

Fuzzy Multiple Heuristic Ordering for Course Timetabling

Hishammuddin Asmuni, Edmund K. Burke and Jonathan M. Garibaldi

Automated Scheduling, Optimisation and Planning (ASAP) Research Group,

School of Computer Science and Information Technology,

University of Nottingham, Nottingham, NG8 1BB, United Kingdom.

{hba, ekb,jmg}@cs.nott.ac.uk

Abstract

In this paper, we address the issue of ordering events by simultaneously considering three separate heuristics using fuzzy methods. Combinations of the following three heuristic orderings are employed: largest degree, saturation degree and largest enrollment. The fuzzy weight of an event is used to represent how difficult it is to schedule. The decreasingly ordered events are sequentially chosen to be assigned to the last slot with least penalty cost value while the feasibility of the timetable is maintained throughout the process. Uncheduling and rescheduling events is performed until all events are scheduled. The proposed algorithm has been tested on 11 benchmark data sets of course timetabling problems and the results show that this approach can produce good quality solutions with low requirements for rescheduling. Moreover, there is significant potential to extend the approach further by including a larger range of criteria.

1 Introduction

The problem of timetabling courses is of much interest and concern to academic institutions. The basic problem is to allocate a time slot and a room for all lectures within a limited number of permitted time slots and rooms in order to find a conflict free timetable. This assignment process is subject to ‘hard’ constraints which must be satisfied in order to get a feasible timetable, such as that no student is required to attend two events at the same time. In addition, it is also important to build a good quality lecture timetable that considers not only the administration requirements, but also takes into account lecturers’ and students’ preferences. These requirements are generally considered as ‘soft’ con-

straints which are desirable (but not essential) to satisfy.

As this task is time consuming and tedious to do manually, many attempts have been made over the last few decades to generate timetables automatically. With a large number of events needing to be assigned to resources (time slots and rooms) and a list of constraints needing to be satisfied, the search space for this problem is very large.

Various approaches have been investigated and developed for solving course timetabling problems. Such approaches include constraint programming [20], graph colouring [13, 14, 16], case based reasoning [12], hyper-heuristics [11, 13] and various metaheuristic approaches including multi-neighbourhood local search [19], tabu search [2], ant colony optimisation [27], memetic algorithms [22], variable neighbourhood search [1], and the great deluge algorithm [8]. Overviews of the recent state of the art of course timetabling can be found in a variety of papers (refer to [10, 18, 23, 26]).

Since being introduced by Zadeh in 1965 [30], fuzzy methodologies have been successfully applied in a wide range of real world applications. In scheduling and timetabling applications, fuzzy methodologies have been implemented in other timetabling problems such as aircrew rostering [28], driver scheduling [21] and nurse rostering [5].

In the context of academic timetabling, fuzzy methods have been used for measuring the problem similarity in case based reasoning by Yang and Petrovic [29]. In [24], Petrovic *et al.* employed fuzzy methodologies to measure the satisfaction of various soft constraints. The authors describe how they modeled two soft constraints, namely *constraint on large exam* and *constraint on proximity of exams*, in the form of fuzzy linguistic terms and defined the related rule set. A memetic algorithm was then implemented to evaluate the approach on 12 benchmark examination timetabling problem instances.

Approaches which order the events prior to assignment to a time slot have been discussed by several authors including Boizumault *et al.* [6], Brailsford *et al.* [7], Burke *et al.* [9], Burke and Newall [14], Burke and Petrovic [15] and Carter *et al.* [17]. The main principle is that events are ordered by certain heuristics before each event is sequentially chosen to be assigned to a time slot. This ordering represents how difficult it is to schedule the events. The idea is that by assigning the most difficult events first it is likely that we can avoid generating unfeasible solutions. Here, a feasible solution means that all events are assigned to time slots without violating any of the specified hard constraints. Many studies have been made into the best way to calculate the ‘difficulty to schedule an event’.

