Non-Linear Great Deluge with Learning Mechanism for Solving the Course Timetabling Problem

Joe H. Obit* Dario Landa-Silva* Djamilah Ouelhadj* Marc Sevaux†

*ASAP Research Group, School of Computer Science, University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
jzh@cs.nott.ac.uk, dario.landasilva@nottingham.ac.uk, dxs@cs.nott.ac.uk

†University of South-Brittany
CNRS, FRE 2734, LESTER, F-56321 Lorient, France
marc.sevaux@univ-ubs.fr

1 Introduction

The course timetabling problem has been tackled using a wide range of exact methods, heuristics and meta-heuristics. In recent years, the term *hyper-heuristic* has emerged for referring to methods that use (meta-) heuristics to choose (meta-) heuristics [8]. Then, a hyper-heuristic is a process which, given a particular problem instance and a number of *low-level heuristics*, manages the selection and acceptance of the low-level heuristic to apply at any given time, until a stopping condition is met. Low-level heuristics are simple local search operators or domain dependent heuristics. Typically, a hyper-heuristic is meant to search in the space of heuristics instead of searching in the solution space directly. One of the main challenges in designing a hyper-heuristic method is to manage the low-level heuristics with minimum parameter tuning.

Early research work on hyper-heuristics focused on the development of advanced strategies for choosing the heuristics to be applied at different points of the search. For example, Soubeiga [24] used a choice function that incorporates principles from reinforcement learning. That choice function rewards or penalises the low-level heuristics according to their success in finding a better solution. Another mechanism based on tabu search was proposed by Burke et al. [9] in which a tabu list is used to prevent (for a number of iterations) the acceptance of low-level heuristics with poor performance. Ross et al. [20] used a learning classifier system to learn which heuristics were more useful than others when tackling bin packing problems. Other hyper-heuristic approaches include the GA-based hyper-heuristic by Cowling et al. [14], the case-based hyper-heuristic approach by Burke et al. [11] and the ant-based hyper-heuristic by Burke et al. [12]. Also, researchers have proposed different acceptance criteria to drive the selection of low-level heuristics within a hyper-heuristic framework. For example, Soubeiga [24] used a simulated annealing acceptance criterion, Ayob and Kendall [5] used a Monte Carlo acceptance criterion while Kendall and Mohamad [15] used the great deluge acceptance criterion.

In this paper, we propose an approach that uses a learning mechanism and a non-linear great deluge acceptance criterion in order to choose which low-level heuristic to apply while solving course timetabling problem instances. Section 2 describes the course timetabling problem tackled in this work. Section 3 reviews previous meta-heuristic and hyper-heuristic methods used to tackle this problem. Section 4 presents the non-linear great deluge hyper-heuristic method proposed in this paper while Section 5 describes and discusses our experimental results. Finally, conclusions and future research are the subject of Section 6.

2 The University Course Timetabling Problem

The university course timetabling problem can be defined as a process of allocating, subject to predefined constraints, a set of limited timeslots and rooms to courses, while satisfying as nearly as possible a set of desirable objectives. In the timetabling problem, constraints can be divided into two categories: hard and soft constraints. A timetable is said to be feasible (usable) if no hard constraints are violated. However, soft constraints may be violated and the objective is to minimise their violation in order to increase the quality of the timetable. The course timetabling problem is very complex (as discussed by Cooper and Kingston [13]) and common to a wide range of educational institutions. The manual process of preparing the timetable is tedious, time consuming and yet not guaranteed to produce a timetable free of conflicts.

Several formulations of the course timetabling problem exist in the literature. We adopt the one by Socha et al. [22] and the corresponding benchmark data sets in order to test the algorithm proposed in this paper. More formally defined, the course timetabling problem solved in this paper consists of the following: n events $E = \{e_1, e_2, \ldots, e_n\}$, k timeslots $T = \{t_1, t_2, \ldots, t_k\}$, m rooms $R = \{r_1, r_2, \ldots, r_m\}$ in which events can take place, a set F of room features satisfied by rooms and required by events, and a set S of students. Each room has a limited capacity and each student attends a number of events. The problem is to assign n events to k timeslots and m rooms in such a way that all hard constraints are satisfied and the violation of soft constraints is minimised. The benchmark data set proposed by Socha et al. [22] involves 11 instances which are split according to their size into 5 small, 5 medium and 1 large. For the small instances, $n = 100$, $m = 5$, $|S| = 80$, $|F| = 5$. For the medium instances, $n = 400$, $m = 10$, $|S| = 200$, $|F| = 5$. For the large instance, $n = 400$, $m = 10$, $|S| = 400$, $|F| = 10$. For all instances, $k = 45$ (9 hours in each of 5 days).

