# Examination Timetabling Using Late Acceptance Hyper-heuristics

Ender Özcan, Yuri Bykov, Murat Birben and Edmund K Burke

***Abstract* — A hyperheuristic is a high level problem solving methodology that performs a search over the space generated by a set of low level heuristics. One of the hyperheuristic frameworks is based on a single point search containing two main stages: heuristic selection and move acceptance. Most of the existing move acceptance methods compare a new solution, generated after applying a heuristic, against a current solution in order to decide whether to reject it or replace the current one. Late Acceptance Strategy is presented as a promising local search methodology based on a novel move acceptance mechanism. This method performs a comparison between the new candidate solution and a previous solution that is generated *L* steps earlier. In this study, the performance of a set of hyper-heuristics utilising different heuristic selection methods combined with the Late Acceptance Strategy are investigated over an examination timetabling problem. The results illustrate the potential of this approach as a hyper-heuristic component. The hyper-heuristic formed by combining a random heuristic selection with Late Acceptance Strategy improves on the best results obtained in a previous study.**

## I. Introduction

HYPER-HEURISTICS represent a class of techniques for solving *difficult* optimisation problems [1, 2]. A high level (meta-)heuristic that performs a search over a set of low level (meta-)heuristics rather than the solutions can be referred to as a hyper-heuristic. An increasing interest in hyper-heuristics is observed as illustrated in Table I, since they provide a simple to implement, yet a powerful framework to deal with the complexities of the real-world problems.

The low level heuristics can be *constructive* that are used to build solutions, or *perturbative* that are used to modify a given candidate solution (or solutions). For example, Burke et al. [3] uses constructive graph colouring heuristics for solving exam timetabling. On the other hand, Kendall and Mohamad [11] employ Great Deluge based hyper-heuristic that manages perturbative heuristics for solving a channel assignment problem.

A research interest to the examination timetabling problem is motivated by two reasons. First, this problem is practically important, and second, it appears as a challenging direction for theoretical studies. In fact, a tight competition between high educational institutions (e.g., universities) forces the administration to seek for additional ways of attraction new students, and of course, they are keen on increasing the variety of course modules. However, having quite restricted resources the first priority task is the effectiveness of their management where an efficient timetabling plays a major role.

On the other hand, different instances of examination timetabling problem can be considered as good benchmarks for testing, investigating and comparing a variety of optimisation techniques. These problems are usually large-scale, highly constrained and extremely difficult to solve ones. They belong to the class of NP-complete constraint optimisation problems [28], hence there is no existent method, which can produce an optimal solution in a polynomial time. As an alternative, different heuristic (including meta-heuristic and hyper-heuristic) techniques might be used here to search for an attainable minimum (near-optimum solution).

There is empirical evidence that the choice of heuristic selection and move acceptance might affect the performance of a hyper-heuristic [6]. In this study we propose to combine a novel move acceptance strategy referred to as Late Acceptance Strategy by Burke and Bykov [29] with different heuristic selection methods generating different hyper-heuristics. Their performances are compared to identify the best heuristic selection choice for this move acceptance

TABLE I
SOME PREVIOUS APPLICATIONS OF HYPER-HEURISTICS

| Application (s) | Reference (s) |
|---|---|
| Benchmark function optimisation | [4-6] |
| Bin packing | [7-10] |
| Channel assignment | [11] |
| Chicken catching and transportation | [12] |
| Component placement sequencing | [13] |
| Course timetabling | [14, 15] |
| Examination timetabling | [4, 15, 16] |
| Nurse rostering | [14] |
| Open shop scheduling | [17] |
| Orc quest, logistics domain | [18] |
| Presentation scheduling | [19] |
| Sales summit scheduling | [20] |
| Space allocation | [21, 22] |
| Trainer scheduling problem | [23-26] |
| Vehicle routing | [27] |

Ender Özcan, Yuri Bykov and Edmund K Burke are with the Automated Scheduling, Optimisation and Planning Research Group, School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom (phone: 115-846-6569; fax: 115-846-7877; e-mail: {exo, yxb and ekb}@cs.nott.ac.uk).

Murat Birben is with Yeditepe University, Department of Computer Engineering, Inonu Mahallesi, Kayisdagi Caddesi, Kadikoy, Istanbul 34755, Turkey (e-mail: mbirben@cse.yeditepe.edu.tr).

strategy for solving an examination timetabling problem as formulated by Ozcan and Ersoy [30].

In Section II, a broad overview of literature on perturbative hyper-heuristics and examination timetabling is provided. Some selected hyper-heuristic components used during the experiments will be discussed in detail. In Section III, the Late Acceptance Strategy is explained. In Section IV, the examination timetabling problem at Yeditepe University is described and the details of hyper-heuristics for solving the problem are provided. Section V discusses the experimental settings and computational results. Finally, conclusions are presented in Section VI.

## II. BACKGROUND

### A. Perturbative Hyper-heuristics

Hyper-heuristics are the emerging class of optimisation tools [1, 2]. There are a number of different hyper-heuristic frameworks that allow a high level heuristic to perform search over a space of low level heuristics. A bilevel perturbative hyper-heuristic framework is illustrated in Figure 1. Here, a high level heuristic manages a set of low level perturbative (improvement, variation) heuristics. This framework is used in our study. In a single point search, an initial solution ($s_0$) passes through two main stages iteratively. In the first stage, a heuristic selection is performed. Next, the chosen heuristic ($H_i$) applied to the current solution ($s$) and a new candidate solution is generated ($s' \leftarrow H_i(s)$). After that a decision is made whether to accept or reject the new move. If it is accepted, the new candidate solution replaces the current solution ($s \leftarrow s'$). This process continues until a certain set of termination criteria is satisfied. Hence, a hyper-heuristic will be denoted as a pair form this point onward: "heuristic selection method"–"move acceptance mechanism".

