# Late Acceptance-Based Selection Hyper-heuristics for Cross-domain Heuristic Search

Warren G. Jackson, Ender Özcan and John H. Drake
School of Computer Science
University of Nottingham
Jubilee Campus
Wollaton Road
Nottingham, NG8 1BB, UK
Email: {psxwgj, ender.ozcan, psxjd2} @nottingham.ac.uk

***Abstract*—Hyper-heuristics are high-level search methodologies used to find solutions to difficult real-world optimisation problems. Hyper-heuristics differ from many traditional optimisation techniques as they operate on a search space of low-level heuristics, rather than directly on a search space of potential solutions. A traditional iterative selection hyper-heuristic relies on two core components, a method for selecting a heuristic to apply at a given point and a method to decide whether or not to accept the result of the heuristic application. Raising the level of generality at which search methods operate is a key goal in hyper-heuristic research. Many existing selection hyper-heuristics make use of complex acceptance criteria which require problem specific expertise in controlling the various parameters. Such hyper-heuristics are often not general enough to be successful in a variety of problem domains. Late Acceptance is a simple yet powerful local search method which has only a single parameter to control. The contributions of this paper are twofold. Firstly, we will test the effect of the set of low-level heuristics on the performance of a simple stochastic selection mechanism within a Late Acceptance hyper-heuristic framework. Secondly, we will introduce a new class of heuristic selection methods based on roulette wheel selection and combine them with Late Acceptance acceptance criteria. The performance of these hyper-heuristics will be compared to a number of methods from the literature over six benchmark problem domains.**

## I. Introduction

Solving real-world NP-hard [1] optimisation problems such as timetabling, bin-packing and vehicle routing is an ongoing research challenge. Finding the optimal solution to such problems by exhaustively enumerating the search space is infeasible in practice for many cases due to the exponential amount of time that is required to do so. Heuristic and meta-heuristic methods are often applied to these problems sacrificing some guarantee of solution quality for an improvement in the time taken to execute. Hyper-heuristics are a class of high-level search techniques which aim to raise the level of generality at which search methods operate [2], [3]. Unlike traditional search methods which operate on a search space of solutions, hyper-heuristics operate on a search space of heuristics or heuristic components. Cowling et al. [4] presented the first work to use the term 'hyper-heuristic' in the field of combinatorial optimisation however methods which show hyper-heuristic properties can be traced back to the early 1960's [5].

Hyper-heuristic methodologies are split into two main categories, methods which select from a set of low-level heuristics which to apply at a given time in a search and those methods which generate new heuristics from existing heuristics or heuristic components [6], [7]. Here we are concerned with the first category, selection hyper-heuristics. Selection hyper-heuristics have previously been applied to a wide range of real-word optimisation problems such as bin packing [8], [9], dynamic environments [10], [11], knapsack problems [12], scheduling [4], [5], [8], [13], [14], timetabling [9], [13], [15], [16], [17], [18], [19], [20] and vehicle routing [8], [21], [22].

The HyFlex [23] framework was initially developed for the first Cross-domain Heuristic Search Challenge (CHeSC) [8] and is a software framework *"designed to enable the development, testing and comparison of iterative general-purpose heuristic search algorithms (such as hyper-heuristics)"*. This framework provides six pre-implemented problem domains allowing researchers to concentrate on the development and analysis of high-level search methodologies for cross-domain search rather than on the implementation details of various problem domains and low-level heuristics.

In this paper, we will introduce a number of new heuristic selection strategies and compare their performance to a baseline Simple Random selection strategy when paired with Late Acceptance move acceptance criteria over a number of problem domains provided by the HyFlex framework. Özcan et al. [15] used Late Acceptance as a move acceptance criteria in hyper-heuristics applied to examination timetabling benchmarks. This work suggested that Late Acceptance could be successful when paired with Simple Random selection however it was unable to perform as well when used with 'intelligent' selection methods such as Choice Function and Reinforcement Learning. We show that it is possible to design simple 'intelligent' selection methods which can outperform random heuristic selection over a number of benchmarks.

## II. Selection Hyper-heuristics

Özcan et al. [24] decomposed single-point search selection hyper-heuristics into two key components; a selection mechanism and a move acceptance criteria. Hyper-heuristics of this nature will often be referred to as *selection method-acceptance criteria* in this paper herein. In such a framework, selection hyper-heuristics have an iterative cycle between heuristic selection and move acceptance. Operating on a single solution, a low-level heuristic is selected and applied at each point before

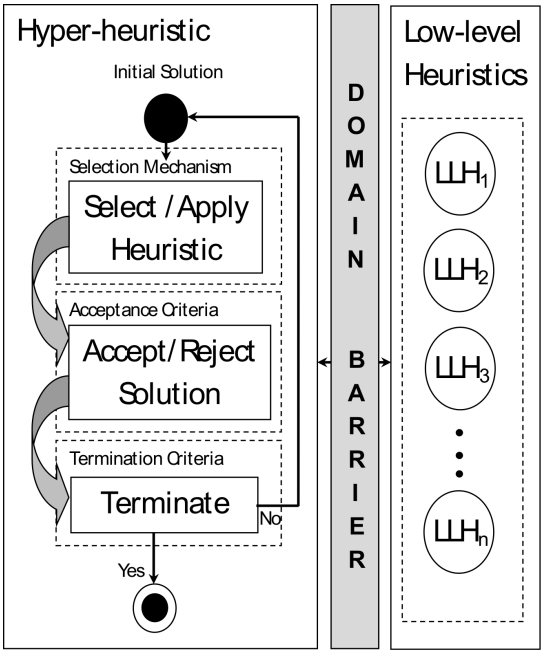


Fig. 1. A traditional selection hyper-heuristic framework

a decision is made whether to accept the modification made by the selected low-level heuristic. This process is repeated until some termination criteria is met. A traditional single-point search selection hyper-heuristic framework is shown in Figure 1. Section II-A and Section II-B will introduce a number of standard selection methods and move acceptance criteria from the literature respectively.