There are **4 hard constraints**: 1) a student cannot attend two events simultaneously (events with students in common must be timetabled in different timeslots); 2) only one event can be assigned per timeslot in each room; 3) the room capacity must be equal to or greater than the number of students attending the event in each timeslot; 4) the room assigned to the event must satisfy the features required by the event. There are **3 soft constraints**: 1) students should not have exactly one event timetabled on a day; 2) students should not attend more than two consecutive events on a day; 3) students should not attend an event in the last timeslot of the day.

3 Summary of Related Work

Meta-heuristics have been used successfully to solve some of the course timetabling problems described above. Socha et al. first proposed an *MAX-MIN* ant system [22] and then later an ant colony

system [23] in which artificial ants follow a construction graph to build a timetable. Rossi-Doria et al. [21] compared the performance of several meta-heuristics to solve this problem. The methods compared were: evolutionary algorithm, ant colony optimisation, iterated local search, simulated annealing, and tabu search. No best results were reported by Rossi-Doria et al. as the intention was to assess the strength and weaknesses of each algorithm. Asmuni et al. [4] implemented fuzzy multiple heuristic ordering in which fuzzy logic was used to establish the ordering of events prior to be timetabled. Abdullah et al. [1] proposed versions of variable neighbourhood search while Abdullah et al. [2] applied a randomised iterative improvement approach using a composite of eleven neighbourhood structures in exploring the current solution. Later, Abdullah et al. [3] presented a hybrid approach combining a mutation operator with their previous randomised iterative improvement procedure. Recently, a non-linear great deluge algorithm (NLGD) was proposed by Landa-Silva and Obit [16]. That method produced new best results in 4 out of 11 problem instances. Finally, McMullan [18] proposed an extended great deluge algorithm (EGD), which allows re-heating similar to simulated annealing, and found new best results for the 5 medium instances.

Hyper-heuristics have also been investigated for solving this course timetabling problem. Burke et al. [9] applied a choice function hyper-heuristic which also uses a tabu list to guide the iterative application of a set of simple local search heuristics. Rattadilok et al. [19] proposed a distributed choice function hyper-heuristic and implemented two designs based on a parallel architecture: hierarchical and agent-based. Burke et al. [10] proposed a graph-based hyper-heuristic in which a tabu search procedure is used to change the permutations of six graph colouring heuristics before applying them to construct a timetable. The key feature of that approach is to find good ordering of constructive heuristics to schedule the events. Bai et al. [6] developed a simulated annealing hyper-heuristic which selects low-level heuristics based on a stochastic ranking mechanism.

4 The Non-linear Great Deluge Hyper-heuristic

In this paper, we extend our non-linear great deluge algorithm (NLGD) [16] following the hyper-heuristic methodology, i.e. we incorporate a mechanism to select the low-level heuristics to apply at each step of the search process. That is, while in a NLGD meta-heuristic *candidate solutions* are accepted or not based on the great deluge criterion, in the proposed Non-Linear Great Deluge Hyper-heuristic (NLGDHH) it is *candidate low-level heuristics* which are accepted or not, i.e. the method operates in the heuristic search space.

Figure 1 illustrates the proposed method in which the low-level heuristics are local search operators which explore the solution space while the learning mechanism and the non-linear great deluge acceptance criterion explore the heuristic space. At present, we are working with few low-level heuristics, but given the promising results so far, we intend to incorporate a larger number of low-level heuristics and of different type in the extended version of this work. We use the non-linear great deluge criterion because of its simplicity and less dependent nature upon parameter tuning compared to simulated annealing [7, 16]. The low-level heuristics implemented in this work are listed below. These heuristics are based on random search but always ensuring the satisfaction of hard constraints.