Cowling, Kendall and Soubeiga [20] explored a set of hyperheuristics combining Simple Random (SR), Random Descent, Random Permutation, Random Permutation Descent, Greedy (GR) and Choice Function (CF) heuristic selection methods with two simple move acceptance strategies: accept All Moves (AM) and Only Improving moves (OI). Simple Random selects a heuristic randomly from $k$ given low level heuristics. The Greedy (GR) method invokes $k$ heuristics successively using the same candidate solution and compares the quality of $k$ new solutions and selects the heuristic that generates the solution with highest quality. Choice Function (CF) is based on a scoring mechanism that statistically evaluates individual and pair-wise performances of the low level heuristics. The time passed since the last invocation of the selected heuristic is also weighed in the scoring function. The heuristic with the maximal score is selected for employment. Cowling, Kendall and Soubeiga observed that the Choice Function–All Moves hyperheuristic performed better than the rest.

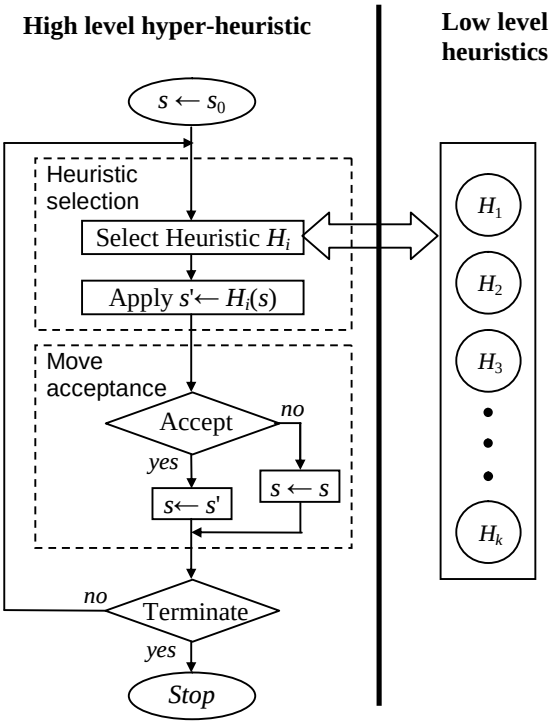


Fig. 1. A bilevel hyper-heuristic framework

Nareyek [18] explored Reinforcement Learning (RL) for choosing heuristics during the search process. A utility value (score) is assigned to each heuristic. If a selected heuristic improves the current solution, its utility is increased by a given rate (*positive adaptation*). Otherwise, it is decreased using another rate (*negative adaptation*). The utility values are allowed to change within an interval of [0, *number-of-heuristics*]. At each step, a heuristic is selected based on the utility values. Different strategies can be used for this purpose. For instance, the *max* strategy chooses the heuristic with the maximal utility value, whereas *fair random* (roulette wheel strategy) arranges the probability of a heuristic being selected proportional to its utility value over the sum of utility values. Nareyek combined this strategy with All Moves acceptance and investigated combinations of different adaptation schemes and heuristic selection strategies for solving Orc quest and logistics domain problems. The results show that the additive reward (+1) and subtractive punishment (–1) and using the maximal utility (max) are viable choices for the Reinforcement Learning heuristic selection method.

Burke, Kendall and Soubeiga [14] proposed a hyperheuristic that combined reinforcement learning mechanism with tabu-search (TABU) as a heuristic selection method. Each heuristic is associated with a rank and manipulated in a similar manner as the utility values in [18]. All Moves was used as a move acceptance strategy. A variable length tabu list holds the heuristics that should be avoided. At each step, the heuristic with the maximum rank,

which is not in the tabu list is selected and applied to the candidate solution. Whenever there is an improvement or an equal quality solution is generated, the heuristics in the tabu list are released. If the move is a worsening move, then the selected heuristic is queued into the tabu list. The experimental results show that this hyper-heuristic delivers a similar or even better performance as compared to the custom-made meta-heuristics for solving nurse rostering and course timetabling problem instances.

### B. *Examination Timetabling Problem*

The goal of examination timetabling is to allocate a given set of events (exams) to available resources (timeslots, rooms, etc.) subject to a set of constraints. Two different types of constraints can be identified: *hard* and *soft* constraints. Hard constraints are required to be satisfied, and only a timetable without violations of hard constraints can be considered as *feasible*. Soft constraints represent the preferences that should be satisfied as many as possible (usually a timetable without violations of soft constraints does not exist).

The real-world timetabling requirements are unique for each particular university and usually reflect the conflicting priorities of the different participants of the educational process [31]. However, the most significant requirements were consolidated into a certain common scheme. It was observed that the most general hard constraint is that no student has to sit two exams at the same time. This goal is equivalent to the classical graph colouring problem [32], which causes a major hardship of Examination Timetabling. The most common soft constraint is the spreading exams over the examination session. However, different versions of this problem use different ways of calculation the spreading as well as other soft constraints.

During the years virtually every optimisation technique was applied to Exam Timetabling Problems. In a way, the timeline of Exam Timetabling studies traces the progress of optimisation algorithms and computer hardware. Thus, the earlier studies were mostly focused on various constructive methods, from the simplest heuristics as in [33] and [34], up to quite advanced ones (see [35, 36]).

