A New Neural Network Based Construction Heuristic for the Examination Timetabling Problem

P.H.Corr, B.McCollum, M.A.J.McGreevy, P.McMullan

Queens University of Belfast, Northern Ireland
p.corr@qub.ac.uk

Abstract. This paper examines the application of neural networks as a construction heuristic for the examination timetabling problem. Building on the heuristic ordering technique, where events are ordered by decreasing scheduling difficulty, the neural network allows a novel dynamic, multi-criteria approach to be developed. The difficulty of each event to be scheduled is assessed on several characteristics, removing the dependence of an ordering based on a single heuristic. Furthermore, this technique allows the ordering to be reviewed and modified as each event is scheduled; a necessary step since the timetable and constraints are altered as events are placed. Our approach uses a Kohonen self organising neural network and is shown to have wide applicability. Results are presented for a range of examination timetabling problems using standard benchmark datasets.

Introduction

The examination timetabling problem is principally concerned with the scheduling of a number of events into a restricted number of time-periods, subject to a set of constraints [1]. Conflicting events which have students in common must not be scheduled into the same time-period. This essential condition is a *hard constraint*. Another such hard constraint is that seating capacity much not be exceeded in any time-period (the so called capacitated problem). A timetable satisfying all hard constraints with all events scheduled is a *feasible* solution. While there may be many feasible solutions to a given problem, the timetabler's task is to find a good quality solution, satisfying as many desirable conditions as possible. These desirable conditions, or *soft constraints*, often vary between data sets but typically involve placing events such that each student has a reasonable gap between any two exams. These conditions are rarely, if ever, completely satisfied, and often vary extensively between data sets. The degree to which the soft constraints are met - and hence the quality of the timetable - is measured by a cost function or penalty function. Therefore, the principal objective of the timetabler is to construct a timetable with an acceptably low, ideally optimum, penalty function.

Over the years various researchers have considered several methods of timetable construction. A detailed discussion of these techniques is given by Carter et al [2] and an excellent review of the field is provided by Burke et al [3].

Early approaches include techniques such as Graph Colouring and Constraint Programming [3]. Subsequently, metaheuristic approaches have been used to help improve the solution. Simulated annealing [4], Tabu search [5] and other techniques such a Great Deluge [6] and Memetic Algorithms [7] have all proved useful. In general metaheuristic approaches have performed well on benchmark problems though they are time consuming compared with graph colouring based approaches. Another avenue of research can be found in evolutionary and genetic approaches [8, 9]. Hybrid approaches involving combinations of heuristics and metaheuristics such as genetic algorithms and hill climbing techniques [10, 11] have produced good results on benchmark datasets. Other successful approaches taken include multi-criteria approaches [12], constraint based techniques [13], case based reasoning [14], and recently hyper-heuristics [15]. Recent papers note that on the whole, methods used to tackle the examination problem tend to use problem specific information and heuristics in particular.

Construction of a timetable is normally accomplished in two phases. An initial timetable is built using a *construction heuristic*; this initial timetable is then enhanced using an *improvement heuristic* [2]. Heuristic ordering is one approach to the construction phase. Events are ordered in decreasing order of perceived scheduling difficulty and then placed sequentially into available positions in the timetable within the conditions imposed by the hard constraints. If necessary, events are left unscheduled at this initial stage rather than violating the constraints with a high penalty being attributed to the incomplete timetable. Heuristic ordering may take either a static or a dynamic approach to construction. In the static approach a complete ordering is established at the start of the construction phase prior to scheduling and remains constant throughout the process. Events may be ordered using a single heuristic such as; largest degree, weighted degree, or exam size, all of which contribute to scheduling difficulty. In the dynamic approach, the order in which events are placed may change as the timetable is built. For example, if events are scheduled by the number of available slots remaining in the timetable (saturated degree) the placement order will change as events are placed in the timetable.

In recent years, improvement heuristics have received much more attention from researchers than the construction phase. As outlined above, a wide range of different techniques have been applied, such as: local search, simulated annealing, tabu-search, genetic algorithms, great deluge algorithm, and hybridised methods. The research reported here focuses on the construction phase and seeks to improve the quality of the initial timetable produced prior to the application of an improvement heuristic.

