In: B.L. Golden, S. Raghavan and E.A. Wasil (eds.). The Next Wave in Computing, Optimization, and Decision Technologies. Conference Volume of the 9th informs Computing Society Conference. 79-91. published by Springer. Jan 2005.

HYBRID GRAPH HEURISTICS WITHIN A HYPER-HEURISTIC APPROACH TO EXAM TIMETABLING PROBLEMS

Edmund Burke[1], Moshe Dror[2], Sanja Petrovic[1] and Rong Qu[1]

1Automated Scheduling Optimization & Planning Group, School of CSiT, University of Nottingham, Nottingham, NG8 1BB, UK;

2Department of Management Information Systems Eller College of Business and Public Administration, University of Arizona, Arizona 85721

Abstract: This paper is concerned with the hybridization of two graph coloring heuristics (Saturation Degree and Largest Degree), and their application within a hyper-heuristic for exam timetabling problems. Hyper-heuristics can be seen as algorithms which intelligently select appropriate algorithms/heuristics for solving a problem. We developed a Tabu Search based hyper-heuristic to search for heuristic lists (of graph heuristics) for solving problems and investigated the heuristic lists found by employing knowledge discovery techniques. Two hybrid approaches (involving Saturation Degree and Largest Degree) including one which employs Case Based Reasoning are presented and discussed. Both the Tabu Search based hyper-heuristic and the hybrid approaches are tested on random and real-world exam timetabling problems. Experimental results are comparable with the best state-of-the-art approaches (as measured against established benchmark problems). The results also demonstrate an increased level of generality in our approach.

Key words: case based reasoning, exam timetabling problems, graph heuristics, hyper-heuristics, knowledge discovery, tabu search.

1. INTRODUCTION

1.1 Timetabling Problems

Timetabling problems have been attracting the attention of the scientific research community across Artificial Intelligence and Operational Research for more than 40 years[1-10].

A general timetabling problem includes assigning a set of events (exams, courses, sports matches, meetings, etc) into a limited number of timeslots (time periods), while satisfying a set of constraints. These constraints are usually grouped into two types, which are described below:

- Hard constraints which cannot be violated under any circumstances. For example, a person cannot be assigned to two different events at the same time. Solutions which do not violate any of the hard constraints are called feasible solutions.
- Soft constraints are desirable but not essential. In most real world situations no solutions can be found which satisfy all of stipulated soft constraints.

In the early days of timetabling research, graph heuristics[11, 12, 13] were widely studied. They represent simple techniques but tend to be impractical for complex problems (at least when implemented on their own). Integer linear programming[14] is an exact solution method which tends to be computationally very expensive when solving large timetabling problems. Over the years, constraint programming methods have also been investigated at some length[15, 16, 17]. Meta-heuristics[18] have been shown to be very successful on a variety of timetabling problems. Examples include Tabu Search[19, 20], Simulated Annealing[21, 22] and Evolutionary Algorithms[23, 24]. Other new methods studied for timetabling problems include Case Based Reasoning[25] (for educational timetabling[26, 27, 28] and nurse scheduling[29]).

The work presented in this paper investigates the benefits of hybridizing two well-studied graph heuristics[11, 12, 13] by using a hyper-heuristic on timetabling. The term hyper-heuristic can be taken as a ‘heuristic that searches for heuristics’[30]. A hyper-heuristic searches a space of heuristics rather than problem solutions. Our hyper-heuristic approach searches from a set of lower level heuristics according to different problem solving situations that might occur, and then applies those heuristics to the particular problem in hand. Different higher level heuristics/techniques employed within a hyper-heuristic framework include Case Based Reasoning[25], choice functions[31] and meta-heuristic methods[32, 33].

Many of the current state of the art approaches in exam timetabling employ specially tailored heuristic/meta-heuristics methods[20, 22, 34-39]. This kind of approach is also typical for other scheduling problems. The purpose of this paper is to describe our initial attempts at developing an approach which is fundamentally more general than the above methods. The goal is not necessarily to ‘beat’ those methods but to obtain comparable results by only employing general methods that can ‘pick’ appropriate heuristics and which would be applicable to a broader range of problems.

