparable to the behavior of hyper-heuristics. Fisher and Thompson (1963) contend that a great accomplishment in production scheduling can be achieved by combining scheduling rules, as opposed to utilizing them separately. It is possible to demonstrate that the effective research in hyper-heuristics began with this paper (Fisher and Thompson, 1963).

2.2. Classification of Hyper-heuristic approaches

Two primary groups of hyper-heuristics have been distinguished by Burke et al. (2010). The first is heuristic selection, which comprises methods for picking or selecting low-level heuristics that already exist, and the second is heuristic creation, which generates heuristics from the parts of various low-level heuristics that already exist. The distinction between the constructive and perturbative search paradigms is the next step in this dimension (Burke et al., 2013)

However, there are also two major classes when considering the quantity of solutions used during the search process. These are multi-point-based search, which operates on several solutions, and single solution, which acts on a single-point-based search.

According to Özcan et al. (2013), a hyper-heuristic is a learning algorithm if it makes use of evaluation throughout the search process. They divide the feedback received during learning into two classes: online learning and offline learning, depending on where it came from. Hyperheuristics for online learning involve

learning while solving problems. In offline learning, on the other hand, the system learns from a collection of training cases with the intention of generalizing to solve problems in cases that have not yet been observed.

2.3 Multi-meme Memetic Algorithms

From a conceptual standpoint, metaheuristics are utilized as a template for addressing challenging combinatorial optimization issues and aid in the creation of search algorithms. Metaheuristics cannot be categorized as universal techniques since they require knowledge particular to the issue domain. There are two different kinds of metaheuristics based on how many points they use during the search process. These metaheuristics are based on both single and multiple points. Another name for multi-point-based metaheuristics is population-based metaheuristics. Genetic algorithms are one type of potential solution used by population-based metaheuristics (Burke and Kendall, 2005). In an evolutionary cycle, genetic algorithms iteratively improve population quality using a set of genetic operators. The best one may survive, or the old person may be replaced by kids. Crossover, mutation, and local searches are among names for these operators. Local searches are employed to intensify the solution, whereas crossover and mutation heuristics are used to avoid local optima (Gendreau and. Potvin, 2005). The balance between diversification and intensity is a critical decision that impacts an individual's level of fitness.

2.3.1 Multi-meme memetic algorithm taxonomy

Memetic algorithms are divided into two categories by Ong et al. (2006): the process of adaptation, also known as the adaptation type (Hinterding et al., 1997; Eiben et al., 1999) and the adaptation level, which modifies the selection of memes in adaptive memetic algorithms. Another name for adaptation level is meme history knowledge. The taxonomy of adaptive memetic algorithm techniques in use is shown in Figure 1.

Adaptive Type		Adaptive Level		
		External	*Local*	*Global*
Static		Basic meta-Lamarckian learning / Simplerandom		
Adaptive	*Qualitative Adaptation*		Randomdescent / Randompermdescent	Tabu-search
	Quantitative Adaptation		Sub-Problem Decomposition/ Greedy	Straightchoice/ Rankedchoice/ Roulettechoice/ Decompchoice/ Biased Roulette Wheel
Self-Adaptive			Multi-memes/ Co-evolution MA	

Figure 1: A Classification of Memes Adaptation in Adaptive Memetic Algorithms (Ong et al., 2006)

This taxonomy divides co-evolution memetic algorithms (Smith et al., 2002; Smith, 2003) and multi-meme memetic algorithms (Krasnogor et al., 2002; (Krasnogor, 2002) into self-adaptive categories since the memes are coded as a component of the population and also go through conventional evolution. Additionally, a multi-meme memetic algorithm or a memetic algorithm is categorized based on the decision-making process based on the degree of adaption. This meme selection procedure is local if it just takes into account a portion of historical knowledge. As a result, this taxonomy places our multi-meme memetic algorithm in the local level and self-adaptive category.

2.4 Heuristic selection methodologies

Many hyper-heuristic approaches that are divided into two categories can be found in the literature. These categories can be divided into two groups: those based on constructive low-level heuristics and those based on perturbative low-level heuristics. Starting with a blank solution, the constructive low-level heuristics construct attempts to gradually develop the entire solution (Özcan et al., 2010). They have had success using them to solve combinatorial optimization issues. Bin-packing (Ross et al., 2003) timetabling (Terashima-Marin et al., 1999; Asumuni et al., 2007; Qu, et al., 2008) production scheduling (Vazquez-Rodriguez et al., 2007) and reducing stock (Terashima-Marin et al., 2005) are a few examples of these issues.

Within the second category, methods utilizing perturbative low-level heuristics attempt to identify a feasible starting solution using some straightforward mechanisms; these can be determined arbitrarily or through the application of a fundamental constructive heuristic. Subsequently, the solution is attempted to be improved by applying shift and swap perturbations (Özcan et al., 2010). In other words, they pick or choose from a group of neighborhoods that can yield better results than the initial, full answer. Perturbative hyper-heuristics are sometimes referred to as improvement hyper-heuristics by Özcan et al. (2005). Improvement hyper-heuristics like this have been effectively used to solve real-world issues including staff scheduling (Cowling et al., 2001), timetabling (Terashima-Marin et al., 2005), and (Burke et al., 2003). In most cases, perturbative hyper-heuristics are used on a single candidate solution. These hyper-heuristics aim to improve a particular answer.

Most search operations in a perturbative hyper-heuristic framework are carried out with a single candidate solution. Iteratively, these hyper-heuristics aim to enhance a given solution by averaging two successive steps: heuristic selection and move acceptance, as seen in Figure 1. Using a chosen heuristic (or heuristics), a candidate solution (St) at a specified time (t) is disturbed into a new solution (or solutions). After this stage is complete, the found solution is either rejected or accepted using a technique known as move acceptance. Until a predetermined halting requirement is satisfied, this procedure is repeated (Özcan et al., 2010). Information related to the problem cannot be transferred from the problem domain to the hyper-heuristic layers due to a domain barrier.