The era of meta-heuristic studies was started with a particular attention to Genetic Algorithms. This technique was applied to Exam Timetabling in pioneering publications of Corne et al. [37] and Burke et al. [38]. These studies were expanded by the application of Memetic Algorithm [39, 40] and a multi-objective evolutionary algorithm (MOEA) by Paquete and Fonseca [41]. The examples of further applications of these methods can be found in [42] and [43]. Ozcan and Ersoy [30] generalized their previous study in Alkan and Ozcan [44] and proposed an extended framework for designing violation directed adaptive operators, which perform a search over the constraint oriented neighbourhoods. A memetic algorithm utilizing such a hill-climber is implemented as a problem solver in a tool called FES. FES is the first tool that supports timetabling markup language (TTML) [45] and accepts input in that format.

In parallel with the evolutionary methods the performance of different iterative search meta-heuristics was widely studied for examination timetabling, such as simulated annealing [46] and tabu search [47]. The investigations of this family of techniques are still continued involving innovative search methodologies such as GRASP (Greedy Randomized Adaptive Search Procedure) [48], trajectory-based multi-objective search [49], Ahuja-Orlin's very large neighbourhood search [50], the generalization of the great deluge algorithm called as Flex-Deluge Algorithm [16]. The most recent example is the Late Acceptance Strategy by Burke and Bykov [29]. Additionally, a variety of other optimisation methods were applied to examination timetabling including constraint satisfaction techniques [51], case based reasoning [15, 52], fuzzy methods [53], ant colony [53] as well as different hybrid methods [55, 56].

A special attention in the recent years is paid to the application of different hyperheuristic approaches to examination timetabling. In Bilgin, Ozcan and Korkmaz [4] tested a set of hyperheuristics that combine heuristic selection and move acceptance mechanisms. Burke et al. [3] investigated a hyperheuristic based on a tabu search mechanism that assigns proper graph colouring heuristics for constructing an examination timetable.

The significantly larger list of publications as well as more detailed information about examination timetabling studies can be found in a number surveys starting from Carter [57] and continued by Burke et al. [58], Shaerf [59], Burke and Petrovic [60]. The most recent survey is provided by Qu et al. [61].

## III. Late Acceptance Strategy

*Late Acceptance Strategy* (LAS) is a new and original general purpose meta-heuristic technique. It was proposed by Burke and Bykov [29] who studied it over a set of examination timetabling problems. Although nowadays the investigations of the properties of LAS are still in their earlier stages, this method has already showed quite promising performance. The presented research has been done in course of the current investigations of the properties of this new method.

Late Acceptance Strategy belongs to the family of iterative search techniques but it employs an advanced acceptance mechanism. It is atypical to the most of the existing search meta-heuristics (such as Hill-Climbing, Simulated Annealing, etc.) where at each iteration; a new generated candidate solution is compared with a current one. In contrast, the main idea of LAS is to compare the candidate solution with the one which was "current" several iterations earlier. Correspondingly, each current solution is used for comparison not at the immediate, but at some later iteration. The "delay" in the comparison inspires the name of this new meta-heuristic and also enables the use the simplest greedy

acceptance rule. The current version of Late Acceptance Strategy compares the cost functions of the candidate and its "late" competitor and only candidates with better (or equal) cost are accepted.

The implementation of the above idea appears as a list (fitness array), which contains cost functions of current solutions during a number of recent iterations. In order to maintain the list of an invariable length $L$, at each iteration, the algorithm inserts the value of the current cost into the beginning of the list and simultaneously removes the last element from the end. An example of the execution of 3 consecutive iterations of LAS is shown in Fig. 2.

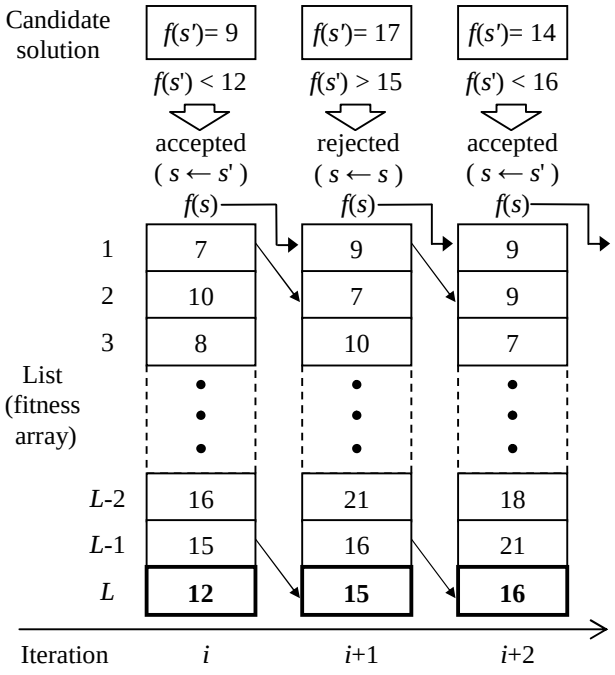


Fig. 2. An illustration of how LAS executes for a minimization problem.

In this example the candidate solutions at $i^{th}$ and $(i+2)^{th}$ iterations are accepted as their cost is less than the value from the end of the list. Correspondingly, the $(i+1)^{th}$ candidate is rejected (its cost is higher than the $L^{th}$ element of the list). Here the main rule of LAS is highlighted that the current (not the candidate) cost is inserted into the list. When the candidate is rejected, the current cost remains equal to the previous one and therefore the list contains a series of same values.