1.2 Case Based Reasoning (CBR)

CBR[25] is a knowledge-based technology that solves the problems in hand (target cases) by using knowledge obtained by solving previous similar problems. In a CBR system, a case base stores a set of previously solved problems with their good solutions or problem solving strategies (called source cases). A similarity measure, usually defined as a formula, is used to assess the similarities between the target case and source cases. The good solutions or problem solving strategies of the most similar source case are reused to tackle the target case.

The basic idea of CBR is to avoid solving new problems from scratch when the knowledge of solving similar problems is available. Our previous work using CBR on course and exam timetabling has presented successful results, either by reusing good partial solutions of problems whose constraints are structurally similar with current problems[26, 27], or by reusing good heuristics in similar problem solving situations[28]. This has provided the foundation for the research presented in this paper.

The next section presents our Tabu Search based hyper-heuristic (TSHH) on two graph heuristics. The results obtained by TSHH are utilized to propose two hybrid graph heuristics (including one which employs CBR). This is followed by experiments on both random and real-world problems. We conclude by briefly discussing the impact of the work and potential future research directions.

2. A TABU SEARCH BASED HYPER-HEURISTIC

In our previous work CBR was studied as the higher level searching technique to suggest different constructive heuristics during the exam timetabling problem solving process[28]. At each step of the solution construction, we select the heuristic (stored in the case base) that made the least penalty schedule in a previous *similar* situation. Employing this knowledge can help in finding good heuristics in new similar situations and in generating better quality solutions compared with those generated using single heuristics.

2.1 Tabu Search in a Hyper-heuristic Framework

In this paper we will employ Tabu Search as the higher level searching algorithm within a hyper-heuristic methodology that searches for the best combinations of heuristics (heuristic lists) for constructing the solutions for exam timetabling problems. This means that the heuristic list found not only represents good heuristics at each particular step (as before[28] where the least penalty schedules are made), but also it represents problem solving context (heuristics used and costs occurred before the current step, etc). A Tabu Search methodology within a hyper-heuristic framework has already been demonstrated as a successful, general methodology across very different problems (for course timetabling and nurse rostering[32]).

The search space of the TSHH consists of all of the possible permutations of the Saturation Degree (SD) and Largest Degree (LD), as shown below. Starting from an initial list of heuristics, a move within the TSHH is to change one of the heuristics in the heuristic list. The heuristics in the list are then employed, one by one, to construct the solution for the problem. The objective of TSHH is to find the heuristic list that generates the best quality solutions. TSHH stops after a certain number of iterations (5 times the number of the exams in the problem being considered).

SD and LD are two widely studied graph heuristics for applications to timetabling problems[11, 12, 13]. They are sequential methods that order the exams to be scheduled according to the difficulty of scheduling them. They then assign them one by one into feasible timeslots without violating any hard constraint and with the lowest penalty (i.e. the lowest total number of violations of soft constraints). They can be described as follows:

- Largest Degree (LD): Exams are ordered decreasingly by the number of conflicts they have with other exams. This heuristic aims to schedule the most conflicting exams first.
- Saturation Degree (SD): Exams that are not yet scheduled are ordered increasingly by the number of feasible timeslots available at that time. The priorities of the exams thus change dynamically according to the situations encountered at each step of the solution construction.

The heuristic lists selected to construct the solutions may not generate feasible solutions once they are performed, because the moves in the TSHH concern the changes of heuristics in the heuristic list, not the actual assignment of each exam. The search space of the TSHH is thus very large, containing a large number of non-valid heuristic lists. We add three mechanisms into the TSHH to reduce the size of the search space. They are described below:

- The parts of heuristic lists that generate infeasible assignment are stored in the searching process of the TSHH. At each move, the heuristic list selected will be checked before it is applied to see if it contains any stored infeasible heuristic lists. For example, if a heuristic list with the part 'SD LD ..' is stored because LD make an infeasible schedule at that step, all the heuristic lists selected later such as 'SD LD SD ..' or 'SD LD LD ..' will be ignored in the searching process. This mechanism significantly cuts the size of the search space by ignoring non-valid sections.
- At each step of the solution construction, we schedule a number of exams at once (we choose 3 here) by the given heuristic in the heuristic list. This is motivated by the observation[28] that the heuristics in the best heuristic

lists tend to switch to others after a number of events have been scheduled. This mechanism also significantly reduces the size of the search space of TSHH.
- The initial heuristic list of TSHH is set as a list of SD only. We observe that in most cases SD is superior to LD, thus it is expected that the appearance of SD will be higher than that of LD in the best heuristic lists.

2.2 Experiments on Random Data Sets

The data sets we use are generated by using the same process as that of Carter et al[38] which simulates real-world exam timetabling problems. Each time a student is created and *r* exams are assigned (following a discrete uniform distribution in [2, 6]). This process is repeated until the defined density of conflict matrix is met, which is calculated as the number of conflicts among exams to the total number of exams. This generates 6 types of problems of 200 to 400 exams with density of 0.05, 0.15 and 0.25, namely ‘200-5’, ‘200-15’, ‘200-25’, ‘400-5’, ‘400-15’ and ‘400-25’.

The hard constraints consider the ‘conflict’ between exams with students in common. The soft constraints under consideration concern spreading out the students’ exams evenly. The cost function that evaluates the solutions is the same as that of Carter et al[38]. The objective is to minimize the cost per student. For each problem type, 20 distinct problems are tested on using the SD and LD heuristics alone, and the TSHH. The average costs and time spent by these approaches are presented in Table 1.

Table 1. SD, LD alone, two hybrid approaches and TSHH on random problems

problem	SD		LD		SD+23%LD		SD+CBR		TSHH		
	cost	time	cost	time	cost	time	cost	time	cost	time	density
200-5	10.07	1.25	10.21	0.02	9.97	0.97	10.01	2.39	**9.63**	195	0.24
200-15	9.47	4.48	9.42(18)	0.05	9.46	3.35	9.40	7.32	**9.21**	604	0.23
200-25	5.92	12.49	6.13(14)	0.08	5.90	10.00	5.88	28.31	**5.81**	3881	0.23
400-5	9.33	12.41	9.32(16)	0.06	9.30	9.66	9.28	22.62	**9.05**	5011	0.26
400-15	4.72	76.7	4.70(16)	0.31	4.67	57.08	4.68	73.26	**4.52**	20074	0.24
400-25	3.70	185.84	n/f(20)	0.59	3.70	158.16	3.70	204.27	**3.68**	73462	0.26

The best results in the table are presented in bold. For all the problems, the TSHH works much better than using SD and LD alone. The values in ‘()’ represents the number of times infeasible solutions are obtained by the corresponding approach. Compared with SD or LD alone, TSHH takes a much longer time, especially for larger problems with higher densities. Note that in real world situations, exam timetables are usually generated weeks (or months) in advance and thus the time does not usually have much impact on the usefulness of the methodology.

The column under the title of ‘density’ in Table 1 presents the average densities of the LD in the best heuristic lists obtained by TSHH. We can see that for all of the problem types, the densities of LD in the best heuristic lists are almost the same, ranging from 0.23-0.26. This motivated us to build one of the hybrid approaches, whose details are presented in the next section.

3. HYBRID GRAPH BASED APPROACHES

By investigating the TSHH, we propose two hybrid approaches. They are SD injected with LD by 23%, and by a CBR system built using knowledge discovery techniques that extract the knowledge of TSHH. They are presented in the following two sub-sections.

3.1 SD Injected with 23% LD

In the first hybrid approach, LD is randomly injected into the heuristic list (of SD) to form 23% of it. This hybridization is proposed by using the density of LD that appears in the best heuristic lists. Our aim is to investigate whether such a heuristic list is good for all of the problem instances.

The results of this approach on all types of problems are presented in the column entitled SD+23%LD in Table 1. They are compared with the results obtained by using SD and LD heuristics alone. Please note that it presents the cost per student thus small differences from different approaches may indicate large differences in costs.