In [17], Carter *et al.* used four different types of heuristic ordering to rank the exams in decreasing order to estimate how difficult it is to schedule each of the exams. They considered largest degree, saturation degree, largest weighted degree and largest enrollment. These heuristics were used individually each time they wanted to order the exams. Burke and Newall [14] applied an adaptive heuristic technique in which they start ordering by a particular heuristic and then alter that heuristic ordering to take into account the penalty that exams are imposing upon the timetable. Recently, Burke *et al.* [13] proposed a new hyper-heuristic approach for solving timetabling problems. Instead of using a single heuristic to find solutions for course and examination timetabling problems, a sequence of heuristics is applied. The authors used tabu search and deepest descent local search in order to find the best list of heuristics to guide the sequential constructive algorithm in finding the ‘best solution’ for each problem instance.

In our previous papers [3, 4], a fuzzy methodology was used to rank exams based on an assessment of how difficult they were to schedule, taking into account multiple heuristics. We showed that when two heuristic orderings were simultaneously considered to rank the events, better results were obtained compared to single heuristic [4]. We also implemented three heuristic orderings simultaneously, and a comparison was made between fuzzy ordering with single and linear combination of heuristic orderings [3]. All this previous work has been concerned with the problem of creating timetables for *examinations*.

In this paper we apply the same underlying methodologies (i.e. fuzzy multiple heuristic orderings) to a novel context; that of *course timetabling*. We apply the same algorithms to

create fuzzy inferencing systems as in [3, 4], with a different penalty function to capture the different domain characteristics.

In the following section, the proposed algorithm and the experiments carried out are explained in detail. Section 3 describes the problem definition and the results obtained. These results are then discussed in Sections 4.

2 Methods

2.1 The Basic Sequential Construction Algorithm

There is a well known analogy between a basic version of the timetabling problem (no soft constraints) and the graph colouring problem (see [9]). Indeed, some of the best known timetabling heuristics are based upon graph colouring heuristics and these can be employed within a basic and simple timetabling algorithm (see Figure 1). We consider five different versions of the basic algorithm, which employ five different heuristic orderings respectively and require the following steps to assign all events to a time slot.

First, the events are ordered (most difficult first) by applying one of the heuristic orderings. The following heuristics are employed as ordering criteria:

Largest Degree (LD) First. The degree of an event is simply a count of the number of other events which conflict in the sense that students are enrolled in both events. This heuristic orders events in terms of those with the highest degree first.

Largest Enrollment (LE) First. The number of students enrolled for each event is used to order the events (the highest number of students first).

Least Saturation Degree (SD) First. The number of time slots available is used to order the events. The basic motivation is that events with less time slots available are more likely to be difficult to be scheduled. The fewer time slots that are available, the higher up the ordering is the event.

Largest Coloured Degree (LCD) First
This heuristic is based on LD. For this heuristic, only events which already assigned to the schedule are considered as the events which will cause conflict.

```

Sort unscheduled events using selected heuristic ordering;
k := number of events to be inserted before recalculate the events ordering
While there exist unscheduled events
  For e := 1 to k
    Select an event with the highest difficulty to be scheduled, event';
    If found clash free time slot
      Insert event' into the last time slot with least penalty;
      Remove event' from unscheduled event list;
    Else
      Find time slots where event' can be inserted with minimum number of
      scheduled events need to be removed from the time slot;
      If found more than one time slot with the same number of scheduled
      events need to be removed
        Select a time slot randomly from the candidate list of time slots, ts;
      End if
      c := number of events in time slot ts that conflict with event';
      For m := 1 to c
        Select event[m];
        If found another time slot with minimum cost to move event[m]
          Move event[m] to the time slot;
        else
          Bump back event[m] to unscheduled event list;
        End if
      End for
      Insert event' to time slot ts;
      Remove event' from unscheduled event list;
    End If
  End For
Sort unscheduled events using selected heuristic ordering;
End while

```

Figure 1: Pseudo code for general framework of sequential construction algorithm

Weighted Largest Degree (WLD) First

This heuristic also based on LD. Beside the number of events in conflict, the total number of student involved in the conflict are taken into account as well.

Then, events are selected sequentially from the ordering and assigned a valid time slot that will cause the minimum penalty cost for that event. In the selection of clash free time slot with least penalty cost, if several time slots are available, then the last available time slot in the list will be selected. If no clash free time slot is available, some of the events that have already been scheduled earlier need to be reshuffled by moving the scheduled events that conflicted with the new event to another valid time slot in order to create a clash free time slot for the current considered event.