H1: selects 1 event at random and assigns it to a feasible random pair timeslot-room.
H2: selects 2 events at random and swaps their timeslot-room while ensuring feasibility.
H3: selects 3 events at random and exchanges timeslot-room at random while ensuring feasibility.

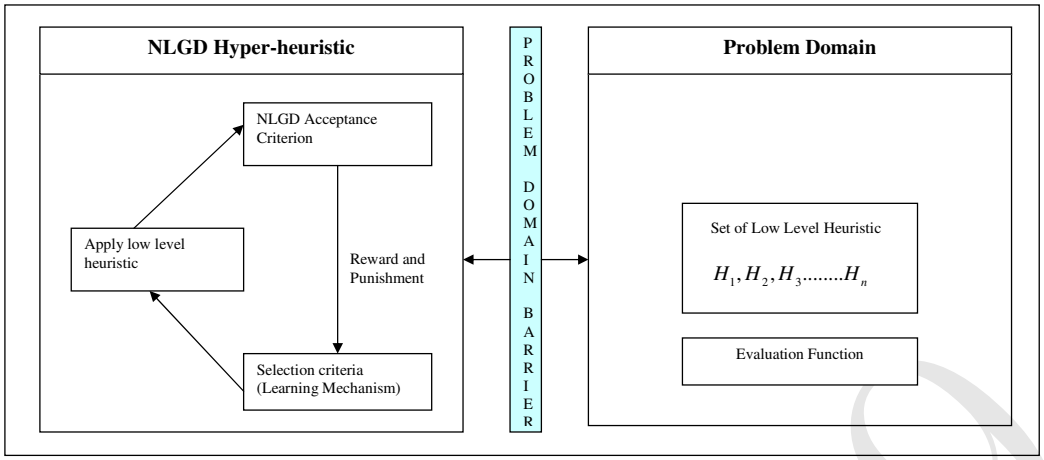

Figure 1: Non-Linear Great Deluge Hyper-heuristic Approach

4.1 Non-linear Great Deluge (NLGD) Acceptance Criterion

The NLGD acceptance criterion refers to accepting improving and non-improving low-level heuristics depending of the performance of the heuristic and the current water level B. Improving heuristics are always accepted while non-improving ones are accepted only if the detriment in quality is less than or equal B. The initial water level is usually set to the quality of the initial solution and then decreased by a non-linear function proposed in our previous work [16] as follows:

$$B = B \times (\exp^{-\delta(rnd[min,max])}) + \beta \tag{1}$$

The various parameters in Eq. (1) control the speed and the shape of the water level decay rate. Parameter β influences the shape of the decay rate and it represents the minimum expected penalty corresponding to the best solution. The role of parameters min and max is to control the speed of the decay rate. Therefore, for higher values of min and max, the water level decreases more rapidly and hence, improvements to the solution quality are also achieved faster. However, the search could get stuck and to avoid this, it is necessary sometimes to relax the water level. When the water level is about to converge to the current penalty cost, the algorithm then allows the water level to go up.

We set $\delta = 5 \times 10^{-7}$ and $\beta = 0$ for all datasets. The reason for setting $\beta = 0$ is that we want B to reach the value of zero by the end of the search. If for a given problem, the minimum penalty that should achieved is lets say 100, then β should be set around that value. If there is no previous knowledge on the minimum penalty expected (best expected fitness), then we suggest to tune β through preliminary experimentation for the problem in hand. The values of min and max in Eq. (1) are set according to the current penalty cost. When the penalty cost is more than 20 we use $min = 80000$ and $max = 90000$. When the penalty cost goes below 20 we use $min = 20000$ and $max = 30000$. When the $range < 1$ (range is the difference between the water level B and the current penalty), B is increased by a random number within the interval $[B_{min}, B_{max}]$, we call this mechanism *floating B*. For small and medium problem instances the interval used is $[0.85, 1.5]$ while for the large problem instance the interval used is $[1, 5]$.