Compared with the results obtained by using SD and LD alone, the hybrid approaches perform on all of the problems, except for ‘400-25’, where the same results (3.70) are obtained by SD+23%LD and SD+CBR. For problem type ‘400-25’, LD failed to obtain feasible solutions for any pf the problem instances (indicated by ‘n/f’ in the table). The values in Table 1 present the average penalties for only the feasible solutions obtained.

Among the approaches, LD takes the least amount of time as the ordering of exams will not change during the problem solving process. The time spent by SD+23%LD is less than that of SD, which orders the exams that are not yet scheduled at each step of the problem solving process. This hybrid approach is superior to SD on both the solution quality and the problem solving time.

3.2 SD Injected with LD by CBR

We present another hybrid approach using a CBR system developed by investigating the heuristic lists obtained from the TSHH described above.

The idea is to inject the LD by the suggestions from the CBR system which stores the appropriate heuristics in different problem solving situations. At each step of the scheduling process, the current problem solving situation is input into the CBR system as the target case, and the most similar source case is retrieved. The heuristic of this retrieved source case is employed to select and schedule the exams in the next step of solution construction.

In the CBR system, we use a list of feature-value pairs to represent the cases. The similarity measure employs a nearest neighborhood approach to sum up the differences of values for each pair of features in the target and source cases. The most similar source cases will be retrieved and the corresponding heuristics will be suggested for use in the next step of scheduling.

Knowledge discovery is carried out by using the best heuristic lists obtained from the TSHH. The objective is to discover the most relevant features to be used in the list of features to represent cases so that the correct heuristic can be selected by CBR. We collected a set of initial training features that describe the problem solving situations. They can be grouped into two types, which are presented below.

1. Simple features: this can also be grouped into two types:
   - Features that describe the problems. These include: the no. of exams, students, timeslots, the total no. of conflicts among all the exams, the density of the conflict matrix, the no. of the conflicts for the most conflicting exams and the no. of the most conflicting exams.
   - Features with values that are changeable during problem solving. These include: the no. of exams that have been scheduled in a particular timeslot, the heuristic employed before the current step, the increased penalty occurred by the last step schedule, and the cost of the partial solution concerning the violations of only the soft constraints.
2. A Combination of the simple features: the ratios between each pair of the simple features.

At each particular scheduling step during the TSHH, the problem solving situations (values of all of the training features presented above) are recorded for each problem solved. These situations along with the best heuristics at that step in the process form the *cases* to be used in the knowledge discovery process. All of the cases obtained are randomly divided into two groups: one group is stored as source cases in the case base and the other will be the training cases used just in the knowledge discovery process for training purposes.

A Tabu Search (not the same one used in TSHH but just for knowledge discovery) is used to discover the best feature list for the case representation. All of the possible feature lists form the search space of Tabu Search. An initial feature list is randomly selected and a move is a change of a feature and its weight. All of the training cases (whose heuristic is already obtained beforehand by TSHH) are input one by one into the system. If the heuristic of the most similar source case retrieved (by similarity measure upon the set of features for the training) is the same as that of the training case obtained beforehand, it will be seen as a successful retrieval. The total number of the successful retrievals indicates the system performance. The objective of the Tabu Search is to find the feature list upon which the highest system performance is obtained for all of the training cases. The most relevant features found by the first stage of knowledge discovery are: 'the increase in penalty in the previous step of scheduling' with weight 100, and 'the number of exams already scheduled at that step' with weight 1.

The second stage of knowledge discovery aims to refine the case base. The best feature list is obtained from the first stage each time a source case is removed from the case base. If the system performance is decreased, the removed source case will be added back into the case base as it contains useful information for the heuristic selection in that particular problem solving situation; otherwise the source case will be removed permanently as it contains either redundant or wrong information that is harmful for the heuristic selection.