In practical implementation the insertion/removing procedure can be simplified by just exchanging the value of the fitness array element, which is calculated as a virtual beginning/end (see more details in [29]). The authors also give recommendations regarding the initialization of the array and termination criteria. At the beginning of the search all array elements are the same and are equal to the initial cost. Similar to other search methods, the execution of the algorithm can be run until convergence (no improvement for a high number of iterations). However, this algorithm tends to make an extremely slow improvement at the last stage of the search, so it could be reasonable to employ some other termination criteria, such as fixing the total execution time. The convergence issues of this algorithm are currently under investigation and could be considered as a subject of future work.

The existence of the fitness array indicates that LAS follows the idea of Laguna and Glover [62] of the "intelligent" use of the information collected during the search. The most popular interpretation of this idea is the Tabu Search, which operates with the list of recent moves. However, the nature of the lists and the ways of operation are different in Late Acceptance Strategy and Tabu Search and they can be seen as absolutely different methods.

The main advantage of Late Acceptance Strategy is its simplicity: it is easy for implementation and quite straightforward in tuning. De-facto, the performance of this algorithm depends on a single genuine parameter: length of the list $L$. This indicates the overall robustness of the method and high immunity against possible mistakes in parameterization (which can appear when, for example, defining a cooling schedule.). The dependence of the performance of the method on the length $L$ was investigated in [29]. It was argued that with $L$=1, the Late Acceptance Strategy degenerates into a pure Hill-Climbing. This method is known to be very fast, but hardly able to produce a good quality results. The further experiments showed that with the increasing of $L$ the performance of the method is changing. The search becomes slower, but the quality of result increases. This is quite logical, as with longer $L$, more worsening moves are accepted, so the search has less chance to be stuck in local minima. With relatively long $L$ (several thousands) this algorithm becomes powerful enough to be competitive with the best existing meta-heuristics. It was able to produce very strong results during a relatively short running time (several minutes).

## IV. SOLVING AN EXAMINATION TIMETABLING PROBLEM

### *A. Examination Timetabling Problem at Yeditepe University*

The set of constraints for an examination timetabling problem differs from institution to institution. There are many variants as discussed in [61]. In this study, a capacitated examination timetabling problem variant at Yeditepe University as introduced in [30] is used. The problem requires arrangement of a set of exams for a given number of days. Only three consecutive exams are allowed to be arranged in a day. A list of students and the exams that they take are provided as input. This unique problem imposes the following constraints:

i) The exams that each student takes must be assigned to different timeslots.
ii) The total number of students taking an exam at a timeslot is not allowed to exceed a predetermined capacity.
iii) If a student is scheduled to take two exams in the same day, the exams must not be assigned to successive timeslots.

A direct representation is used to represent a candidate solution encoding all events and their mappings onto a given timetable of size 3×*days*. The following fitness function is used to evaluate a given solution *S*:

$$fitness(S) = -1/\ (\forall i \sum w_i\ V_i(S)+1), \quad (1)$$

where $i$ is the constraint type, $V$ measure the number of violations in $S$ due to the $i^{th}$ constraint type and $w_i$ is the weight of the given constraint type.

### B. Low Level Heuristics

Both heuristic selection and move acceptance are equally important while designing a hyper-heuristic. In this study, Simple Random, Greedy, Reinforcement Learning, Reinforcement Learning with Tabu Search and Choice Function heuristic selection methods are combined with Late Acceptance Strategy under a perturbative hyper-heuristic framework for solving the examination timetabling problem at Yeditepe University. These heuristic selection mechanisms are chosen since they are reported to have potential in [4].

Four perturbative low level heuristics are implemented under the hyper-heuristics for solving the exam timetabling problem. The first one is a random perturbation operator commonly referred to as *mutation* in Genetic Algorithms, while three remaining ones search constraint based neighbourhoods. The mutation makes a pass over each exam in the list sequentially and randomly reschedules it with a probability of (1/*number-of-exams*).

The constraint based perturbative heuristic aims to resolve the conflicts for a corresponding constraint type. During this process two decisions have to be made. The first decision is to choose an exam to reschedule and the second is to choose a timeslot to be assigned for ir. For both tasks, a tournament strategy that chooses an item based on the number of conflicts with a tour size of 2 is used. The heuristics for resolving the constraints (i) and (iii) execute in a similar way. The tournament strategy selects an exam with the maximal number of conflicts among two randomly drawn exams for rescheduling. Then, the same strategy is employed selecting a timeslot that minimizes the number of conflicts between two randomly chosen exams. The heuristic for resolving the constraint (ii) selects a timeslot with the largest number of capacity conflicts using tournament. An exam with the maximum number of attendants is chosen for rescheduling using a tournament over the set of exams scheduled for the selected timeslot. After a tournament, a timeslot that minimizes the number of attendants is assigned to the selected exam.

## V. EXPERIMENTS

### A. Experimental Setup and Data

Toronto benchmark collection, provided by Carter et al. [64], is a well known data set in the exam timetabling community. This set contains 13 real world problems. As the problem formulation presented in this study is unique, Toronto benchmark data are extended with new characteristics as illustrated in Table II. This version of the data set is initially used by Bilgin et al. [4].

