Fuzzy Multiple Heuristic Ordering for Examination Timetabling

Hishammuddin Asmuni[1], Edmund K. Burke[1], Jonathan M. Garibaldi[1], and Barry McCollum[2]

[1] School of Computer Science and Information Technology,
University of Nottingham, Jubilee Campus, Wollaton Road,
Nottingham, NG8 1BB, UK
{hba,ekb,jmg}@cs.nott.ac.uk
[2] School of Computer Science,
Queen's University Belfast,
Belfast BT7 1NN, UK
b.mccollum@qub.ac.uk

Abstract. In this paper, we address the issue of ordering exams by simultaneously considering two separate heuristics using fuzzy methods. Combinations of two of the following three heuristic orderings are employed: largest degree, saturation degree and largest enrollment. The fuzzy weight of an exam is used to represent how difficult it is to schedule. The decreasingly ordered exams 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. Unscheduling and rescheduling exams is performed until all exams are scheduled. The proposed algorithm has been tested on 12 benchmark examination timetabling data sets and the results show that this approach can produce good quality solutions. Moreover, there is significant potential to extend the approach by including a larger range of heuristics.

1 Introduction

Examination timetabling is essentially the problem of allocating exams to a limited number of time periods in such a way that none of the specified hard constraints are violated. A timetable which satisfies all hard constraints is often called a *feasible* timetable. In addition to the hard constraints, there are often many soft constraints whose satisfaction is desirable (but not essential). The set of constraints which need to be satisfied is usually very different from one institution to another as reported by Burke *et al.* [12]. However, a common hard constraint across all problem instances is the requirement to avoid any student being scheduled for two different exams at the same time.

In practice, each institution usually has a different way of evaluating the quality of a feasible timetable. In many cases, the measure of quality is calculated based upon a penalty function which represents the degree to which the soft constraints are satisfied.

Over the years, numerous approaches have been investigated and developed for exam timetabling. Such approaches include constraint programming [30, 8, 26, 34, ?], graph colouring [21, 16, 15], case based reasoning [18], hyper-heuristics [15] and various metaheuristic approaches including greedy local search [23, 19], genetic algorithms [13, 28], tabu search [29], simulated annealing [41], the great deluge algorithm [9], and hybridized methods [34] which draw on two or more of these techniques. Discussions about other approaches can be found in papers by Bardadym [4], Burke *et al.* [14], Burke and Petrovic [17], Carter [20], Carter and Laporte [22], De Werra [27], Petrovic and Burke [35] and Schaerf [39].

Since being introduced by Zadeh in 1965 [43], fuzzy methodologies have been successfully applied in a wide range of real world applications. In scheduling and timetabling applications, fuzzy evaluation functions have been utilised in a number of different applications. For example, Dahal, Aldridge and McDonald [25] considered such approaches in generator maintenance scheduling, and Abboud *et al.* [1] used fuzzy target gross sales (fuzzy goals) to find 'optimal' solutions of a manpower allocation problem, where several company goals and salesmen constraints need to be considered simultaneously. Fuzzy methodologies have been investigated for other timetabling problems such as aircrew rostering [40], driver scheduling [31] and nurse rostering [3].

In the specific context of examination timetabling, fuzzy methods have been implemented for measuring the problem similarity in case based reasoning by Yang and Petrovic [42]. In this work, a fuzzy similarity measure is used to retrieve a good heuristic ordering for a new problem based on comparison with previous problems that are stored in the case base. The selected heuristic ordering is then applied to the new problem for generating an initial solution before applying the Great Deluge Algorithm in the improvement stage. Their results indicated that the performance of this algorithm is better when this fuzzy similarity measure is applied in the initialisation stage compared to other initialisation approaches.

In [37], Petrovic *et al.* employed fuzzy methodologies to measure the satisfaction of various soft constraints. The authors described 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 their approach on the same 12 benchmark problem instances that are considered in this paper.

Approaches which order the events prior to assignment to a period have been discussed by several authors including Boizumault *et al.* [5], Brailsford *et al.* [6], Burke *et al.* [11], Burke and Newall [16], Burke and Petrovic [17] and Carter *et al.* [21]. In the context of the same benchmark data sets used in our experiments, this sequencing strategy has been implemented by Carter *et al.* [21], Burke and Newall [16] and Burke *et al.* [15]. In [21], the authors 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. Then, the exams were selected sequentially and assigned to a time slot that satisfied all the specified constraints. If no clash free time slot was found, backtracking was implemented. The process was continued until all the exams were scheduled and a feasible solution was produced. Burke and Newall [16] 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 [15] 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 constructive algorithm in finding the 'best solution' for each problem instance.

In this paper, a fuzzy methodology is used to rank exams based on an assessment of how difficult they are to schedule taking into account multiple heuristics. This paper is motivated by the observation that the consideration of more than one heuristic to rank the exams may lead to rankings that better reflect the actual difficulty of placing the exam, as several factors are simultaneously taken into account. The fuzzy multiple heuristic ordering method described in this paper should not be confused with multi-criteria approaches to examination timetabling, such as those described in Arani and Lotfi [2], Burke *et al.* [10], Lotfi and Cerveny [33], and Petrovic and Bykov [36]. In our approach, two heuristic orderings are simultaneously considered to rank the exams, whereas in [2, 10, 33, 36] they employ multi-criteria approaches to evaluate timetabling solutions and describe approaches which can handle this.

In the following section, the proposed algorithm and the experiments carried out are explained in detail. Section 3 describes the results obtained. These results are discussed and some concluding comments presented in Sections 4 and 5 respectively.

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 [11]). 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 Fig. 1). We consider three different versions of the basic algorithm, which employ three different heuristic orderings respectively and require the following steps to assign all exams to a time slot. First, the exams are ordered (most difficult first) by applying one of the ordering heuristics. The following heuristics are employed as ordering criteria:

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

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

Least Saturation Degree (SD) First. The number of time slots available is used to order the exams. The basic motivation is that exams 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 exam.