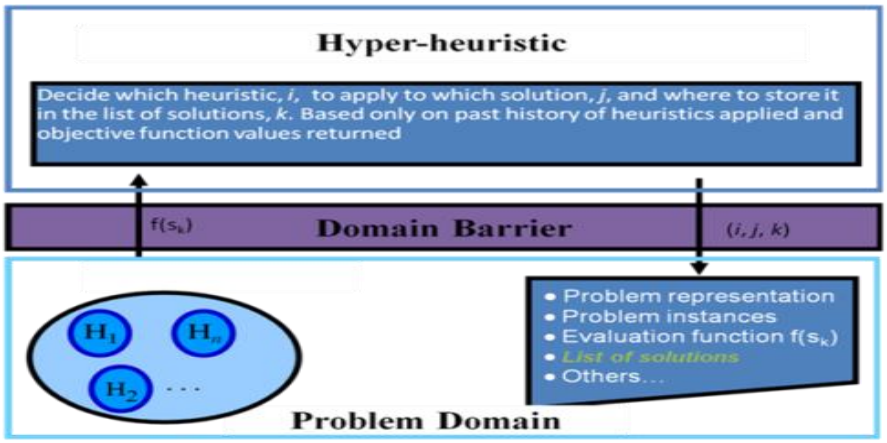

Figure 2. Hyper-heuristic Domain Barrier (Burke et al., 2009)

2.4.1 Simple Random

Simple Random uses a uniform random distribution to randomly select low-level heuristics. This heuristic selection process lacks a learning mechanism because no information is kept (Neri and Cotta, 2012)

2.4.2 Reinforcement Learning

The ability to be applied to a broad range of problem cases, some of which may come from other problem domains with unique features, is one of the primary characteristics of hyperheuristics. For hyper-heuristics to choose the best heuristic during the selection phase, machine learning processes are therefore necessary. One of the current hyper-heuristics for learning is reinforcement learning (Kaelbling, et al., 1996; Sutton and Barto, 1998). The reinforcement learning process chooses an action to raise or reduce a value in exchange for a long-term reward, working in tandem with the environment to change its state (Özcan et al., 2010). For this reason, for any low-level heuristic, a learning hyper-heuristic updates a utility value that is acquired through a predefined system of rewards and punishments.

2.4.3 Modified Choice Function

Low-level heuristics are scored by the Choice Function based on three different variables. The strategy that depends on these scores helps choose the low-level heuristic that will be used. First, the low-level heuristic's prior performance (f1) is noted. The low-level done more recently is given more weight. Equation 1's formula is then used to calculate the value for each heuristic:

$$f_1(h_j) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \qquad (1)$$

Equation 1 : Choice Function f1

Where Tn(hj) is the time taken to invoke the low-level heuristic for each precedent appeal n of the low-level heuristic hj, α is a value between 0 and 1 that has a higher influence to late performance, and ln(hj) is the difference in the evaluation function.

Second, f2 value looks for any pairwise relationships between heuristics at the low-level. Equation 2's formula is used to determine f2 values when the hj low-level heuristic is used immediately following hk.

$$f_2(h_k, h_j) = \sum_n \beta^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)} \qquad (2)$$

Equation 2: Choice Function f2

where Tn(hk, hj) is the amount of time needed to call a low-level heuristic for each precedent appeal, and In(hk, hj) is the difference in evaluation function. The low-level heuristic hj's n and β have a stronger effect on late performance when they have a value between 0 and 1.

Thirdly, a value f3 is computed that permits the selection of all low-level heuristics. The time elapsed since the low-level heuristic was selected by the Choice Function is represented by f3.

$$f_3(h_j) = \tau(h_j) \quad (3)$$

Equation 3: Choice Function f3

Following the computation of these three f, a score is assigned independently to each low-level heuristic using the Choice Function F formula, as indicated by Equation 4:

$$F(h_j) = \alpha f_1(h_j) + \beta f_2(h_k, h_j) + \delta f_3(h_j) \quad (4)$$

Equation 4: Choice Function Formula [10]

where f1 and f2 are weighted by α and β, giving an intensified heuristic search process. To provide sufficient diversification, δ weights f3. These variables in the Choice Function were chosen based on the author's prior experimentation knowledge. They also demonstrated how well Choice Function works in conjunction with the All Move acceptance technique.

The weight of f1 and f2, as well as the values of α and β, are taken into account by Drake et al. (2012), in their suggested Modified Choice Function as a single parameter that is employed for intensification. This parameter was given the name ϕ. This value will also be used, similar to the original Choice Function created by Cowling et al. (2001), to assign late performance a greater noteworthy relevance. The weight of f3, the parameter that governs the degree of diversification of the low-level heuristic, will be displayed as δ in the updated version. For each low-level heuristic hj, the Ft score in the Modified Choice Function will be determined using the formula in Equation 5:

$$F_t(h_j) = \phi_t f_1(h_j) + \phi_t f_2(h_k, h_j) + \delta_t f_3(h_j) \quad (5)$$

Equation 5: Modified Choice Function Formula [33]

3. Implementation and Methodology

3.1. HyFlex

A group at The University of Nottingham's Department of Computer Science has created an object-oriented framework known as HyFlex for testing hyper-heuristic algorithms. The purpose of this framework's design and development was to enable the CHeSC2011 Cross-domain Heuristic Challenge. Creating a standard framework for testing and comparing different cross-domain algorithms is one of the key goals of this framework. HyFlex comprises six domains. These fields include the following: traveling salesman issue, bin-packing (one-dimensional) flow shop, personnel scheduling, and maximum satisfiability, or MAX-SAT. Developers can use this framework to test their algorithms' performance directly on past challenge participants (The University of Nottingham, 2021).

3.2 Implementation

Simple Random, Reinforcement Learning, and Modified Choice Function are the three hyper-heuristic selection methods that the memes regulate. They also determine the mutation rate and depth of search for each method. Table 1 shows how memes are organized.

Table 1: Representation of Memes

	memes		Possible values
	0	Hyper-Heuristic Selection Method	{0: SR, 1: RL, 2: M CF}
SR	1	Mutation Rate	{0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8 4: 1}
	2	Depth of Search	{0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8 4: 1}
RL	3	Mutation Rate	{0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8 4: 1}
	4	Depth of Search	{0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8 4: 1}
M CF	5	Mutation Rate	{0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8 4: 1}
	6	Depth of Search	{0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8 4: 1}

A framework called HyFlex (The University of Nottingham, 2021) is employed. Although it offers nine distinct domains, only the six that were utilized at CHeSC2011 were actually used. HyFlex offers a partition between domains and algorithms. The individuals' fitness values and the kinds of low-level heuristics are the only pieces of information that can go between levels. Only the low-level heuristic's type—whether it's a mutational, crossover, or local search—is disclosed. Both mutation and ruin and recreate heuristics are included in mutational heuristics. The number of low-level heuristics of each kind varies between domains (The University of Nottingham, 2021).