TABLE II
CHARACTERISTICS OF THE EXPERIMENTAL DATA

| Instance | Exams | Density | Days | Capacity |
|---|---|---|---|---|
| car92 I | 543 | 0.14 | 12 | 2000 |
| car91 I | 682 | 0.13 | 17 | 1550 |
| ear83 I | 190 | 0.27 | 8 | 350 |
| hecs92 I | 81 | 0.42 | 6 | 650 |
| kfu93 | 481 | 0.06 | 7 | 1955 |
| lse91 | 381 | 0.06 | 6 | 635 |
| pur93 I | 2419 | 0.03 | 10 | 5000 |
| rye92 | 486 | 0.07 | 8 | 2055 |
| sta83 I | 139 | 0.14 | 4 | 3024 |
| tre92 | 261 | 0.06 | 10 | 655 |
| uta92 I | 622 | 0.13 | 12 | 2800 |
| ute92 | 184 | 0.08 | 3 | 1240 |
| yor83 I | 181 | 0.29 | 7 | 300 |

Pentium IV 3 GHz LINUX machines having 2 Gb memories are used during the experiments. Each hyper-heuristic experiment with a problem instance is repeated fifty times. The value of $L$ is fixed as 500 in all experiments. The search is terminated whenever the execution time exceeds 600 CPU seconds or there is no violation, in order to achieve a fair comparison between all algorithms.

### B. Computational Results

The performances of five late acceptance hyper-heuristics, namely; Simple Random–LAS, Greedy–LAS, Reinforcement Learning–LAS, Choice Function–LAS and Reinforcement Learning with Tabu Search–LAS are compared and the results are summarised in Table III and Figure 3. Each hyper-heuristic is ranked from 1 to 5 for each problem instance with respect to the best candidate solution achieved in fifty runs, where 1 indicates the best performing approach, while 5 is the worst performing one. The ties in case of similar performance are taken into consideration during ranking. For example, if the 2nd and 3rd ranking items have similar performance, their ranks are provided as 2.5 for both. The Simple Random–LAS hyper-heuristic performs the best with an average ranking of 2.35 over all problems

(Figure 3), whereas the Greedy–LAS hyper-heuristic performs the worst. Moreover, Simple Random–LAS generates the best results in 5 out of 13 problem instances.

TABLE III
PERFORMANCE COMPARISON OF LAS BASED HYPER-HEURISTICS EACH USING A DIFFERENT HEURISTIC SELECTION METHOD BASED ON RANKINGS. THE BEST APPROACHES ARE MARKED IN BOLD.

| Instance | CF | GR | RL | SR | TABU |
|---|---|---|---|---|---|
| car-f-92 | 4 | 5 | 3 | 2 | **1** |
| car-s-91 | 4 | 5 | **1** | 2.5 | 2.5 |
| ear83 I | 4 | **1** | 2 | 5 | 3 |
| hecs92 I | 2 | **1** | 3 | 5 | 4 |
| kfu93 | 4 | 5 | 3 | 2 | **1** |
| lse91 | 2 | 5 | 4 | **1** | 3 |
| pur93 I | **1** | 5 | 4 | 2 | 3 |
| rye92 | 3 | 5 | 2 | **1** | 4 |
| sta83 I | **2** | 4.5 | **2** | **2** | 4.5 |
| tre92 | 3 | 5 | 2 | **1** | 4 |
| uta92 I | 3 | 5 | 2 | 4 | **1** |
| ute92 | 4.5 | 3 | 4.5 | **1** | 2 |
| yor83 I | 5 | **1** | 4 | 2 | 3 |

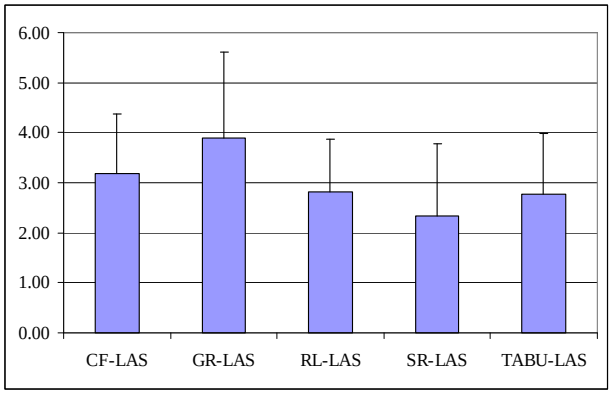


Fig. 3. Average ranking and the standard deviation of each LAS based hyper-heuristic

As another evaluation criterion, the student's t-test is also performed. The results indicate that the performance differences between Simple Random–LAS versus Reinforcement Learning–LAS, Reinforcement Learning with Tabu Search–LAS and Choice Function–LAS are statistically significant for {tre92}, {pur93 I} and {car92 I, tre92}, respectively, within a confidence interval of 95%. Simple Random–LAS performs significantly better than Greedy-LAS almost for all problem instances, except sta83 I and ute92. For these problem instances, Simple Random–LAS delivers a slightly better performance. The Greedy heuristic selection method selects the most improving heuristic to be invoked at each step. The overall hyper-heuristic framework turns into a hill climbing approach. It seems that the Greedy method can not compensate the time it looses while invoking all heuristics and gets stuck at local optima.

The experimental results in [4] show that Choice Function and a generic Simulated Annealing move acceptance yields the best results over the same examination timetabling problem. A final comparison is performed between this hyper-heuristic and the Simple Random–LAS as illustrated in Table IV. Simple Random–LAS generates the best performance that is statistically significant almost for all problem instances, except hecs92 I, sta83 I and ute92. Both hyper-heuristics have a similar performance for sta83 I and ute92. On the other hand, Choice Function–Simulated Annealing performs significantly better only for solving hecs92 I.