Then, exams are selected sequentially from the ordering and assigned a valid time slot that will cause the minimum penalty cost for that exam. If no clash free time slot is available, the exam is skipped and the process continued with the next exam. The skipped exams are then revisited and a process for scheduling the unscheduled exams is carried out (see Fig. 2).

```
Sort unscheduled exams using selected heuristic ordering;
Insert exams into the last timeslot with least penalty;
While there exist unscheduled exam
   Perform the process for scheduling the unscheduled exams;
   Sort unscheduled exams using selected heuristic ordering;
End while
```

Fig. 1. Pseudo code for general framework of sequential construction algorithm

The sequential construction algorithm used here is similar to the approach applied by Carter *et al.* [21] with some modification. Basically, there are three differences between these two algorithms. The first difference is in the initial stage of the algorithm. In our algorithm we apply the heuristic ordering to *all* exams, whereas Carter *et al.*'s algorithm first finds the maximum-clique of examinations and assigns them to different time slots, and then applies heuristic ordering to the remaining exams. The second difference is in the selection of a free time slot. A search is carried out to find the clash free time slot with least penalty cost in order to assign each exam to a time slot. In our algorithm, if several time slots are available, then the last available time slot in the list will be selected. (It was found that the choice of assigning exams to the last available time slot or the first available time slot did not make much difference, as the main purpose of this was simply to spread out the student's timetable.) In contrast, Carter *et al.* chose the first clash free time slot found in which to assign the exam.

Thirdly, for reshuffling a scheduled exam, we randomly select a time slot from the list of time slots with the same minimum number of scheduled exams that needed to be 'bumped back', whereas Carter *et al.* used *minimum disruption cost* to break any ties. A detailed overview of the 'unschedule and reschedule scheduled exams' algorithm is shown in Fig. 2.

2.2 The Fuzzy Multiple Heuristic Ordering

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)

```
k := number of unscheduled exams;
For u := 1 to k
    Select exam[u];
    Find timeslots where exam[u] can be inserted with minimum number of
    scheduled exams need to be removed from the timeslot;
    If found more than one slot with the same number of scheduled exams
    need to be removed
      Select a timeslot randomly from the candidate list of slots, ts;
    End if
    c :=  number of exam in timeslot $ts_u$ that conflict with exam[u];
    For m := 1 to c
       Select exam[m];
      If found another timeslot with minimum cost to move exam[m]
        Move exam[m] to the timeslot;
      else
        Bump back exam[m] to unscheduled exam list;
      End if
    End for
    Insert exam[u] to timeslot $ts_u$;
    Remove exam[u] from unscheduled exam list;
End for
```

Fig. 2. Pseudo code for rescheduling the scheduled exams

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. A fuzzy set A of a universe of discourse X (the range over which the variable spans) is characterised by a *membership function* $\mu_A : X \rightarrow [0, 1]$ which associates with each element x of X a number $\mu_A(x)$ in the interval $[0, 1]$, with $\mu_A(x)$ representing the *grade of membership* of x in A. The precise meaning of the membership grade is not rigidly defined, but is supposed to capture the 'compatibility' of an element to the notion of the set.

Fuzzy systems are used for representing and employing knowledge that is imprecise, uncertain, or unreliable. They usually consist of four main interconnected components: an input fuzzifier, a set of rules, an inference engine, and an output processor (defuzzifier). Rules which connect input variables to output variables in 'IF ... THEN ...' form are used to describe the desired system response in terms of *linguistic* variables (words) rather than mathematical formulae. The 'IF' part of the rule is referred to as the 'antecedent', the 'THEN' part is referred to as the 'consequent'. The number of rules depends on the number of inputs and outputs, and the desired behaviour of the system. Once the rules have been established, such a system can be viewed as a non-linear mapping from inputs to outputs. It is not appropriate to present a full description of the functioning of fuzzy systems here; the interested reader is referred to Cox [24] for a simple treatment or Zimmerman [44] for a more complete treatment.

The fuzzy inference process is illustrated for a three-rule system based on two input variables, LD and LE. Each of the input and output variables are associated with three linguistic terms; fuzzy sets corresponding to meanings of *small*, *medium* and *high*. These membership functions are chosen arbitrarily to span the universe of discourse of the variable. A rule set connecting the input variables (LD and LE) to a single output variable, *examweight*, is constructed.

The following three rules are used to illustrate the behaviour of this example system (note that this is only an illustrative example; the membership functions and rules used in each actual experiment are described in Section 2.3 below):

Rule 1: IF (*LD* is *small*) AND (*LE* is *medium*) THEN (*examweight* is *small*)
Rule 2: IF (*LD* is *medium*) AND (*LE* is *medium*) THEN (*examweight* is *medium*)
Rule 3: IF (*LD* is *medium*) AND (*LE* is *high*) THEN (*examweight* is *high*)

The first stage is to normalise the input values within the range $[0, 1]$. The transformation is as follows:

$$v' = \frac{(v - minA)}{(maxA - minA)}$$

where v is the actual value in the initial range $[minA, maxA]$. For example, if $v = 10$ in $[0, 20]$, the normalized value v' is 0.5 in the new range $[0, 1]$.

Fig. 3 illustrates the inferencing of this system (a Mamdani inference process) with normalised values for *LD* and *LE* of 0.4 and 0.65, respectively. For each rule in turn, the fuzzy system operates as follows. The input component ('fuzzifier') computes the membership grade for each input variable based on the membership functions defined. That is, in *Rule 1*, the membership grade is computed for *LD* in the fuzzy set *small* and for *LE* in the fuzzy set *medium.* As shown in the figure, the determined grades of membership for each input variable are:

$$\mu_{small}(LD = 0.4) = 0.15, \quad \text{and}$$
$$\mu_{medium}(LE = 0.65) = 0.6$$

With these fuzzified values, the inference engine then computes the overall truth value of the antecedent of the rule (*Rule 1*) by applying the appropriate fuzzy operators corresponding to any connective(s) (*AND* or *OR*). In the example, the fuzzy *AND* operator is implemented as a minimum function:

$$\begin{aligned} \textbf{Rule 1} \quad & \text{IF } (LD \text{ is } small) \text{ AND } (LE \text{ is } medium) \\ \mu_{Rule1} &= \mu_{small}(LD = 0.4) \;\wedge\; \mu_{medium}(LE = 0.65) \\ &= min(0.15, 0.6) \\ &= 0.15 \end{aligned}$$

Next, the inference engine applies the implication operator to the rule in order to obtain the fuzzy set to be accumulated in the output variable. In this case, inferencing is implemented by truncating the output membership function at the level corresponding to the computed degree of truth of the rule's antecedent. The effect of this process can be seen in the *consequent* part of *Rule 1* in which the membership function for the linguistic term *small* was truncated at the level of 0.15. The same processes are applied to the rest of the rules in turn.

