General Employee Timetabling

Efficient Generation of Cyclic Schedules.

Rafael Hope

Department of Computer Science,
Hedmark University College, 2450 Rena, Norway
rafaelh@hihm.no

Abstract.

Cyclic schedules, which are commonly used in the organizing of Norwegian healthcare service institutions, are daily rotating with three main shifts (day, afternoon and night) or days-off. Cyclic schedules are arranged weekly periodically. A period consists of a certain number of weeks, w_1, w_2 ,…, w_n, following each other with w_1 following w_n.
Generating high-quality schedules can be done stepwise by first registering the basic inputs, followed by generating the basic schedule and finishing up with altering and tuning the basic schedule until it is acceptable for all parties involved. The quality of a schedule depends on how well it matches the several rules and constraints imposed by different interests such as legal requirements, social welfare demands, local adaptations, agreements with workers associations. The construction of schedules for personnel in healthcare institutions involves at least 35 constraints and therefore it is hard to solve.
The basic inputs are workforce-matrices showing the weekly working hours for each employee, workload-matrices showing for each day in a period, how many employees are needed on each shift. The working hours of the most common tree main shifts are needed for the comparison of these two matrices. Team-scheduling are often used, some time in combination with self-scheduling. For each team, the basic schedule is constructed on the corresponding basic inputs. At this stage the employees included are each treated as being available for a time equal to the average for all employees in the current team, which means that they all have the same basic schedule with different offsets. The basic schedule is then individualized according to the difference in available working hours for each employee in the team by swapping of shifts and off-days. Finally the schedule must be tuned by introducing new shifts to match the exact available working hours for each employee.
The algorithm presented in this paper generates basic schedules interactively with the user, the human decision maker. The user can alter the basic schedule in any way she wants at the risk of breaking constraints.
The algorithm is based on simulation of the traditional manual way of constructing shifts schedules. It has been implemented object-oriented based on analyses using the techniques from Unified Modeling Language (UML), designing several Use-Cases and corresponding classes. The main Use-Cases are obtained from a commonly recommended method for the manual construction of schedules: (1) place the working weekends and the off-weekends, (2) place the remaining night-shifts, (3) place the remaining off-days, (4) place the remaining evening-shifts and (5) place the remaining day-shifts. Each Use-Case has been thoroughly analyzed and the constraints involved have been identified. Several classes have been constructed with methods implemented with these constraints. The algorithm uses these classes.
Generated shift schedules from real-world problems and from examples found in literature have been tested against the constraints criteria and found to be of acceptable quality. The algorithm has been tested on generated basic inputs and has so far proved to be of order $O(n^2)$, where $n <= 10000$ is the number of weeks in the generated schedule. The algorithm has been implemented in a shift scheduling system, which has not yet been commercialized.

Keywords: cyclic schedule, Norwegian healthcare service, constraints, algorithm, simulation, Use-Case.

Practical Timetabling

Agent Technology for Timetabling

P. De Causmaecker, P. Demeester, Y. Lu and G. Vanden Berghe

KaHo Sint-Lieven
Information Technology
Gebr. Desmetstraat 1
9000 Gent
Belgium
{patdc, peterdm, yang.lu, greetvb}@kahosl.be

Abstract. Agent technology promises possibilities for the design of distributed systems, the design of open systems and the reduction of information overload for the users. Timetabling applications must operate in heterogeneous environments at large organisations where users cannot be bothered with all the details influencing their work schedules. It is a domain where, at least at the system level, many separately developed systems must interact. In this paper, we investigate how agent technology can play a part in timetabling applications.
Timetabling problems are distributed by nature but performance also contributes to the distributiveness. The performance issues are related to the complexity of the timetabling problem and the need to respond within a limited time. The problem becomes distributed when departments of a different speciality are interconnected and have to gear their timetables to one another.
Timetabling systems will often function as components in larger settings. As such, they have to interact with a diversity of existing solutions and should thus best be build as open systems. Data are stored in databases; business systems may contain relevant information. The results of the timetabling operations must be fed back to these information providers because these systems ultimately control the operations.
Timetabling systems need as much detail as possible while they must report on the resulting timetable to users with a limited scope. Both channels must be guarded against information overload. The user should not be prompted to specify information that is not available and he should not be confronted with information he cannot interpret. On the other hand the system should try to understand what the user really wants, present a suitable timetable and provide comments that are easily accessible to that user.
The three mentioned key issues in timetabling systems are competences for which software agents are particularly appropriate. We review the available technology for distribution, interconnection and information overload and report on some experiments.

Keywords: timetabling, agent technology

Overview

Although the terminology was introduced earlier, agents were pioneered from 1994 on as a possible solution for the information overload problem [29]. Agent technology has become an important domain of research in Artificial Intelligence. The agents build profiles of their users and gather knowledge about their domain of application in an autonomous way. Examples are the shop agent modelling customers [1], the user interface agent searching for related files while following the reader of documents [27], the browsing assistant [44], an agent assisting in selling and buying goods [9], etc.

It was soon realised that agents could be used to model autonomous components in distributed systems. Many researchers tried to pinpoint their essential properties [4, 19, 23, 24, 30, 35, 37, 45]. Prominent characteristics are autonomy, interactivity, adaptivity… The agent model allows describing a system as a society of agents working together. The location where the agent is operating becomes immaterial, and a middleware environment can tackle aspects of distribution over physical computers [36]. Load balancing is achieved in a decentralised way by collaboration in the agent community [8, 20, 21].

Apart from using distribution in order to reach better performance when more CPUs are available, agents enable the description of solutions for essentially distributed problems. The distribution may be caused by the geographic separation between the involved departments. However, the separation may just as well stem from the business the departments are involved in [5]. On the other hand, it is not immediately clear how to develop algorithms based on agent technology. One advantage of such algorithms would certainly be that distributed operation requires no modification. There are examples of distributed algorithms for distributed constraint programming [32, 39] and for vehicle routing [5, 39, 40]. For reviews we refer to [28,42, 46]. Further examples and models are in [13,38,41, 43].

Today, a firm technological basis for agent technology is available. Several organisations defined standards (FIPA [18]) and developed platforms and tools (JADE, ZEUS, Fipa-OS, JACK,…). Connectivity with legacy applications enhances the practical usability [3, 15, 16, 17, 26, 31, 33, 34]. The communication between agents and other systems is supported by the development of domain frameworks [25] of which domain ontology or a communication ontology is an essential part. Compliance to such framework ensures that agents will be able to interact with other compliant applications. Several tools are available for ontology development [2, 14, 22].

The central part of a typical timetabling system is the computational algorithm. Timetabling problems are recognised to be NP complete, even in their simplest form [10]. This has fostered an intensive search for efficient heuristics. A lot of research has been carried out in order to find and classify the most relevant constraints [6], but still, solving a specific timetabling problem is a highly interactive process with diverse and geographically separated partners involved [7]. In the daily practice of a developer/operator of a timetabling system, the interaction with these involved parties is found to be an important issue.

The application of agent technology for timetabling problems can be investigated at all the aforementioned levels. Timetabling applications function in a distributed environment. Information retrieval must happen through interaction with legacy soft-

ware or with human users [11,12]. The availability of computing power in the network introduces possibilities for performance boosting. At this level we will need distributed algorithms. Openness and communication with foreign systems, and the exchangeability of timetabling information can be shown to be relevant for timetabling systems. Furthermore, we investigate possibilities for limitation of information overload and information gathering at the user interface level. We did two prototype developments to test and illustrate part of what was mentioned above.

References

[1] http://www.amazon.com/

[2] S. Bechhofer, I. Horrocks, C. Goble, R. Stevens: OilEd: A Reason-able Ontology Editor for the Semantic Web, Proceedings of KI, Joint German/Austrian conference on Artificial Intelligence, Vienna, Springer Verlag LNAI Vol. 2174, p 396-408, 2001

[3] F. Bellifemine, A. Poggi, G. Rimassa: JADE – A FIPA-compliant agent framework, CSELT internal technical report 1999

[4] J. C. Brustoloni: Autonomous Agents: Characterization and Requirements, Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University, 1991

[5] H-J. Bürckert, K. Fischer, G. Vierke: Holonic Transport Scheduling with TELETRUCK, Applied Artificial Intelligence 14(7), 697-726, 2000

[6] E. Burke, D. Elliman, P. Ford, R. Weare: Examination timetabling in British universities: A survey, Selected papers from the first international conference on the Practice and Theory of Automated Timetabling, 76-90, 1995

[7] M. W. Carter: A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo, Selected papers from the third international conference on the Practice and Theory of Automated Timetabling, Springer 2000, 64-82.

[8] A. Chavez, A. G. Moukas, P. Maes: Challenger: A Multiagent System for Distributed Resource Allocation, Proceedings of the International Conference on Autonomous Agents, Marina Del Ray, California, 1997

[9] A. Chavez, P. Maes: Kasbah: An Agent Marketplace for Buying and Selling Goods, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, April 1996

[10] T. B. Cooper, J. H. Kingston: The Complexity of Timetable Construction Problems, Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling, 1995

[11] P. De Causmaecker, P. Demeester, Ph. De Pauw-Waterschoot and G. Vanden Berghe: Agent Assistance in Lab Session Planning, In Workshop notes Agents in Industry at The Fourth International Conference on Autonomous Agents 2000, Barcelona.

[12] P. De Causmaecker, P. Demeester, Ph. De Pauw-Waterschoot, G. Vanden Berghe: Agent Assistance in a Lab Session Planning, SCI'2000/ISAS'2000, Orlando 2000

[13] R. Diekmann, J. Simon: Problem Independent Distributed Simulated Annealing and its Applications, Applied Simulated Annealing, R.V.V. Vidal, Lecture Notes in Economics and Mathematical System, vol. 396, 17-44, 1993

[14] A. Farquhar, R. Fikes, J. Rice: The Ontolingua Server: a Tool for Collaborative Ontology Construction, 1996, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/farquhar/farquhar.html

[15] T. Finin, Y. Labrou, J. Mayfield: KQML as an agent communication language, In Software agents, Bradshaw JM (ed.), MIT Press, 291-316, 1997.

[16] FIPA, Specification, Version 2. Part 2. Agent Communication Language. Foundation for Intelligent Physical Agents, 1997

[17] P.D. O'Brien and R.Nicol: FIPA: toward a standard for Intelligent Agents, BT Technical Journal, Vol.16, July 1998.