TABLE IV
PERFORMANCE COMPARISON OF SIMPLE RANDOM–LAS AND CHOICE FUNCTION–SIMULATED ANNEALING [4] HYPER-HEURISTICS BASED ON T-TEST OVER EACH PROBLEM INSTANCE.

| Instance | Performance | Confidence Interval |
|---|---|---|
| car-f-92 | SR-LAS > CF-SA | 0.99 |
| car-s-91 | SR-LAS > CF-SA | 0.99 |
| ear83 I | SR-LAS > CF-SA | 0.99 |
| hecs92 I | CF-SA > SR-LAS | 0.99 |
| kfu93 | SR-LAS > CF-SA | 0.99 |
| lse91 | SR-LAS > CF-SA | 0.99 |
| pur93 I | SR-LAS > CF-SA | 0.99 |
| rye92 | SR-LAS > CF-SA | 0.99 |
| sta83 I | SR-LAS ≈ CF-SA | 0.82 |
| tre92 | SR-LAS > CF-SA | 0.99 |
| uta92 I | SR-LAS > CF-SA | 0.99 |
| ute92 | SR-LAS ≈ CF-SA | 0.22 |
| yor83 I | SR-LAS > CF-SA | 0.99 |

">" indicates "is better than" and the performance difference for the given approach is statistically significant within the given confidence interval. "≈" indicates "is slightly better than" and the performance difference is not significant.

## VI. CONCLUSION

It has been observed in the previous studies that different combinations of heuristic selection and move acceptance methods in a perturbative hyper-heuristics framework might generate different performances. In this study, the best heuristic selection match is investigated for a recently proposed move acceptance strategy, referred to as Late Acceptance Strategy over a set of examination timetabling problems. The results indicate the success of hyper-heuristic combination of Simple Random heuristic selection and Late Acceptance Strategy. Learning mechanisms based on reinforcement learning or statistical analyses do not function well in combination with the late acceptance strategy. Reinforcement Learning, Reinforcement Learning with Tabu Search and Choice Function heuristic selection mechanisms

make use of a short term memory. These elaborate schemes score each heuristic based on their previous and current performances. The delay within the acceptance mechanism seems to cause the learning mechanisms to respond to the changes in the search environment less rigorously than expected. The experimental results show that Reinforcement Learning–LAS, Reinforcement Learning with Tabu Search–LAS and Choice Function–LAS hyper-heuristics still have potential. As a future work, different learning rates and memory lengths for these heuristic selection methods can be investigated. The Simple Random–Late Acceptance Strategy hyper-heuristic outperforms the previous best hyper-heuristic for solving the examination timetabling problem.

## REFERENCES


[1] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in: Glover F, Kochenberger G (eds) *Handbook of Metaheuristics*, Kluwer, pp. 457-474, 2003.

[2] P. Cowling , G. Kendall, E. Soubeiga, "Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation," in EvoWorkShops, Springer, *Lecture Notes in Computer Science*, vol 4193, pp. 1-10, 2002.

[3] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, "A graph-based hyper heuristic for educational timetabling problems," *Eur. J. of Op. Res.* 176: 177-192, 2007.

[4] B. Bilgin, E. Ozcan, E. E. Korkmaz, "An experimental study on hyperheuristics and exam scheduling," in Proc. of the 6th Int.l Conf. on the Practice and Theory of Automated Timetabling, pp.123–140, 2006.

[5] E. Ozcan, B. Bilgin, E. E. Korkmaz , "Hill climbers and mutational heuristics in hyperheuristics," in Runarsson, T., Beyer, H.G., Burke, E., J.Merelo-Guervos, J., Whitley, D., Yao, X. (eds.) The 9th Int. Conf. on Parallel Problem Solving from Nature, Reykjavik, Iceland. *LNCS*, vol. 4193. pp. 202–211, 2006.

[6] E. Ozcan, B. Bilgin, E. E. Korkmaz , "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, 12:1, pp. 3-23, 2008.

[7] A. Cuesta-Canada, L. Garrido, H. Terashima-Marin, "Building hyper-heuristics through ant colony optimisation for the 2d bin packing problem," KES'05, *LNCS*, vol. 3684, pp. 654–660, 2005.

[8] E. K. Burke, M. R. Hyde, G. Kendall, "Evolving bin packing heuristics with genetic programming," in Parallel Problem Solving from Nature - PPSN IX, *LNCS*, vol. 4193, pp. 860–869, Springer-Verlag, 2006.

[9] P. Ross, S. Schulenburg, J. G. Marin-Blázquez, E. Hart, "Hyperheuristics: learning to combine simple heuristics in bin-packing problems," in: Proc. of the Genetic and Evo. Comp. Conf., New York, July 9-13, pp. 942–948, 2002.

[10] P. Ross, S. Schulenburg, J. G. Marin-Blázquez, E. Hart, "Learning a procedure that can solve hard bin-packing problems: a new ga-based approach to hyper-heuristics," in Proc. Genetic and Evo. Comp., Springer, *LNCS*, vol. 2724, pp. 1295–1306, 2003.

[11] G. Kendall, M. Mohamad, "Channel assignment in cellular communication using a great deluge hyper-heuristic," in Proc. of the 12th IEEE Int. Conf. on Networks, vol. 2, pp. 769–773, 2004.