Finally, all the truncated output membership functions are aggregated together to form a single fuzzy subset (labelled as *Final Output* in Fig. 3) by taking the maximum across all the consequent sets. A further step (known as

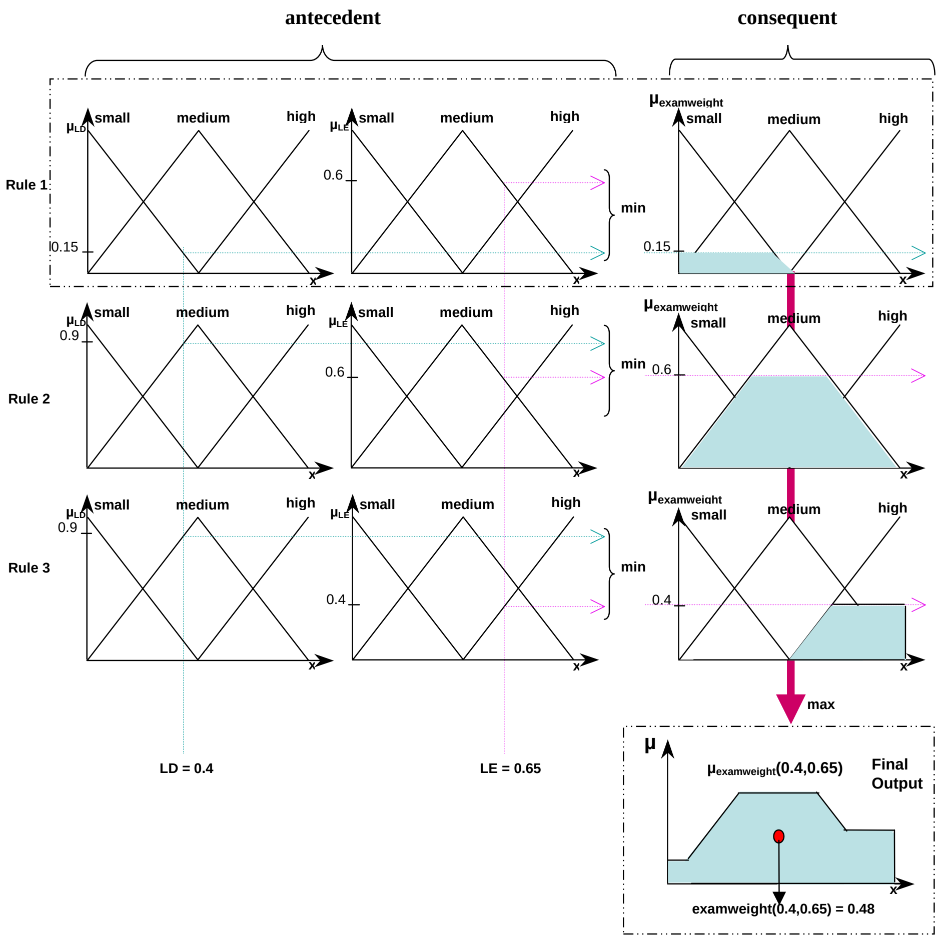

Fig. 3. A three-rule Mamdani Inference process

'defuzzification') is then performed if (as is usual) the final fuzzy output is to be translated into a crisp output. We applied a common form of this process, termed 'centre of gravity defuzzification' as it is based upon the notion of finding the centroid of a planar figure, as given by:

$$\sum_i \frac{\mu(x_i) \cdot x_i}{\mu(x_i)}$$

In the example of Fig. 3, the output for the fuzzy system (that represents how difficult the exam is to be scheduled) is 0.48 for the given inputs (i.e an exam with *LD* and *LE* of 0.4 and 0.65, respectively).

All exams in the given problem instance are evaluated using the same fuzzy system, and the sequential constructive algorithm uses the crisp output of each exam for ordering all exams. The exam with the biggest crisp value is selected to be scheduled first, and the process continues until all the exams are scheduled without violating any of the hard constraints.

2.3 Description of Experiments

A number of experiments were carried out in which progressively more sophisticated fuzzy mechanisms were created to order the exams. In each experiment this ordering is simply inserted into the basic general algorithm presented in Fig. 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 exams in the problem instances were ordered based on a single heuristic ordering. All the exams were then selected to be scheduled based on this ordering.

Fixed Fuzzy LD+LE Model Next, a fixed fuzzy model that took into account multiple heuristic ordering was implemented. Two out of the three ordering heuristics described in Section 2.1, namely *largest degree* (LD) and *largest enrollment* (LE) were selected as input variables. The membership functions used in this experiment are shown in Fig. 4. The choice of these membership functions was based on 'trial and error' to test how the algorithm would work when exams were ordered with the aid of fuzzy reasoning.

The fuzzy rules used in this experiment are shown in Table 1. For simplicity, the fuzzy rules are expressed as a linguistic matrix (see [32]). In such a linguistic matrix, the left-most column and the first row denote the variables involved in the antecedent part of the rules. The second column contains the linguistic terms applicable to the input variable shown in the first column; those in the second row correspond to the input variable shown in the first row. Each entry in the main body of the matrix denotes the linguistic values of the consequent part of a rule. For instance, the bottom-right entry in Table 1 is read as *"IF LD is* *high AND LE is high THEN examweight is very high"*. The same representation is also used to express the fuzzy rule sets for the experiments explained in the following sections. Note that, in addition to the three basic terms, the *hedge* 'very' was utilised to create two extra terms for the output variable. The 'very' hedge squares the membership grade $\mu(x)$ at each x of the fuzzy set for the term to which it is applied. Thus the membership function of the fuzzy set for 'very small' is obtained by squaring the membership function of the fuzzy set 'small'.