4.2 Learning Mechanism

A simple learning mechanism (adapted from Bai et al. [6]) guides the selection of low-level heuristics during the search. Initially, all low-level heuristics have the same probability to be selected for exploring the solution space. The learning mechanism tunes the priorities of the low-level heuristics as the search progresses so that the algorithm tries to learn which low-level heuristic to use in order to better explore the solution space. In this paper, we investigate two types of learning mechanisms: *learning with static memory length* and *learning with dynamic memory length* as described below.

4.2.1 Learning with Static Memory Length

In each iteration, a low-level heuristic i is selected with probability p_i given by Eq. (2) where n is the number of heuristics and w_i is the weight assigned to each heuristic.

$$p_i = \frac{w_i}{\sum_{i=1}^{n} w_i} \tag{2}$$

Initially, every weight is set to $w_i = 0.01$ but from the first iteration, the algorithm starts to reward or punish the heuristics according the their performance. When the chosen heuristic improves the current solution, a reward of 1 point is given to the heuristic. If the heuristic does not improve the solution, the punishment is to award no points. This amount of reward/punishment never changes. However, the algorithm updates the set of weights w_i in every learning period (lp) given by $lp = max(K/500, n)$ where K is the total number of feasible moves explored.

We use the following counters to track the performance of each low-level heuristic: $Ctotal_i$, is the number of times that heuristic i is called; $Cnew_i$ is the number of times that heuristic i generates solutions with different fitness value; and $Caccept_i$ is the number of times that heuristic i meets the non-linear great deluge acceptance criterion. Each heuristic weight w_i is updated at every learning period lp and normalised by the ratio $Caccept_i/Ctotal_i$ when $range > 1$ and by $Cnew_i/Ctotal_i$ when $range < 1$. At every learning period lp and if $range < 1$, the water level increases to $B = B + rand[B_{min}, B_{max}]$, we call this mechanism *surge B*. We set B_{min} equal to 1 and B_{max} equal to 4 regardless to the size of the dataset. Note that the water level can increase due to the *floating B* (continuous) mechanism or the *surge B* (every lp feasible moves) mechanism.

4.2.2 Learning with Dynamic Memory Length

In each iteration, a low-level heuristic i is selected with probability p_i given by Eq. (3) where n is the number of heuristics, w_i is the weight assigned to each heuristic and $w_{min} = min\{0, w_i\}$.

$$p_i = \frac{w_i + w_{min}}{\sum_{i=1}^{n} w_i + w_{min}} \tag{3}$$

Initially, every weight is set to $w_i = 0.01$ as before but now each w_i is updated every time the algorithm performs a feasible move. When the selected heuristic improves the current solution, the heuristic is rewarded, otherwise the heuristic is punished. The value $\Re_{ij}$ of reward/punishment

applied to heuristic i at iteration j is as given below where $r = 1$, $\Im = 0.1$ and Δ is the difference between the best solution (lowest penalty) so far and the current solution (current penalty).

$$\Re_{ij} = \begin{cases} r & \text{if } \Delta < 0 \\ -r & \text{if } \Delta > 0 \\ \Im & \text{if } \Delta = 0 \text{ and new solution} \\ -\Im & \text{if } \Delta = 0 \text{ and no new solution} \\ 0 & \text{if not elected} \end{cases}$$

Then, at each iteration h, each weight w_i is calculated using Eq.(4) where σ gives the length of the dynamic memory.

$$w_{ih} = \sum_{j=k}^{h} \sigma^j \Re_{ij} \tag{4}$$

In every learning period lp, the algorithm updates σ with a random value in $(0.5, 1.0]$. Here, we also set $lp = max(K/500, n)$ as before. At every learning period lp and if $range < 1$, the water level increases to $B = B + rand[B_{min}, B_{max}]$. We set B_{min} equal to 1 and B_{max} equal to 4 regardless to the size of the dataset.