### A. Heuristic Selection Mechanisms

There are a large number of other heuristic selection methods for which a complete review is far beyond the scope of this paper. Here we will introduce a number of the most popular techniques from the literature.

In their early hyper-heuristic work, Cowling et al. [4] experimented with Simple Random (SR), Greedy and Choice Function (CF) heuristic selection methods. Since that time a large number of papers in the literature have used Simple Random [10], [11], [12], [15], [17], [19], [20], [25], Greedy [4], [10], [15], [17], [20] or Choice Function [10], [12], [15], [17], [20], [26] as heuristic selection methods. Simple Random is a pure stochastic selection method which selects a heuristic to apply randomly from the set of low-level heuristics available. In Greedy selection, all available heuristics are applied to the given solution with the best move made considered by the move acceptance criteria. The Choice Function is a more elegant selection method which gives each heuristic a score based on three measures. The heuristic to apply is then chosen by a strategy based on these scores. The first measure records the previous performance of each heuristic while the second measure captures any pair-wise dependencies between heuristics by monitoring the performance of each heuristic when invoked immediately following the previously selected heuristic. The final measure is simply the time elapsed since the individual heuristic was last selected by the Choice Function.

Reinforcement learning (RL) [27] is a classic machine learning technique explored as a heuristic selection method by Nareyek [28] and a number of other papers in the literature [10], [11], [12], [15], [16], [17], [29]. In such a framework all heuristics are given a utility weight. If a heuristic improves a solution this weight is increased by an amount defined by the chosen adaptation function, conversely if a heuristic does not improve a solution this weight is decreased accordingly. Heuristic selection at the next step of the search is then based on these values.

### B. Move Acceptance Criteria

As with heuristic selection methods there are a large number of move acceptance criteria presented in the literature, many of which are based on general optimisation techniques.

Dueck [30] introduced Great Deluge (GD) as a general optimisation technique and has shown to be a promising acceptance criteria in hyper-heuristics [10], [16], [25]. When used as an acceptance criteria, Great Deluge always accepts improving moves and accepts worsening moves which are below a certain threshold which is lowered over time at a linear rate.

Late Acceptance (LA) is a relatively new general purpose optimisation strategy proposed by Burke and Bykov [31] widely used as an acceptance criteria in selection hyper-heuristics [12], [15], [19], [22]. Late Acceptance promotes a general trend of improvement throughout the search process by comparing a candidate solution to one generated a specified number of steps before kept in memory.

Simulated Annealing (SA) [32] is another generic meta-heuristic technique for optimisation often used as an acceptance criteria in hyper-heuristics [9], [10], [12], [19], [20]. In Simulated Annealing, any move which results in a solution of equal or greater quality than the previous move is accepted. If a move yields a poorer quality solution, the move is accepted probabilistically based on how much poorer the neighbouring solution is and the current temperature (a parameter which decreases over time). The acceptable level of worsening solutions will decrease as the temperature decreases and the probability of moving to a worse solution will reduce over time.

In addition to move acceptance criteria based on generic optimisation techniques, a number of simple criteria have also been proposed. Cowling et al. [4] experimented with accept All Moves (AM) and accept Only Improving moves (OI). Along with accepting Improving or Equal moves (IE) [10], [11], [19] AM [10], [11], [26] and OI [12] have been widely adopted in the literature.

## III. Late Acceptance-based Selection Hyper-heuristics

Late Acceptance (LA) is a generic optimisation method [31] which is an extension of simple hill-climbing. In hill-climbing, a solution is accepted if it is of better quality than the one immediately preceding it, Late Acceptance accepts a solution if it is of better quality than the solution $n$ iterations previously, where $n$ is the size of a memory of previously seen solutions. Late Acceptance has shown to perform as well as other powerful optimisation methods

including Simulated Annealing and Great Deluge [15], [31], but only relies on the setting of one parameter, the length of the memory. As a result it is less problem specific and more general purpose, requiring less parameter tuning than the methods mentioned previously and therefore more independent of human expertise. Although only a single parameter is required, it is still important that this parameter is set up appropriately. The length of the memory can affect performance when using Late Acceptance, if it is too short the search will converge on a sub-optimal point quickly, if the memory is too long the search will stagnate.

Two selection methods will be explored in this paper. The first is Simple Random selection which chooses a heuristic to apply at each step randomly from the set of available low-level heuristics. The second is a new fitness proportionate selection mechanism based on the relative ranking points scoring system previously used by Formula 1 motor racing.