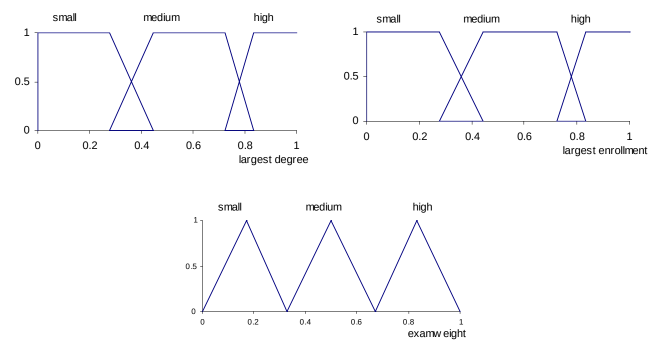

Fig. 4. Membership Functions for Fixed Fuzzy LD+LE Model

Table 1. Fuzzy Rule Set for Fixed Fuzzy LD+LE Model

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

VS:	very small
S:	small
M:	medium
H:	high
VH:	very high

Tuned Fuzzy LD+LE Model 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. For the purpose of finding a better fuzzy model, a relatively straight-forward tuning procedure was implemented in order to investigate whether the initial choice of fuzzy model was appropriate. This tuning procedure was then applied to different combinations of multiple ordering heuristics.

As an initial extension to the *Fixed Fuzzy LD+LE Model*, 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 Fig. 5. Triangular shape membership functions were employed to represent *small*, *medium* and *high*. However, the fuzzy model was then altered 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 (two inputs and one output).

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 data sets that have 400 and fewer exams, and 0.25 increments (i.e. the values 0.0, 0.25, 0.5, 0.75 and 1.0) for data sets that have more than 400 exams. The effect of varying the point *cp* from 0.0 to 1.0 is shown in Fig. 6.

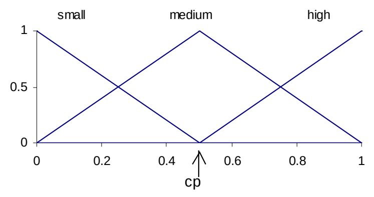

Fig. 5. The Membership Function for Tuned Fuzzy Model

In this experiment, the combination of *LD* and *LE* heuristics were again used as the fuzzy input variables. The fuzzy rule set used is shown in Table 2.

Table 2. Fuzzy Rule Set for Tuned Fuzzy LD+LE Model

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

VS:	very small
S:	small
M:	medium
H:	high
VH:	very high

Tuned Fuzzy SD+LE Model In this experiment, the same approach as above was employed, but now the combination of *SD* and *LE* were used as the fuzzy input variables. A new fuzzy rule set was required as the *SD* heuristic is reversed compared to the *LD* and *LE* heuristics. The fuzzy rule set is presented in Table 3.

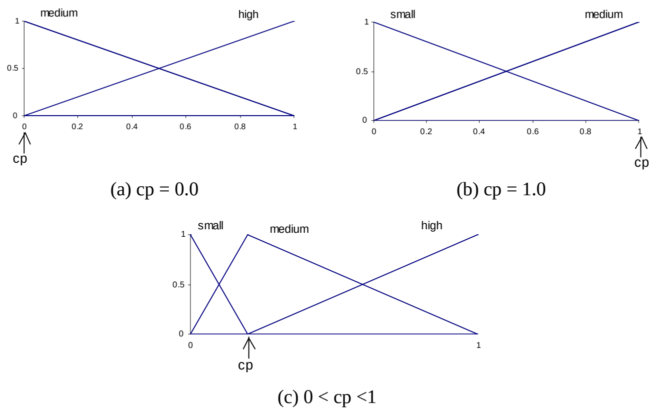

Fig. 6. Range of Possible Membership Functions

Table 3. Fuzzy Rule Set for Tuned Fuzzy SD+LE Model

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

VS:	very small
S:	small
M:	medium
H:	high
VH:	very high

3 Experimental Results

In this section the results obtained in each experiment are presented. In all experiments, the basic algorithm of Fig. 1 was employed. The only difference was the heuristic ordering used. The experiments were carried out with 12 benchmark data sets made publicly available by Carter *et al.* Table 4 reproduces the problem characteristics.

A proximity cost function was used to measure the timetable quality. The maximum capacity for each time slot was not taken into account. Only feasible timetables were accepted. The penalty function is taken from Carter *et al.* [21]. It is motivated by the goal of spreading out each student's examination schedule. If two exams scheduled for a particular student are t time slots apart, the weight is set to $w_t = 2^{5-t}$ where $t \in \{1, 2, 3, 4, 5\}$. The weight is multiplied by the number of students that sit for both of the scheduled exams. The average penalty per student is calculated by dividing the total penalty by total number of students. The following formulation was used (adapted from Burke *et al.* [9]), in which the goal is to minimize:

$$\frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij} w_{|p_j - p_i|}}{T},$$

Table 4. Examination Timetabling Problem Characteristics

Data Set	Number of slots	Number of exams	Number of students	Conflict density
CAR-F-92	32	543	18419	0.14
CAR-S-91	35	682	16925	0.13
EAR-F-83	24	190	1125	0.27
HEC-S-92	18	81	2823	0.42
KFU-S-93	20	461	5349	0.06
LSE-F-91	18	381	2726	0.06
RYE-F-92	23	486	11483	0.08
STA-F-83	13	139	611	0.14
TRE-S-92	23	261	4360	0.18
UTA-S-92	35	622	21266	0.13
UTE-S-92	10	184	2750	0.08
YOR-F-83	21	181	941	0.29

where N is the number of exams, s_{ij} is the number of students enrolled in both exam i and j, p_i is the time slot where exam i is scheduled, and T is the total number of students; subject to $1 \leq |p_j - p_i| \leq 5$.

The algorithm was developed using java based object oriented programming. The fuzzy inference engine developed by Sazonov *et al.* [38] was implemented. The experiments were run on a PC with a 1.8 GHz Pentium 4 and 256MB of RAM. In the case of the *Single Heuristic Ordering* and the *Fixed Fuzzy LD+LE Model* each instance was run five times. In the other experiments (that involved tuning the fuzzy model), the aim was to search for the best fuzzy model to guide the constructive algorithm. In order to reduce the size of the search space, only the membership functions are tuned, whereas the fuzzy rule set is fixed. In this tuning process, for problem instances that have 400 and fewer exams, the algorithm was tested on 1331 (3 variables and 11 options - 11^3) membership function combinations. Problem instances that have more than 400 exams were tested on 125 (3 variables and 5 options - 5^3) membership function combinations. Because of this, each instance was only run twice. For all experiments, only the best results are selected and presented in Table 5.