[18] FIPA, http://www.fipa.org

[19] S. Franklin, A. Graesser: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996

[20] A.P. Greenaway, and G.T. McKee: A Practical Demonstration of the Effect of Malicious Mobile Agents on CPU Load Balancing, ECOOP'99, the 13th European Conference on Object-Oriented Programming, Lisbon, Portugal 14-18 June, 1999

[21] M. Grivas, S.J. Turner: Agent Technology in Load Balancing for Network Applications, Proceedings of the Workshop on Intelligent Agents on the Internet and Web in 4th World Congress on Expert Systems, 1998

[22] W. E. Grosso, H. Eriksson, R. W. Fergerson, S. W. Tu, M. M. Musen: Knowledge modeling at the millennium", the design and evolution of Protege-2000, Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99), Banff, Canada,October, 1999

[23] B. Hayes-Roth: An Architecture for Adaptive Intelligent Systems, Artificial Intelligence: Special Issue on Agents and Interactivity, 72, 329-365, 1995

[24] The IBM Agent, http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm

[25] R. E. Johnson: Framework= (Component + Patterns), Communication of the ACM, Vol. 40, No. 10, 39-42, 1997

[26] Knowledge Query and Manipulation Language, http://www.cs.umbc.edu/kqml/

[27] Letizia, MIT site http://www.mit.edu

[28] N. Lynch: Distributed Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1996

[29] P. Maes: Agents that Reduce Work and Information Overload, Communications of the ACM, Vol. 37, No.7, 31-40, 146, ACM Press, July 1994

[30] P. Maes: Artificial Life Meets Entertainment: Life like Autonomous Agents, Communications of the ACM, 38, 11, 108-114, 1995

[31] OMG's Mobile Agent System Interoperability Facility, http://www.fokus.gmd.de/research/cc/ecco/masif/

[32] A. Meisels, I. Razgon: Distributed Forward Checking with Dynamic Ordering, Proceedings workshop on Collective Search Algorithms, 2001

[33] H. S. Nwana, D. T. Ndumu, L. C. Lee, J. C. Collis: ZEUS: A Toolkit for Building Distributed Multi-Agent Systems, Applied Artificial Intelligence Journal 13 (1/2), 129-185, 1999

[34] OMG, http://www.omg.org

[35] OMG Agent Working Group, Agent Technology, green paper, OMG Document ec/2000-08-01 Version 1.0, 2000

[36] B. Robben: Language technology and metalevel architectures for distributed objects, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 1999

[37] S. J. Russell, and P. Norvig: Artificial Intelligence: A Modern Approach, Prentice Hall, 1995

[38] G. Solotorevsky, E. Gudes, A. Meisels: Distributed Constraint Satisfaction Problems - A Model and Application, Preprint, October 1997

[39] G. Solotorevsky, E. Gudes: Solving a Real-life Nurses Timetabling and Transportation problem using Distributed CSP techniques, AAAI97 Workshop on Constraints and Agents, 1997

[40] R. Slootmaekers, H. Van Wulpen, W. Joosen: Modelling Genetic Search Agents Using a Concurrent Object-Oriented Language, Proceedings of HPCN Europe 1998, LNCS 1401, Amsterdam 1998

[41] T. Starkweather, D. Whitley, K. Mathias: Optimization Using Distributed Genetic Algorithms, 176, PPSN Conference 1, Oct 1990.

[42] G. Tel: Introduction to Distributed Algorithms, Cambridge University Press, 2000

[43] M. Toulouse, T. Crainic, B. Sans: An Experimental Study of Systemic Behavior of Cooperative Search Algorithms, Meta-Heuristics 1997: Theory and Applications, Kluwer Academic Publishers, 373-392, 1999

[44] A. Wexelblat, P. Maes: Using History to Assist Information Browsing, RIAO'97: Computer-Assisted Information Retrieval on the Internet, Montreal, 1997

[45] M. Wooldridge, R. Jennings: Agent Theories, Architectures, and Languages: a Survey, Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1-22, 1995

[46] M. Yokoo, Algorithms for Distributed Constraint Satisfaction: A Review, Autonomous Agents and Multi-Agent Systems, Vol.3, No.2, 189-212, 2000

Possible Models for Timetabling at Tertiary Institutions

T Nepal[1], M I Ally[2]

Department of Computer Studies
ML Sultan Technikon, PO Box 1334, Durban
[1]nepal@wpo.mlsultan.ac.za, [2]allyi@wpo.mlsultan.ac.za

Abstract: This paper deals with the timetabling problem at tertiary institutions. An empirical research was carried out dealing with timetabling at tertiary institutions in Southern Africa. Questionnaires dealing with, inter alia, student population, number of groups, constraints and the method used for timetabling, were sent to the tertiary institutions. There was a response rate of over 80%. Mad it clear that the timetabling problem is still prevalent. An analysis of the questionnaires revealed some very interesting results, namely, that there were various methods used for timetabling and that no institution that responded is using a fully automated system for timetabling. The analysis indicates that a great deal of time and effort is involved in the process, up to 200 person hours in some institutions. It is clear that the present methods do not address all the timetabling problems. This paper highlights the problems that are still prevalent today and presents three possible models for tertiary timetabling. Each model is examined for its uniqueness, range of applicability and areas of success. It could be loading at the limitations of the models.

Keywords: Timetabling, scheduling.

Computing Review Category: F2.2, H4.1

1. Empirical Research

1.1 Introduction

Questionnaires were designed and sent to timetable co-ordinators of the different tertiary institutions. A total of 46 questionnaires were despatched. There was a return rate of over 80% in the case of Universities and Technikons. A Technikon is a South African term associated with the universal term Polytechs. Thirty-six institutions indicated that the timetable was constructed manually; fifteen used a mixed approach using either the trial and error method or the so-called 'blocking system'. In the mixed approach the computer mainly did the allocation of venues. An analysis of the responses to the questionnaire is presented in [1].

1.2 Constraints

1.2.1 The Rooms Problem

Most institutions do not have an unlimited supply of venues, and so these resources and their use must be considered as a constraint in practical solutions. In addition certain specialist rooms e.g. Laboratories, may have to be treated as special cases where a particular group and lecturer may require a specific room. It seems however that the more general rooms problem of having a wide range of venue *sizes* to match with another range of group *sizes* has not been seriously addressed.

1.2.2 Pre-allocations or Pre-assignments

Here group g_i is obliged to meet lecturer l_j at some fixed time p_k. This could arise, for example, when it is desired that students meet a professional from industry for some set period of each day. In the case of inter-departmental servicing, it may be necessary to 'fix' certain times. Servicing here refers to a department that on the basis of its expertise or capacity assists another by providing or all tuition for a course, for example English as a second language.

1.2.3 Unavailabilities.

These occur when a group of students or a lecturer is not available for a particular period eg. Senior students maybe guaranteed certain periods off for vocational training for example trainee nurses and teachers. Similarly other resources, for instance a lecturer or a venue may not be available for some or other reason.

1.2.4 Minimising versus equitation.

There are two possible goals beyond merely creating a feasible timetable, which may be desirable to an institution. The question of minimising involves attempting to minimise some resource (eg. number of rooms used for institutions where venues have a certain cost - such as rental - or alternately minimising the total number of periods needed). In equitation the requirement is for some form of `even spread' - say that group gi needs to meet lecturer lj for 5 periods of Economics classes in a week. To generate a timetable which puts all five periods on Monday would not be acceptable to most institutions. Similarly such equitation is often desired across days, so that the load of staff and students is more or less the same on each day of the week rather than having very heavy (`clustered') days intermingled with very light ('sparse') days, although the former may be requested in some instances.

1.2.5 Preferences

Unlike unavailabilities, where there is a rule involved (lecturer X will *not* be able to give lectures in 4th period on Tuesdays), preferences relate to `nice-to-haves'. Such preferences can arise from personal reasons such as the lecturer who wishes to be free to attend to children at particular times, or on didactic grounds, such as preferences regarding which subjects should follow others in a timetable. Certainly preferences are not vital considerations in creating *feasible* timetables, but approaches which do take account of these will lead to greater user satisfaction and community acceptance.

1.2.6 Miscellaneous constraints.

A number of other considerations may be important, largely functions of the particular institution involved. There is a vast number of requirements specific to institutions and courses and that generalised solutions must of necessity be adapted to individual circumstances. One particularly common constraint here is the need for double and triple periods in a particular subject (often those involved in laboratory work). `Team teaching' (more than one lecturer to a class) and class-splitting (into, say, small tutorial groups) in particular add a great deal of complexity to the problem.

The next section presents possible models for tertiary timetabling. These models have been developed from the approaches currently being used by tertiary institutions. It also presents a heuristic model for automated timetabling.

2. Possible Models for Constructing Tertiary Timetabling

2.1 Introduction

The 'block' and the 'generalised' models are heuristic models for manual timetabling. The block model derives from the block approach to timetabling, while the generalised model derives from the grid, colour card and magnetic board approaches [1]. This section goes on to apply the heuristics developed in the generalised model to an automated model suitable for computer implementation.

2.2 The Block Timetabling Model

The heuristics derived from this model may be presented in a 5 step process:

STEP 1: Gather data and determine user requirements, inputs, and outputs.

The following are typical of the inputs in setting up a block timetable: subjects, group(s), number of lecture periods, tutorial periods, and practical periods per subject, venues and their capacities, and the total number of subjects.

The outputs include a table reflecting the different blocks and the subjects in each block, a table reflecting the periods for each block, and finally a table showing the venues for each subject.

STEP 2: Determine the number of blocks and the number of subjects per block.

The number of blocks is dependent on the total number of subjects and the venues that are available. If there are twenty venues available, then only twenty subjects can be scheduled in any one period. The maximum number of subjects per block in this case will be twenty. Twenty divided into the total number of subjects will determine the number of blocks.

STEP 3: Allocate the subjects to the different blocks based on heuristics supplied by the departments.

The main rule followed in allocating subjects is to group possible non-clashing subjects into blocks as in table 1, so that students are unlikely to choose more than one subject from a block. This blocking, however, leads to one of the major criticisms of this system because invariably some students do want to choose more than one subject but are prevented because of a clash. In order to avoid a clash a subject may appear in more than one block.

STEP 4: Allocate blocks to specific periods based mainly on equitation and other requirements submitted by departments.

It is necessary to ensure that lectures are spread 'evenly' throughout the week. For example, if there are 5 periods allocated to a subject, then, as far as possible allocate one period per day. Do not allocate all the periods in the mornings or all the periods in the afternoon. Some of the periods should be in the mornings while others may in the afternoons (table 2).

STEP 5: Output the results.

Table 1 and Table 2 can now be used as the final timetable. Subject block is determined from table 1 and the scheduled times from table 2.

2.3 A Generalised Model For Timetabling

This section formalises the approach used in the grid, magnetic board, and colour card system. The heuristics are presented in the same format as above.