### A. Relative ranking inspired heuristic selection

The original Formula 1 ranking system (2003-2009) assigns a number of points to different competitors based on their performance. The first place competitor is awarded 10 points, the second 8 points and then each further heuristic is awarded 6, 5, 4, 3, 2, 1 and 0 points respectively. Formula 1 Fitness Proportional Selection (F1FPS) ranks each heuristic from 1 to $N$ where $N$ is the total number of heuristics. The ranks are then mapped to scores based on the Formula 1 ranking system (Rank, Points), $\{(1,10),(2,8),(3,6),(4,5),(5,4),(6,3),(7,2),(8,1)\}$. If a rank is shared by multiple heuristics, the score is evenly distributed to those heuristics. These scores are then used to calculate the probability of selecting each heuristic using Roulette Wheel selection. In Roulette Wheel selection each heuristic $i$ is selected with a probability $p_i$ proportional to its utility weight calculated as:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{1}$$

where the fitness function $f_i$ returns the scores mapped from their respective ranks for each heuristic. A reinforcement learning mechanism is used to update the rank of each heuristic following each iteration of the search process.

A number of variants of this selection method combined with Late Acceptance move acceptance criteria will be tested to take into consideration a number of factors. We will consider combinations of all of the options outlined below.

As we are using a discrete ranking system, some method needs to be determined in order for the reinforcement learning mechanism to rank each heuristic. Initially the ranking of each heuristic is determined arbitrarily as the order of the low-level heuristics in the heuristic set. Subsequent rankings are updated for the heuristic applied at each stage using one of three learning mechanisms described in Table I.

Previous work has shown that counter-intuitively, a reinforcement learning scheme which rewards a heuristic obtaining a worsening solution can outperform simply rewarding improving solutions [33]. In addition to the standard ranking system described above we consider a reverse Formula 1 ranking system. That is, each of the heuristics are ranked in reverse order. Using a reverse technique, we demote heuristics which generate solutions which intensify or diversify the solution too often based on the learning mechanism used. With a reverse approach, acceptance-based and combined learning will demote heuristics which continually improve the solution in hand thus increasing the frequency of a natural diversification of the search. This will hopefully encourage short intervals of intensification and diversification which will favour longer convergence times and better results. Fitness-based acceptance however will promote heuristics that cause micro-reheats, potentially causing too much diversification to occur.

TABLE I. DESCRIPTION OF LEARNING MECHANISMS USED TO UPDATE HEURISTIC RANKINGS

| Learning Mechanism | Heuristic Promotion | Heuristic Demotion |
|---|---|---|
| Acceptance | Promote the heuristic if it generated a candidate solution which was accepted by the move acceptance criteria | Demote the heuristic if it generated a candidate solution which was not accepted by the move acceptance criteria. |
| Fitness | Promote the heuristic if it generated a candidate solution whose fitness was strictly less than the fitness of the solution previously accepted by the move acceptance criteria. | Demote the heuristic if it generated a candidate solution whose fitness was greater than or equal to the fitness of the solution previously accepted by the move acceptance criteria. |
| Combined | Promote the heuristic if it generated a candidate solution whose fitness was strictly less than the fitness of the solution previously accepted by the move acceptance criteria or if it generated a solution which was accepted by the move acceptance criteria. | Demote the heuristic if it generated a candidate solution whose fitness was greater than or equal to the fitness of the solution previously accepted by the move acceptance criteria and if it generated a solution which was not accepted by the move acceptance criteria. |

As the standard Formula 1 ranking system allocates scores to the top 8 ranked contestants, heuristics which are ranked $\geq 9^{th}$ position will experience starvation from the set of heuristics which can be chosen from. Unless a particular heuristic is especially bad, this starvation is detrimental to the search procedure as any one of these might become a good heuristic to use in a later stage of the search. Therefore two schemes are tested, the first stays true to the original rankings and gives heuristics $\geq 9^{th}$ position a score of 0. The second scheme instead assigns a score of 1 to all heuristics $\geq 9^{th}$ position giving all heuristics a positive probability of selection.

If multiple heuristics share the same ranking, methods to distinguish between the two should exist, we consider a 'fair' system and an 'unfair' system to manage these cases. In the 'fair' system the scores for those heuristics are evenly distributed. Some low-level heuristics are very fast operations and can have 10s of thousands of applications per minute. Sharing the scores between equal ranks in this way is a very expensive operation relative to the time taken to apply a heuristic. To try and produce a faster score allocation method such that the number of iterations of the hyper-heuristic in a fixed time frame will be greatly increased, an 'unfair' scoring system is used. The 'unfair' score allocation assigns ranks to heuristics as they appear when ordered by their reinforcement

fitness scores. This means that for example, if two heuristics both share 1st place, the first heuristic found will gain a score associated with rank 1 and the second heuristic found will gain a score associated with rank 2.

## IV. Experimentation and Results

In order to test and compare our hyper-heuristics we will use the benchmarks provided by the HyFlex framework. The HyFlex [8] framework (Available to download from http://www.asap.cs.nott.ac.uk/external/chesc2011/index.html) offers a common framework in which to test heuristic search algorithms over a broad set of problem domains. This framework was used to support an international research competition, the Cross-domain Heuristic Search Challenge (CHeSC) [34]. In total six problem domains are provided; boolean satisfiability (MAX-SAT), one dimensional bin packing, personnel scheduling and permutation flow shop, the vehicle routing problem (VRP) and the travelling salesman problem (TSP). For each problem domain a set of black-box low-level heuristics are defined and characterised as either ruin-recreate, mutation, local search or crossover.