The column entitled SD+CBR in Table 1 presents the results obtained by the hybrid SD with CBR on the same problems tested using other approaches. We can see that it outperforms SD and LD alone on all problems except on problem '400-25', where the same result is obtained by SD alone. Compared with SD+23%LD, it obtained slightly better solutions but with longer time occurred on searching the case base. On all of the problem types except '400-25', both of the two hybrid approaches outperform the SD and LD alone. Among all of the approaches tested TSHH works the best. Note, though, that the two hybrid approaches are much quicker than the TSHH although (as we already mentioned) time is not usually a critical issue in exam timetabling.

These results show that by embedding knowledge of employing different heuristics during problem solving, the hybrid approaches work better than those of the single heuristics. The hybrid approaches have the ability to choose appropriate heuristics in different situations thus have significant potential for being more generally applicable than the current state of the art.

4. EXPERIMENTS ON REAL-WORLD PROBLEMS

We carried out another set of experiments on 4 real-world benchmark exam timetabling problems presented by Carter et al[38]. These problems cover a range of characteristics (i.e. on number of exams and conflict matrix density). Table 2 presents the best results for these 4 problems by all of the approaches presented above, except LD which failed to generate feasible solutions for all of the problems.

Table 2. SD, two hybrid approaches, TSHH and state-of-the-art on real-world problems

problem	ute92	uta93	sta83	ear83
SD	37.63	n/f	191.93	n/f
SD+23% LD	38.37	5.09	n/f	n/f
SD+CBR	37.53	5.06	173.82	47.42
TSHH	35.40 (0.37)	4.52 (0.12)	158.2 (0.07)	45.60 (0.48)
Asmuni et al[34]	27.78	3.57	160.42	37.02
Burke et al[35]	25.7	3.4	159.1	35.4
Burke & Newall[36]	25.83	**3.20**	168.73	37.05
Caramia et al[37]	**24.4**	3.5	158.2	**29.3**
Carter et al[38]	25.8	3.5	161.5	36.4
Casey & Thompson[22]	25.4	n/f	**134.9**	34.8
Di Gaspero & Schaerf[20]	31.3	4.5	166.8	46.7
Merlot et al[39]	25.1	3.5	157.3	35.1

We can observe that for real-world problems, the results obtained from the hybrid approaches show different characteristics compared with those on random data sets. The reason may be that the knowledge discovered from the random data sets may not cover enough problem solving situations of real world problems with different characteristics.

SD+CBR shows promising results and is reasonably reliable over both random and real-world problems, as the injection of LD is made by using knowledge concerning different problem solving situations and thus can help to solve more types of problems. However, to be able to solve more types of problems, the CBR system needs to be trained to store more knowledge of problem solving over a wider range of problems (including both the random and real-world problems) with a variety of problem features.

One observation is that the densities of LD (presented in '( )' in Table 2) in the best heuristic lists found are different for different problems, and none of them has a value that is within 0.23-0.26. Thus SD+23%LD will not be the appropriate approach for solving the real-world problems presented here.

The TSHH shown in Table 2 outperforms all of the other approaches described in this paper on real problems. Table 2 also presents the current best published results on these benchmarks by 8 other approaches reported in the literature[20, 22, 34-39]. The best results are presented in bold. Note that TSHH gets into the same region as these sophisticated approaches which are 'tailor made' for exam timetabling.

Also note that except Carter et al[38] and Asmuni et al[34] all the other approaches are improving approaches that are based on initial solutions obtained beforehand. Our approaches are simple constructive methods that are independent of the initialization process and obtained comparable results from the reported approaches on the benchmark problems, for which most simple constructive methods failed to obtain feasible solutions. Moreover, this approach is far simpler and more generic than those approaches. It selects appropriate simple heuristics during the search process. These simple heuristics can be employed in many other timetabling and scheduling problems.

5. CONCLUSIONS AND FUTURE WORK

The overall goal of this paper is to investigate the development of approaches/systems which can operate at a higher level of generality than current approaches/systems. The TSHH uses only very simple heuristics (SD and LD) and clearly outperforms the heuristics on their own and the other two hybrid approaches we have described in this paper.