STEP 1: Gather data and determine user requirements, inputs, and outputs.

The first step before any timetabling commences is to determine whether timetable is for a department, faculty, or the institution as a whole. This will determine the amount of inter-departmental and inter-faculty co-ordination that will be required. The next step is to obtain the necessary information from the relevant departments.

The information required from the academic departments includes the diplomas or degrees, groups, subjects, levels, number of students (estimated), lecturers, and number of lectures, tutorials, and practicals. These should also include any special requirements such as special venues pre-assignments, preferences. A sample request form is given in table 3.

The details regarding the venues available need to be requested from the relevant administrative department. The venue details include the room number, type, capacity, and possibly its location. A sample venue availability request sheet is given in table 4. These have to be sent to the relevant department for the necessary venue information. Similar forms can be for specialist venues.

STEP 2: Find the best case allocations based on heuristics.

The following heuristics are used for timetabling: unavailabilities are allocated first, then pare-allocations, then combined groups, then triple periods, then double periods, then split groups, then single periods, and finally preferences if they can be accommodated.

The unavailabilities are entered in respective tables 5, 6, and 7. This is followed by the preallocation entries. The combined groups are then examined and a search is carried out for the most appropriate slot. As appropriate slots are found, entries are made in the respective tables. This process continues until all the allocations are made.

STEP 3: If OK then goto 4
Else make adjustment to initial entries or requirements and retry.

If all the allocations are not made then it will be necessary to make adjustments to some of the previous entries because of possible clashes and other problems (eg. unavailability of lecturer or venue). Clashes can be reduced by examining the group, lecturer, and room free slots and making changes in order to accommodate the clashes. Assume that a particular group is scheduled to have an Accounting 1 lecture on Monday, period 5. Information Systems I taken by J Bloggs is still to be scheduled for this group. It's discovered that the group is free during period 7 and J Bloggs is free during period 5. There are two options: either move Accounting 1 to period 7 if possible or make J Bloggs free during period 7 by making changes to his timetable. This process continues until a feasible timetable is achieved.

STEP 4: **Output the results**

The completed table 5 is the composite personal timetable, table 6 the composite group timetable, and table 7 the room loading schedule.

2.4 An Automated Heuristic Model

2.4.1 Introduction

Manual timetabling is a time consuming task. It is difficult to cater for preferences. It is even more difficult to make changes to the timetable once it is finalised. There is therefore a need for an automated timetabling system. This section presents an automated model for timetabling and a prototype of the model is currently being constructed.

2.4.2 Algorithm

```
Gather data
Input user requirements
Equitation: = 100%
REPEAT
        Preferences := 100%
        While (NOT ACCEPTABLE-SOLUTION-FOUND) DO BEGIN
                While (NOT ALL-ORDERINGS-TRIED) AND (NOT-ACCEPTABLE SOLUTION-FOUND)
                DO
                        TRY-AN-ORDERING;
                IF (NOT ACCEPTABLE-SOLUTION-FOUND) THEN
                        PREFERENCES := PREFERENCES - 10
        END;
        IF (NOT ACCEPTABLE-SOLUTION-FOUND) THEN
                        EQUITATION: = EQUITATION -10
UNTIL (ACCEPTABLE-SOLUTION-FOUND OR (EQUITATION < 0))
```

2.4.3 User Requirements

The procedure for data collection is the same as that for the generalised model. The requirements can be divided into necessary and desirable ones, also described as hard and soft constraints. The hard constraints must be satisfied while the soft constraints are negotiable and thus desirable. The hard constraints determine the feasible solutions and the soft constraints are criteria for optimization [3].

The goal of the automated timetabler should be to create a schedule for the next cycle, viz. term, semester, or year. A sub goal would be to optimize the use of time and room resources thereby providing community satisfaction [2].

The following user requirements were derived from a literature study [2, 3, 4, 5, 6] and interviews with timetabling authorities from five different institutions viz. M L Sultan Technikon, Port Elizabeth Technikon, Rhodes University, Pretoria University and Natal University:

1. The data should be complete and there should be *total* correspondence of the timetable to the curriculum. The curriculum considered here are the set of lectures to be scheduled with specified multiplicities per whole planning period (usually a week, fortnight, or term). The curriculum items can be defined as follows: <group, subject, venue, lecturer, multiplicity>.

Pre-assignments of some lectures to specified timeslots may be needed according to the requirements determined for individual constraints, for example a computer laboratory may be available at a particular day and period.

2. The sequencing requirements should be clearly stated. In some cases, there should be continuity of lectures for all students meaning the absence of gaps or "windows" i.e., when a group or lecturer has no lectures during some timeslot, and has lectures before and after this timeslot on a particular day. In other cases, the non-continuity of lectures may be preferred i.e., the preference to have 'some' gaps instead of a straight series of lectures.

The sequencing requirements should reflect the logical structure of courses as defined by the curriculum for lectures on different subjects, or lessons of different types. For example, in food technology, the theoretical lectures should come before their practical application in the laboratory.

3. The maximum number of lectures per day for each lecturer and group should be specified. The distribution of lectures should also be specified. For example, the lectures may be distributed uniformly over the week i.e., not all mornings or not all afternoons but some sort of even spread. Attempts may be made to distribute subjects according to didactical, psychological and ergonomic requirements. Where possible alternate 'difficult' and relatively 'easy' subjects, and schedule laboratory work in the afternoons.

4. The user must specify which groups must be divided into subgroups for tutorials and those that must be combined for common lectures.

2.4.4 Processing

STEP 1: Determine resource usage in terms of rooms and lecturers.

This model assumes that venue allocation is an integral part of the timetabling process. It is obvious that the timetabling task becomes easier if excess capacity is available and so venue allocation can be treated separately.

The first rule is to determine resource usage in terms of venues and lecturers and variance from norms in respect of groups, lecturers, and number of periods allocated.

If the venue utilization is more than 100% i.e., the number of periods to be allocated is more than the venue space available, then it is not possible to continue with the timetable process. Heads of departments should then be requested to reduce, where possible, the number of periods allocated for subjects. Exception reports should also reflect whether lecturer or groups are loaded more than the norm in terms of number of periods.

STEP 2: Find allocations based on heuristics.

A basic heuristic approach may be as follows: unavailabilities are allocated first, followed by pare-allocations, then combined groups, then triple periods, then double periods, then split groups, then single periods. Attempts to allocate at maximum preference levels first.

In the above process, the sequencing, distribution, and equitation criteria should be taken into account. Timeslots may be chosen at random within the criteria already established (eg., not all afternoon or not all morning periods).

STEP 3: If OK then goto 4
Try an ordering and retry.

Factors such as equitation, sequencing, distribution, other preferences and requirements are negotiable and begin with a preference level of 100% initially. Preferences in each category have to be prioritized. The preference with least priority should be adjusted first and its preference level should be decreased by 10% and the program rerun

STEP 4: If OK then goto 7
Else preferences - 10 goto 2.

If a suitable timetable is not achieved then the preference level is decreased by a further 10% and rerun until the preference level drops to 0%.

STEP 5: If OK then goto 7
Else equitation - 10 goto .

The equitation with the next lower priority is then considered and the process repeated. If a suitable timetable is not achieved then the equitation level is decreased by a further 10% and rerun.

STEP 6: If OK then goto 7
Else goto 2.

This process continues until all preferences in a particular category are considered. The preferences in the lowest category are then considered and the process repeated.

STEP 7: Output the results.

If a solution has not been found, and if all orderings have not been tried then change the orderings i.e., if the Computer Skills 1 triple was considered before the Chemistry triple then change the orderings and examine the results. After the orderings for the triples have been considered then change the orderings of the doubles and so on. The order of the heuristics in step 2.2 does not change. Instead the subject order within heuristics stops (eg, order of triples) changes.

2.4.5 Output Requirements

1. The output should contain no conflicts, thus there should be a one to one correspondence in a curriculum (group, subject, venue, lecturer)

2. The assignment of lectures to venues that are unacceptable should be avoided and the allocation of lectures to groups or lecturers when they are unavailable should also be avoided.

3. Where a campus is distributed over a wide geographical area, it is important to minimise travel time..

4. The outputs from the system should include the lecturers' personal timetables, the group timetables and the room loading schedules.

2.5 Conclusion

Three models have been proposed for timetabling, viz., the block model, the generalised model, and the automated model. The block model takes about 100 - 200 person hours to develop and is used mainly by universities. Once it is finalised the same timetable is used each year. The major criticism against this model is that it is inflexible and that resources are not fully utilised. A student cannot choose more than one subject from a block nor can a lecturer lecture in more than one subject from a block.

The generalised model is used mainly by Technikons. This method uses a trial and error method for allocating periods to lecturers, groups and venues. This method is time consuming as a new timetable is constructed each year. It is difficult to make changes once the timetable is finalised. Unlike the block approach, the generalised approach is more flexible in that it can cater for new requirements, because a new timetable is constructed each year.

The block model and the generalised model have their limitations. The use of resources such as lecturers and venues are not maximised. The construction of the timetable is time consuming. These timetables do not always offer a free choice of subjects or cater for lecturer preferences. The automated model provides a base for a computerised solution.

References

1 Nepal, T. 1994. An Investigation into Timetabling in Tertiary Institutions in South Africa. Unpublished Masters project, University of South Africa.

2 Martinsons M G, Cheong, K, Chow P K O & Lo V H Y, 1993. Academic Timetabling With a Microcomputer-Based Expert Systems Shell. *l. of Artificial Intelligence in Education.* Vol. 4, 4, 333 - 356.

3 Bardadym V A, 1995. Computer-Aided School And University Timetabling - The New Wave. *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling.* 253 - 268.

4 de Werra D, 1995. Some Combinatorial Models for Course Scheduling. *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling.* 1 - 20.

5 Rankin R C, 1995. Memetic Timetabling in Practice. *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling.* 45 - 56.

6 Cheng C, Kang L, Leung L & White M, 1995. Investigations of a Constraints Logic Programming Approach to University Timetabling. *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling.* 82 - 93.