The well-established heuristic ordering approach has proved very effective and offers a firm foundation for further development. However, a feature of the approach is that the order in which events are scheduled is typically determined based on a single criteria. In reality, timetabling is a multi-criteria problem. For example, recent work by Asmuni et al [16] has shown how fuzzy techniques that incorporate characteristics from a number of established heuristic orderings can be used in establishing the initial order. Another approach explored in recent research has focused on heuristic adaptability, in which the scheduling order is adapted to suit the problem leading to an improvement in the quality of the initial timetable [17]. Heuristic adaptability also introduces a degree of generality to the system since the solution, as it develops, adapts to the environment. Each time an event is scheduled, the timetabling environment has been altered. In essence, an available position has been removed from the timetable, resource availability is reduced and a new and more difficult problem has been created. It is into this modified environment, a partially completed timetable, that the remaining events must be placed. As the timetable is generated, the scheduling order must be reviewed after every placement, and modified if necessary, to ensure that the most difficult exam is scheduled at each stage.

In this paper we propose a novel, neural network based multi-criteria methodology for dynamically modifying scheduling order during the construction of an initial examination timetable.

The Timetabling System

In order to investigate the effectiveness of the neural network as a multi-criteria adaptive scheduling component the construction of a feasible timetable is viewed conceptually as a two stage process. Firstly, the neural network ranks all remaining exams by perceived difficulty and chooses the most difficult to be scheduled next. A placement component then places the chosen exam in the timetable. Following placement, the process repeats – the remaining exams are ranked by difficulty and the most difficult is placed in the timetable – until all exams have been placed or the chosen exam cannot be placed. The quality of the final timetable is determined by a penalty function which measures the extent to which each student’s exams are spread across the available periods. Clearly, before the neural network can rank exams by difficulty it must be appropriately trained. Details of both network training and the penalty function used are given later.

The Neural Network

The system is based on a Kohonen self-organising neural network. As illustrated in figure 1, the network consists of two layers of processing elements or neurons; an input layer and a mapping layer. Neurons (prototypes) in the mapping layer are spatially arranged as a 2-D grid of five neurons by eight. The network employs an unsupervised learning algorithm in which it is not necessary to know in advance the 'correct' output for a given input. Once trained the organised network topology reflects the statistical regularities of the input data. Inputs (feature vectors) are projected onto the prototypes in the mapping layer such that the topology of the input space is preserved.

During training the Kohonen layer undergoes a self-organising process in which a two-dimensional map is produced representing the higher dimensional input space. An essential feature of the map produced is that it preserves the topology of the input space in that inputs which are ‘close together’ in input space are mapped to points ‘close together’ on the Kohonen layer. In effect, points on the Kohonen map represent prototypes, or cluster centres, for the feature vectors used during training. Thus, a feature vector input to the trained network will be represented by a single prototype on the mapping layer.

The choice of inputs and the extent to which the data used to train the network is an accurate and adequate representation of the problem are crucial to the success of the network. The purpose of the network is to determine the difficulty of each event such that the most difficult exam can be scheduled at each point during the construction phase. As such, it is important that the feature vectors that form the input to the network reflect those characteristics of the problem which contribute to scheduling difficulty. At any point during the construction of the timetable the concept of difficultly is highly dependent upon both the data set and the current state of the timetable. An examination may be perceived as 'easy' or 'difficult' based on characteristics such as, for example, the number of conflicts (degree), the number of students enrolled (exam size) or the number of available slots left (saturation degree). The input feature vector used in this work contains both static components, such as degree, weighted degree and exam size, and a dynamic component reporting the current state of the timetable.