3.2 Algorithms

Algorithm 1: Simple Random - OI + Reinforcement Learning – OI + Modified Choice Function AM

Create a population of populationSize with random individual

Initialise all individuals

Create a two dimmentional array *memeplex* with size of populationSize by seven

For *i*=0 to *populationSize* **do**

// Randomly fill heuristic selection method (0: SR, 1: RL, 2: M CF)

Fill *memeplex* for each individual with random values from heuristic selection methods

// Randomly fill *mutation_rate* and *depth_of_search* (0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8, 4: 1)

Fill *memeplex* for each individual with random values from heuristic selection methods

End For

While termination criterion is not met **do**

Parent1 ← Select Parent (Population, tour-size)

//Select a different parent than Parent1

Parent2 ← Select Parent (Population, tour-size)

offspring ← Apply a random a crossover operator (Parent1, Parent2)

bestParentID ← set the parents ID who has best function value

// Inherit best parents memes to offspring

offspring memes ← bestParent's memes

If *memeplex[bestParentID][0]== 0:*

Apply ***Simple Random Only Improvement*** Heuristic Selection Method

Else if *memeplex[bestParentID][0]== 1:*

Apply ***Reinforcement Learning Only Improvement*** Heuristic Selection Method

Else if *memeplex[bestParentID][0]== 2:*

Apply ***Modified Choice Function*** Heuristic Selection Method

end if

Replacement: Replace the worst individual by offspring

end while

The first algorithm initializes each individual after generating a random population of size ten. After each person has been initialized, a two-dimensional array known as a memeplex is produced. The mutation rate, search depth, and hyper-heuristic that will be applied to each individual are predetermined. Next, a parent who has picked a tournament with a two-tour option is selected. After the first parent, a second parent is selected. With parent 1 and parent 2, a random crossover low-level heuristic is used, and an offspring is produced. The

child inherits the memes of the parent who is the fittest. The child is randomly assigned to the meme of one parent if both parents have the same fitness score.

The Simple Random Only Improvement (SR-OI) hyper-heuristic is selected if the hyper-heuristic selection meme is zero. In case number one and number two are selected, the Modified Choice Function All Move and Reinforcement Learning Only Improvement (RL-OI) hyper-heuristic, respectively. The heuristics do not exchange any information.

If there is any improvement, the offspring in the SR-IO is subjected to a randomly chosen low-level heuristic for mutations, and if not, a randomly chosen local search is used. If there is no improvement, the low-level heuristics for local search and/or mutation are not applied to the progeny. Then the child takes the position of the worst person. Following all procedures, each meme may undergo a mutation at a rate of 0.2 innovation to achieve meme diversification. Every low-heuristic (LLH) in the RL-OI has a score, and the LLH with the highest score is selected. Once more, only enhanced solutions are approved.

Algorithm 2: Simple Random - OI + Reinforcement Learning – GD + Modified Choice Function AM

Create a population of populationSize with random individual

Initialise all individuals

Create a two dimmentional array *memeplex* with size of populationSize by seven

For *i*=0 to *populationSize* **do** // Randomly fill heuristic selection method (0: SR, 1: RL, 2: M CF)

Fill *memeplex* for each individual with random values from heuristic selection methods

// Randomly fill *mutation_rate* and *depth_of_search* (0: 0.2, 1: 0.4, 2: 0.6, 3: 0.8, 4: 1)

Fill *memeplex* for each individual with random values from heuristic selection methods

End For

While termination criterion is not met **do**

Parent1 ← Select Parent (Population, tour-size) //Select a different parent than Parent1

Parent2 ← Select Parent (Population, tour-size)

offspring ← Apply a random a crossover operator (Parent1, Parent2)

bestParentID ← set the parents ID who has best function value // Inherit best parents memes to offspring

offspring memes ← bestParent's memes

If *memeplex[bestParentID][0]== 0:*

Apply ***Simple Random Only Improvement*** Heuristic Selection Method

Else if *memeplex[bestParentID][0]== 1:*

Apply ***Reinforcement Learning Great Deluge*** Heuristic Selection Method

Else if *memeplex[bestParentID][0]== 2:*

Apply ***Modified Choice Function*** Heuristic Selection Method

end if

Replacement: Replace the worst individual by offspring

end while

Every setting in the second algorithm was the same as in the first, with the exception that the Great Deluge was used as the acceptance criterion for reinforcement learning. The hyper-heuristics were still not exchanging information with each other.

4. Results and Analysis

The first Cross-domain Heuristic Search Challenge (CHeSC 2011) was organized by the Automated Scheduling Optimisation & Planning Group, or ASAP, at The School of Computer Science at The University of Nottingham. The purpose of the challenge was to bring together experts from various fields, including computer science, operational research, and artificial intelligence, to evaluate their advanced methodologies. Every issue domain has a few low-level heuristics available. These particular low-level heuristics should be managed by the high-level algorithm in order to tackle various problems from various fields. Each domain has its own set of low-level heuristics, but the hyper-heuristic, the governing high-level algorithm, must be distinct. Stated differently, more than one low-level heuristic from a separate issue domain should be controlled by the same algorithm.

Additionally, ASAP offers a common software framework (HyFlex) designed in Java to handle a variety of combinatorial optimization issues in diverse fields. HyFlex offers a number of techniques for producing and assessing solutions. In addition, the hyper-heuristics can apply move operators, hill-climbers (local searches), ruin and recreate heuristics, and mutation operators to the solutions.

4.1. Formula One Scoring System

Every hyper-heuristic in the CHeSC 2011 has completed 31 runs on five distinct examples from each of the six problems. Next, each problem instance's median score was taken into account. The Formula One scoring methodology is then used to order these scores. The top hyper-heuristic receives 10 points for each instance, followed by eight for the second, six for the third, and five, four, three, two, and one points for the other hyper-heuristics. From the ninth-best score, every other score is zero (The University of Nottingham, 2021).