5 Computational Experiments and Results

To evaluate the performance of the proposed algorithm, we conducted a range of experiments using the standard course timetabling benchmark instances proposed by Socha et al. [22]. For each problem instance we executed our algorithm 10 times. The stopping condition was a maximum of 500K feasible moves or maximum computation time t_{max} or achieving a penalty value of zero, whatever was sooner. For small instances we set $t_{max} = 0$ as the algorithm takes less than 2500 seconds (42 minutes) to explore the 500K feasible moves and most of the times achieving penalty zero. For medium instances we set $t_{max} = 3$ hours and the algorithm is capable of exploring around 300K feasible moves in that time. For the large instance we set $t_{max} = 5$ hours and the algorithm is capable of exploring between 100K and 250K feasible moves in that time. Our previous NLGD meta-heuristic [16] was not able to improve results even after extending the execution time. However, the approach proposed here is now able to find better solutions thanks to the learning mechanism that selects low-level heuristics accurately to further improve the solution quality.

We first compare the proposed NLGDHH (with static and with dynamic memory length) to previous great deluge meta-heuristics in order to assess the contribution of the non-linear acceptance criterion and the learning mechanism. Table 1 shows the results obtained by the non-linear great deluge hyper-heuristic with static (NLGDHH-SM) and with dynamic (NLGDHH-DM) memory, the extended great deluge (EGD) [18], the non-linear great deluge (NLGD) [16], the evolutionary non-linear great deluge (ENLGD) [17] and the conventional great deluge (GD). We can see in Table 1 that NLGDHH-SM mostly outperforms NLGDHH-DM in terms of the number of best solutions found across all instances. Both variants of the proposed method obtain equal or better results than the other approaches, except for instances M5 and L where EGD found better solutions. However, NLGDHH-SM produced better solutions for 8 out of the 11 instances. In fact, NLGDHH-SM

improved the solutions by 11.25% for M1, 21.9% for M2, 1.44% for M3, and 37.5% for M4. The average improvements are 9.86%, 23.52%, 8.82% and 27.11% for M1, M2, M3 and M4 respectively. For M5, EGD produced an improvement of 16.98% over NLGDHH-SM while the average improvement was 13.51%. For the large instance, the best result obtained by EGD is 6.05% better and in average 8.45% better than NLGDHH-SM. The overall performance of both NLGDHH-SM and NLGDHH-DM is quite good according to these results.

Table 1: Comparison of the proposed great deluge based hyper-heuristic and other great deluge methods from the literature.

Instance	NLGDHH-SM		NLGDHH-DM		EGD		NLGD	ENLGD	GD
	Best	Avg	Best	Avg	Best	Avg	Best	Best	Best
S1	0	0.4	0	0.9	0	0.8	3	0	17
S2	0	0.7	0	1.5	0	2	4	1	15
S3	0	1	0	1.9	0	1.3	6	0	24
S4	0	1.2	0	1.4	0	1	6	0	21
S5	0	0	0	0.4	0	0.2	0	0	5
M1	71	91.4	88	107	80	101.4	140	126	201
M2	82	89.4	88	103.1	105	116.9	130	123	190
M3	137	147.8	112	148.7	139	162.1	189	185	229
M4	55	79.3	84	97.3	88	108.8	112	116	154
M5	106	138.4	103	130	88	119.7	141	129	222
L	777	911.1	915	1017.4	730	834.1	876	821	1066

We now compare the proposed NLGDHH to other hyper-heuristics reported in the literature. Table 2 presents results obtained by NLGDHH-SM, NLGDHH-DM, the choice function hyper-heuristic (CFHH) [9], the case-based hyper-heuristic (CBHH) [10], the simulated annealing hyper-heuristic (SAHH) [6] and the distributed-choice function hyper-heuristic (DCFHH) [19]. We see that the proposed method finds equal or better solutions for 10 out of the 11 instances. For all small instances, both NLGDHH-SM and NLGDHH-DM are able to find the optimal solutions. For all medium instances, the NLGDHH variants achieve a significant improvement over the other hyper-heuristics. The NLGDHH approaches are also quite competitive in the large instance when compared to the results obtained by SAHH.

Table 2: Comparison of the proposed great deluge based hyper-heuristic and other hyper-heuristics from the literature.