[12] W. Hart, , P. M. Ross, J. Nelson, " Solving a real-world problem using an evolving heuristically driven schedule builder," *Evolutionary Computation*, (6)1, 61–80 (1998)

[13] M. Ayob, G. Kendall, "A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine," in Proc. of the Int. Conf. on Intelligent Technologies, Chiang Mai, Thailand, pp.132–141, 2003.

[14] E. K. Burke, G. Kendall, E. Soubeiga, "A tabu-search hyper-heuristic for timetabling and rostering," *J.l of Heuristics* 9(6), 451–470, 2003.

[15] E. K. Burke, S. Petrovic, R. Qu, "Case Based Heuristic Selection for Timetabling Problems," *J. of Scheduling* 9(2), pp. 115-132, 2006.

[16] E. K. Burke, Y. Bykov. "An adaptive flex-deluge approach to university exam timetabling," *INFORMS J. on Computing*, submitted for publication. 2008a

[17] H.-L. Fang, P. M. Ross, D. Corne, "A promising hybrid ga/heuristic approach for open-shop scheduling problems," in Cohn, A. (ed) Proc. of the 11th Eur. Conf. on Artificial Intelligence, John Wiley and Sons Ltd., pp. 590–594, 1994.

[18] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in: Metaheuristics: Computer Decision Making, pp. 523–544. Kluwer Academic Publishers, 2004.

[19] E. K. Burke, G. Kendall, D. L. Silva, R. O'Brien, E. Soubeiga, "An ant algorithm hyperheuristic for the project presentation scheduling problem," in Proc. of the IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, pp. 2263–2270, 2005.

[20] P. Cowling, G. Kendall and E. Soubeiga, "A hyper-heuristic approach to scheduling a sales summi," in: E. K. Burke and W. Erben (eds.) The 3rd Int. Conf. on the Practice and Theory of Automated Timetabling, Konstanz, Germany, 16-18 August 2000, *Lecture Notes in Artificial Intelligence*, vol. 2079, pp. 176–190, 2001.

[21] E. K. Burke, D. L. Silva, E. Soubeiga, "Multi-objective hyper-heuristic approaches for space allocation and timetabling," in T. Ibaraki, K. Nonobe, M. Yagiura (eds.) Meta-heuristics: Progress as Real Problem Solvers, selected papers from the 5th Metaheuristics International Conference (MIC 2003), Springer, pp. 129-158, 2005.

[22] K. Dowsland, E. Soubeiga, E. K. Burke, "A simulated annealing hyper-heuristic for determining shipper sizes," *Eur. J. of Op. Res.* (179)3, 759–774, 2005.

[23] P. Cowling, G. Kendall and L. Han, "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem," in Proc. of the 2002 Congress on Evolutionary Computation, pp. 1185–1190, 2002.

[24] P. Cowling, G. Kendall and L. Han, "An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem," in Proc. of the 4th Asia-Pacific Conference on Simulated Evolution And Learning, Orchid Country Club, Singapore, pp. 267–271, 2002.

[25] G. Kendall and L. Han, "Investigation of a tabu assisted hyper-heuristic genetic algorithm," in Proc. of Congress on Evolutionary Computation, vol. 3, pp. 2230–2237, 2003.

[26] G. Kendall and L. Han, "Guided operators for a hyper-heuristic genetic algorithm," in Tamás D Gedeon and Lance Chun Che Fung (eds.) Proc. of AI-2003: Advances in Artificial Intelligence. The 16th Australian Conference on Artificial Intelligence, Perth, Australia 3-5 Dec 2003, *Lecture Notes in Artificial Intelligence*, vol. 2903, pp. 807–820. Springer (2003b)

[27] R. Bai, E. K. Burke, M. Gendreau and G. Kendall, "A simulated annealing hyper-heuristic: adaptive heuristic selection for different vehicle routing problems," in Proc. of the 3rd Multidisciplinary Int. Conf. on Scheduling: Theory and Applications, Paris, France, August 28-31, pp. 67-70, 2007.

[28] S. Even, A. Itai and A. Shamir, "On the complexity of timetable and multi-commodity flow," *SIAM J. on Computing*, vol. 5, pp. 691-703, 1976.

[29] E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for exam timetabling problems," presented at PATAT 2008 Conference, Montreal, Canada, August 18-22, 2008b.

[30] E. Ozcan, E. Ersoy, "Final Exam Scheduler - FES," in Proc. of 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp 1356-1363, 2005.

[31] E. K. Burke, D. Elliman, P. Ford, R. Weare, "Examination timetabling in British universities: a survey," in *Lecture Notes in Computer Science*, vol. 1153, E. K. Burke and P. Ross, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 1996, pp. 76-90.

[32] Leighton, F. T.: A graph coloring algorithm for large scheduling problems. J. of Res. of the National Bureau of Standards, 84:489–506 (1979)

[33] A. J. Cole, "The preparation of examination timetables using a small-store computer," *The Computer J*. vol. 7, pp. 117-121, 1964.

[34] S. Broder, "Final examination scheduling," *Communications of the ASM*, vol. 7, pp.494-498, 1964.

[35] T. Arani, V. Lotfi, "A three-phased approach to final exam scheduling," *IIE Trans.*, vol. 21, pp. 86-96, 1989.