If this process fails to find a clash free time slot for the current considered event, after all attempts to reshuffle the scheduled events, a time slot is selected randomly from the list of time slots with the same minimum number of scheduled events that need to be 'bumped back'. All the conflicting scheduled events are removed from the selected time slot and the new event is inserted. All the removed events are then bumped back to the unscheduled event list. The whole process continues until all the events are scheduled. In the rest of this paper, this process is referred to as the '*rescheduling procedures*'.

2.2 Description of Experiments

A number of experiments were carried out in order to compare the single heuristic ordering

and simultaneous multiple heuristic orderings. In each experiment this ordering is simply inserted into the basic general algorithm presented in Figure 1.

2.2.1 Single Heuristic Ordering

In order to provide a comparative test, the algorithm was initially run without implementing fuzzy ordering. That is, in this approach, the events in the problem instances were ordered based on a single heuristic ordering. All the events were then selected to be scheduled based on this ordering.

2.2.2 Fuzzy Multiple Heuristic Orderings

In many decision making environments, it is often the case that several factors are simultaneously taken into account. Often, it is not known which factor(s) need to be emphasised more in order to generate a better decision. Somehow a trade off between the various (potentially conflicting) factors must be made. The general framework of fuzzy reasoning facilitates the handling of such uncertainty.

Fuzzy modeling can be thought of as the task of designing a fuzzy inference system. The selection of important parameters for the inference system is crucial as the overall system behaviour is highly dependent on a large number of factors such as how the membership functions are chosen, the number of rules involved, the fuzzy operator used, and so on.

Based on our observations of implementing fuzzy multiple heuristic orderings on examination timetabling problems (see [3, 4]), we found that in many cases considering three heuristic orderings simultaneously will produce better solutions compared to single heuristic ordering and two heuristic orderings. Inspired by that finding, in this present work the focus is on creating a fuzzy inferencing system based on three of the five single heuristics, Largest Degree (*LD*), Largest Enrollment (*LE*) and Least Saturation Degree (*SD*). These three heuristics were selected as these were the ones that featured in our work on examination timetabling, and based on the fact that the design of a fuzzy system utilising all five heuristics would have been intractable (if the same tuning methodology had been utilised). As explained below, each variable has 11 options of the membership function's shape. For fuzzy system with 3 input variables, the search in tuning process needs to explore 11^3 (1331) combinations of membership functions. If we consider a fuzzy system with 5 input variables, the tuning process would need to explore 11^5 (161,051) combinations. As we have 11 data sets on which the system is run, experiments with 5 heuristic orderings would take months to finish.

A restricted form of exhaustive search was used to find the most appropriate shape for the fuzzy membership functions in the system. There are very many alternatives that may be used when constructing a fuzzy model. In our implementation, we arbitrarily restricted the search based on the membership functions as shown in Figure 2. Triangular shape membership functions were employed to represent three linguistic terms; *small*, *medium* and *high*. The membership function was tuned by moving the point *cp* along the universe of discourse. This single point corresponded to the right edge for the term *small*, the centre point for the term *medium* and the left edge for the term *high*. Thus, there was one *cp* parameter for each fuzzy variable (three inputs and one output).

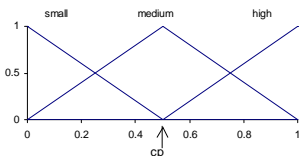


Figure 2: The triangular membership function for *Fuzzy Multiple Heuristic Ordering*

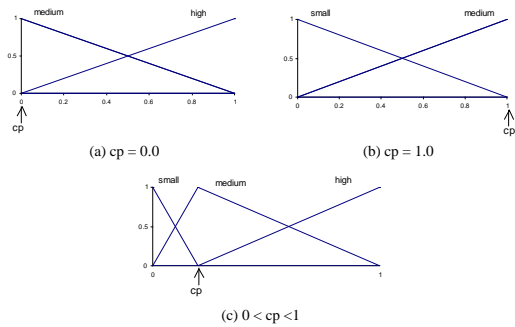


Figure 3: Range of possible membership functions

A search was then carried out to find the best set of *cp* parameters. During this search, each point *cp* (for any of the fuzzy variables) can take a value between 0.0 and 1.0 inclusive. Increments of 0.1 were used (i.e. the values 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0) for all data sets. The effect of varying the point *cp* from 0.0 to 1.0 is shown in Figure 3. The solution quality calculation described in Section 3.1 is used as the criteria to determine the fitness of the membership function combinations. Table 1 illustrates the fuzzy rule set, consisting of 27 fuzzy rules connecting the three input variables (*LD*, *LE* and *SD*) to a single output variable, *eventweight*. Note that, in addition to the three basic terms, the *hedge* ‘very’ was utilised to create two extra terms for the output variable.