	NLGDHH-SD	NLGDHH-DM	CFHH	CBHH	SAHH	(DCFHH)
S1	0	0	1	6	0	1
S2	0	0	2	7	0	3
S3	0	0	0	3	1	1
S4	0	0	1	3	1	1
S5	0	0	0	4	0	0
M1	**71**	88	146	372	102	182
M2	**82**	88	173	419	114	164
M3	137	**112**	267	359	125	250
M4	**55**	84	169	348	106	168
M5	106	**103**	303	171	106	222
L1	777	915	1166	1068	**653**	-

Finally, we compare the results obtained by the proposed algorithm to the best results reported in the literature for the subject problem. The first two columns in Table 3 show the best results obtained by the algorithm proposed here while the third column shows the best known results and the corresponding approaches. It should be noted that although a timetable with zero penalty exists for each problem instance (the data sets were generated starting from such a timetable [22]), to the best of our knowledge no heuristic method has found the ideal timetable for the medium and large instances. Hence, these data sets are still very challenging for heuristic search methods. For all small instances, both approaches NLGDHH-SM and NLGDHH-DM produced optimal solutions. For medium instances, NLGDHH-SM improved the best solutions of M1, M2, M3, and M4 while NLGDHH-DM improved the best solution of M1, M2, M3, and M4. For the large instance, neither NLGDHH-DM nor NLGDHH-DM improved the best solution reported but they are very competitive.

Table 3: Comparison of the proposed great deluge based hyper-heuristic to the best results reported in the literature for the Course Timetabling Problem of Socha et al. [22].

	NLGDHH-SM	NLGDHH-DM	Best Known
S1	0	0	0 (VNS-T)
S2	0	0	0 (VNS-T)
S3	**0**	0	0 (CFHH)
S4	0	0	0 (VNS-T)
S5	**0**	0	0 (MMAS)
M1	**71**	87	80 (EGD)
M2	**82**	88	105 (EGD)
M3	137	**112**	139 (EGD)
M4	**55**	84	88 (EGD)
M5	106	103	**88** (EGD)
L1	777	915	**529**(HEA)

NLGDHH-SM is the Non-Linear Great Deluge Hyper-heuristic with fixed memory length
NLGDHH-DM is the Non-Linear Great Deluge Hyper-heuristic with dynamic memory length
MMAS is the MAX-MIN Ant System in [22]
CFHH is the Choice Function Hyper-heuristic in [9]
VNS-T is the Hybrid of VNS with Tabu Search in [1]
HEA is the Hybrid Evolutionary Algorithm in [2]
EGD is the Extended Great Deluge in [18]

6 Conclusions

We have developed a heuristic approach that uses a learning mechanism and a non-linear great deluge acceptance criterion to manage the selection of low-level heuristics during the search process. The method focuses on trying to choose the most appropriate heuristic in each step of the search and hence it follows the *hyper-heuristic* concept. We appplied the proposed method to well-known instances of the university course timetabling problem proposed by Socha et al. [22]. The experimental results showed that the proposed non-linear great-deluge hyper-heuristic (NLGDHH) was able to find new best solutions for 4 out of the 11 problem instances compared to results reported in the literature. However, for the large instance, the algorithm produced only competitive results. We believe that for very large search spaces, the learning mechanism becomes less effective. Our future work contemplates the decomposition of large problems into smaller ones where the proposed algo-

rithm seems to be very effective. We also want to incorporate a larger number of low-level heuristics and perhaps some more specialised operators. Another issue that requires further investigation is the robustness of the learning mechanism with respect to the various algorithm parameters.

References

[1] S. Abdullah, E.K. Burke, B. McCollum. An investigation of variable neighbourhood search for university course timetabling. In: *Proceedings of the 2nd Multidisciplinary Conference on Scheduling: Theory and Applications*, pages 413-427, NY, USA, 2005.

[2] S. Abdullahm, E.K. Burke, B. Mccollum. Using a randomised iterative improvement algorithm with composite neigjhbourhood structures for university course timetabling. In: *Metaheuristics - progress in complex systems optimization*, Springer, pages 153-172, 2007.

[3] S. Abdullah, E. Burke, B. McCollum. A hybrid evolutionary approach to the university course timetabling problem. In: *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 1764-1768, 2007.

[4] H. Asmuni, E. Burke, J. Garibaldi. Fuzzy multiple heuristic ordering for course timetabling. In: *Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI 2005)*, pages 302-309, 2005.