The heuristic selection methods described here represent a framework which can easily be applied to other timetabling and scheduling problems. They take simple heuristics and we demonstrated that those heuristics can be better employed by intelligent selection at appropriate points in the solution construction process. These methods are comparable to the bespoke methods even though the overall goal of this approach is to be more generally applicable rather than to produce the ‘best’ results on benchmark problems. Note also that the methods employed here use only the generally applicable graph coloring heuristics that can be easily employed for many timetabling and scheduling problems. The work can be extended in two ways: 1) extra graph heuristics can be added to the framework to give more choices; and 2) the same framework can be extended to other scheduling problems as little domain specific knowledge is employed. The searching time of TSHH needs further improvement upon larger problems with a higher number of constraints. This may be investigated and compared on other meta-heuristics such as Simulated Annealing and evolutionary approaches, etc.

For the randomly generated data sets, the two hybrid approaches produce better results than those obtained by using the graph heuristics alone, meaning that the knowledge extracted from the TSHH on random data helps in solving problems, avoiding the time and effort required for development of problem specific algorithms for timetabling problems. SD+CBR is able to provide appropriate heuristics within particular problem solving situations using the knowledge discovered beforehand, enabling it to underpin a more general approach for a wider range of problem types. More dedicated knowledge discovery techniques and machine learning methods can be investigated to discover more accurate knowledge within the critical area of learning in hyper-heuristic methodology for solving general timetabling problems.

REFERENCES

1. V. Bardadym. Computer-Aided School and University Timetabling: The New Wave. In: [2], pp. 22-45. (1995).
2. E. Burke and P. Ross eds. Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, LNCS 1153, Springer-Verlag, 1996).
3. E. Burke and M. Carter eds. Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling, LNCS 1408. (Springer-Verlag, 1998).
4. E. Burke and W. Erben, W. eds. Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling, LNCS 2079, (Springer-Verlag, 2001).
5. E. Burke, K. Jackson, J Kingston and R. Weare. Automated Timetabling: The State of the Art. The Computer Journal, **40**(9): 565-571, (1997).
6. E. Burke and P. Causmaecker, eds. Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, LNCS 2740. (Springer-Verlag, 2003).
7. M. Carter and G. Laporte. Recent Developments in Practical Examination Timetabling. In: [2], pp. 3-21. (1995).
8. M. Carter and G. Laporte. Recent Developments in Practical Course Timetabling. In: [3], pp. 3-19.
9. A. Schaerf. A Survey of Automated Timetabling. Artificial Intelligence Review. **13**(2): 87-127. (1999).
10. E. Burke and S. Petrovic, Recent Research Directions in Automated Timetabling. EJOR, **140**(2): 266-280. (2002).
11. D. Brelaz, New Methods to Color the Vertices of a Graph. Communications of the ACM, **22**(4): 251-256. (1979).
12. de Werra, Graphs, Hypergraphs and Timetabling. Methods of Operations Research. 49: 201-213. (1985).
13. E. Burke, J. Kingston and D. de Werra, Applications to Timetabling, Handbook of Graph Theory, (J. Gross and J. Yellen eds.), pp. 445-474, (Chapman Hall/CRC Press, 2003).
14. M. Carter, A Lagrangian Relaxation Approach to the Classroom Assignment Problem. IFOR **27**(2): 230-246. (1986).
15. B. Deris, S. Omatu, H. Ohta and D. Samat. University Timetabling by Constraint-based Reasoning: A Case Study. JORS. 48(12): 1178-1190. (1997).
16. K. Nonobe T. and Ibaraki. A Tabu Search Approach to the Constraint Satisfaction Problem as a General Problem Solver. EJOR. 106: 599-623. (1998).
17. D. Banks, P. Beel and A. Meisles. A Heuristic Incremental Modelling Approach to Course Timetabling. Proceedings of the Canadian Conference on Artificial Intelligence, pp. 16–29. (1998).
18. F. Glover, and G. Kochenberger, Handbook of Metaheuristics, Kluwer. 2003.
19. D. Costa. A Tabu Search for Computing an Operational Timetable. EJOR. 76: 98-110. (1994).
20. L. Di Gaspero and A. Schaerf, Tabu Search Techniques for Examination Timetabling, In: [4], pp. 104-117. (2000).
21. K. Dowsland, Off the Peg or Made to Measure", In: [3], 37-52. (1998).