In our implementation, the input values were normalised within the range $[0, 1]$. The transformation is as follows:

$$v' = \frac{(v - \min A)}{(\max A - \min A)}$$

where v is the actual value in the initial range $[\min A, \max A]$. For example, if $v = 10$ in $[0, 20]$, the normalised value v' is 0.5 in the new range $[0, 1]$. Standard Mamdani style fuzzy inference was implemented with standard Zadeh (min-max) operators. Centroid defuzzification was utilised to obtain a single crisp (real) value for *eventweight*, that represents the difficulty of scheduling the event.

All events in the given problem instance were evaluated using the same fuzzy system, and the sequential constructive algorithm used the crisp output of each event for ordering all events. The event with the biggest crisp value was selected to be scheduled first, and the process continued until all the events were scheduled without violating any of the hard constraints.

Table 1: Fuzzy rule set for *Fuzzy Multiple Heuristic Ordering*

LE	LD								
	S			M			H		
	SD			SD			SD		
	S	M	H	S	M	H	S	M	H
S	S	VS	VS	S	S	VS	M	S	S
M	S	S	VS	H	M	M	H	M	M
H	H	S	S	H	M	M	VH	H	M

(VS=very small, S=small, M=medium)
(H=high, VH=very high)

Table 2: Course timetabling problem characteristics

Data sets	No. of events	No. of rooms	No. of students	No. of features
Small1	100	5	80	5
Small2	100	5	80	5
Small3	100	5	80	5
Small4	100	5	80	5
Small5	100	5	80	5
Medium1	400	10	200	5
Medium2	400	10	200	5
Medium3	400	10	200	5
Medium4	400	10	200	5
Medium5	400	10	200	5
Large	400	10	400	10

3 Implementation

3.1 Problem Definition

Table 2 reproduces the characteristics of the data sets that were used for these experiments. These problems deal with the assignment of courses into time slots such that rooms do not violate any of the following hard constraints:

1. No student is required to attend more than one course at the same time
2. A course can only be scheduled to a room which satisfies the features required by the course
3. A course can only be scheduled to a room which has enough room to accommodate all students registered for it
4. Only one course can be scheduled in one room at any time slot

Any solutions which do not violate any of the above hard constraints are defined as feasible solutions. Only feasible solutions are accepted. Besides these hard constraints, the solutions should also try to satisfy the following soft constraints:

1. No student should be scheduled to attend only one course on a day
2. No course should be scheduled at the last time slot of the day for any student
3. No student should be scheduled to attend more than two courses consecutively in any one day

An attempt is made to best satisfy these soft constraints, but they are not compulsory. The quality of any feasible solution is measured by simply summing the number of students that fail to satisfy the soft constraints. Hence, the less the number of students that violate the soft constraints, the better the solution quality.

The timetable is developed for one week, from Monday to Friday. For each day, there are 9 time slots available. Hence, the number of time slots available is 45 x *number of rooms*.

3.2 Experimental Results

In order to reduce the computational time, the number of ‘*rescheduling procedures*’ allowed was limited to 500 for small and medium data sets, whereas for large data set it was limited to 1000 times. This meant that during the search for a solution, if too many events needed to be reshuffled, the fuzzy model that was currently under consideration was skipped and the solution for that fuzzy model was treated as an infeasible solution. A new fuzzy model was then tested. This was because, from observation, it was found that in many cases good quality solutions were usually produced when only a small number of ‘*rescheduling procedures*’ were required. The same setting for the maximum number of required ‘*rescheduling procedures*’ was implemented for *Single Heuristic Ordering*.

The algorithm was developed using Java based object oriented programming. The fuzzy inference engine developed by Sazonov *et al.* [25] was implemented. The experiments were run on a PC with a 1.8 GHz Pentium 4 and 256MB of RAM. Each heuristic ordering was run only once for each of the problem instances.