Appendix

BLOCK A	BLOCK B	BLOCK C	BLOCK D	E	F	G	H
Economics I	Education I	Afrikaans I	Afrikaans IB				
Education I	Education II	Arabic II	Arabic I				
German I	English I	Economics I	Education I				
Hindi III	Geog I	French II	French III				
Latin IB	Geog II	Zulu III	Latin III				
Maths IA	Hindi I	Drama III	Zulu IA				
Maths IB	Zulu II	English I	Zulu IB				
Maths II	Maths III	German III	History II				
Arabic III	Stats I	Latin II	Greek III				

Table 1: GROUPS OF SUBJECTS IN SPECIFIC BLOCKS

PERIOD	TIMES	MON	TUES	WED	THUR	FRI
1	08H00 - 08H40	A	B	B	E	C
2	08H45 - 09H25	B	A	C	D	A
3	09H30 - 10H10	C	D	G	H	F
4	10H30 - 11H10	E	G	A	B	D
5	11H15 - 11H55	H	F	K	C	E
6	12H00 - 12H40	K	K	K	C L	
7	13H15 - 13H55	COMMON BREAK				
8	14H00 - 14H40	I	E	D	F	H
9	14H45 - 15H25	G	H	F	G	I
10	15H30 - 16H10		J	G		

Table 2: TIMESLOTS FOR EACH BLOCK OF SUBJECTS

TIMETABLE REQUIREMENTS FOR (DEPT.)

SUBJECT	DEGREE/ DIPLOMA	GROUP/S	NO. OF STUDENTS	LECTURER	PERIODS T	P	Tuts

UNAVAILABILITIES:
PRE-ASSIGNMENTS/PRE-ALLOCATIONS:
EQUITATION:
SEQUENCING:
DISTRIBUTION:
OTHER PREFERENCES:
ADDITIONAL REQUIREMENTS:

* All requirements and each preference level must be prioritized.
* The preferences above are listed in order of decreasing priority.

Table 3: REQUIREMENTS REQUEST FORM

DETAILS OF VENUES							
TYPE	VENUES						
LECTURE VENUES (capacity)	ROOM NUMBERS						
500 seater	Hall						
300	DG-1						
200							
...							

Table 4: Venue Availability Form

Monday		LECTURERS					
P	TIMES	EA	ED	KBC	RSS	SWM	VPL
1	08H00 - 08H40						
2	08H45 - 09H25						
3	09H30 - 10H10			* IS1 IT1/FIS1 DG-1			
4	10H30 - 11H10						
...	.....						
...	.....						
10	15H30 - 16H10						

Table 5: COMPOSITE LECTURERS' TIMETABLE

* IS1 - subject; IT1/FIS1 - group/s; DG-1 - lecture room

Monday		GROUPS/subjects					
P	TIMES	IT1	IT2	IT3	FIS1	FIS2	FIS3
1	08H00 - 08H40						
2	08H45 - 09H25						
3	09H30 - 10H10	IS1 KBC DG-1			IS1 KBC DG-1		
4	10H30 - 11H10						
...	.....						
...	.....						
10	15H30 - 16H10						

Table 6: COMPOSITE GROUP TIMETABLES

P	TIMES	VENUES and capacities					
		DG-1 (300)	D1-2 (120)	D2-6 (70)	D5-13 (60)	B4-15 (40)	T1-30 (30)
1	08H00 - 08H40						
2	08H45 - 09H25						
3	09H30 - 10H10	IS1 IT1/FIS1 KBC					
4	10H30 - 11H10						
...	.....						
...	.....						
10	15H30 - 16H10						

Table 7: ROOM LOADING

Using Web Standards for Timetabling

P. De Causmaecker, P. Demeester, Y. Lu and G. Vanden Berghe

KaHo Sint-Lieven
Information Technology
Gebr. Desmetstraat 1
9000 Gent
Belgium
{patdc, peterdm, yang.lu, greetvb}@kahosl.be

With the advent of the Semantic Web we examine how the technologies that lie at its basis can be applied for timetabling. In this paper, we give some directions for using these recently developed techniques. Many researchers spend a lot of time re-implementing and modifying already existing heuristics in order to make them suitable for their particular problem. We put now forward how the Semantic Web could interpret problem specific knowledge, and thus improve the communication between heuristic approaches and any problem specification.
The paper starts with a rather extensive introduction to concepts of the Semantic Web that we will use later on. The ideas behind the Semantic Web lead us to proposing a model for future timetabling development, which is based on loosely coupled components that communicate with each other by sending XML messages. The advantage of such a component model is that only a few components need to be adapted when transferring it from one timetabling application to another.

Keywords: timetabling, Semantic Web, web standards

1. Introduction

Since the term was first introduced in 1999 by Berners-Lee in 'Weaving the Web' [1], the Semantic Web gained a lot of popularity. Several international conferences are currently devoted to the subject[1] and even DARPA[2] and the European Community sponsor a lot of research in this domain. It is the intention that the Semantic Web will be the successor of the current web. At present the Semantic Web is still a vision since it is not yet realised.
The terminology used in timetabling approaches is often very confusing (scheduling, rostering, timetabling,…[2]). However most of the timetabling problems have a similar goal. They all aim at assigning activities to time slots subject to constraints. Timetabling problems are very hard to solve and there exist lots of different approaches for generating good quality solutions.

[1] See http://www.semanticweb.org/resources.html#events for a list.
[2] DARPA: Defense Advanced Research Projects Agency (see http://www.darpa.mil/)

In this paper we indicate how the technologies involved in the Semantic Web can be used in the domain of timetabling. We introduce the Semantic Web in Section 2 and in Section 3 we explain some acronyms of the Semantic Web. Section 4 puts forward how timetabling could benefit from the Semantic Web. A few possible applications are discussed in Section 5 and we conclude in Section 6.

2. The Semantic Web

There exist numerous definitions of the Semantic Web, which mean more or less the same. For this paper, we have selected two. The first one is from Berners-Lee and the second one is taken from the W3C Semantic Web Activity Statement.

> "*The Semantic Web is a web of data, in some ways like a global database*"[3].

> "*The Semantic Web is a vision: the idea of having data on the web defined and linked in a way, that it can be used by machines - not just for display purposes, but for automation, integration and reuse of data across various applications.*" [4]

The current web was designed to improve the exchange of information between different people [5]. Unfortunately machines cannot easily access this information since there is too much overhead involved: the content is not cleanly separated from the instructions on how to present content. An HTML document mixes content and layout instructions. Another problem is that a word can have different meanings. Any search engine on the web at present suffers badly from this defect, and a lot of effort is spent trying to remedy it. Using languages that allow automatic interpretation of content can solve this problem at a fundamental level.

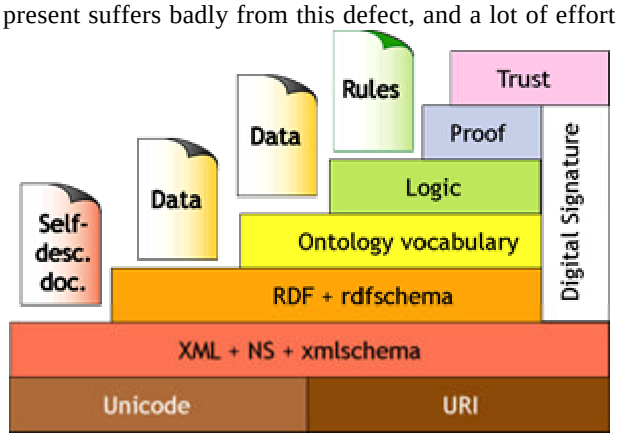

Figure 1: The semantic web architecture as proposed by Tim Berners-Lee

It is the goal of the Semantic Web to make information on the web machine-processable. A lot of things which currently need human aid can be automated in this way, e.g. making a synthesis of information that is found in different documents, filtering the results of a search engine, etc. Tim Berners-Lee - director of the W3C -, James Hendler - responsible for agent-based computing research at DARPA - and Ora Lassila - chief scientist of Nokia and co-

author of the RDF specification - [6] describe a hypothetical situation – automating the process of making an appointment with a physician - that could be real when the Semantic Web is ready. To bring this situation to a favourable conclusion, the authors use software agents, which roam the Semantic Web looking for information. The Semantic Web will be the natural habitat of software agents.

3. The different layers of the Semantic Web

In order to realise the Semantic Web, Berners-Lee proposed an architecture [7]. Since this architecture is neatly layered it is sometimes called the 'layer cake' (Figure 1).
Some of the ideas and techniques will be applied in timetabling and therefore we discuss the different layers very briefly. More detailed information can be found in [8, 9, 10].

3.1 First layer

The lowest layer consists of Uniform Resource Identifiers (abbreviated as URIs) and Unicode[3]. URIs are used to identify objects on the Web. These objects can be anything: a web page, a paragraph from a web page, a MPEG movie, an MP3 sound fragment, a JPEG image, a test bed for timetabling problems, algorithms, rooms, lecturers etc. Once something has a URI it is said to be "on the Web". A well-known form of a URI is the Uniform Resource Locator (URL), this is the string which is entered in the address bar of a browser.

3.2 Second layer

XML handles the syntax of documents in the second layer. XML [11] is an acronym for eXtensible Markup Language. XML is more than the successor of HTML[4]. XML offers the possibility for users to create documents using their own

[3] "Unicode is an entirely new idea in setting up binary codes for text or script characters. Officially called the Unicode Worldwide Character Standard, it is a system for "the interchange, processing, and display of the written texts of the diverse languages of the modern world." It also supports many classical and historical texts in a number of languages.
Currently, the Unicode standard contains 34,168 distinct coded characters derived from 24 supported language scripts. These characters cover the principal written languages of the world. Additional work is underway to add the few modern languages not yet included.
Also see the currently most prevalent script or text codes, ASCII and extended binary-coded decimal interchange code (EBCDIC)." [taken from the whatis.com website, http://whatis.techtarget.com/definition/0,,sid9_gci213250,00.html]

[4] HyperText Markup Language

structure and syntax. Unlike in HTML, which only contains a limited number of tags, XML allows to invent tags suitable for a particular problem.

Example 1: this XML document contains data about the authors of an article. The tags and order of the tags is something we invented.

```
<?xml version="1.0"?>
<article>
     <title>Using web standards for timetabling</title>
     <author>Sep Ducamaercke</author>
     <author>Per Tedesem</author>
     <author>Guy Lan</author>
     <author>Venghed Banger</author>
     <affiliation>KaHo Sint-Lieven, Gent</affiliation>
</article>
```

From the tags we use it is clear what this XML document is about.

In order to avoid confusion, which is not unlikely if everybody invents different tags, XML schemas are used. These schemas determine how the content should be formatted in order to create a syntactically valid XML document. Such document can be exchanged between people, companies or applications. If it satisfies a particular schema constructed by an individual/company, it is guaranteed that it is understandable by this individual/company.