For each hyper-heuristic we will test a subset of 3 different instances for each domain, using three pre-defined seeds $\{1234, 5678, 101010\}$ and taking the median score for each instance to compare between different hyper-heuristics. The instances chosen for each domain are; MAX-SAT $\{0, 10, 11\}$, bin packing $\{0, 10, 11\}$, personnel scheduling $\{0, 10, 11\}$, flow shop $\{0, 10, 11\}$, VRP $\{0, 1, 2\}$ and TSP $\{0, 8, 9\}$. Each instance is allowed to run for 10 competition minutes as defined by the CHeSC rules. For the Late Acceptance move acceptance criteria, three memory lengths are tested $\{50, 1000, 10000\}$.

To compare hyper-heuristics, a modified version of the Formula 1 points system was used which was designed to compensate for the number of hyper-heuristics being compared. This was because the Formula 1 ranking system was designed for the number of contestants in the formula 1 racing competition, therefore, comparison of smaller sets of hyper-heuristics could cause all hyper-heuristics to obtain scores which otherwise would not happen in a larger set. This could allow hyper-heuristics which under-perform in certain areas to gain an unfair advantage. In the unmodified version of Formula 1 ranking, the top 8 contestants obtain scores based on their ranks. From rank $1 \rightarrow 8$, the score allocation is $\{10, 8, 6, 5, 4, 3, 2, 1\}$. In the original Formula 1 scoring system there were around 22 contestants competing in any one race. This meant that roughly the top $1/3^{rd}$ of contestants gained scores while the remaining $2/3^{rds}$ obtained a zero score. Inspired by this we define a generalised pattern of the relations of scores to ranks:

- Contestant $c_1$ scores $\lceil \frac{n}{3} \rceil + 2$
- Contestant $c_2$ scores $\lceil \frac{n}{3} \rceil$
- Contestant $c_i$ scores $max((\lceil \frac{n}{3} \rceil - i) + 1, 0)$ for $i \geq 3$

We will use a generalised *Compensative Formula 1 Ranking System* which follows the rules outlined above. For example if we have 12 hyper-heuristics to compare, the ones ranked $1 \rightarrow 4$ for a given problem instance would gain scores $\{6, 4, 2, 1\}$ for their respective ranks $\{1, 2, 3, 4\}$.

### A. Simple Random - Late Acceptance hyper-heuristics with differing heuristic subsets

Simple Random selection has shown previously to work well when paired with Late Acceptance move acceptance criteria in hyper-heuristics [15]. Previous work has suggested that reducing the set of heuristics from which to choose can lead to improved performance [35]. In light of this, we will test the SR-LA hyper-heuristic over a number of heuristic subsets and analyse the performance. In total four heuristic subsets will be tested, three are static heuristic sets while the fourth is a dynamic set updated during the search. The first subset is simply all of the available heuristics. In the case of crossover heuristics the operator is applied using a single parent solution given as the current solution. The second is using only the available mutation heuristics and the third the set of heuristics which show mutational behaviour i.e. $mutation \cup ruin\text{-}recreate$. The final heuristic set which is inspired by the success of selecting and applying a mutational heuristic followed by a pre-defined hill climber before deciding whether to accept a move in the work of Özcan et al [36] over a number of other hyper-heuristic frameworks. Mutational operators alone are often only effective for certain problem domains, while other problem domains require local search heuristics. Therefore a combination of both was designed such that to begin with, only mutational heuristics exist to prevent fast convergence in the search for problem domains that actually only need mutational heuristics. If no solutions are accepted by the late acceptance move acceptance method for $\frac{1}{2}$ of the length of the memory, then it is possible that no further improvements can be made using only mutational heuristics, therefore a random local search heuristic is added to the set of heuristics to be chosen from in an attempt to generate accepting solutions and progress the search. Upon each injection of a local search heuristic, the counter for non-accepting solutions is reset to zero and restarted such that multiple more local search heuristics may be added if necessary. For each heuristic subset, three different lengths of memory for the Late Acceptance criteria are used {50, 1000, 10000}.

The results of stochastic selection of the various heuristic subsets using the Compensative Formula 1 Ranking System are shown in Table II.

TABLE II. Comparison of Simple Random - Late Acceptance hyper-heuristics with different low-level heuristic sets using Compensative Formula 1 Ranking

| Heuristic Subset Methods | Memory Length | | |
|---|---|---|---|
| | 50 | 1000 | 10000 |
| All Heuristics | 29.00 | 51.00 | 62.00 |
| Mutation Heuristics | 2.00 | 2.00 | 12.50 |
| Heuristics with Mutational Properties | 12.00 | 10.50 | 4.50 |
| Heuristics with Mutational Properties and Local Search Injection | 17.50 | 24.50 | 4.50 |

For all memory lengths, stochastic selection of All Heuristics outperformed all other heuristic subsets. In addition to this it is observed that for smaller memory lengths, Mutational Properties with Local Search Injection proved to outperform Mutation Heuristics and Heuristics with Mutational Properties showing that inclusion of local search heuristics within a late acceptance hyper-heuristic are beneficial for a general-