The experimental results are shown in Table 3. The best results amongst the heuristic orderings implemented are highlighted in bold font. The ‘-’ in Table 3 indicates that no feasible solution was generated within the specified maximum number of ‘*rescheduling procedures*’. It can be seen that the *Fuzzy Multiple Heuristic Ordering* has outperformed all *Single Heuristic Ordering* in all tested problem instances.

In term of feasibility, the *Fuzzy Multiple Heuristic Ordering* managed to produce feasible solutions for all data sets, whereas the best *Single Heuristic Ordering*, SD, only managed to produce 10 feasible solutions out of 11 data sets. Other *Single Heuristic Orderings* were worse. Moreover, no *Single Heuristic Ordering* was able to produce a feasible solution for the *Large* problem instance.

Table 3: Comparison of solution quality between *Single Heuristic Ordering* and *Fuzzy Multiple Heuristic Ordering*

Data Sets	Best	Single Heuristic				
	Fuzzy	LD	SD	LCD	LE	WLD
Small1	10	78	31	48	79	80
Small2	9	45	44	55	34	52
Small3	7	28	30	42	41	27
Small4	17	42	50	48	51	48
Small5	7	41	29	74	43	47
Medium1	243	423	345	433	465	445
Medium2	325	-	398	-	-	-
Medium3	249	-	298	-	-	-
Medium4	285	-	403	-	-	-
Medium5	132	296	252	307	399	445
Large	1138	-	-	-	-	-

Table 4: Comparison of number of ‘*rescheduling procedures*’ required to produce the solutions shown in Table 3

Data Sets	Best	Single Heuristic				
	Fuzzy	LD	SD	LCD	LE	WLD
Small1	0	0	0	0	0	0
Small2	0	0	0	0	0	0
Small3	0	0	0	0	0	0
Small4	0	0	0	0	0	0
Small5	0	0	0	0	0	0
Medium1	0	40	0	122	60	59
Medium2	0	-	0	-	-	-
Medium3	0	-	0	-	-	-
Medium4	1	-	0	-	-	-
Medium5	0	2	0	51	41	40
Large	307	-	-	-	-	-

Table 4 summarises the performance of each heuristic ordering in terms of the number of ‘*rescheduling procedures*’ required. It can be seen that, all heuristics obtained the solutions for the small size problem instances without any ‘*rescheduling procedures*’. On the other hand, only the *Fuzzy Multiple Heuristic Ordering* and *Single Heuristic Ordering SD* managed to find solutions for all of the medium size problem instances without any ‘*rescheduling procedures*’ (except for *Medium4* problem instance in which the *Fuzzy Multiple Heuristic Ordering* needed to perform the ‘*rescheduling procedures*’ for one event). However, for the *Large* problem instance the *Fuzzy Multiple Heuristic Ordering* needed to perform the ‘*rescheduling procedures*’ 307 times before it found the solution, whereas *Single Heuristic Ordering SD* was unable to find a feasible solution (refer to Table 4).

4 Discussion and Future Work

The main objective of this research is to investigate the potential of implementing fuzzy systems in solving timetabling problem. As fuzzy approaches have successfully been applied to many problem which involve multi criteria decision

making, we have proposed a new heuristic that utilises fuzzy reasoning to assess the difficulties of scheduling events into feasible time slots such that several heuristic orderings are simultaneously considered.

In current practice, the decision to choose which event is the most difficult to schedule is usually based only on a single heuristic ordering. This selection decision is important because the order in which events are selected will influence the search algorithm in finding a feasible solution. Graph based heuristic orderings (as explained in Section 2.1) are also usually applied individually to measure the difficulty of the event to be scheduled.

We do not expect to generate solutions that will outperform the solutions produced by *iterative improvement* methods on the same benchmark data sets, but we would expect that a more ‘appropriate’ ordering of events can be generated if more than one heuristic ordering is used to measure the difficulty of the event to be scheduled. By providing a good ordering of events, the sequential constructive algorithm will hopefully be guided to search for better quality solutions. Looking at the quality of the produced solutions summarised in Table 3, for all test instances of small and medium size, the *Fuzzy Multiple Heuristic Ordering* resulted in better solutions compared to any of the single heuristic orderings. For the *Large* data set, a feasible result was obtained only when the *Fuzzy Multiple Heuristic Ordering* was implemented.