4.2 Results of the Algorithms

The competition's ranking mechanism assigns a number to each hyper-heuristic based on its relative performance. Three distinct algorithms have been created and examined in this study. An Intel i5 fifth generation 2.7GHZ CPU with four cores and a 3MB L3 cache was used for all tests. To be fair with the 600 seconds of competition running time, the benchmarking tool that ASAP gave offers 464 seconds for each run.

Eleven runs were scheduled for each method, across six distinct domains with five distinct examples each. The entire algorithm test took 42.53 hours with these parameters.

4.2.1 Results of the First Algorithm

The first multi-meme memetic algorithm combines low-level heuristic values for mutation rates and search parameter depth with hyper-heuristic approaches such as Simple Random Only Improvement, Reinforcement Learning Only Improvement, and Modified Choice Function All Moves. As in (Özcan et al., 2013). the population size is ten, and the initialization of each member of the population is done at random. This population also has a memeplex allocated to it, which governs the attributes of each member, including the hyper-heuristic and its parameters.

Following eleven runs, the median values are determined. The findings are then compared to the performance of prior competitors using the Formula One scoring system, which is available on the CHeSC 2011 website.

Figures 3 and 4 illustrate how the created algorithm failed to receive a single point in either Bin Packing or MAX-SAT. placed sixth with twelve points in Flow Shop, eighth with ten points in TSP, ninth with six points in VRP, and twelfth with three points in personnel scheduling. Figures 5, 6, 7, and 8 show the rankings for Flow Shop, TSP, VRP, and Personnel Scheduling, respectively.

When compared to other competitors, this algorithm's overall performance is appalling. With only thirty-one points altogether, this algorithm finished the competition in thirteenth place. This is depicted in figure 9.