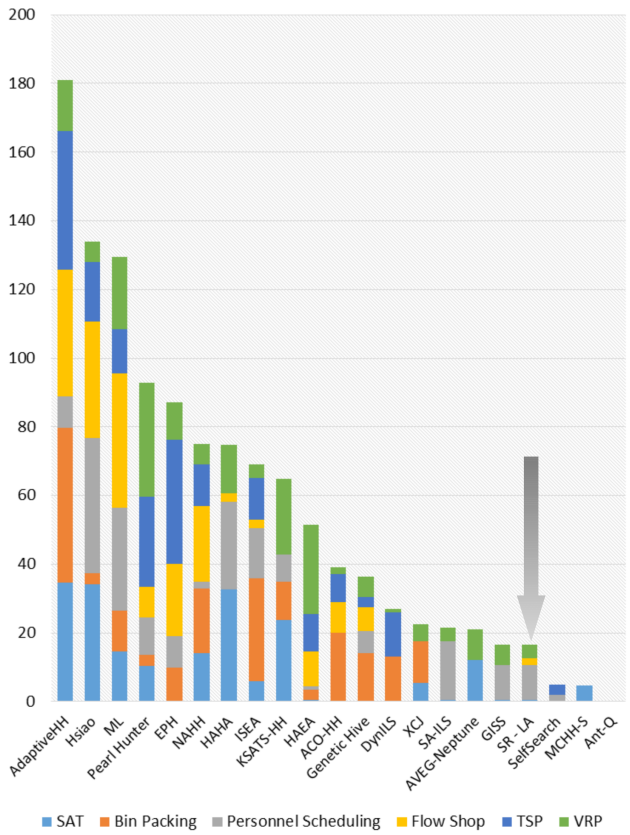


Fig. 2. Comparison of Simple Random - Late Acceptance hyper-heuristic with All Heuristics and CHeSC 2011 competitors

purpose hyper-heuristic. In general, stochastic selection over a subset of heuristics was found not to be as powerful as stochastic selection over all heuristics. This is positive in terms of generality as the hyper-heuristic framework used can make use of a larger set of low-level heuristics.

The best performing hyper-heuristic from this set is SR-LA with a list length of 10,000, operating over the set of all available instances. The performance of this hyper-heuristic against the original competitors is shown in Figure 2. In order to perform this comparison the hyper-heuristic is tested with the same rules as the original CHeSC competition, performing 31 runs of the hyper-heuristic over each problem instance using a selection of 5 problem instances from each of the available problem domains (30 instances in total). The median score of each of the 31 runs for each instance is taken and the original Formula 1 scoring system applied to the hyper-heuristic competitors (21 in total). The hyper-heuristic is given 10 competition minutes to solve each instance.

From this we see that SR-LA outperforms 3 of the competition entrants, finishing 17th out of 21 hyper-heuristics. This hyper-heuristic scores well in the personnel scheduling (10 points) and vehicle routing problem (4 points) domains scoring 16.6 points in total over all six problem domains using the original Formula 1 ranking system.

### *B. Formula 1 ranking inspired fitness proportional selection with Late Acceptance hyper-heuristics*

A number of variants of the F1FPS heuristic selection method are described in Section III-A. In total there are four parameters to consider for the fitness proportionate selection heuristic selection method. In addition to this we will consider the same three different memory lengths for Late Acceptance as described in Section IV-A.

Each hyper-heuristic will be labeled following a convention indicating which particular parameters have been chosen. The first character will represent whether the hyper-heuristic uses the normal (N) Formula 1 ranking system or the reverse (R) Formula 1 ranking system which allocates ranks in reverse order. The second character corresponds to the 'fair' (F) or 'unfair' (U) distribution of scores for heuristics with equal ranks. The third character denotes the type of learning mechanism used, this is either acceptance-based (A), fitness-based (F) or a combination of the two (C). The final character indicates whether heuristics with a rank $\geq 9^{th}$ are allocated a score of zero (0) or (1). As an example, the selection method labeled *F1FPS_NFA0* would be rank each heuristic using the standard ordering, in the case of two heuristics sharing a ranking the scores would be distributed fairly, an acceptance-based learning mechanism is used to promote/demote heuristics and all heuristics with a rank $\geq 9^{th}$ are allocated a score of zero. Essentially each different combination of these parameters represents a different hyper-heuristics. Each of these parameters will be isolated at some point during the following sequence of results to ascertain whether these components have a bearing on the overall performance of a hyper-heuristic.

*1) Comparison of Formula 1 Parameters with acceptance-based learning mechanism:* Here we will fix the learning mechanism used to decide whether to promote or demote a heuristic to acceptance-based learning. The acceptance-based learning mechanism strategy promotes or demotes heuristics based on if the candidate solution produced is accepted by the move acceptance criterion. Using the 18 instances over six problem domains described at the beginning of Section IV, the hyper-heuristics are ranked for each instance using the Compensative Formula 1 Ranking System. Each hyper-heuristic is again tested with three different memory lengths for Late Acceptance and allowed 10 competition minutes to solve each instance. The results are shown in Table III.