This is consistent with our previous implementation of *Fuzzy Multiple Heuristic Ordering* on examination timetabling problems (see [3, 4]). Hence, these observations seem to indicate that this *Fuzzy Multiple Heuristic Ordering* approach may be applicable to a wider range of timetabling and scheduling problems.

Table 5 show the comparison of our best results to the approaches of other researchers applied to the same data sets. However, it has to be kept in mind that our method uses a simple constructive initial solution, compared to the other methods which are iterative improvement approaches (except Burke *et al.* [13]). Burke *et al.* [11] and Socha *et al.* [27] start finding the solution by constructing an infeasible initial solution and then iteratively improving the timetable within a limited number of evaluations. Abdullah and Burke [1] start with feasible solutions and use a *variable neighbourhood search* with randomised iterative improvement

Table 5: Comparison of solution quality with other results in literature

Data Set	FMHO	NS	GHH	THH	RRLS	AMM
		[1]	[13]	[11]	[27]	[27]
Small1	10	0	6	1	8	1
Small2	9	0	7	2	11	3
Small3	7	0	3	0	8	1
Small4	17	0	3	1	7	1
Small5	7	0	4	0	5	0
Medium1	243	242	372	146	199	195
Medium2	325	161	419	173	202.5	184
Medium3	249	265	359	267	-	248
Medium4	285	181	348	169	177.5	164.5
Medium5	132	151	171	303	-	219.5
Large	1138	-	1068	1166	-	851.5

FMHO - Fuzzy Multiple Heuristic Ordering

NS - Neighbourhood Search

GHH - Graph-Based Hyperheuristic

THH - Tabu-Search Hyperheuristic

RRLS - Random Restart Local Search

AMM - Ant MAX-MIN Algorithm

techniques to improve the solutions. Burke *et al.* [13] used a sequence of heuristic orderings to construct the initial solution and applied steepest descent local search to improve the solution.

Although our approach did not perform particularly well for small size problem instances, it is evident that our results are comparable to the other approaches for the medium and large size problem instances. Indeed, for *Medium5* data set we obtained the best result amongst the compared approaches.

We think it is unfair to compare our constructive approach with the iterative techniques. It is more interesting to compare with Burke *et al.*'s [13] approach because they used a sequence of heuristic orderings to construct the solution whereas we use several heuristic orderings simultaneously to construct the timetable. When comparing these two constructive approaches, our approach produced better results for all of the medium size problem instances, but slightly worse solutions were obtained for small and large size problem instances. We believe these initial solutions can be easily improved by applying a simple optimisation algorithm.

This paper has established that there is significant potential in taking this approach further by adding more heuristics. Future work could use other possible combinations of heuristic ordering listed in Section 2.1.

Acknowledgements

This research is supported by the Universiti Teknologi Malaysia (UTM) and the Ministry of Science, Technology and Innovation (MOSTI) Malaysia.

References

- [1] S. Abdullah and E. K. Burke. Using a randomised iterative improvement algorithm to improve the solution quality of course timetabling. 2005. Accepted to be published in MIC2005.
- [2] R. Alvarez-Valdes, E. Crespo, and J. M. Tamarit. Assigning students to course sections using tabu search. *Annals of Operations Research*, 96:1 – 16, Jan 2000.
- [3] H. Asmuni, E. K. Burke, and J. M. Garibaldi. A comparison of fuzzy and non-fuzzy ordering heuristics for examination timetabling. In A. Lotfi, editor, *Proceeding of 5th International Conference on Recent Advances in Soft Computing 2004*, pages 288–293, 2004.
- [4] H. Asmuni, E. K. Burke, and J. M. Garibaldi. Fuzzy multiple ordering criteria for examination timetabling. In E. K. Burke and M. Trick, editors, *Proceeding of 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 51–65, 2004.
- [5] H. M. Aufm Hofe. Solving rostering tasks by generic methods for constraint optimization. *International Journal of Foundations of Computer Science*, 12(5):671–693, 2001.
- [6] P. Boizumault, Y. Delon, and L. Peridy. Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26(2):217–233, 1996.
- [7] S. C. Brailsford, C. N. Potts, and B. M. Smith. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119:557581, 1999.
- [8] E. K. Burke, Y. Bykov, J. P. Newall, and S. Petrovic. A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research*, 13(2):139–151, 2003.
- [9] E. K. Burke, D. De Werra, and J. Kingston. *Handbook of Graph Theory*, chapter Applications in Timetabling, pages 445–474. Chapman Hall, CRC Press., 2003.
- [10] E. K. Burke, K. Jackson, J. H. Kingston, and R. F. Weare. Automated university timetabling: The state of the art. *Computer Journal*, 40:565–571, 1997.
- [11] E. K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, Dec 2003.