For comparison, the best results obtained by Carter *et al.* [21] when using various different heuristics to order the exams are shown in column 2 of Table 5. The results obtained for our three varieties of *Single Heuristic Ordering* are presented in columns 3,4 and 5. The results obtained for the *Fixed Fuzzy LD+LE Model* are shown in column 6. In general, these results are worse than for the best *Single Heuristic Ordering*, except for the STA-F-83 data set, where the fixed fuzzy model obtained the best result. This observation suggested that there might be promise in the fuzzy approach and prompted us to undertake further investigations with tuned fuzzy models. The results for the *Tuned Fuzzy LD+LE*

Table 5. Experimental results for single and fuzzy heuristic orderings

Data Set	Carter *et al.* [21]	Single Heuristic Ordering LD	LE	SD	Fixed Fuzzy LD+LE Model	Tuned Fuzzy LD+LE Model	Tuned Fuzzy SD+LE Model
CAR-F-92	6.2	5.56	5.03	5.50	5.65	4.62	**4.56**
CAR-S-91	7.1	6.38	5.90	5.91	6.31	5.60	**5.29**
EAR-F-83	36.4	40.58	45.88	49.10	48.14	38.41	**37.02**
HEC-S-92	10.8	14.98	14.94	14.27	16.93	12.53	**11.78**
KFU-S-93	14.0	18.63	16.46	18.60	18.29	16.53	**15.81**
LSE-F-91	10.5	15.08	14.52	13.46	16.84	12.35	**12.09**
RYE-F-92	7.3	12.95	11.12	11.60	12.98	11.75	**10.38**
STA-F-83	161.5	173.09	171.87	178.24	161.21	**160.42**	160.75
TRE-S-92	9.6	10.98	9.93	10.81	10.36	9.05	**8.67**
UTA-S-92	3.5	4.48	4.78	3.83	5.16	3.87	**3.57**
UTE-S-92	25.8	35.19	28.80	33.14	30.54	28.65	**28.07**
YOR-F-83	41.7	45.60	43.53	45.27	46.41	41.37	**39.80**

Model are shown in column 7 and those for the *Tuned Fuzzy SD+LE Model* in column 8.

The best fuzzy results obtained in Table 5 are highlighted in bold font. The corresponding membership functions of the fuzzy model which obtained the best result for each data set are presented in Fig. 7 and Fig. 8. It can be seen that the membership functions differ in each case — i.e. there is no generic fuzzy model which suits all the data sets.

4 Discussion

Amongst the three single heuristics, it would appear that LE is the 'best' in this context as it produced the best solution for 8 out of the 12 data sets, compared to only one for LD (for EAR-F-83) and three for SD (for HEC-S-92, LSE-F-91 and UTA-S-92). It also can be seen that, when compared to Carter *et al.*'s best results, our simplified version of their algorithm produced worse results in 10 out of the 12 data sets, but a slightly better timetable was obtained for the CAR-F-92 and CAR-S-91 cases. The *Fixed Fuzzy LD+LE Model* only achieves a better result than the best *Single Heuristic Ordering* in 1 out of the 12 data sets (STA-F-83). However, the rules and membership functions for this initial fuzzy model were completely arbitrary, so it may be surprising that it achieved a best result even once.

It is evident that the *Tuned Fuzzy LD+LE Model* produced better results than the *Fixed Fuzzy LD+LE Model* in all cases. Although entirely expected, this observation was taken as confirmation that the fuzzy system was capturing meaningful information and that the tuning procedure, although not finding the truly optimal fuzzy model (in the sense of the globally best set of membership functions for the given set of rules and other fixed aspects of the fuzzy system),

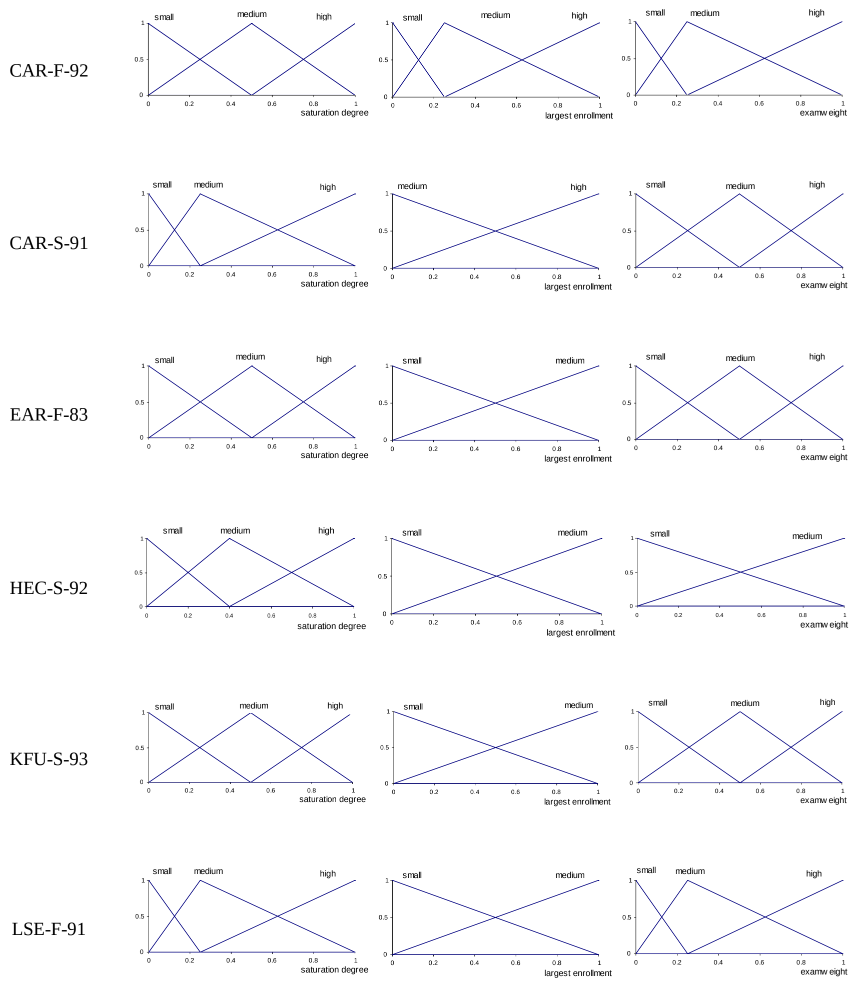

Fig. 7. Best fuzzy model for data sets CAR-F-92, CAR-S-91, EAR-F-83, HEC-S-92, KFU-S-93 and LSE-F-91