TABLE III. COMPARISON OF F1FPS HYPER-HEURISTICS WITH ACCEPTANCE LEARNING USING THE COMPENSATIVE FORMULA 1 RANKING SYSTEM

| Hyper-Heuristic | Memory Length | | |
|---|---|---|---|
| | 50 | 1000 | 10000 |
| F1FPS_NFA0 | 0.50 | 0.50 | 0.50 |
| F1FPS_NFA1 | 0.50 | 0.50 | 0.50 |
| F1FPS_NUA0 | 42.00 | 67.00 | 50.00 |
| F1FPS_NUA1 | 39.50 | 64.75 | 72.00 |
| F1FPS_RFA0 | 11.50 | 11.00 | 14.50 |
| F1FPS_RFA1 | 11.50 | 11.00 | 14.50 |
| F1FPS_RUA0 | 24.50 | 38.00 | 49.25 |
| F1FPS_RUA1 | 18.00 | 60.75 | 99.25 |

These results suggest that assigning a score of 1 for heuristics with rank $\geq 9^{th}$ tends to offer better performance

than assigning a score of 0 for a large memory length and that the opposite is true for a small memory length. In general, with the exception of F1FPS_NUA0, a higher memory length is favoured by this set of hyper-heuristics. Unfair score allocation also yields much better results than the corresponding fair hyper-heuristic. This could be attributed to the fact that the unfair score allocation procedure does not suffer from the same computational overheads that the fair score allocation procedure does. As a result a higher number of heuristic applications are possible within in the same time frame. In the case of fair ranking the reverse ranking method always yielded better hyper-heuristics. In contrast, for unfair ranking the reverse ranking method yields worse results with the exception of F1FPS_RUA1 with memory length 10,000 which actually outperforms all other hyper-heuristics in this set.

*2) Comparison of learning mechanisms:* In order to compare the three learning mechanisms, the order of the ranking system and the fairness of score distribution were fixed to N and F respectively. Again these tests are performed over the same 18 instances with the parameters as described previously using three different memory lengths for the Late Acceptance move acceptance criteria. The results are presented in Table IV.

TABLE IV. COMPARISON OF DIFFERING LEARNING MECHANISMS USING THE COMPENSATIVE FORMULA 1 RANKING SYSTEM

| Hyper-Heuristic | Memory Length | | |
|---|---|---|---|
| | 50 | 1000 | 10000 |
| F1FPS_NFA0 | 0.00 | 0.00 | 0.00 |
| F1FPS_NFA1 | 0.00 | 0.00 | 0.00 |
| F1FPS_NFF0 | 19.00 | 27.50 | 33.50 |
| F1FPS_NFF1 | 13.00 | 20.50 | 34.50 |
| F1FPS_NFFA0 | 0.00 | 0.00 | 0.00 |
| F1FPS_NFFA1 | 0.00 | 0.00 | 0.00 |

These results show that the fitness-based learning mechanism greatly outperforms the acceptance and fitness acceptance learning mechanisms for this set of hyper-heuristics. In line with the results observed in Section IV-B1 the hyper-heuristics with the largest memory size perform well.

*3) Comparison of hyper-heuristics using the fitness-based learning mechanism:* As shown in Section IV-B2 the fitness-based learning mechanism was shown to outperform the acceptance-based and combined learning mechanisms. Here we will combine this learning mechanism with the top four combinations of parameters found using the acceptance-based learning mechanism shown in Table III. Table V shows the results of these experiments.

TABLE V. COMPARISON OF HYPER-HEURISTICS UTILISING FITNESS-BASED LEARNING USING THE COMPENSATIVE FORMULA 1 RANKING SYSTEM

| Hyper-Heuristic | Memory Length | | |
|---|---|---|---|
| | 50 | 1000 | 10000 |
| F1FPS_NUF0 | 7.00 | 12.00 | 0.00 |
| F1FPS_NUF1 | 15.00 | 20.50 | 28.00 |
| F1FPS_RUF0 | 14.00 | 25.80 | 44.80 |
| F1FPS_RUF1 | 10.30 | 30.20 | 27.30 |

Again a large memory size for the Late Acceptance mechanism is performing well in addition to reversing the scores assigned to each heuristic based on ranking. Unlike some of the previous results however, a hyper-heuristic which allocates a score of zero to heuristics ranked $\geq 9^{th}$ clearly outperforms all of the other competitors.

*4) Comparison of best hyper-heuristics:* As a final comparison of the best derived hyper-heuristics, the best hyper-heuristics from the immediately preceding sections (IV-B1,IV-B2,IV-B3) were compared with each other and the current best hyper-heuristic found in Section IV-A, Simple Random - Late Acceptance with All Heuristics. The results using the Compensative Formula 1 Ranking System are shown in Table VI.

TABLE VI. FORMULA 1 SCORES FOR COMPARISON OF CURRENT BEST HYPER-HEURISTICS

| Hyper-Heuristic | Memory Lengths | | |
|---|---|---|---|
| | 50 | 1000 | 10000 |
| Simple Random - Late Acceptance | 14.00 | 16.80 | 21.30 |
| F1FPS_RUA1 - Late Acceptance | 5.00 | 13.00 | 48.70 |
| F1FPS_NFF1 - Late Acceptance | 11.00 | 17.00 | 28.20 |
| F1FPS_RUF0 - Late Acceptance | 4.00 | 20.70 | 34.30 |

In this comparison, we see that it is necessary to have a large memory size in order for the fitness proportionate hyper-heuristics to outperform Simple Random selection. The best hyper-heuristic found uses Formula 1 fitness proportionate selection with reverse ranking, unfair score allocation, an acceptance-based learning mechanism and assigns scores of 1 to heuristics ranked $\geq 9^{th}$ (abbreviated to F1FPS_RUA1 - Late Acceptance). As the best performing hyper-heuristic overall we compare F1FPS_RUA1 - Late Acceptance to the CHeSC competitors in Figure 3. These results are calculated using the original Formula 1 scoring system on the median results of 31 runs on each of 30 instances taken from the six problem domains.