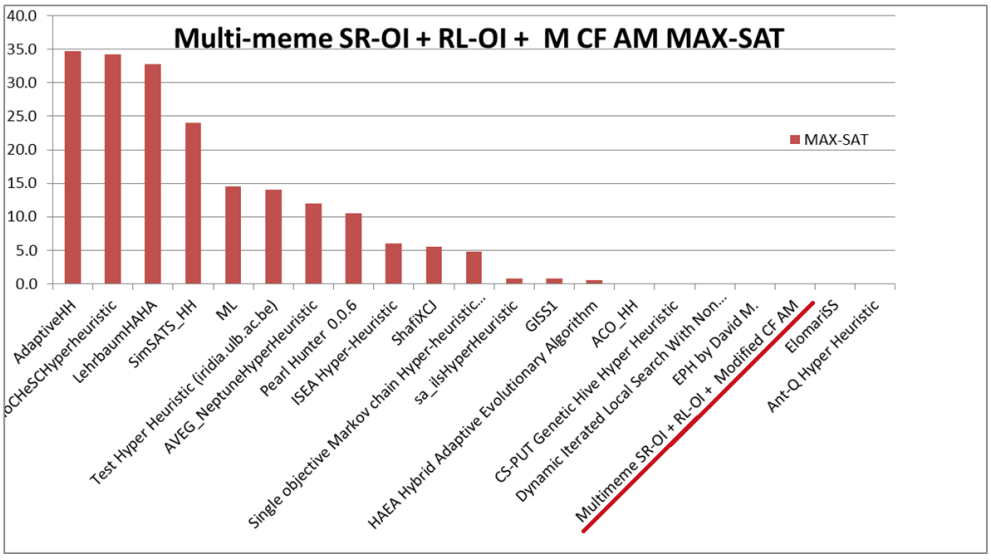

Figure 3: Multi-meme SR OI RL OI M CF AM MAX-SAT Results

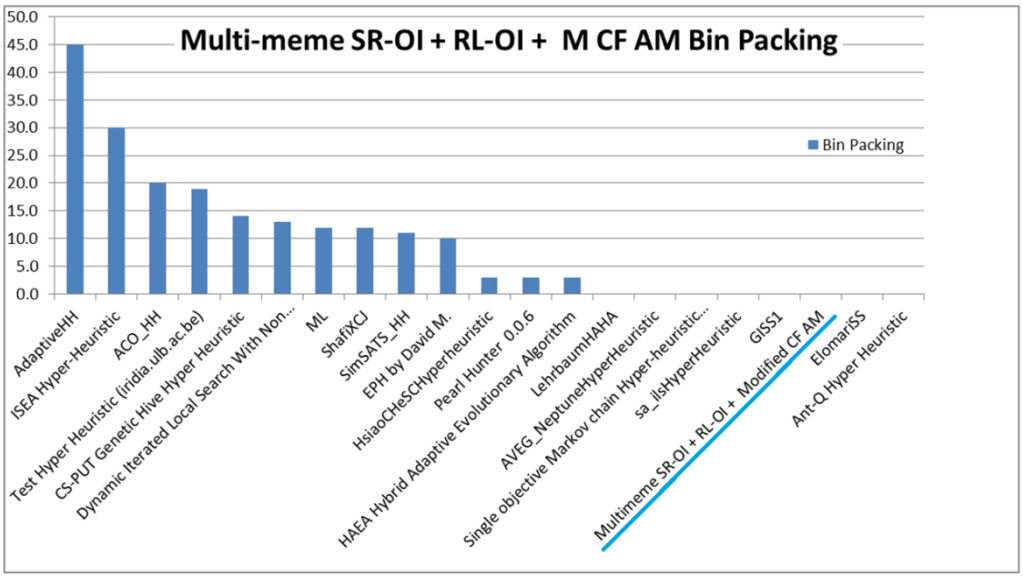

Figure 4: Multi-meme SR OI RL OI M CF AM Bin Packing Results

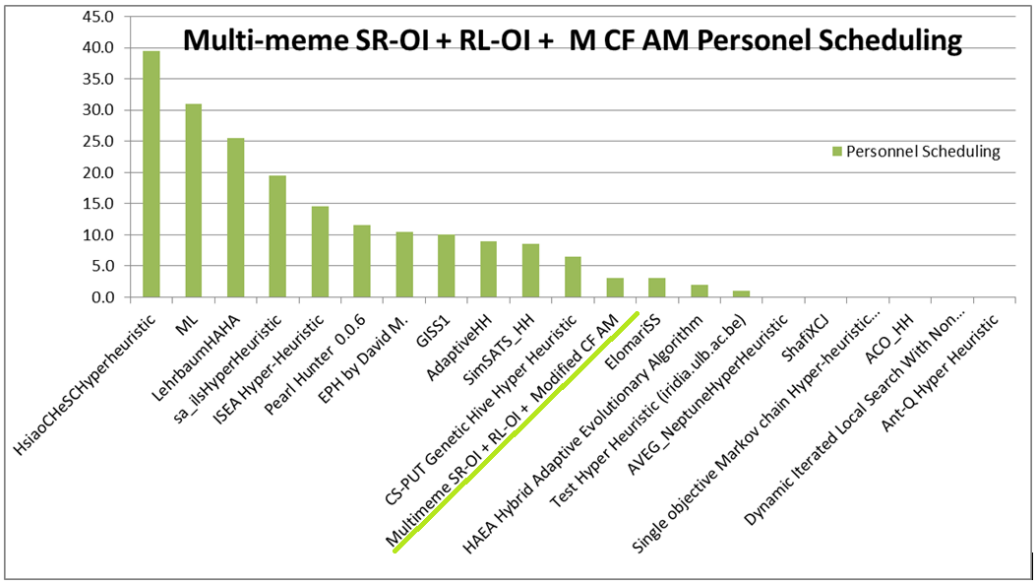

Figure 5: Multi-meme SR OI RL OI M CF AM Personnel Scheduling Results

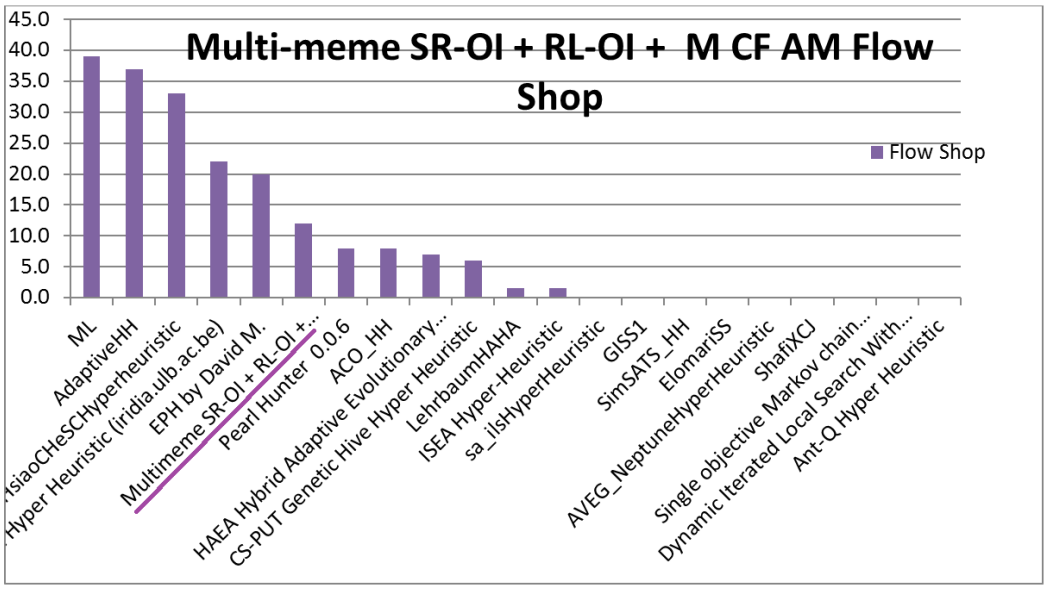

Figure 6: Multi-meme SR OI RL OI M CF AM Flow Shop Results

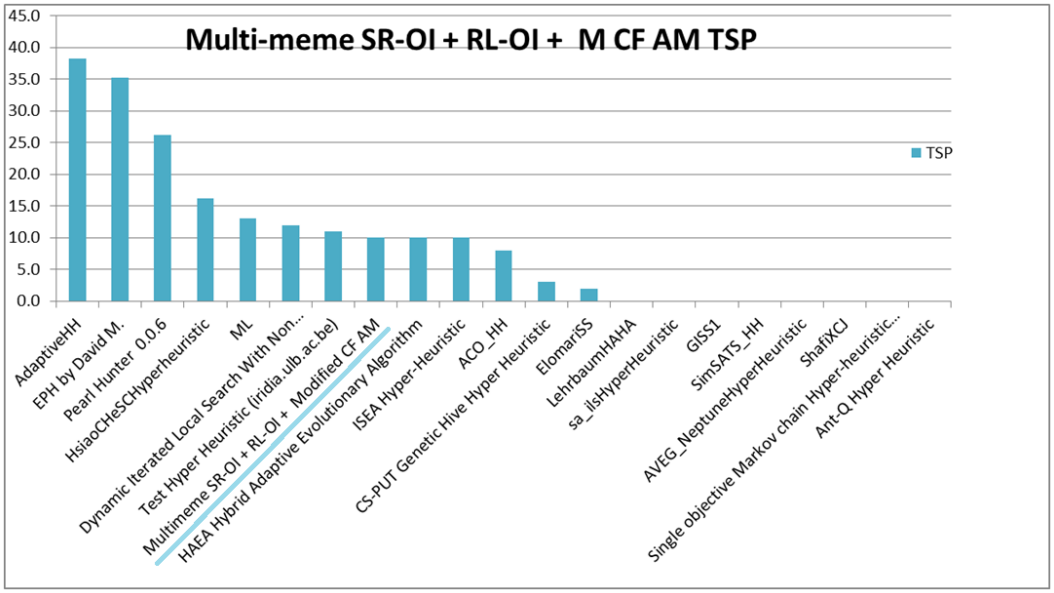

Figure 7: Multi-meme SR OI RL OI M CF AM TSP Results

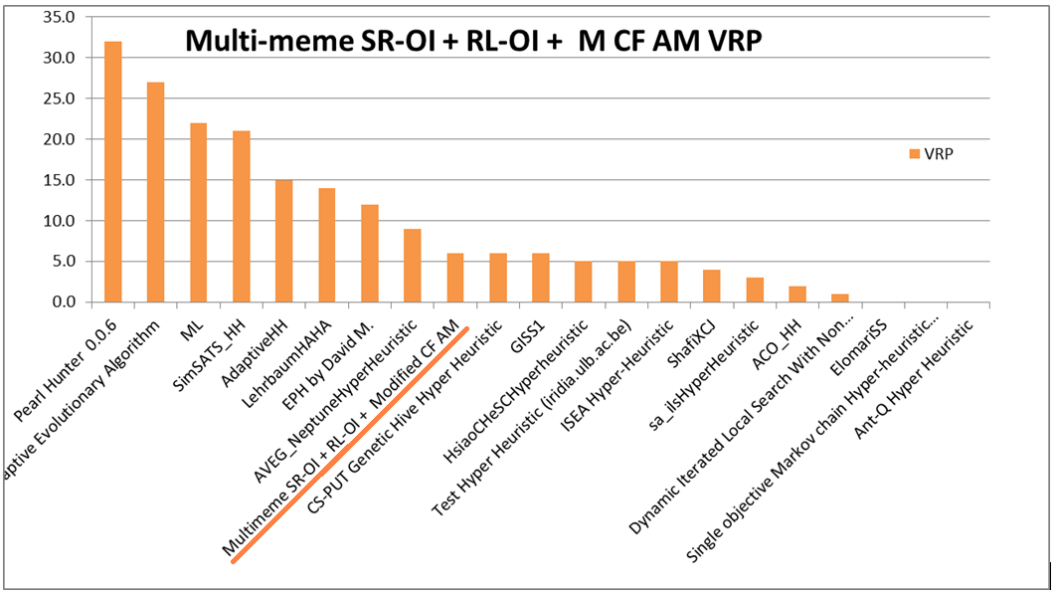

Figure 8: Multi-meme SR OI RL OI M CF AM VRP Results

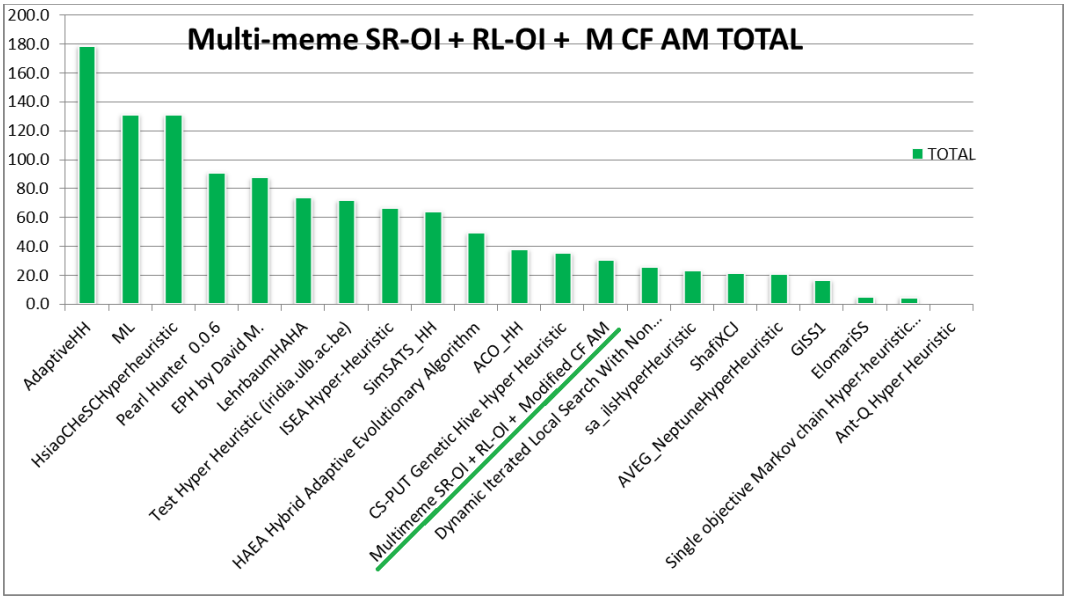

Figure 9: Multi-meme SR OI RL OI M CF AM Overall Results

Co-evolution of memes was managed using the basic inheritance mechanism in the first algorithm. Reward Learning Only Improvement was the hyper-heuristic that was most frequently selected. The least chosen hyper-heuristic was the Simple Random Only improvement one. The mutation rate and search depth that the mechanism has mostly chosen are 0.2 and 0.4, respectively.

Table 2 : Detailed scores for Hyper-Heuristic 1 (Algorithm 1) with compare to other competitors

	Multimeme SR-OI + RL-OI + MCF With NO information Sharing between Heuristics							
Rank	**Heuristic Name**	**MAX-SAT**	**Bin Packing**	**Personnel Scheduling**	**Flow Shop**	**TSP**	**VRP**	**TOTAL**
1	AdaptiveHH	**34.8**	**45.0**	9.0	**37.0**	**38.3**	15.0	**179.0**
2	ML	14.5	12.0	**31.0**	**39.0**	13.0	**22.0**	**131.5**
3	HsiaoCHeSCHyperheuristic	**34.3**	3.0	**39.5**	**33.0**	16.3	5.0	**131.0**