- [12] E. K. Burke, B. MacCarthy, S. Petrovic, and R. Qu. Case-based reasoning in course timetabling: An attribute graph approach. In D. W. Aha and I. Watson, editors, *Practical Aspects of Declarative Languages: First International Workshop, PADL'99, San Antonio, Texas, USA*, volume 2080 of *LNCS*, pages 90–104. Springer-Verlag, Berlin, Heidelberg, 2001.
- [13] E. K. Burke, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper heuristic for timetabling problems. *European Journal of Operational Research*. To appear 2005.
- [14] E. K. Burke and J. P. Newall. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research*, 129:107–134, 2004.
- [15] E. K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140:266–280, 2002.
- [16] M. W. Carter. A comprehensive course timetabling and student scheduling system at the university of waterloo. In E. K. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III (PATAT 2000, Konstanz Germany, August, selected papers)*, volume 2079 of *Lecture Notes in Computer Science*, pages 64–82. Springer-Verlag, Berlin Heidelberg, 2001.
- [17] M. W. Carter, G. G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 47:373–383, 1996.
- [18] M. W. Carter and G. Laporte. Recent developments in practical course timetabling. *Lecture Notes in Computer Science*, 1408:3–19, 1998.
- [19] L. D. Gaspero and A. Schaerf. Multi-neighbourhood local search with application to course timetabling. *LNCS*, 2740:262 – 275, Aug 2003.
- [20] H.-J. Goltz and D. Matzke. University timetabling using constraint logic programming. In G. Gupta, editor, *Practical Aspects of Declarative Languages: First International Workshop, PADL'99, San Antonio, Texas, USA*, volume 1551 of *Lecture Notes in Computer Science*, pages 320–334. Springer-Verlag GmbH, Berlin, Heidelberg, New York, Jan 1999.
- [21] J. Li and R. S. K. Kwan. A fuzzy genetic algorithm for driver scheduling. *European Journal of Operational Research*, 147:334–344, 2003.
- [22] E. Ozcan and A. Alkan. Memetic algorithms for timetabling. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 1796–1802, Canberra, 8-12 Dec. 2003. IEEE Press.
- [23] S. Petrovic and E. K. Burke. *Ch. 45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter University Timetabling. CRC Press, 2004.
- [24] S. Petrovic, V. Patel, and Y. Yang. University timetabling with fuzzy constraints. In E. K. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V (PATAT 2004, Pittsburg USA, August 2004, selected revised papers)*, LNCS. Springer-Verlag, Berlin Heidelberg New York, to appear.
- [25] E. S. Sazonov, P. Klinkhachorn, H. V. S. Gangarao, and U. B. Halabe. Fuzzy logic expert system for automated damage detection from changes in strain energy mode shapes. *Non-destructive Testing and Evaluation*, 18(1):1–20, 2002.
- [26] A. Schaerf. A survey of automated timetabling. *Artificial Intelligent Review*, 13:87–127, 1999.
- [27] K. Socha, J. Knowles, and M. Sampels. A MAX-MIN Ant System for the university timetabling problem. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 - Third International Workshop on Ant Algorithms*, Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany, 2002.
- [28] D. Teodorovic and P. Lucic. A fuzzy set theory approach to the aircrew rostering problem. *Fuzzy Sets and Systems*, 95:261–271, 1998.
- [29] Y. Yang and S. Petrovic. A novel similarity measure for heuristic selection in examination timetabling. In E. K. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V (PATAT 2004, Pittsburg USA, August 2004, selected revised papers)*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg New York, to appear.
- [30] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.