These results are greatly improved over the results of the Simple Random - Late Acceptance hyper-heuristic shown in Figure 2. F1FPS_RUA1- Late Acceptance finished 10th out of the 21 competitors scoring 54.2 points. This hyper-heuristic performs particularly well in the personnel scheduling (24 points) and MAX-SAT (24.2 points) and also scores in permutation flow shop and VRP (3 points in each domain).

## V. CONCLUSIONS AND FUTURE WORK

In this work, we have analysed the effect of differing sets of low-level heuristics on a Simple Random - Late Acceptance hyper-heuristic and introduced a new class of heuristic selection mechanisms. In addition to this we have introduced a generalised version of the Formula 1 ranking system which can be used to measure the performance of any number of competing techniques. It has been observed that reducing the heuristic search space can be detrimental to the performance of a Simple Random - Late Acceptance hyper-heuristic. A large memory size for hyper-heuristics using Late Acceptance move acceptance works well using the selection methods presented here. In line with previous work in the literature although slightly counter-intuitive, reversing the scores assigned to a heuristic (i.e. giving the best performing hyper-heuristic the lowest score and vice versa) can lead to improved performance offering sufficient diversification to prevent the search from stagnating. We have presented results for a simple stochastic

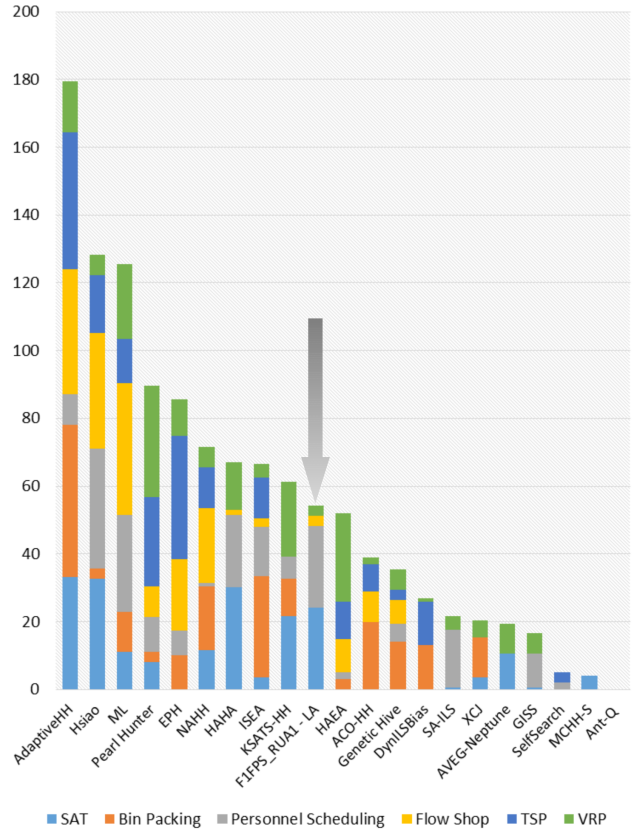


Fig. 3. Comparison of F1FPS_RUA1 - Late Acceptance hyper-heuristic with All Heuristics and CHeSC 2011 competitors

hyper-heuristic and a fitness proportionate selection mechanism inspired by the Formula 1 ranking system using Late Acceptance over a set of six benchmark problem domains. Swan et al. [37] compared a number of variants of Late Acceptance as an acceptance criteria applied to the Daily Car-Pooling problem. These variants modify the values stored within the memory of the Late Acceptance mechanism after a given number of idle iterations. We are currently in the process of including these variants within the existing framework in order to assess their performance.

## References


[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[2] P. Ross, "Hyper-heuristics," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Technologies*, E. K. Burke and G. Kendall, Eds. Springer, 2005, ch. 17, pp. 529–556.

[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*, 2013.

[4] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2000)*, ser. LNCS, E. K. Burke and W. Erben, Eds., vol. 2079. Konstanz, Germany: Springer, 2001, pp. 176–190.

[5] M. Fisher and G. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Factory Scheduling Conference*, Carnegie Institute of Technology, 1961.

[6] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, *Handbook of Metaheuristics 2nd ed.* Springer, 2010, ch. A Classification of Hyper-heuristics Approaches, pp. 449–468.

[7] E. Özcan and A. J. Parkes, "Policy matrix evolution for generation of heuristics," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 2011–2018.

[8] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. Burke, "Hyflex: A benchmark framework for cross-domain heuristic search," in *Proceedings of Evolutionary Computation in Combinatorial Optimization (EvoCOP 2012)*, ser. LNCS, vol. 7245. Malaga, Spain: Springer, 2012, pp. 136–147.

[9] R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, and B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support," *4OR: A Quarterly Journal of Operations Research*, vol. 10, no. 1, pp. 43–66, 2012.

[10] B. Kiraz, A. S. Uyar, and E. Özcan, "Selection hyper-heuristics in dynamic environments," *Journal of the Operational Research Society*, 2013, to appear.

[11] M. Köle, S. Etaner-Uyar, B. Kiraz, and E. Özcan, "Heuristics for car setup optimisation in torcs," in *Proceedings of 12th Annual Workshop on Computational Intelligence (UKCI 2012)*, Edinburgh, UK, 2012, pp. 1–8.

[12] J. H. Drake, E. Özcan, and E. K. Burke, "Controlling crossover in a selection hyper-heuristic framework," School of Computer Science, University of Nottingham, Tech. Rep. No. NOTTCS-TR-SUB-1104181638-4244, 2011.