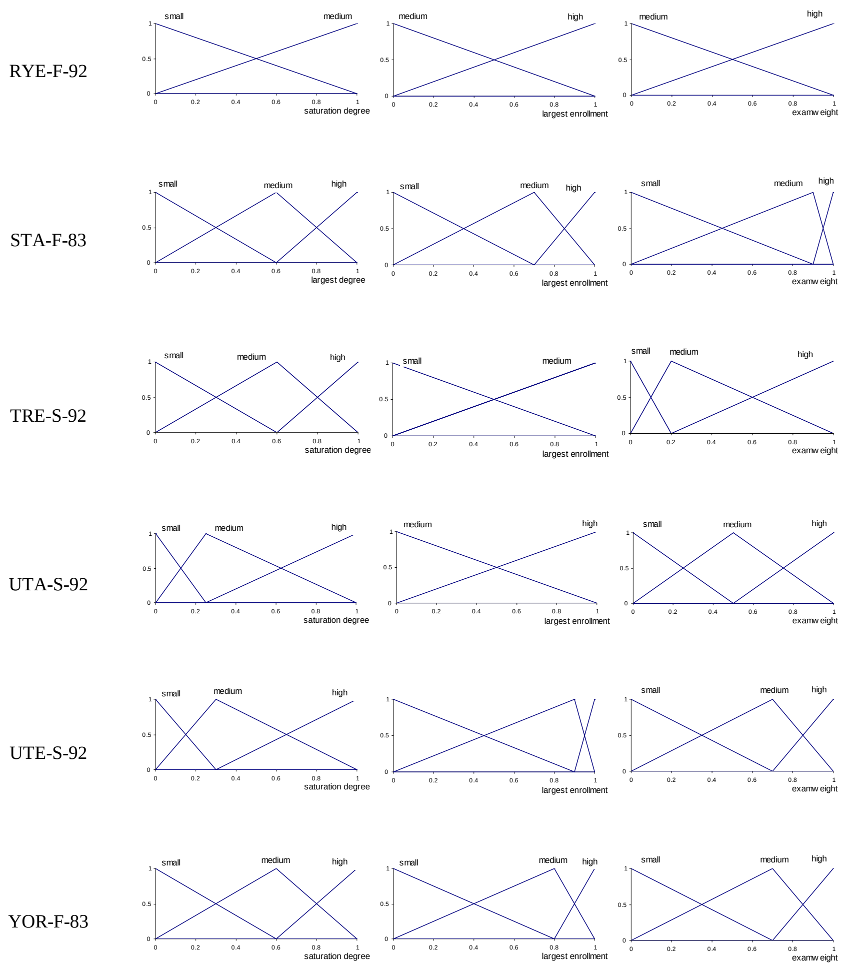

Fig. 8. Best fuzzy model for data sets RYE-F-92, STA-F-83, TRE-S-92, UTA-S-92, UTE-S-92 and YOR-F-83

was operating successfully. In comparison with best *Single Heuristic Ordering*, the *Tuned Fuzzy LD+LE Model* obtained better results in all cases except for the KFU-S-93, RYE-F-92 and UTA-S-92 data sets.

The *Tuned Fuzzy SD+LE Model* went on to produce better results than the *Tuned Fuzzy LD+LE Model* for all cases except the STA-F-83 data set. When compared to Carter *et al.*'s original results, the tuned fuzzy models operating on two heuristics simultaneously (taking the best tuned fuzzy model for each data set) obtained better results for 5 out of the 12 data sets. These were the CAR-F-92, CAR-S-91, STA-F-83, TRE-S-92 and YOR-F-83 data sets. Although these results have since been bettered by many authors (see the discussion of Table 6 below), these have been based on iterative improvement techniques rather than the constructive approach employed by Carter *et al.* and ourselves.

Initially, the choice to use a combination of the LD and LE heuristics was based on the fact that these heuristics are static in the sense that they only have to be calculated once at the beginning of the ordering process. In contrast, the SD heuristic must be recalculated after each exam is assigned to a slot. Thus, it was felt that tuning the fuzzy model based on the LD+LE combination would be quicker. The choice to use the SD+LE combination in the subsequent model was based on the observation that the LE heuristic ordering, when used alone, obtained the minimum penalty cost for 8 out of the 12 data sets while the SD heuristic ordering obtained the minimum cost for 3 out of 12. Thus it was felt that these offered the most promising combination of two heuristics.

The design of the fuzzy rule sets was based on three assumptions:

– if LD is *High* then examweight is *High*
– if LE is *High* then examweight is *High*
– if SD is *Small* then examweight is *High*

However, it must be emphasized that the rule sets specified in Tables 1 to 3 are only one possible instance (in the case of each experiment) out of a very large number of alternatives. Due to the very large number of degrees-of-freedom in any fuzzy model, it is very rare that the first fuzzy system constructed will perform at an acceptable level. Usually some form of optimisation or performance tuning of the system will need to be undertaken. The most significant influences on performance of a fuzzy system are likely to be the number and location of the membership functions and the number and form of the rules. In our implementation, the number and form of the rules are kept fixed in all cases. Although the fuzzy membership functions were, to a certain extent, tuned to obtain good performances, there was no attempt in the current work to tune the rule sets. It is highly likely that, given sufficient time to perform the tuning, a set of fuzzy rules leading to better performance of the fuzzy models could be obtained.