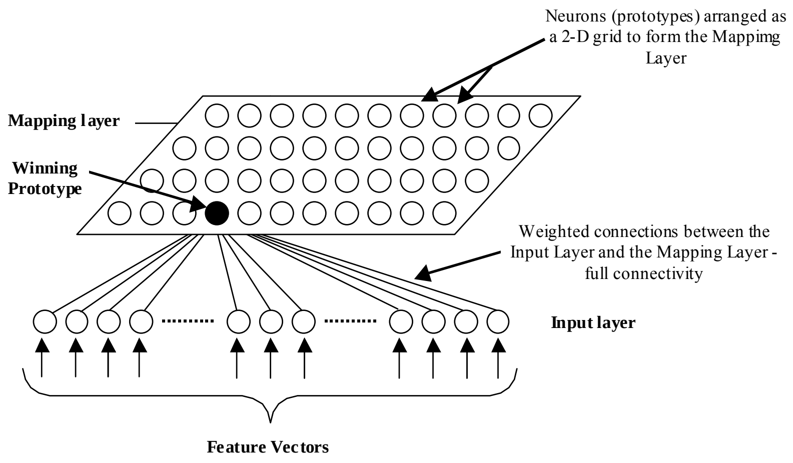
Neurons (prototypes) arranged as a 2-D grid to form the Mappimg Layer
Mapping layer
Winning Prototype
Weighted connections between the Input Layer and the Mapping Layer - full connectivity
Input layer
Feature Vectors

Fig. 1. Schematic of the Kohonen Network. Further information on the Kohonen network and neural networks in general may be found in Haykin [18]

The Training Process

Having defined the input feature vector it is necessary to establish a corpus of vectors which are representative of the particular timetabling problem and which may be used to train the network. In defining the training data it is essential that the corpus should capture the complexities of the scheduling problem. In this work the training data was generated by building a series of timetables using a random ordering heuristic. An event is chosen at random and an attempt made to schedule it using the placement system described above. Should placement succeed, the characteristics (of both the event and the current state of the timetable) are recorded and stored to the training corpus as a valid feature vector. Nothing is stored should placement fail. In this way a corpus of feature vectors is constructed containing examples of successful scheduling situations for the problem in hand. This approach allows each exam to be encountered in a variety of ordering positions; scheduled early, mid or late in the construction. The training corpus then contains a wide spread of possible scenarios that may arise during the course of building the timetable.

The Kohonen network is trained using a competitive learning algorithm. As input data is presented to the network the neurons on the mapping layer compete amongst each other for activation, resulting in a winning neuron. The weights associated with this winning neuron are then adjusted as dictated by the learning algorithm to align more closely with the input [18]. Through this process the neurons on the mapping layer become tuned to particular input patterns. The mapping layer is initially arranged as a two dimensional lattice of neurons as shown in figure 1. As the neurons become tuned, and patterns are identified, they arrange topologically so that their position is representative of the input characteristics.

As training progresses, a two-dimensional, topological preserving map of the input space is formed, made up of prototypes representing a range of inputs. The essence of the methodology then is to label each prototype represented in the Kohonen layer with a relative scheduling difficulty. Since the Kohonen network uses an unsupervised learning algorithm it is not necessary to know *a priori* how difficult it is to actually schedule the event represented by each of the feature vectors. However, it is a fundamental assumption in this work that feature vectors (events) that map to the same prototype on the Kohonen mapping layer, and are therefore ‘close together’ in input space, will have a similar scheduling difficulty.

Since all components in the feature vector are individually positively correlated with perceived difficulty it is possible to allocate a relative difficulty to each prototype based on a simple linear distance measure based on the normalised values of the prototype’s weighted inputs. Prototypes with the largest value represent the most difficulty exams to schedule; prototypes with the smallest value represent the easiest exams to place. This method presupposes that all features contribute equally to scheduling difficulty. In reality, some features are more influential than others and some can exhibit non-linear relationships with scheduling difficulty. The relative importance of the features and their non-linear relationships must be accounted for by a pre-processing stage prior to input to the network.

Construction of a timetable

Construction of a timetable can begin once the network has been trained and the prototypes labelled. Scheduling begins with an empty timetable. A feature vector is generated for each event to be placed and presented in turn to the trained network. The order of presentation is irrelevant. The network will map each input to one of the forty prototypes. Since each prototype is labelled with perceived relative difficulty, it is relatively straightforward to find all those events which are perceived to be most difficult at this stage. One of this group of events is chosen to be placed; at the moment the choice is random since each event in the group is assumed to be equally difficult. The chosen event is placed using the placement algorithm already described.

When an event is successfully placed, the resulting change in the timetable can increase the scheduling difficulty of events yet be timetabled. These changes are captured in the updated feature vectors which are again presented to the network as the first stage in choosing the next event to be scheduled. And so the process continues with the scheduling difficulty of the remaining events changing and adapting as more events are scheduled until either a feasible solution is obtained or the selected event cannot be placed.