4	Pearl Hunter 0.0.6	10.5	3.0	11.5	8.0	**26.3**	**32.0**	**91.3**
5	EPH by David M.	0.0	10.0	10.5	20.0	**35.3**	12.0	**87.8**
6	LehrbaumHAHA	**32.8**	0.0	**25.5**	1.5	0.0	14.0	**73.8**
7	Test Hyper Heuristic (iridia.ulb.ac.be)	14.0	19.0	1.0	22.0	11.0	5.0	**72.0**
8	ISEA Hyper-Heuristic	6.0	**30.0**	14.5	1.5	10.0	5.0	**67.0**
9	SimSATS_HH	24.0	11.0	8.5	0.0	0.0	21.0	**64.5**
10	HAEA Hybrid Adaptive Evolutionary Algorithm	0.5	3.0	2.0	7.0	10.0	**27.0**	**49.5**
11	ACO_HH	0.0	**20.0**	0.0	8.0	8.0	2.0	**38.0**
12	CS-PUT Genetic Hive Hyper Heuristic	0.0	14.0	6.5	6.0	3.0	6.0	**35.5**
13	**Multimeme SR-OI + RL-OI + Modified CF AM**	**0.0**	**0.0**	**3.0**	**12.0**	**10.0**	**6.0**	**31.0**
14	Dynamic Iterated Local Search With Non Improvement Bias	0.0	13.0	0.0	0.0	12.0	1.0	**26.0**
15	sa_ilsHyperHeuristic	0.8	0.0	19.5	0.0	0.0	3.0	**23.3**
16	ShafiXCJ	5.5	12.0	0.0	0.0	0.0	4.0	**21.5**
17	AVEG_NeptuneHyperHeuristic	12.0	0.0	0.0	0.0	0.0	9.0	**21.0**
	Colours representation : **First** **Second** **Third**							

4.2.2 The Results of the Second Algorithm

With the exception of the Reinforcement Learning acceptance criterion, every setting in the second algorithm was unchanged. The Great Deluge mechanism is employed rather than settling for only an improved solution. The hyper-heuristics were still not exchanging information with each other. Memes have been passed down through the application of the Simple Inheritance Mechanism. Put another way, the child has inherited the memes of the parent who is the fittest.