In Table 5, we have demonstrated that, in all cases, tuning the fuzzy model produces better results, as might be expected. This confirms our hypothesis that simultaneous ranking of multiple heuristic ordering *can* produce better results. The fact that the best fuzzy results are all obtained using different fuzzy membership functions, as shown in Figs. 7 and 8, means that no *generic* fuzzy model has been obtained at this stage. Such a generic model would be necessary if the approach is to be applied quickly and efficiently to novel data sets. The lack of such a generic fuzzy model may cast doubt regarding the usability and flexibility of this approach. This indicates that care must be taken when applying fuzzy techniques: it is certainly not the case that just because it is fuzzy it is necessarily better.

Table 6 shows the performance of our algorithm in comparison with selected recently published results on Carter *et al.*'s benchmarks. The best result amongst the compared techniques for each data set is highlighted in bold font. Collectively, these results have been selected to show the best known results for each data set. Although our algorithm has not beaten the best known result for any data set, its performance is broadly competitive with the others in the sense that it it *not* the worst in 6 out of the 12 data sets. It is also worth pointing out that our algorithm produces solutions for all 12 data sets, and that in two of the cases where ours produces the worst result, at least one of the other papers did not quote any result. However, it has to be kept in mind that our method is a simple constructive initial solution, compared to the other methods which are iterative improvement approaches. Although our results are well behind more recent results, especially those of Caramia *et al.*, interestingly our fuzzy constructive algorithm can beat Caramia's results for data sets CAR-F-92, CAR-S-91 and TRE-S-92.

Table 6. Results Comparison

Data Set	Our Best Results	Burke and Newall [7]	Burke *et al.* [9]	Caramia *et al.* [19]	Casey and Thompson [23]	Merlot *et al.* [34]
CAR-F-92	4.56	**4.10**	4.2	6.0	4.4	4.3
CAR-S-91	5.29	**4.65**	4.8	6.6	5.4	5.1
EAR-F-83	37.02	37.05	35.4	**29.3**	34.8	35.1
HEC-S-92	11.78	11.54	10.8	**9.2**	10.8	10.6
KFU-S-93	15.81	13.90	13.7	13.8	14.1	**13.5**
LSE-F-91	12.09	10.82	10.4	**9.6**	14.7	11.0
RYE-F-92	10.35	-	8.9	**6.8**	-	8.4
STA-F-83	160.42	168.73	159.1	158.2	**134.9**	157.3
TRE-S-92	8.67	8.35	**8.3**	9.4	8.7	8.4
UTA-S-92	3.57	**3.20**	3.4	3.5	-	3.5
UTE-S-92	27.78	25.83	25.7	**24.4**	25.4	25.1
YOR-F-83	40.66	37.28	36.7	**36.2**	37.5	37.4

Finally, some remarks should be made concerning the time required for our algorithm. In doing so, it is vital that a distinction must be made between the time taken to perform the tuning of the fuzzy models and the time taken to construct a solution once each fuzzy model is fixed. Once the fuzzy model is fixed, the time taken to construct a solution is no longer (in a practical sense) than the time taken when using a single heuristic ordering — that is, the additional time taken for the fuzzy system to perform its ordering is negligible. Indeed, there is some evidence (which we are investigating further at present) that, once the fuzzy model is fixed, solutions are constructed more quickly using the fuzzy ordering. It seems that this may be due to the lack of required backtracking when the fuzzy ordering is used. However, the time taken in tuning each fuzzy model is very significant. Of course, if a generic fuzzy model could be found — that is a single fuzzy model that produces good quality initial solutions for all data sets (including the 12 benchmark data sets used here and novel data sets) — then the approach could be widely adopted, with significant impact.

5 Conclusions

As far as the authors are aware, no other published work has described the exploration of fuzzy methodologies for simultaneously ordering exams in the construction of examination timetables. In this study, we have investigated a fuzzy methodology to use multiple heuristic ordering simultaneously. Our evaluation indicates that better solutions can be produced by this approach when compared against each of the heuristics alone. This is the key point of the paper. Our method does not produce any of the best benchmark results, but we only used a limited number of heuristics to demonstrate the potential. This paper has established that there is significant potential in taking this approach further by adding more heuristics.

We are encouraged by these promising initial results and aim to extend this work further. Future research avenues may include:

– investigating other combinations of heuristic ordering (using combinations of three or more heuristics),
– investigating different sets of fuzzy rules and fuzzy membership functions,
– exploring the use of more sophisticated optimization algorithms when tuning these and other fuzzy models, and
– testing the algorithms on course timetabling problems.

Acknowledgements. This research work is supported by the Universiti Teknologi Malaysia (UTM) and the Ministry of Science, Technology and Innovation (MOSTI) Malaysia. The authors would also like to thank the four anonymous referees for their helpful suggestions.

References

1. N. Abboud, M. Inuiguichi, M. Sakawa, and Y. Uemura. Manpower allocation using genetic annealing. *European Journal of Operational Research*, 111:405–420, 1998.
2. T. Arani and V. Lotfi. A three phased approach to final exam scheduling. *IIE Transactions*, 21(1):86–96, 1989.
3. 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.

4. V.A. Bardadym. Computer aided school and university timetabling : The new wave. In E. K. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling I (PATAT 1995, Edinburgh, Aug/Sept, selected papers)*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer-Verlag, Berlin Heidelberg New York, 1996.
5. P. Boizumault, Y. Delon, and L. Peridy. Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26(2):217–233, 1996.
6. S.C. Brailsford, C.N. Potts, and B.M. Smith. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119:557 581, 1999.
7. E. K. Burke and J. P. Newall. Enhancing timetable solutions with local search methods. In E. K. Burke and P.D. Causmaecker, editors, *Practice and Theory of Automated Timetabling IV (PATAT 2002, Gent Belgium, August, selected papers)*, volume 2740 of *Lecture Notes in Computer Science*, pages 195–206. Springer-Verlag, Berlin Heidelberg New York, 2003.
8. Edmund K. Burke and Peter Ross, editors. *Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August 29 - September 1, 1995, Selected Papers*, volume 1153 of *Lecture Notes in Computer Science*. Springer, 1996.
9. E.K. Burke, Y. Bykov, J. Newall, and S. Petrovic. A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, 36(6):509–528, June 2004.
10. E.K. Burke, Y. Bykov, and S. Petrovic. A multicriteria approach to examination timetabling. 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 118–131. Springer-Verlag, Berlin Heidelberg New York, 2001.
11. 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.
12. E.K. Burke, D.G. Elliman, P.H. Ford, and R.F. Weare. Examination timetabling in British universities - a survey. In E.K. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling I (PATAT 1995, Edinburgh, Aug/Sept, selected papers)*, volume 1153 of *Lecture Notes in Computer Science*, pages 76–90. Springer-Verlag, Berlin Heidelberg New York, 1996.
13. E.K. Burke, D.G. Elliman, and R.F. Weare. A hybrid genetic algorithm for highly constrained timetabling problems. In *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA'95, Pittsburgh, USA, 15th-19th July 1995)*, pages 605–610, San Francisco, CA, USA, 1995. Morgan Kaufmann.
14. 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.
15. E.K. Burke, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper heuristic for timetabling problems. *European Journal of Operational Research*. Accepted to appear 2005.
16. E.K. Burke and J.P. Newall. Solving examination timetabling problems through adaption of heuristic orderings. *Annals of Operations Research*, 129:107–134, 2004.
17. E.K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140:266–280, 2002.
18. E.K. Burke, S. Petrovic, and R. Qu. Case based heuristic selection for timetabling problems. *Journal of Scheduling*, 2005. to appear.