Results

The proposition to use a neural network as a critical component in a multi-criteria adaptive scheduling system is entirely new. In order to accurately evaluate the contribution of the network to the overall scheduling task it is important that the experimental system is kept stable and simple. To that end, a two-phase iterative timetable construction system was developed as outlined above. Determining the order in which exams should be placed in the timetable is the responsibility of the first phase. In the second phase a placement system schedules exams in the chosen order. In all of the results reported below, timetables are produced by a construction heuristic only; improvement heuristics have not been used.

A relatively straightforward placement system is used in this work. When an exam is to be placed all remaining slots in the timetable which do not contravene a hard constraint are considered. The exam is placed in the slot which contributes least to the overall penalty. Should more than one timetable slot meet this criteria, the exam is placed randomly in one of these slots. Importantly, recursive backtracking is not used during timetable construction; once placed, an exam cannot be moved. This simplistic placement regime is necessary to ensure that the impact of the neural network component is clearly visible and, in the context of proving the neural network based approach, can be evaluated without masking by an unnecessarily complex placement algorithm.

Establishing Feasibility of the Method

The first experimental task was to verify that the neural network could act as a multi-criteria adaptive component in a scheduling system. Carter's collection of benchmark examination datasets was used for this purpose[1].

Each dataset was ordered by degree, weighted degree and exam size. These static orderings were passed to the placement system and exams scheduled in the established order. To enable a range of possible timetables, exams were placed randomly in the timetable with the only proviso being that placement did not break a hard constraint. It is practically impossible to generate a feasible timetable using such a simplistic placement mechanism. A number of runs were made for each ordering and the number of unplaced exams was recorded.

An eight-by-five Kohonen network was then constructed and trained for each dataset as described above. A number of timetables were constructed for each dataset. For each run, the number of unplaced exams was recorded. Again, the objective is not to construct a feasible timetable but to determine the contribution of the neural network.

Results of the experiment are shown in table 1.

[1] Benchmark datasets may be downloaded from ftp://ftp.mie.utoronto.ca/pub/carter/testprob

Table 1. Number of unplaced events in the construction of a timetable

Data Set	By Degree	By Size	By W.Degree	Best Result	Best Result using NN
CAR-F-92	15	17	19	15	1
CAR-S-91	22	21	15	15	5
EAR-F-83	8	13	14	8	3
HEC-S-92	3	8	6	3	0
KFU-S-93	13	12	8	8	3
LSE-F-91	7	5	7	5	0
STA-F-83	31	31	30	30	22
UTA-S-92	6	7	7	7	1
UTE-S-92	9	7	11	5	0
TRE-S-92	10	10	5	6	0
YOR-F-83	15	28	26	15	10

In all cases, use of the network component has reduced the number of unplaced events, sometimes significantly, when compared to the use of established event ordering heuristics based on a single criterion. Indeed, despite the highly restrictive placement algorithm, use of the neural network to order events for placement generated feasible timetables for four of the datasets. It was not possible to produce a feasible timetable for any of the datasets using traditional ordering heuristics.

Refining the Methodology

Having established the feasibility of the neural network based methodology the task now is to tune the method so that high quality feasible timetables can be produced for all datasets. For this it is necessary to introduce a penalty function so that the quality of the final timetable can be determined and results compared with those of other researchers.

The penalty function is motivated by the goal of spreading out each student's examination schedule. If two exams, *i* and *j*, scheduled for a particular student are *t* time slots apart, the weight is set to

$$w_t = 2^{5-t} \quad \text{where} \quad t \in \{1,2,3,4,5\} \qquad \textbf{(1)}$$

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 *T*. The goal is to minimise the following formulation:

$$\frac{1}{T}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} s_{ij}\, w_{|p_j - p_i|} \qquad \textbf{(2)}$$

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, subject to $1 \leq |p_j - p_i| \geq 5$

A number of refinements were made to the methodology. In particular, the network training regime was revised. Training data was originally generated using the method outline above in which events are chosen at random and a feature vector added to the training corpus if that event can be scheduled. This method takes no account of difficulty during training and can result in feature vectors representing intrinsically easy events (e.g. small events with low degree) scheduled early in the process being added to the training corpus. Similarly, it is inevitable that the training data will contain feature vectors representing intrinsically difficult events (e.g. large events with high degree) scheduled late in the process. Neither of these eventualities is likely to occur while the timetable is being constructed. Consequently, it can be argued that the network should not be trained with such unlikely exemplars.