[5] M. Ayob, G. Kendall. An investigation of an adaptive scheduling approach for multi-head placement machines. In: *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003)*, pages 363-380, Nottingham, UK, 2003.

[6] R. Bai, E.K. Burke, G. Kendall, B. McCollum. Memory length in hyper-heuristics: An empirical study. In: *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CISched2007)*, pages 173-178, Hawaii, USA, 2007.

[7] E. Burke, Y. Bykov, J.P. Newall, S. Petrovic. A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research (YUJOR)*, 13(2):139-151, 2003.

[8] E. Burke, K. Hart, G. Kendall, J. Newall, P. Ross, S. Schulenburg. Hyper-Heuristic: an emerging direction in modern search technology. In: Fed Glover, Gary A. Kochenberger (eds.) *Handbook of Meta-heuristics*, pages 457-474, Kluwer Academic Publishers, 2003.

[9] E. Burke, G. Kendall, E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9:451-470, 2003.

[10] E. Burke, B. McCollum, A. Meisels, S. Petrovic, Q. Rong. A graph based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176:177-192, 2007.

[11] E. Burke, S. Petrovic, R. Qu. Case based heuristic selection for timetabling problems. *Journal of Scheduling*, 9(2):115-132, 2006.

[12] E.K. Burke, G. Kendall, J.D. Landa-Silva, R. O'Brien, E. Soubeiga. An ant algorithm hyper-heuristic for the project presentation scheduling problem. In: *Proceedings of the 2005 IEEE*

Congress on Evolutionary Computation (CEC 2005), Volume 3, pages 2263-2270, Edinburgh, Scotland, 2005.

[13] T.B. Cooper, H. Kingston. The complexity of timetable construction problems. In: *Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT 1995)*, LNCS, 1153, Springer, pages 283-295, 1996.

[14] P. Cowling, G. Kendall, L. Han. An investigation of a hyper-heuristic genetic algorithm applied to a trainer scheduling problem. In: *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 1185-1190, Honolulu, Hawaii, 2002.

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

[16] D. Landa-Silva, J.H. Obit. Great deluge with nonlinear decay rate for solving course timetabling problems. In: *Proceedings of the 2008 IEEE Conference on Intelligent Systems (IS 2008)*, IEEE Press, pages 8.11-8.18, 2008.

[17] D. Landa-Silva, J.H. Obit. Evolutionary nonlinear great deluge for university course timetabling. To appear In: *Proceedings of the 2009 International Conference on Hybrid Artificial Intelligence Systems (HAIS 2009)*, 2009.

[18] P. McMullan. An extended implementation of the great deluge algorithm for course timetabling. In: *Proceedings of the 2007 International Conference in Computational Science (ICCS 2007)*, LNCS 4487, Springer-Verlag, pages 538-545, 2007.

[19] P. Rattadilok, A. Gaw, R. Kwan. Distributed choice function hyper-heuristics for timetabling and scheduling. In: *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2004)*, 2004.

[20] P. Ross, S. Schulenburg, J. Marin-Blazquez, H. Hart. Hyper-heuristic: Learning to combine simple heuristic in bin-packing problems. In: *Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 942-948, New York, USA, 2002.

[21] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. Gambardella, J. Knowles, M. Manfrin, L. Mastrolilli, B. Paechter, L. Paquete, T. Stuetzle. A comparion of the performance of different metaheuristics on the timetabling problem. In: *Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002)*, LNCS 2740, Springer, pages 330-352, 2003.

[22] K. Socha, J. Knowles, M. Samples. A max-min ant system for the university course timetabling problem. In: *Ant Algorithms: Proceedings of the Third International Workshop (ANTS 2002)*, LNCS 2463, pages 1-13, Springer, 2002.

[23] K. Socha, M. Sampels, M. Manfrin. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In: *Applications of Evolutionary Computing: Proceedings of the 2003 EvoWorkshops*, LNCS 2611, Springer, pages 334-345, 2003.

[24] E. Soubeiga. Development and application of hyper-heuristic to personnel scheduling. *PhD thesis*, School of Computer Science, University of Nottingham, UK, 2003.