19. M. Caramia, P. DellOlmo, and G.F. Italiano. New algorithms for examination timetabling. In S. Naher and D. Wagner, editors, *Algorithm Engineering 4th Int. Workshop, Proc. WAE 2000 (Saarbrucken, Germany, September)*, volume 1982 of *Lecture Notes in Computer Science*, pages 230–241. Springer-Verlag, Berlin Heidelberg New York, 2001.
20. M. W. Carter. A survey of practical applications of examination timetabling algorithms. *Operation Research*, 34(2):193–202, 1986.
21. 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.
22. M.W. Carter and G. Laporte. Recent development in practical examination timetabling. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling I (PATAT 1995, Edinburgh, Aug/Sept, selected papers)*, volume 1153 of *Lecture Notes in Computer Science*, pages 3–21. Springer-Verlag, Berlin Heidelberg New York, 1996.
23. S. Casey and J. Thompson. GRASPing the examination scheduling problem. In E.K. Burke and P.D. Causmaecker, editors, *Practice and Theory of Automated Timetabling IV (PATAT 2002, Gent Belgium, August, selected papers)*, volume 2740 of *Lecture Notes in Computer Science*, pages 232–244. Springer-Verlag, Berlin Heidelberg New York, 2003.
24. E. Cox and M. O'Hagen. *The Fuzzy Systems Handbook : A Practitioner's Guide to Building, Using and Maintaining Fuzzy Systems*. AP Professional, Cambridge, MA, 1998.
25. K.P. Dahal, C.J. Aldridge, and J.R. McDonald. Generator maintenance scheduling using a genetic algorithm with a fuzzy evaluation function. *Fuzzy Sets and System*, 102:21–29, 1999.
26. A.J. Davenport and E.P.K. Tsang. Solving constraint satisfaction sequencing problems by iterative repair. In *The First International Conference on The Practical Application of Constraint Technologies and Logic Programming (PACLP)*, pages 345–357, London, April 1999.
27. D. De Werra. An introduction to timetabling. *European Journal of Operational Research*, 19:151 162, 1985.
28. S. Deris, S. Omatu, H. Ohta, and P. Saad. Incorporating constraint propagation in Genetic Algorithm for university timetabling planning. *Engineering Applications of Artificial Intelligence*, 12:241–253, 1999.
29. L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. 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 104–117. Springer-Verlag, Berlin Heidelberg New York, 2001.
30. Christelle Guéret, Narendra Jussien, Patrice Boizumault, and Christian Prins. Building university timetables using constraint logic programming. In Burke and Ross [8], pages 130–145.
31. J. Li and R.S.K. Kwan. A fuzzy genetic algorithm for driver scheduling. *European Journal of Operational Research*, 147:334–344, 2003.
32. M. H. Lim, S. Rahardja, and B. H. Gwee. A GA paradigm for learning fuzzy rules. *Fuzzy Sets and Systems*, 82:177–186, 1996.
33. V. Lotfi and R. Cerveny. A final exam-scheduling package. *Journal of the Operational Research Society*, 42(3):205–216, 1991.

34. L.T.G. Merlot, N. Boland, B.D. Hughes, and P.J. Stuckey. A hybrid algorithm for examination timetabling problem. In E. K. Burke and P.D. Causmaecker, editors, *Practice and Theory of Automated Timetabling IV (PATAT 2002, Gent Belgium, August, selected papers)*, volume 2740 of *Lecture Notes in Computer Science*, pages 207–231. Springer-Verlag, Berlin Heidelberg New York, 2003.
35. S. Petrovic and E. K. Burke. *Ch. 45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter University Timetabling. to be published by CRC Press, April 15, 2004., 2004.
36. S. Petrovic and Y. Bykov. A multiobjective optimisation technique for exam timetabling based on trajectories. In E.K. Burke and P.D. Causmaecker, editors, *Practice and Theory of Automated Timetabling IV (PATAT 2002, Gent Belgium, August, selected papers)*, volume 2740 of *Lecture Notes in Computer Science*, pages 179–192. Springer-Verlag, Berlin Heidelberg New York, 2003.
37. S. Petrovic, V. Patel, and Y. Yang. University timetabling with fuzzy constraints. *Accepted for PATAT V selected volume*, 2005.
38. 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.
39. A. Schaerf. A survey of automated timetabling. *Artificial Intelligent Review*, 13:87–127, 1999.
40. D. Teodorovic and P. Lucic. A fuzzy set theory approach to the aircrew rostering problem. *Fuzzy Sets and Systems*, 95:261–271, 1998.
41. J.M. Thompson and K.A. Dowsland. A robust simulated annealing based examination timetabling system. *Computers Oper Res.*, 25(7/8):637–648, 1998.
42. Y. Yang and S. Petrovic. A novel similarity measure for heuristic selection in examination timetabling. *Accepted for PATAT V selected volume*, 2005.
43. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
44. H. J. Zimmerman. *Fuzzy Set Theory and Its Applications.* Kluwer Academic Publishers, 3rd edition, 1996.