Example 2: An XML Schema that corresponds to the XML document of Example 1.
The schema describes how the XML document from Example 1 should be formatted. The document consists of a *root* element `article`, which has `title`, `author` and `affiliation` as *children*, all having `string` as data type. In XML Schema, a large set of simple data types are already built in [12]. Any data type that is not built in can be constructed using the `complexType` element. The element `author` can occur several times since it contains the attribute `maxOccurs` that has the value `unbounded`.
Suppose the computer system of a conference on timetabling uses such a schema to classify the submitted articles. If every submitted paper is accompanied by an XML document satisfying this schema, it is guaranteed that the computer system can read each document and classify it accordingly.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="article">
  <xs:complexType>
    <xs:sequence>
       <xs:element name="title" type="xs:string"/>
       <xs:element name="author" type="xs:string"
        maxOccurs="unbounded"/>
       <xs:element name="affiliation" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

A personal meaning can be attached to documents by user-defined tags.
Since the second layer builds on the lowest layer, it is possible to uniquely identify tags. URIs are used for that purpose. By identifying tags an "XML Namespace" [13] will be created. Namespaces created by other people can be reused.
XML is thus used to describe the content of a problem. Note however that XML says nothing about how to display content. In contrast with HTML, content and presentation are separated. XSL (eXtensible Style Language) or CSS (Cascading Style Sheets) are used for presenting XML content.

3.3 Third layer

The next layer offers the opportunity to make 'machine-processable' statements. The technology behind it is called Resource Description Framework (RDF) [14]. The goal of RDF is to add formal semantics to the web [15], which is done by defining a data model. Three object types are used in the data model:

- Resources: an object that can be identified by a URI
- Properties: a specific characteristic that can be used to describe a resource
- Statements: a triple consisting of a resource, a property and a value of that property

The RDF description model uses triples, which consist of a subject, a predicate and an object. These triples are known as statements. Instances of this description model can be viewed as directed or labelled graphs.

> **Example 3**: An RDF statement can provide information about an article that is written by Sep Ducamaercke: '*ArticleX has creator Sep Ducamaercke.*'
> In this case 'ArticleX' is the resource (or the subject), 'creator' is the predicate (or the property) and 'Sep Ducamaercke' is the object. This can be depicted as in Figure 2.
> Note that the resource is represented by an URI.

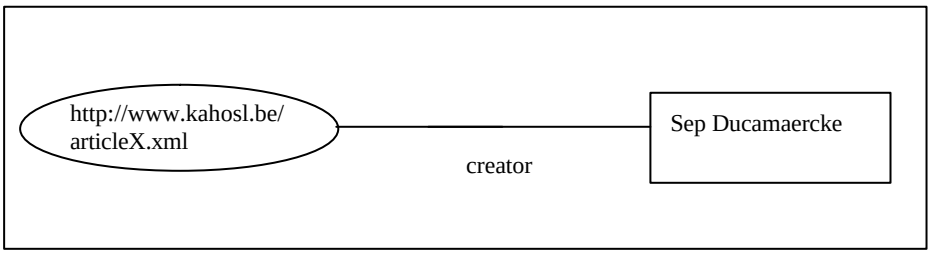

Figure 2: An RDF example

RDF provides only a model for representing metadata[5]. One possible representation is the labelled graph representation of Figure 2, another is a triple list. Many other

[5] metadata is data about data.

syntactic representations are possible. An obvious candidate for an alternative representation is XML. With XML, instances of the model can be stored in files and exchanged between users. RDF cannot replace XML. It builds a layer on top of XML, in order to enable interoperable exchange of semantic information.

> **Example 4**: the RDF statement of Example 3 can be 'translated' in XML. Note that we introduce two namespaces of which only one has a prefix (`rdf`). Instead of defining a completely new concept, we re-use the concept `Description` that is already defined in this namespace. The namespace without prefix is called the default namespace.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://sep.ducamaercke.net/rdfexamples">
  <rdf:Description about="http://www.kahosl.be/articleX.xml">
     <creator>Sep Ducamaercke</creator>
  <rdf:Description>
<rdf:RDF>
```

An object-oriented type system is situated on top of the basic RDF model: RDF Schema (sometimes abbreviated as RDFS). RDF Schema is a data-typing model for RDF [16]. Mark however that the functions which RDF Schema and XML Schema have are not the same [15]:
XML schema is used to control the syntax of an XML document, while RDF schema only says something about the interpretation of RDF statements.

3.4 Fourth layer

The next layer is the ontology layer. It is the layer where at present most of the research concerning Semantic Web happens. Fensel et al. [17] present a possible reason why ontologies are nowadays popular: "*ontologies are becoming popular largely because of what they promise: a shared and common understanding that reaches across people and application systems*".
A common accepted definition of an ontology is:

> "*an ontology is a formal, explicit specification of a shared conceptualisation*" [18].

A conceptualisation is a way of thinking in a particular domain. This is typically expressed as a set of concepts (entities, attributes, processes), their definitions and their inter-relationships.

One of the interesting applications of ontologies is the translation service based on an ontology, which maps different data structures onto areas where no standard ontologies exist or where people need to translate their own terminology into the standard. This translation service covers structural, semantic in addition to language

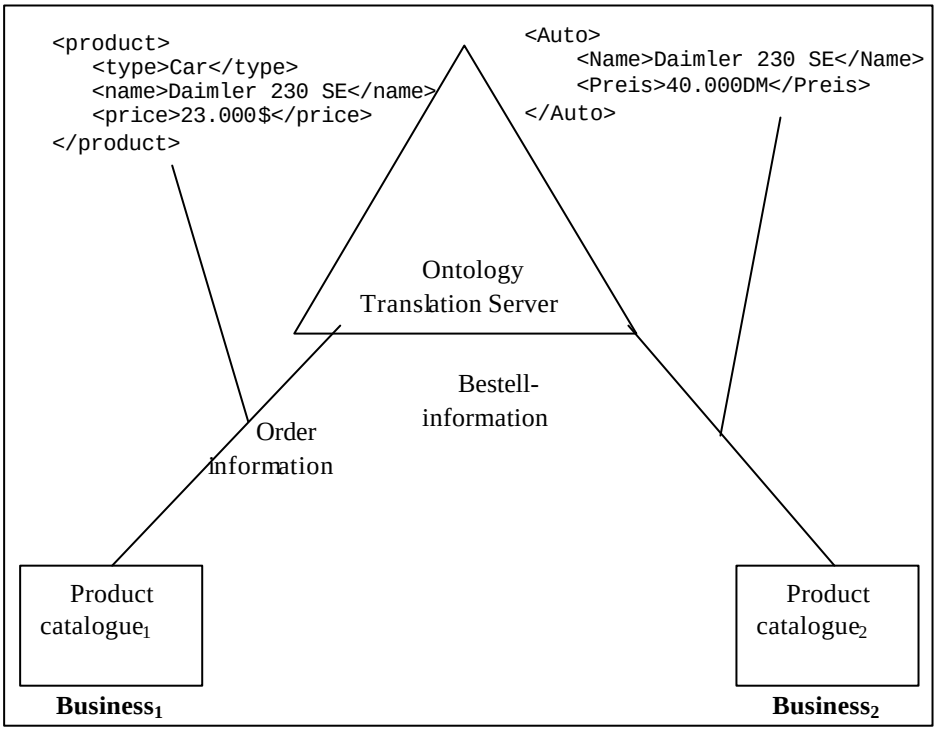

Figure 3: translation of structure, semantics and language. Example taken from [17]

differences [17] (see Figure 3).

Although ontology languages (LOOM [19], POWERLOOM [20], KIF [21], CYCL [22], Ontolingua [23], etc) were created years before the term Semantic Web was invented, the two most important languages are based on RDF Schema. The creators of both ontology languages quickly realised that it was better to co-operate than to compete. At this moment they are working on a unified ontology language called DAML+OIL, which is a junction of the two original names. DAML is funded by DARPA and OIL is funded by the EC. DAML+OIL still applies RDF Schema but allows more expressiveness: inverses, unambiguous and unique properties, restrictions on properties, disjoint unions,… can be defined. For more details consult [24].

As part of the DAML programme people from the academic world are working on DAML-S, an ontology of services. Services refer to dynamic web sites which allow actions or changes in the world [25]. In order to use services, software agents need to interpret a description of the web service. They must also know how the service can be accessed. The goal of the DAML-S project is to establish the description based on

DAML. Some of the web service tasks, which will be automated once DAML-S is fully functional, are:

- automated web service discovery,
- execution,
- interoperation,
- composition and
- execution monitoring [26].

There is an effort from the industry (especially from IBM, MicroSoft, HP, etc) to develop standards for the description of web-based services: UDDI, WSDL, WSFL, SOAP, .NET, etc. We will briefly introduce these services by explaining SOAP as an example. SOAP [27] stands for Simple Object Access Protocol and it was originally developed by DevelopMentor, Microsoft, and UserLand Software. This protocol specifies how to exchange information in a decentralised, distributed environment. It is operating system and programming language independent. Applications written in Java and running on a SUN-OS environment can easily co-operate with a computer program written in C++ running on a WINTEL environment. SOAP uses HTTP as transport protocol and XML messages to exchange information. There are already some web sites, which offer "web services". The XMethods website [28], for instance, offers simple applications, such as the conversion of currencies or playing chess against a computer, which can be invoked remotely. There will probably be companies in the future, which will offer rather complex applications on payment, evocable from any computer. At present, these services still imply human involvement: users still have to look manually which methods to invoke, which services to use, etc. It is the aim that, once the above-mentioned industry standards are fully functional, all this happens automatically.
These industry standards evolved from current web technologies and they focus on technologies, that will be used on a short term. Since these technologies are not based on languages like DAML, and thus lack expressiveness, they are for humans rather than for software agents. For a further discussion on this subject we refer to [29].

3.5 Fifth layer

The fifth layer and the sixth layer still have to be developed. The logic layer provides a language for describing sets of deductions, which can be derived from a collection of data. It is the purpose to determine how new facts can be derived from the world described in the ontology layer. The people involved in DAML are now working on DAML-L. It is a logical language with well-defined semantics and the ability to express propositional Horn clauses [30], which enable a compact representation of constraints and rules for reasoning.

3.6 Sixth layer

The proof layer describes the steps taken to reach a conclusion from the facts. These proofs can then be passed around and verified, and can be used as short cuts to new facts in the system without repeating the deductions.

3.7 Seventh layer

This top layer is not a physical layer. It considers the confidence, which will exist on the web once the lower layers are ready. It is actually a metaphor, which tells that received data on the Semantic Web can be trusted. In order to reach this trust, people should attach a digital signature (see Figure 1) to their documents.

4 The Semantic Web and timetabling

It is our opinion that the above-described technologies can be useful in the domain of timetabling and that timetabling can gain profit from the Semantic Web once it is in place. In this paragraph we will give some directions for applications of these technologies. We will do this on the basis of the above-mentioned layers. For each of the interesting layers (from the second up to and including the fourth layer) we will explain how these technologies can be applied in timetabling.

4.1 Second layer

At the essential, basic level we have to define a format or language for representing instances of a timetable. Burke et. al. [31] mention requirements which should be fulfilled to talk about a standard data format for timetabling. Such a format should be:

- general, express all kinds of timetabling problems [32],
- complete, express these problems in full detail [32],
- accessible, easy to translate to and from [32].

By catching these requirements in XML, we nowadays have parsing tools available, while the files can be generated by almost every database. This massive support from the software industry is also an important reason to express the elements of the language STTL [32] in XML.

In [31] the authors note the following:

"*Programs will be necessary to convert data in a non-standard format into the standard data format. In order to encourage the development of consistent conversion programs we will publish a standard interface. This might then be used to interface with a common conversion application which would be used as the standard front-end to the conversion utility.*"[31]

These requirements are fulfilled by XML through the introduction of the Document Object Model (DOM) [33]. "The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page." This interface allows a programmer to manipulate the content of an XML document: elements can be added, deleted, moved, etc.
In the conclusion of [31] it is mentioned:

"*Many formats used by timetabling applications will have been designed with concerns for minimising data storage requirements or to facilitate the fast processing of data. This means that such formats would not be suitable as a standard format since generality or readability of data (by humans) would be forfeited. Our aim, however, is to provide generality; allowing the maximum number of data instances to be represented as is practicable.*"[31]

This pre-formulates one of the postulates of the Semantic Web. Since XML is simply text (because it is based on Unicode) it is readable by humans. Sometimes this is seen as a disadvantage since data cannot be saved in a binary format but only in text format and this is not economical in storage space.

One extra advantage that web technologies can bring to the timetabling domain lies in its intrinsic focus on interactivity. Users should be allowed to express their goals and their interpretations of soft constraints as freely as possible. By providing an ontological description of the domain, the resulting XML files may become as rich in information content as possible. Among the available technologies to realise this, we certainly want to mention fuzzy logic and fuzzy set theory [34, 35, 36].

4.2 Third layer

The technologies of this layer allow us to make instances of timetabling resources. Every resource can be given a unique identifier. We consider using the following resources:

- Students, classes, rooms, teachers,…(see later)
- Documents with timetables, requirement descriptions, goals,…

In the following example, the instances `Sep Ducamaercke, Per Tedesem, Guy Lan, Venghed Banger` are further described using `rdf:type`. Remark however that we use `daml:Class` to describe to which class every instance belongs. This is done since the RDF language does not have this kind of statement.