[36] M. W. Carter, G. Laporte, J. W. Chinneck, "A general examination scheduling system," *Interfaces*, vol. 24, pp. 109-120, 1994.
[37] D. Corne, H. L. Fang, C. Melish, "Solving the module exam scheduling problem with genetic algorithms," in *Proceedings of the 6th International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, P. W./H. Chung, G. Lovergrove, M. Ali, Eds. Gordon and Breach Science Publishers, 1993, pp. 370-373.
[38] E. K. Burke, D. G. Elliman, R. F. Weare, "A genetic algorithm for university timetabling," presented at the AISB Workshop on Evolutionary Computing, Leeds, UK, April 1994.
[39] E. K. Burke, J. P. Newall, R. Weare, "A memetic algorithm for university exam timetabling," in *Lecture Notes in Computer Science*, vol. 1153, E. K. Burke and P. Ross, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 1996, pp. 241-250.
[40] E. K. Burke, J. P. Newall, "A multi-stage evolutionary algorithm for the timetabling problem," *The IEEE Transactions on Evolutionary Computation*, vol. 3(1), pp. 63-74, 1999.
[41] L. F. Paquete, C.M. Fonseca, "A study of examination timetabling with multiobjective evolutionary algorithms," in *Proc 4th Metaheuristics International Conference*, 2001, pp. 149-154.
[42] W. Erben, "A grouping genetic algorithm for graph colouring and exam timetabling," in *Lecture Notes in Computer Science*, vol. 2079, E. K. Burke and W. Erben, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2001, pp. 132-156.
[43] P. Cote, T. Wong, R. Sabouri, "Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem," in *Lecture Notes in Computer Science*, vol. 3616, E. K. Burke and M. Trick, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2005, pp. 151-168.
[44] A. Alkan, E. Ozcan, "Memetic algorithms for timetabling," in: Proc. of IEEE Congress on Evolutionary Computation, pp. 1796–1802, 2003
[45] E. Ozcan, "Towards an xml based standard for timetabling problems: TTML," in: Multidisciplinary Scheduling: Theory and Applications, Springer Verlag, 163–187, 2005.
[46] J. M. Thompson, K.A. Dowsland, "Variants of simulated annealing for the examination timetabling problem," *Annals of Operations Research*, vol. 63, pp. 105-128, 1996.
[47] L. Di Gaspero, A. Schaerf, "Tabu search techniques for examination timetabling," .in *Lecture Notes in Computer Science*, vol. 2079, E. K. Burke and W. Erben, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2001, pp. 104-117.
[48] S. Casey, J. Thompson, "GRASPing the examination scheduling problem," in *Lecture Notes in Computer Science*, vol. 2740, E. K. Burke and P. De Causmaecker, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2003, pp. 232-246.
[49] S. Petrovic and Y. Bykov, "A multiobjective optimisation technique for exam timetabling based on trajectories," in *Lecture Notes in Computer Science*, vol. 2740, E. K. Burke and P. De Causmaecker, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2003, pp. 181-194.
[50] S. Abdullah, S. Ahmadi, E.K.Burke, M. Dror, "Investigating Ahuja-Orlin's large neighbourhood search for examination timetabling," *OR Spectrum*, vol. 29, pp. 351-372, 2007.
[51] H. T. Marin, "Combinations of gas and csp strategies for solving examination timetabling problems," Ph. D. Thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey, 1998.
[52] S. Petrovic, Y. Yang, M. Dror, "Case-based selection of initialisation heuristics for metaheuristic examination timetabling," *Exp. Sys. with App*. (33)3, 772–785, 2007.
[53] H. Asmuni, E.K. Burke, J. Garibaldi, "Fuzzy multiple ordering criteria for examination timetabling," .in *Lecture Notes in Computer Science*, vol. 3616, E. K. Burke and M. Trick, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2005, pp. 344-354.
[54] Eley, M.: Ant algorithms for the exam timetabling problem. In: The 6th International Conference on Practice and Theory of Automated Timetabling, pp. 167–180. (2006)
[55] L. Merlot, N. Boland, B. Hughes, P. Stuckey, "A hybrid algorithm for the examination timetabling problem," in *Lecture Notes in Computer Science*, vol. 2740, E. K. Burke and P. De Causmaecker, Eds. Springer-Verlag, Berlin, Heidelberg, New York, 2003, pp. 207-231.
[56] M. Caramia, P. Dell'Olmo, G. Italiano, "Novel local-search based approaches to university examination timetabling," *INFORMS J. on Computing*, vol. 20, pp. 86-99, 2008.
[57] M. W. Carter, "A survey of practical applications of examination timetabling algorithms," *Operations Research*, vol. 34(2), pp. 193-201, 1986.
[58] E. K. Burke, J. Kingston, K. Jackson, R. Weare, "Automated university timetabling: The state of the art," *The Computer J*., vol. 40(9), pp. 565-571, 1997.
[59] A. Schaerf, "A survey of automated timetabling," *Artificial Intelligence Review*, vol. 13(2), pp. 187-127, 1999.
[60] E. K. Burke, S. Petrovic, "Recent research directions in automated timetabling," *Europ. J. Oper. Res.*, vol. 140, pp. 266-280, 2002.
[61] R. Qu, E. K. Burke, B. McCollum, L. Merlot, S. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling* DOI: 10.1007/s10951-008-0077-5
[62] M. Laguna and F. Glover, "What is tabu search?" *Colorado Business Review*, Vol. 6, pp. 5-12, 1996.
[63] M. W. Carter, G. Laporte, S. T. Lee, "Examination timetabling: algorithmic strategies and applications," *J. of the Op. Res. Soc*. 47, 373–383, 1996.