22. S. Casey, J. Thompson, A Hybrid Algorithm for the Examination Timetabling Problem. In: [6], pp. 205-230. (2002).
23. E. Burke, J. Newall and R. Weare, R. Initialization Strategies and Diversity in Evolutionary Timetabling. Evolutionary Computation, **6**(1): 81-103. (1998).
24. E. Burke and J. Newall. A Multi-Stage Evolutionary Algorithm for the Timetabling Problem. The IEEE Transactions on Evolutionary Computation. **3**(1): 63-74. (1999).
25. D. Leake ed. Case-based Reasoning: Experiences, Lessons and Future Directions. (AAAI Press, Menlo Park, CA. 1996).
26. E. Burke, B., MacCarthy, S. Petrovic and R. Qu, Structured Cases in Case-Based Reasoning - Re-using and Adapting Cases for Time-tabling Problems. Knowledge-Based Systems, **13**(2-3): 159-165. (2000).
27. E. Burke, B. MacCarthy, S. Petrovic and R. Qu, Multiple-Retrieval Case-Based Reasoning for Course Timetabling Problems. Technical Report NOTTCS-TR-2004-3, School of CSiT, University of Nottingham, U.K. (accepted by JORS, 2004).
28. E. Burke, S. Petrovic and R. Qu, Case Based Heuristic Selection for Examination Timetabling. Technical Report NOTTCS-TR-2004-2, School of CSiT, University of Nottingham, U.K. (To appear in Journal of Scheduling, 2005).
29. S. Petrovic, G. Beddoe and G. Vandem Berghe, Storing and Adapting Repair Experiences in Employee Rostering. In: [6], pp. 148-165. (2003).
30. E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross and S. Schulenburg, Hyper-heuristics: an Emerging Direction in Modern Search Technology. In: F. Glover and G. Kochenberger eds., Handbook of Meta-Heuristics, (Kluwer, 2003), pp. 457-474.
31. G. Kendall, P. Cowling and E. Soubeiga, Choice Function and Random HyperHeuristics. Proceedings of SEAL'02, pp. 667-671. (2002).
32. E. Burke, G. Kendall. G and E. Soubeiga, A Tabu Search Hyperheuristic for Timetabling and Rostering. Journal of Heuristics. **9**(6). (2003).
33. L. Han and G. Kendall. Investigation of a Tabu Assisted Hyper-Heuristic Genetic Algorithm. Congress on Evolutionary Computation, Canberra, Australia, 2230-2237. (2003).
34. H. Asmuni, E. Burke, and J. Garibaldi. Fuzzy Multiple Ordering Criteria For Examination Timetabling. To appear in the 5th International Conference on the Practice and Theory of Automated Timetabling. Pittsburgh, USA. Aug 2004.
35. E. Burke, Y. Bykov, J. Newall, J. and S. Petrovic. A Time-Predefined Local Search Approach to Exam Timetabling Problems. IIE Transactions on Operations Engineering, 36(6), 509-528, (2004).
36. E. Burke and J. Newall, Enhancing Timetable Solutions with Local Search Methods. In: [6], pp. 195-206. (2002).
37. M. Caramia P. Dell'Olmo and G. Italiano, New Algorithms for Examination Timetabling. In: S. Naher and D. Wagner eds. LNCS 1982, pp. 230-241. (2001).
38. M. Carter G. Laporte and S. Lee, Examination Timetabling: Algorithmic Strategies and Applications, JORS, **47**: 373-383. (1996).
39. L. Merlot, N. Boland, B. Hughes and P. Stuckey. A Hybrid Algorithm for the Examination Timetabling Problem. In: E. Burke and P. De Causmaecker (eds.) Proceedings of the 4th International Conference on Practice and Theory of Timetabling, pp. 348-371. (2002).