```
...
<rdf:Description rdf:about="resource.daml#Sep Ducamaercke">
   <rdfs:comment>This is the first instance</rdfs:comment>
   <rdf:type>
      <daml:Class rdf:about="resource.daml#fullprofessor"/>
   </rdf:type>
</rdf:Description>
<rdf:Description rdf:about="resource.daml#Per Tedesem">
```

```
   <rdfs:comment>This is the second instance</rdfs:comment>
   <rdf:type>
      <daml:Class rdf:about="resource.daml#researchassistant"/>
   </rdf:type>
</rdf:Description>
<rdf:Description rdf:about="resource.daml#Guy Lan">
   <rdfs:comment>This is the third instance</rdfs:comment>
   <rdf:type>
      <daml:Class rdf:about="resource.daml#researchassistant"/>
   </rdf:type>
</rdf:Description>
<rdf:Description rdf:about="resource.daml#Venghed Banger">
   <rdfs:comment>This is the fourth instance</rdfs:comment>
   <rdf:type>
      <daml:Class rdf:about="resource.daml#associateprofessor"/>
   </rdf:type>
</rdf:Description>
...
```

These instances in RDF can be saved in files, but a better solution is using a database. At the moment the first RDF APIs appear which allow using RDF statements in programming languages such as Java, C++,…[37, 38, 39].

4.3 Fourth layer

4.3.1 Ontology as a starting point for design

At the previous PATAT conference we presented an ontology for timetabling that was based on the OZONE ontology [40] for scheduling. Developing an ontology for timetabling leads to a better understanding of the problem and this offers the opportunity for integrating separate timetabling systems. The developed ontology can be used in two different ways:

- existing timetabling systems can be mapped to this ontology or
- newly developed timetabling systems can take this ontology as a starting point. Or in other words: an ontology can be used as the basis for the design of a software system.

Both approaches lead to a better co-operation between timetabling systems. Through the mapping, concepts in the original timetabling application can be ‘translated’ to concepts in another application. In this way it is possible to use components/parts from already existing applications for new timetabling problems. It should be possible to only use components of different timetabling applications that suit best. If a particular timetabling application has a clearly arranged GUI and another a satisfying algorithm both components can be incorporated in a new application.
Basing newly developed timetabling applications on a commonly accepted ontology will facilitate the co-operation between different applications.

Through the use of domain ontologies, a domain of discourse can be provided that both computers and developers can comprehend. In this way there is a strict distinction between fundamental domain concepts and problem-solving techniques, which can be applied on these concepts. In modern knowledge engineering, building intelligent systems is seen as the process of designing and assembling domain ontologies, the instantiation of knowledge bases (the RDF files that describe the resources) and of designing domain-independent problem solving methods. [41]

> *"The goal is to transform system development into a matter of selecting, modifying and assembling previously tested and debugged components, rather than to require programming of each new application from scratch. The hope is that, because these components are at a quite high level of abstraction, the assembly of the components can take place more easily than in other cases of software reuse."*[41]

By incorporating domain ontologies, the domain-dependent information becomes explicit, accessible and editable.

In the design phase we can also make a distinction between the computational model and the application model. The application model provides the full specification of the problem and allows for maximal expression of the user's intentions. This information has to be filtered when sending it to specific components. This enables future extension of the application model to be extended in the future without disrupting the functionality of the existing components. The computational model can be considered as one of the components, working on a subset of the data and concepts visible and accessible for the user.

4.3.2 The proposed timetabling ontology

As a starting point for a discussion on a commonly accepted ontology for

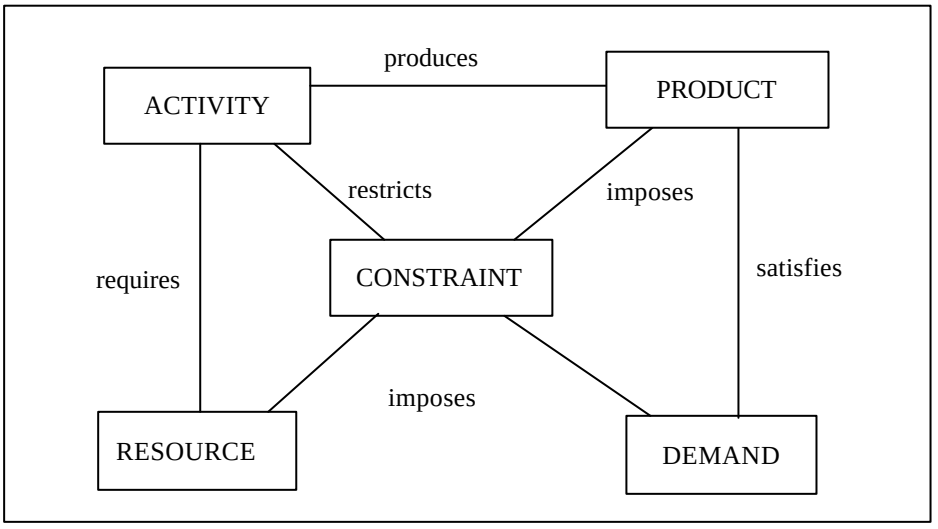

Figure 4: OZONE top-level ontology adapted to timetabling

timetabling we propose the upper level ontology we developed based on the above-mentioned OZONE scheduling ontology (see Figure 4). This upper ontology has 5 central concepts, namely:

- ACTIVITY: a process, which can be executed over a time interval, is represented by an activity. Resources are needed to execute it. This execution depends on and affects the current state of these resources.
- CONSTRAINT: restriction on the set of values that can be assigned to a variable.
- DEMAND: a request for services (products) that the system provides is a demand.
- PRODUCT: A product is a good or service provided by a system. Through the execution of a set of activities the product is realised.
- RESOURCE: an entity that supports or enables the execution of activities is a resource. The ultimate timetabling (or scheduling) tool is making efficient use of resources in support of multiple, competing activities.

4.3.3 Example of an ontology: the resource ontology

Starting from these very high-level concepts the goal is to identify the different kinds of constraints (type and temporal constraints for example), resources (re-usable and consumable resources,…), demands, meetings and activities.
As an example we give the resources ontology (constructed in UML) which is based on our institute (see Figure 5). The names reflect the meaning of the resources.

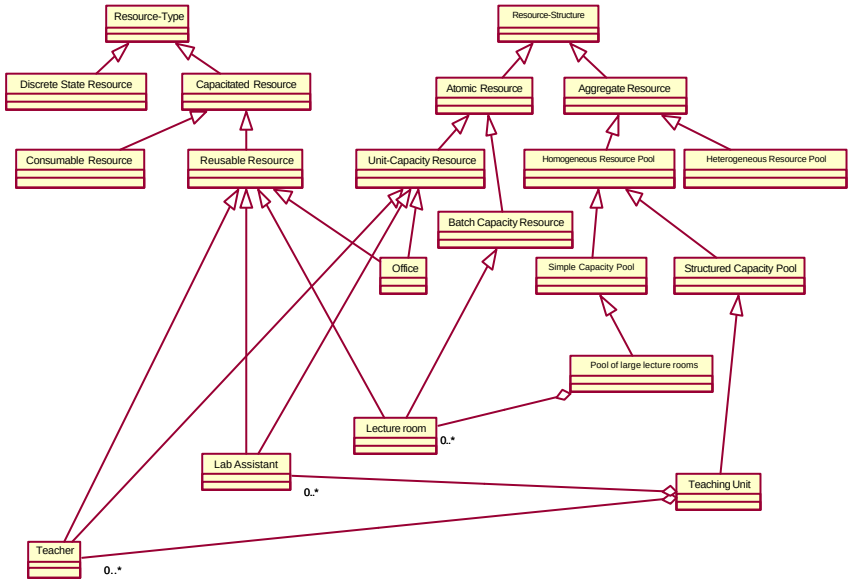

Figure 5: Example of an ontology for resources in our polytechnic constructed in UML based on OZONE [40].

We have opted to use time not as a (consumable) resource.

4.3.4 Relation with STTL

The elements in Kingston's STTL [32] can be reformulated as an ontology. In fact,

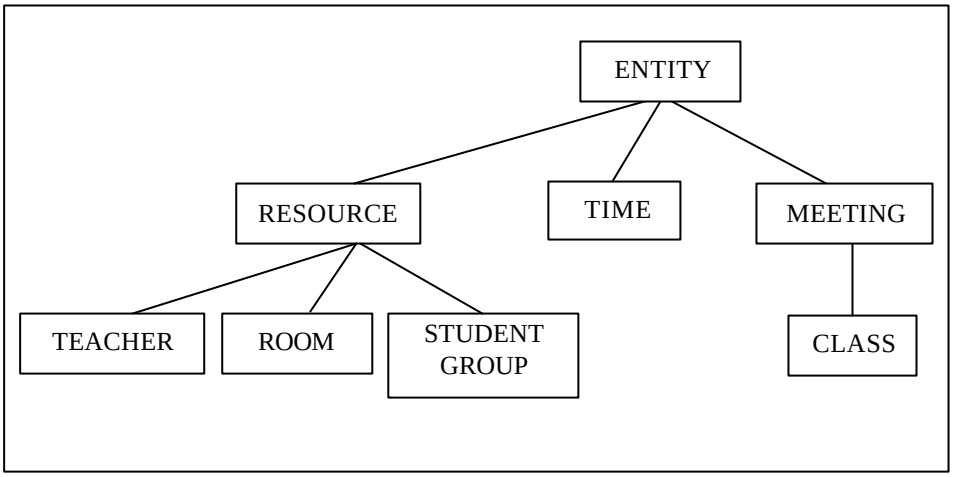

Figure 6: Kingston's implicit timetabling ontology

the timetabling ontology he uses implicitly can be represented as in Figure 6.

After agreeing on the concepts, the definitions of these concepts and the relations between the concepts, this ontology can be 'translated' to a formal language such as DAML+OIL.
In the following example we 'translated' some of the concepts of Figure 5. There are different prefixes (`daml`, `rdfs` and `oiled`). DAML+OIL took the best of all available technologies: it uses parts of RDF and RDFS. Only the prefix `oiled` needs some explanation: this prefix is generated by the OilEd [42] editor we use to construct the ontology.