Numerous researchers have demonstrated the effectiveness of the Great Deluge acceptance mechanism in the literature (Özcan et al., 2010). This investigation demonstrated the effectiveness of this technique once more. The updated algorithm became the best hyper-heuristic in this field with a score of 37.5 in MAX-SAT. Not only has it improved, but it has also received scores of 10, 14, 5, 20, and 16 in bin packing, PS, TSP, and VRP, in that order. But in Flow Shop, the score dropped to six.

The improved algorithm now has an overall score of 104.1, and the hyper-heuristic has moved up to fourth place. The results are shown in Table 3.

The percentage of hyper-heuristic selection was similarly impacted by this modification. While Reinforcement Learning received the highest selection rate (45.33%) in the first hyper-heuristic, it received a higher percentage (51.44%) in the second hyper-heuristic.

Table 3: Detailed scores for Hyper-Heuristic 2 (Algorithm 2) with compare to other competitors

	Multi-meme SR-OI + RL-GD + Modified CF With NO Information Sharing between Heuristics							
Rank	**Multimeme SR-OI + RL-GD + Improved CF**	**MAX-SAT**	**Bin Packing**	**Personnel Scheduling**	**Flow Shop**	**TSP**	**VRP**	**Total**
1	AdaptiveHH	**29.9**	**43.0**	9.0	**36.0**	**36.3**	14.0	**168.2**
2	HsiaoCHeSCHyperheuristic	**29.9**	3.0	**37.5**	**33.0**	16.3	5.0	**124.7**
3	ML	10.5	10.0	**29.5**	**39.0**	12.0	20.0	**121.0**
4	**Multimeme SR-OI + RL-GD + Modified CF**	**37.6**	**10.0**	**14.5**	**6.0**	**20.0**	**16.0**	**104.1**
5	Pearl Hunter 0.0.6	7.5	3.0	11.5	8.0	**25.3**	**30.0**	**85.3**
6	EPH by David M.	0.0	9.0	9.5	20.0	**33.3**	12.0	**83.8**
7	Test Hyper Heuristic (iridia.ulb.ac.be)	11.5	19.0	1.0	22.0	11.0	5.0	**69.5**
8	LehrbaumHAHA	26.9	0.0	**24.0**	2.8	0.0	14.0	**67.8**
9	ISEA Hyper-Heuristic	3.5	**29.0**	14.5	3.5	9.0	4.0	**63.5**
10	SimSATS_HH	19.9	11.0	7.5	0.0	0.0	**21.0**	**59.4**

11	HAEA Hybrid Adaptive Evolutionary Algorithm	0.0	2.0	2.0	9.3	10.0	**26.0**	**49.3**
12	ACO_HH	0.0	**20.0**	0.0	8.3	7.0	1.0	**36.3**
13	CS-PUT Genetic Hive Hyper Heuristic	0.0	13.0	6.5	7.0	2.0	6.0	**34.5**
14	Dynamic Iterated Local Search With Non Improvement Bias	0.0	12.0	0.0	0.0	11.0	0.0	**23.0**
15	AVEG_NeptuneHyperHeuristic	10.5	0.0	0.0	0.0	0.0	9.0	**19.5**
16	sa_ilsHyperHeuristic	0.3	0.0	16.0	0.0	0.0	3.0	**19.3**
17	ShafiXCJ	3.5	11.0	0.0	0.0	0.0	4.0	**18.5**
	Colours representation :		**First**	**Second**	**Third**			

5. Conclusion

Automated techniques that can be used to a wide range of problem situations from various areas are provided by hyper-heuristics. Using instances of various instances of different domains from HyFlex, this study examined three separate hyper-heuristics that incorporate three different acceptance mechanisms while manipulating the mutation rate and depth of search parameters of various low-level heuristics. Two distinct hyper-heuristic combinations were examined. Thirty distinct cases originating from six distinct domains were employed. A multi-meme memetic strategy was employed to regulate the attributes of each individual, including the depth of search, mutation rate, and hyper-heuristic selection procedure. A mutation with a probability of 0.2 is applied to every meme in order to boost diversification. These two algorithms' outputs were carefully examined and contrasted with the outcomes of previous CHeSC2011 participants.

The results have shown that, second algorithm outperformed not only the first implemented algorithm, but also lots of competitors of CHeSC2011.

6. Further Studies

It was not possible to evaluate various combinations of heuristic selection techniques and acceptance mechanisms because the testing of the suggested algorithms required over 200 hours. Memes were employed in this study to choose hyper-heuristic selection techniques, with the parameters being restricted to search depth and mutation rate.

Different memes can be used to regulate the acceptance mechanisms and heuristic selection techniques in future research. This will broaden the range of mechanisms that can be used to various problem scenarios. Put differently, this will enable the hyper-heuristic to combine the Simple Random heuristic selection approach with several acceptance methods, such as All Moves, Only and Equally Improvement, Great Deluge, Simulated

Annealing, and so on. As a result, several acceptance mechanisms may be quickly implemented for every heuristic selection technique.

References

A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithm," IEEE Trans. Evol. Comput., vol. 3, pp. 124–141, Jul. 1999.

Asmuni, H., Burke, E. K., Garibaldi, J. M., & McCollum, B. (2007). A novel fuzzy approach to evaluate the quality of examination timetabling. In Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT'06), LNCS, 3867, (pp. 327-346), Springer.

Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E and Woodward J (2009), Exploring hyper-heuristic methodologies with genetic programming. In: Mumford C and Jain L (eds). Computational Intelligence: Collaboration, Fusion and Emergence, Intelligent Systems Reference Library. Springer: New York, pp 177–201.