New training data was generated for each of the datasets. In each case events were ordered by each of the established ordering heuristics before placement. A number of random orderings were retained in generating the training data. In addition, the placement system used was modified such that the chosen event was placed by least cost. For each dataset, a new trained network was developed and used in the construction of a timetable. The results are shown in table 2.

Table 2. Best cost achieved for each dataset using the revised training method. The ease with which timetables can be generated and an indication of time taken is also shown. The work was carried out using a standard desktop PC with AMD Athlon (tm) XP 1800+ 1.54GHz processor and 256MB RAM.

Data Set	**Proportion of feasible timetables**	**Best Cost**	**Average time to produce a timetable(sec)**	**Best Reported Results [16]**
CAR-F-92	0.002	6.2456	10.51	4.1
CAR-S-91	0.008	7.2129	17.28	4.65
EAR-F-83	0.0004	49.4436	2.03	29.3
HEC-S-92	0.095	13.57	0.60	9.2
KFU-S-93	0.008	19.9	3.17	13.5
LSE-F-91	0.0504	14.9938	2.25	9.6
STA-F-83	0.24275	159.2831	0.96	134.9
UTA-S-92	0.04	4.489	14.04	3.2
UTE-S-92	0.30436	31.25	1.31	24.4
TRE-S-92	0.0584	10.7791	3.01	8.3
YOR-F-83	0	1 unplaced	2.07	36.2

With the modified training regime feasible timetables were constructed for all datasets with the exception of YOR-F-83. For some datasets generating a feasible timetable is relatively straightforward, for others it is problematic. For example, 30% of attempts to generate a timetable for the UTE-S-92 dataset result in a feasible solution. With the exception of the YOR-F-83 dataset, EAR-F-83 was found to be most difficult with only 0.04% of attempts resulting in a feasible timetable.

The costs recorded for each dataset represent the value of the penalty function at the end of the construction phase only; improvement heuristics have not been used in this work. As such, our results are not directly comparable with other published results for these datasets, invariably recorded after an improvement phase. Our primary motivation in this work is to prove the effectiveness of the neural network as a multi-criteria, adaptive construction heuristic; not necessarily to obtain best results on these test datasets. With this proviso, best published results are also shown in table 2.

Application to of the Methodology to Capacitated Data

The datasets above do not contain constraints on the seating available in each period. Such uncapacitated data is useful for developing and proving the methodology but most real-world problems will have a limited set of rooms of varying capacities available. In reality, different institutions must satisfy a range of different constraints in generating an institution-wide timetable [19].

The neural network methodology has been applied to a rich dataset from the University of Nottingham – Nottingham 94. This dataset contains many constraints additional to those found in the benchmark datasets used above. Extra conditions include; specific period assignments, room assignments, timetabling events in a particular order, the requirement for some events to be placed in a room of their own and groups of events to be scheduled together in the same period/room. A neural network was trained, using the technique presented above, with all of these conditions treated as hard constraints. This presents a much more realistic, highly constrained scheduling problem than that posed by the benchmark datasets considered previously. Again, without the use of recursive backtracking or an improvement heuristic the neural network based system succeeded in constructing valid timetables but only by contravening the desirable condition that conflicting events should be scheduled at least one period apart.

It is important to note that the methodology used to order events for placement has not changed from that used with the benchmark datasets. The only component of the timetabling system which is, of necessity, tailored to the institution is the placement system; this component must be tuned such that all institution-specific hard constraints are respected when events are scheduled.

This is a significant result. Taken with the results on the benchmark datasets it illustrates that the neural network methodology has general applicability across a range of data and can be used successfully in real timetabling situations. In essence, the neural network provides a generalisation technique, designed to recognise patterns in the data that may be exploited in the generation of high quality timetables. This provides a high degree of generality resulting in a methodology which is largely independent of institution or dataset.