```
<daml:Class rdf:about="resource.daml#teachingassistant">
   <rdfs:label>teachingassistant</rdfs:label>
   <rdfs:comment><![CDATA[]]></rdfs:comment>
   <oiled:creationDate>2002-04-17T09:26:59Z</oiled:creationDate>
   <oiled:creator><![CDATA[peter]]></oiled:creator>
   <rdfs:subClassOf>
      <daml:Class rdf:about="resource.daml#teacher"/>
   </rdfs:subClassOf>
</daml:Class>
<daml:Class rdf:about="resource.daml#lecturer">
   <rdfs:label>lecturer</rdfs:label>
   <rdfs:comment><![CDATA[]]></rdfs:comment>
   <oiled:creationDate>2002-04-17T09:29:25Z</oiled:creationDate>
   <oiled:creator><![CDATA[peter]]></oiled:creator>
   <rdfs:subClassOf>
      <daml:Class rdf:about="resource.daml#teacher"/>
   </rdfs:subClassOf>
```

```
</daml:Class>
<daml:Class rdf:about="resource.daml#lecture room">
   <rdfs:label>lecture room</rdfs:label>
   <rdfs:comment><![CDATA[]]></rdfs:comment>
   <oiled:creationDate>2002-04-16T15:38:25Z</oiled:creationDate>
   <oiled:creator><![CDATA[peter]]></oiled:creator>
   <rdfs:subClassOf>
      <daml:Class rdf:about="resource.daml#reusable resource"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <daml:Class rdf:about="resource.daml#batch capacity resource"/>
   </rdfs:subClassOf>
</daml:Class>
<daml:Class rdf:about="resource.daml#associateprofessor">
   <rdfs:label>associateprofessor</rdfs:label>
   <rdfs:comment><![CDATA[]]></rdfs:comment>
   <oiled:creationDate>2002-04-17T09:28:41Z</oiled:creationDate>
   <oiled:creator><![CDATA[peter]]></oiled:creator>
   <rdfs:subClassOf>
      <daml:Class rdf:about="resource.daml#teacher"/>
   </rdfs:subClassOf>
</daml:Class>
```

This 'translated' ontology can then be used on the Semantic Web. In this way, it becomes conceivable that we succeed to design systems that can perform as follows. Suppose some software agent is roaming the Semantic Web to find a solution for a specific timetabling problem. Imagine that this agent finds a web page about timetabling, where the creator added a link to a timetabling ontology, and that this page contains a solution to the problem. On the basis of this ontology the software agent can interpret the page and use it to solve his own problem. Hendler [43] describes a number of agents that are roaming on the Semantic Web.

5 Discussion

It is our opinion that future timetabling applications should be built with the ideas of the Semantic Web in mind. At present we are working on a project [44] which will establish a timetabling application consisting of separate components working together. These components are linked together through the use of an ontology. Montana [45] introduced a web-based timetabling application, the only similar approach we are aware of.

Although university and exam timetabling problems are occurring all over the world [46, 47, 48, 49], they do not have a generally satisfying solution. That is because the demands, resources and constraints change from university to university. This bold statement is the standing ground for the upper-level ontology we created for the timetabling problem (see Figure 4). We will develop different separate components, such as algorithmic, demand, resource and constraint components. These components may then be published as web services. Applications can call these components

through SOAP. This allows running these components on a distributed environment. A CPU-intensive algorithmic component could, for instance, run on a fast UNIX server, while the other components can run on a simple WINTEL combination. For the components, schemas can be defined, such that data offered to the components are in a format that the algorithmic component can 'understand'. Using a DOM parser, the results of the algorithm can be transferred to XML documents. By applying XSL and CSS these XML results can then be presented in a browser.

Each personnel scheduling problem has specific constraints and thus requires adapted approaches to solve it [50, 51, 52, 53]. In different sectors, we distinguish a different attitude towards cyclical schedules, overtime, locations, qualifications, etc. Just like in the university timetabling problem discussed in the previous paragraph, the existing knowledge and approaches could become available for any personnel scheduling environment by making use of the Semantic Web.

Ontologies have been used to support fast application generation [54]. Instead of starting from scratch when developing a timetable application that suits the demands, it might be easier to adapt or extend the domain ontology in order to satisfying the preferences. In this way you really can reuse software components, since domain ontologies and generic problem solving methods provide the right kind of abstraction for building timetabling applications. In contrast with traditional OO techniques in which methods (program code) and classes and instance objects (the representation of the domain model) are closely interrelated, these are strictly separated by using domain ontologies and problem-solving methods. This minimises the effort of updating the timetabling applications when requirements change [41]. Another advantage is that any new algorithm with better performance can be plugged in as a problem-solving method. Finally, using ontology in system design makes it function as a breeding ground from which formal domain descriptions may originate. These can then be used for communication between independently developed systems.

6 Conclusion

By applying the ideas of the Semantic Web to the timetabling domain, researchers will have more opportunities to concentrate on new technologies. Instead of repeating and adapting already existing solutions to their problems, they will focus fully on the development and improvement of search algorithms.
The Semantic Web will contribute to simplifying the very time-consuming optimisation problems. By using a common ontology, timetabling problems will easily be recognised, classified and left with the care of software agents that will search appropriate solution techniques on the web. Applying these web technologies will require an effort from the users. They will either have to make their own ontology, refer to, or modify an already existing ontology. In order to make their data available on the web, it has to be machine-readable. However, the overall profit will be high. The available problem ontology will enable widespread co-operation with other systems.

References

[1] T. Berners-Lee, *Weaving the Web*, Harper Business, 1999, 006251587X.

[2] Wren, A. *Scheduling, timetabling and rostering - a special relationship?* in: E.K. Burke & P. Ross (editors) The Practice and Theory of Automated Timetabling, pp. 45-75 Springer-Verlag. 1996.

[3] T. Berners-Lee, *Semantic Web Roadmap, Internal Note*, World Wide Web Consortium, 1998. (http://www.w3.org/DesignIssues/Semantic.html)

[4] W3C Semantic Web Activity Statement, 2001, http://www.w3.org/2001/sw/Activity

[5] T. Berners-Lee, *The World Wide Web: Past, Present and Future*, August 1996. (http://www.w3.org/People/Berners-Lee/1996/ppf.html)

[6] T. Berners-Lee, James Hendler and Ora Lassila, *The Semantic Web*, Scientific American, May 2001. (http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html)

[7] T. Berners-Lee, *Semantic Web*, presentation at XML 2000 conference, 2000. (http://www.w3.org/2000/Talks/1206-xml2k-tbl/)

[8] S. B. Palmer, *The Semantic Web, Taking Form.* (http://infomesh.net/2001/06/swform/)

[9] T. Gardner, *An Introduction to Web Services* (http://www.ariadne.ac.uk/issue29/gardner/)

[10] A. Swartz and J. Hendler, *The Semantic Web: A Network of Content for the Digital City* (http://blogspace.com/ rdf/SwartzHendler)

[11] T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 2000. (http://www.w3.org/TR/REC-xml)

[12] D. C. Fallside, *XML Schema Part 0: Primer*, W3C Recommendation, 2001. (http://www.w3.org/TR/xmlschema-0/)

[13] T. Bray, D. Hollander and A. Layman, *Namespaces in XML*, World Wide Web Consortium Recommendation, 1999. (http://www.w3.org/TR/REC-xml-names/)

[14] O. Lassila and R. R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation, 1999. (http://www.w3.org/TR/REC-rdf-syntax/)

[15] J. Broekstra, M. Klein, S. Decker, D. Fensel and I. Horrocks, *Adding formal semantics to the Web: building on top of RDF Schema* In Proc. SemWeb 2000.

[16] D. Brickley and R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0*, W3C Candidate Recommendation, 2000. (http://www.w3.org/TR/rdf-schema/)

[17] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, *OIL: An Ontology Infrastructure for the Semantic Web*, in IEEE Intelligent Systems & their applications, Vol. 16, No. 2, March/April 2001.

[18] T.R. Gruber, *A Translation Approach to Portable Ontology Specifications*, in Knowledge Acquisition, vol. 5, 1993, pp.199-220.

[19] http://www.isi.edu/isd/LOOM/LOOM-HOME.html

[20] A. Valente, T. Russ, R. MacGregor and W. Swartout, *Building and (Re)Using an Ontology of Air Campaign Planning*, In IEEE Intelligent Systems 14:1, pp. 27-36, Jan-Feb, 1999.

[21] R. Patil, R. Fikes, P. Patel-Schneider, D. McKay, T. Finin, T. Gruber and R. Neches, *The DARPA knowledge sharing effort: Progress report*, 1992.

[22] http://www.cyc.com/cycl.html

[23] A. Farquhar, R. Fikes, and J. Rice, *The Ontolingua server: A tool for collaborative ontology construction*, technical report, Stanford KSL 96-26, 1996.

[24] I. Horrocks, F. van Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, L. Andrea Stein, *DAML+OIL*, March 2001. (http://www.daml.org/2001/03/daml+oil-index.html)

[25] DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), *DAML-S: Semantic Markup for Web Services*, in Proceedings of the International Semantic Web Working Symposium (SWWS). July 30-August 1, 2001.

[26] http://www.daml.org/services/

[27] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer, *Simple Object Access Protocol (SOAP) 1.1*, W3C Note, 2000. (http://www.w3.org/TR/SOAP/)

[28] http://www.xmethods.net/

[29] T. Sollazzo, S. Handschuh, S. Staab, M. Frank, *Semantic Web Service Architecture —Evolving Web Service Standards toward the Semantic Web*, Proc. of the 15th International FLAIRS Conference, Pensacola, Florida, May 16-18, 2002. AAAI Press.

[30] S. A. McIlraith, T.C. Son, H. Zeng, *Mobilizing the Semantic Web with DAML-Enabled Web Services*, in Proceedings of the 2nd International Workshop on the Semantic Web, pp. 82-87, Hongkong, China, May 1, 2001.