Burke, E., Kendall, G., & Soubeiga, E. (2003b), A tabu-search hyper-heuristic for timetabling and rostering. Journal of Heuristics, 9, 451-470, Kluwer Academic Publishers

Cowling, P., Kendall, G., & Soubeiga, E. (2001a). A hyperheuristic approach to scheduling a sales summit. In Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling (PATAT'00), (pp. 176-190), Springer-Verlag.

E. Burke and G. Kendall, Search methodologies: introductory tutorials in optimization and decision support techniques. Springer Science+ Business Media, 2005.

E. Burke and G. Kendall, Search methodologies: introductory tutorials in optimization and decision support techniques. Springer Science+ Business Media, 2005.

E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. ¨Ozcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," Journal of the Operational Research Society, 2013.

E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. ¨ Woodward, "A classification of hyper-heuristic approaches," in Handbook of Metaheuristics, ser. International Series in Operations Research & Management Science, M. Gendreau and J.-Y. Potvin, Eds. Springer US, 2010, vol. 146, pp. 449–468.

E. Özcan, M. Misir , G. Ochoa, E. K. Burke, A Reinforcement Learning - Great-Deluge Hyper-heuristic for Examination Timetabling, International Journal of Applied Metaheuristic Computing, 1(1), pp. 39-59, 2010.

E. Ozcan, S. Asta, and C. Altıntas, "Memetic algorithms for cross domain heuristic search," in Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI 2013), Y. Jin and S. A. Thomas, Eds. Surrey, UK: IEEE Press, 2013, pp. 175–182.

F. Neri and C. Cotta, "Memetic algorithms and memeting computing optimization: A literature review," Swarm and Evolutionary Computation, vol. 2, pp. 1–14, 2012.

Fisher H, Thompson GL (1963) Probabilistic learning combination of local job-shop scheduling rules. In: Muth JF, Thompson GL (eds) Industrial Scheduling, Prentice-Hall, Inc, New Jersey, pp 225-251.

J. E. Smith et al., "Co-evolution of memetic algorithms: Initial investigations," in Parallel Problem Solving From Nature—PPSN VII, G. Guervos et al., Eds. Berlin, Germany: Springer, 2002, vol. 2439, Lecture Notes in Computer Science, pp. 537–548.

J. E. Smith, "Co-evolving memetic algorithms: A learning approach to robust scalable optimization," in IEEE Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press, 2003, vol. 1, pp. 498–505.

John H. Drake, Ender Özcan and Edmund K. Burke An Improved Choice Function Heuristic Selection for Cross Domain Heuristic Search The 12th International Conference on Parallel Problem Solving From Nature (PPSN 2012), Lecture Notes in Computer Science, Volume 7492, pp. 307-316, 2012.

Kaelbling, L. P., Littman, M., & Moore, A. (1996), Reinforcement learning: a survey. Journal of Artificial Intelligence Research, 4, (pp. 237-285).

Kendall G and Mohamad M (2004a), Channel assignment in cellular communication using a great deluge hyper-heuristic. In: Proceedings of the 2004 IEEE International Conference on Network (ICON2004). IEEE: Singapore, pp 769–773.

M. Gendreau and J.-Y. Potvin, "Metaheuristics in combinatorial optimization," Annals of Operations Research, vol. 140, no. 1, pp. 189–213, 2005.

M. Gendreau and J.-Y. Potvin, "Metaheuristics in combinatorial optimization," Annals of Operations Research, vol. 140, no. 1, pp. 189–213, 2005.

N. Krasnogor, "Studies on the Theory and Design Space of Memetic Algorithms," Ph.D., Faculty of Comput., Math., and Eng., Univ. of the West of England, Bristol, U.K., 2002.

N. Krasnogor, B. Blackburne, J. D. Hirst, and E. K. N. Burke, "Multimeme algorithms for the structure prediction and structure comparison of proteins," in Parallel Problem Solving From Nature, 2002, Lecture Notes in Computer Science.

Ozcan, E., Bilgin, B., Korkmaz, E.: Hill Climbers and Mutational Heuristics in Hyperheuristics. In: PPSN IX. pp. 202–211, 2010.

P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech concurrent computation program, C3P Report, vol. 826, p. 1989, 1989.

Qu, R., Burke, E. K., & McCollum, B. (2008a). Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. European Journal of Operational Research, 198(2), 392-404.

R. Dawkins, The Selfish Gene. New York City: Oxford University Press, 1976.

R. Hinterding, Z. Michalewicz, and A. E. Eiben, "Adaptation in Evolutionary Computation: A Survey," in IEEE International Conference on Evolutionary Computation. Piscataway, NJ: IEEE Press, Apr. 1997, pp. 65–69.

Ross P (2005). Hyper-heuristics. In: Burke EK and Kendall G, Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. chap 17, Springer: Berlin, pp 529–556.

Ross, P., Hart, E., Marin-Blazquez, J., & Schulenberg, S. (2003). Learning a procedure that can solve hard bin-packing problems: a new GA-based approach to hyperheuristics. Proceedings of Genetic and Evolutionary Computation Conference (GECCO'2003), (pp. 1295-1306).

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: an introduction. The MIT Press.

Terashima-Marin, H., Moran-Saavedra, A., & Ross, P. (2005). Forming hyper-heuristics with GAs when solving 2D-regular cutting stock problems. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, 2, (pp. 1104-1110), IEEE Press.

Terashima-Marin, H., Ross, P., & Valenzuela-Rendon, M. (1999). Evolution of constraint satisfaction strategies in examination timetabling. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), (pp 635-642).

The University of Nottingham CHeSC2011 website. Retrieved April ,21, 2021 from: http://www.asap.cs.nott.ac.uk/external/chesc2011/hyflex_description.html

Vazquez-Rodriguez, J.A., Petrovic, S. & Salhi, A. (2007). A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup times and uniform machines. In P. Baptiste, G. Kendall, A. Munier-Kordon & F. Sourd (Ed.), the 3rd Multi-disciplinary International Scheduling Conference: Theory and Applications (MISTA'07), (pp. 506-513), Paris, France.

Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu, and Kok-Wai Wong, "Classification of Adaptive Memetic Algorithms: A Comparative Study" IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 36, pp. 141-152 No. 1, February 2006

Yilmaz, A. A., Guzel, M. S., Bostanci, E., & Askerzade, I. (2020). A novel action recognition framework based on deep-learning and genetic algorithms. *IEEE Access*, *8*, 100631-100644.