Conclusions and Further Work

The work presented here has shown the feasibility of using a neural network based methodology as a generally applicable, multi-criteria, adaptive, construction heuristic for the examination timetabling problem. Work is progressing on two fronts; firstly, to refine the method to improve both the proportion of feasible timetables produced and the quality of the final schedule and secondly, to evaluate the applicability of the methodology to related scheduling problems such as course timetabling for example.

References

1 Burke, E.K., Jackson, K.S., Kingston, J.H. and Weare R.F., *Automated Timetabling: The State of the Art,* The Computer Journal, Vol 40, No 9, pp 565-571, 1997.

2. Carter. M.W., and Laporte, G. *Recent developments in practical examination timetabling*. PATAT 1996, Lecture Notes in Computer Science Vol 1153 pp3-21, 1996.

3. Burke, E.K. and Petrovic, S. *Recent research directions in automated timetabling.* European Journal of Operational Research 140 pp 266-280, 2002.

4. Thompson, J.M. and Sowsland, K.A. *A Robust Simulated Annealing Based Examination Timetabling System.* Computers and Operations Research 25(7-8), pp 637-648, 1998.

5. White, G.M. and Xie B.S. *Examination Timetables and Tabu Search with Longer-Term Memory.* PATAT 2001, Lecture Notes in Computer Science Vol 2079, pp 85-103, 2001

6. Burke, E.K. and Newall, J.P. *Enhancing Timetable Solutions with Local Search Methods*. PATAT 2002, Lecture Notes in Computer Science Vol 2740, pp. 195-206, 2003.

7. Burke, E.K., Newall, J.P. and Weare, R.F. *A Memetic Algorithm for University Exam Timetabling.* PATAT 1996, Lecture Notes in Computer Science Vol 1153, pp 241-250, 1996

8. Corne, D., Fang, H.L. and Mellish, C. *Solving the Modular Exam Scheduling Problem with Genetic Algorithms.* In Proc of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert systems, pp 370-373, 1993.

9. Ross, P., Corne, D. and Fang, H.L. *Improving Evolutionary Timetabling and Delta Evaluation and Directed Mutation.* In Y. Davidor, H. P. Schwefel, and R.Manner (eds.), Parallel Program Solving in Nature, Vol. III, pp. 565-566. Berlin: Springer. 1994

10. Burke, E.K., Newall, J.P. and Weare, R.F. *Initialisation Strategies and Diversity in Evolutionary Timetabling.* Evolutionary Computation 6(1), pp 81-103, 1998

11. Burke, E.K. and Newall, J.P. *A Multi-Stage Evolutionary Algorithm for the Timetabling Problem.* IEEE Transactions on Evolutionary Computation 3(1), pp 63-74, 1999.

12. Burke, E.K., Bykov, Y and Petrovic, S. *A Multi-Criteria Approach to Examination Timetabling*, PATAT 1996, Lecture Notes in Computer Science Vol 1153, pp 118-131, 1996.

13. Carter, M.W. and Johnson, D.G. *Extended Clique Initialisation in Examination Timetabling.* Journal of the Operational Research Society 52(5), pp 538-544, 2001.

14. Burke, E.K., Petrovic, S. and Qu. R. *Case-Based Heuristic Selection for examination Timetabling.* Proceedings of the Seal'02 conference, Singapore, pp 277-281, 2002.

15. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S. and Qu, R., *A Graph-Based Hyper-Heuristic for Timetabling Problems*, accepted for publication in the European Journal of Operational Research, to appear 2006.

16. Asmuni, H., Burke, E.K., Garibaldi, J.M. and McCollum, B. *Fuzzy Multiple Heuristic Ordering for Examination Timetabling.* PATAT 2004, Lecture Notes in Computer Science Vol 3616, pp 334-353, 2004.

17. Burke, E.K. and Newall, J.P. *Solving Examination Timetabling Problems through the Adaption of Heuristic Orderings*. Annals of Operational Research 129, pp 107-134, 2004.

18. Haykin, S.S. and Saher S. *Neural Networks: a comprehensive foundation. - 2nd ed..* - Englewood Cliffs, N.J. : Prentice Hall, 1998 . -ISBN 0132733501

19. Burke, E.K., Elliman, D.G., Ford, P. and Weare, R.F. *Examination Timetabling in British Universities: A survey.* PATAT 1996, Lecture Notes in Computer Science Vol 1153 pp76-90. 1996.