[13] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[14] J. Gibbs, G. Kendall, and E. Özcan, "Scheduling english football fixtures over the holiday period using hyper-heuristics," in *Proceedings of Parallel Problem Solving from Nature (PPSN 2011)*, ser. LNCS, R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, Eds., vol. 6238. Kraków, Poland: Springer, 2011, pp. 496–505.

[15] E. Özcan, Y. Bykov, M. Birben, and E. K. Burke, "Examination timetabling using late acceptance hyper-heuristics," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*. Trondheim, Norway: IEEE Press, 2009, pp. 997–1004.

[16] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A reinforcement learning - great-deluge hyper-heuristic for examination timetabling," *International Journal of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39–59, 2010.

[17] E. K. Burke, G. Kendall, M. Misir, and E. Özcan, "Monte carlo hyper-heuristics for examination timetabling," *Annals of Operations Research*, vol. 196, no. 1, pp. 73–90, 2012.

[18] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Applied Intelligence*, vol. 37, no. 1, pp. 1–11, 2012.

[19] P. Demeester, B. Bilgin, P. D. Causmaecker, and G. V. Berghe, "A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice," *Journal of Scheduling*, vol. 15, no. 1, pp. 83–103, 2012.

[20] M. Kalender, A. Kheiri, E. Özcan, and E. Burke, "A greedy gradient-simulated annealing hyper-heuristic for a curriculum-based course timetabling problem," in *Proceedings of 12th Annual Workshop on Computational Intelligence (UKCI 2012)*, Edinburgh, UK, 2012, pp. 1–8.

[21] P. Garrido and C. Castro, "Stable solving of cvrps using hyperheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009)*, F. Rothlauf, Ed. Québec, Canada: ACM, 2009, pp. 255–262.

[22] M. Misir, W. Vancroonenburg, K. Verbeeck, and G. V. Berghe, "A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete," in *Proceedings of the Metaheuristics International Conference (MIC 2011)*, L. D. Gaspero, A. Schaerf, and T. Stützle, Eds., Udine, Italy, 2011, pp. 289–298.

[23] E. K. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, and J. A. Vazquez-Rodriguez, "Hyflex: A flexible framework for

the design and analysis of hyper-heuristics," in *Proceedings of the Multidisciplinary International conference on Scheduling: Theory and Applications (MISTA 2009)*, Dublin, Ireland, 2009, pp. 790–797.

[24] E. Özcan, B. Bilgin, and E. E. Korkmaz, "Hill climbers and mutational heuristics in hyperheuristics," in *Proceedings of the International Conference on Parallel Problem Solving From Nature (PPSN 2006)*, ser. LNCS, T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. M. Guervós, L. D. Whitley, and X. Yao, Eds., vol. 4193. Reykjavik, Iceland: Springer, 2006, pp. 202–211.

[25] B. Bilgin, E. Özcan, and E. E. Korkmaz, "An experimental study on hyper-heuristics and exam timetabling," in *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006)*, ser. LNCS, E. K. Burke and H. Rudová, Eds., vol. 3867. Brno, Czech Republic: Springer, 2006, pp. 394–412.

[26] J. H. Drake, E. Özcan, and E. K. Burke, "An improved choice function heuristic selection for cross domain heuristic search," in *Proceedings of Parallel Problem Solving from Nature (PPSN 2012), Part II*, ser. LNCS, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds., vol. 7492. Taormina, Italy: Springer, 2012, pp. 307–316.

[27] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[28] A. Nareyek, *Metaheuristics: computer decision-making*. Kluwer Academic Publishers, 2001, ch. Choosing Search Heuristics by Non-Stationary Reinforcement Learning, pp. 523–544.

[29] L. D. Gaspero and T. Urli, "Evaluation of a family of reinforcement learning cross-domain optimization heuristics," in *Proceedings of Learning and Intelligent Optimization (LION 2012)*, ser. LNCS, Y. Hamadi and M. Schoenauer, Eds., vol. 7219. Paris, France: Springer, 2012, pp. 384–389.

[30] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *Journal of Computational Physics*, vol. 104, no. 1, pp. 86–92, 1993.

[31] E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for exam timetabling problems," in *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, Montreal, Canada, 2008, p. Extended Abstract.

[32] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[33] E. Demir, T. Bektas, and G. Laporte, "An adaptive large neighborhood search heuristic for the pollution-routing problem," *European Journal of Operational Research*, vol. 223, no. 2, pp. 346–359, 2012.

[34] E. K. Burke., M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. Parkes, and S. Petrovic, "The cross-domain heuristic search challenge - an international research competition," in *Proceedings of Learning and Intelligent Optimization (LION 2011)*, ser. LNCS, C. A. C. Coello, Ed., vol. 6683. Rome, Italy: Springer, 2011, pp. 631–634.

[35] E. Özcan and C. Basaran, "A case study of memetic algorithms for constraint optimization," *Soft Computing*, vol. 13, no. 8-9, pp. 871–882, 2009.

[36] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[37] J. Swan, J. H. Drake, E. Özcan, J. Goulding, and J. R. Woodward, "A comparison of acceptance criteria for the daily car-pooling problem," in *Computer and Information Sciences III - 27th International Symposium on Computer and Information Sciences*, E. Gelenbe and R. Lent, Eds. Paris, France: Springer, 2013, pp. 477–483.