[31] E. K. Burke, P. A. Pepper and J. H. Kingston, *A Standard Data Format for Timetabling Instances*, In Practice and Theory of Automated Timetabling II (selected papers from Proceedings of the Second International Conference on Practice and Theory of Automated Timetabling, Toronto 1997) Springer Lecture Notes in Computer Science 1408, pages 213-222 (1997).

[32] J. H. Kingston, *Modelling Timetabling Problems with STTL*, In Practice and Theory of Automated Timetabling III (selected papers from Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling, Konstanz 2000) Springer Lecture Notes in Computer Science 2079, pages 309-321 (2001).

[33] L. Wood, V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, *Document Object Model (DOM) Level 1 Specification*,Version 1.0, W3C Recommendation, 1998. (http://www.w3.org/TR/REC-DOM-Level-1/)

[34] L. A. Zadeh, *Fuzzy Sets*. Information and Control, vol. 8 pp. 338-353, 1965.

[35] L. A. Zadeh, *Fuzzy Logic*. Computer, april 1998, vol.21, no. 4, pp. 83-93.

[36] L. A. Zadeh, *Knowledge Representation in Fuzzy Logic*. IEEE Transactions on Knowledge and Data Engineering 1(1): 89-100 (1989)

[37] http://www-db.stanford.edu/~melnik/rdf/api.html

[38] http://www.hpl.hp.com/semweb/

[39] http://grcinet.grci.com/maria/www/codipsite/Tools/Components.html

[40] S. F. Smith and M. A. Becker, *An Ontology for Constructing Scheduling Systems*. In Working Notes from 1997 AAAI Spring Symposium on Ontological Engineering, Stanford, CA, March 1997.

[41] M. A. Musen. In J. Cuena et al., Ed., *Knowledge Engineering and Agent Technology*. IOS Press, Amsterdam, 2000.

[42] http://oiled.man.ac.uk/

[43] J. Hendler, *Agents and the Semantic Web*, in IEEE Intelligent Systems & their applications, Vol. 16, No. 2, March/April 2001.

[44] http://project.kahosl.be/cofftea

[45] D. J. Montana, *Optimized Scheduling for the Masses*, Genetic and Evolutionary Computation Conference (GECCO-2001), Workshop on The Next Ten Years of Scheduling Research.

[46] E.K. Burke, J.P. Newall, R.F. Weare, *A Memetic Algorithm for University Timetabling*, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference Edinburgh, Springer, 1995

[47] M.W. Carter, *A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo*, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 64-82

[48] B. Paechter, A. Cumming, M.G. Norman, and H. Luchian, *Extensions to a Memetic Timetabling System*, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, Springer, 1995, 251-265

[49] G.M. White, B.S. Xie, *Examination Timetables and Tabu Search with Longer-Term Memory*, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 85-103

[50] U. Aickelin, K. Dowsland, *Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem*, Journal of Scheduling, Volume 3 Issue 3, 2000, 139-153

[51] E.K. Burke, P. Cowling, P. De Causmaecker, G. Vanden Berghe, *A Memetic Approach to the Nurse Rostering Problem*, Applied Intelligence special issue on Simulated Evolution and Learning, Vol. 15, Number 3, Springer, 2001, 199-214

[52] A. Meisels, E. Gudes, G. Solotorevski, *Combining rules and constraints for employee timetabling*, Journal of Intelligent Systems, Vol. 12, 1997, 419-439

[53] H. Meyer auf'm Hofe, *Solving Rostering Tasks as Constraint Optimization*, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated

Timetabling, Third International Conference, Konstanz, Springer, 2000, 191-212

[54] W. Ceusters, *Terminology and Ontology Management Systems*, Semantic Web and Applications 2002 seminar, Ghent. (http://project.kahosl.be/swa2002/slides/werner%20ceusters/swa2002ceusters.ppt)

A Survey and Case Study of Practical Examination Timetabling Problems

Peter Cowling[*], Graham Kendall[+] and Naimah Mohd Hussin[+]

[+]Automated Scheduling, Optimisation and Planning (ASAP) Research Group
School of Computer Science and Information Technology, University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
Email: {gxk, nbh}@cs.nott.ac.uk
[*]Department of Computing, University of Bradford
Bradford BD7 1DP, UK
Email: Peter.Cowling@scm.brad.ac.uk

1. Introduction

Most academic institutions face the problem of scheduling both courses and examinations in every semester or term. As the difficulty of the problem increases, due to a large number of students, courses, exams, rooms and invigilator constraints, an automated timetabling system that can produce feasible and high quality timetables is often required. Surveys by Comm and Mathaisel (1988), Burke et al (1996) and JISC (Joint Information System Committee) Technology Applications Programme (JTAP 1998) has shown that a lot of automation has taken place over the years and it would be interesting to find out whether the timetables produced are acceptable by people who are directly affected by the timetable. In this paper we gather views from students and invigilators on their current exam or invigilation timetable and what they hope will be taken into account in the next generation of exam timetabling software. We conclude the paper by presenting the very large examination timetabling problem faced by University Technology MARA (UiTM), discussing their examination timetabling processes and the possibility for full computerisation.

2. Questionnaires

In conducting the survey for this paper, two questionnaires were distributed; to students and invigilators. The main objective of these questionnaires was to get the opinion of people who are affected by the schedules they receive every semester. A lot of computerisation has taken place in producing examination timetable and we would like to see whether the timetables produced are acceptable to them.

The student questionnaire results suggest that:

- Almost all students prefer to have at least a gap between exams.

- Students do not like to have weekend exams.
- Some weaknesses of previous examination timetables were an inconsistent gap between two examinations, frequent changes in the timetable, the timetable not updated accurately and in a timely manner, clashing timetables, weekend exams, late timetable delivery and incomplete information such as the exam venue.
- Students felt that the most important consideration while preparing the timetable is to have a uniform distribution of exams over the examination period.

The main objective of the invigilation questionnaire was to ascertain lecturer preferences for their invigilation schedule. The questionnaire suggests that:

- Invigilators prefer to have between 2 - 3 invigilation duties with a one or two day gap between invigilation duties
- Lecturers with other responsibility such as administrative work or research work should be given a reduced number of invigilation duties.

Some weaknesses of the previous schedule as reported by UiTM respondents were:

- Too many chief invigilator assignments (2 out of 3).
- Time clash and last minute changes.
- There are not enough gaps between invigilation duties.
- The invigilation duty is too soon after the lecturers' own paper. This does not allow the lecturers enough time to do their marking and submit their grades within the required time.

3. Examination Timetabling at University Technology MARA (UiTM)

University Technology MARA (UiTM) is the largest university in Malaysia with a total number of students approaching 100,000. The university has 13 branch campuses, one in every state in Malaysia with 144 programs offered by 18 faculties. All the students take the same examination papers and therefore have the same examination timetable.

The Center for Integrated Information System (CIIS), UiTM has the responsibility of planning and managing the overall Information Technology strategy in UiTM and fulfilling administrative and academic needs. The Examination Unit at the Office of Academic Affairs liaises with the Examination Unit from CIIS to manage the preparation of the examination timetable.

Wan Ya and Baharudin (2001) from UiTM uses an examination scheduling program (developed in house about 30 years ago) to produce a first draft of the examination timetable. The total number of students enrolled in Semester 2, 2000/2001 was 94,502, course enrolment was 548,243 and total number of courses to be scheduled

was 2,458. The timetable went through a manual update process by scheduling new courses and removing old one. Apart from the constraints that are common for examination timetabling (Carter 1986; Carter and Laporte 1997; Schaerf 1999), UiTM has to consider the following additional constraints:

- If an exam falls on a state public holiday and there are students from that state sitting for the exam then the exam must be moved to another slot. Malaysia has quite a number of public holidays that are not shared between states.
- Minimise weekend schedules (Friday for certain states and Sunday for other states).
- Data on available space for exams were not considered when moving exams and therefore every faculty, center and branch campus needs to give prompt feedback on space availability for exams already scheduled.

Once the examination timetable is in its final draft, it is sent to all faculties and branch campuses for rooms and invigilators assignment. Some additional constraints for room assignments are:

- Students sitting for the same exam may be assigned to different rooms depending on their group sections.
- To fully utilise available space and to reduce the number of invigilators, more than one exam may be assigned to the same room but the exams should begin and finish at the same time.

With regards to invigilation, data collection is quite difficult when the invigilators are also lecturers serving other faculties. More than one faculty may require the service of the same lecturer and when faculties do their own invigilator scheduling, some clashes may occur across faculties

For invigilator assignment, the examination committee must consider the following constraints:

- Invigilators are not assigned to more than one room in a given time slot.
- Invigilators are assigned fairly i.e. equivalent duties for all invigilators.
- Invigilators do not invigilate their own papers.
- Projects presentation and evaluation are also scheduled during examination period and lecturers should not be scheduled during the project evaluation period.

4. Conclusion

From the survey we have seen that students and invigilators are not satisfied with their current timetable and they would like to see more work being done on improving the gap between two examinations and invigilation duties. Automating the process of timetabling can avoid clashes but it may not take into account other desirable

constraints. Lecturers who have to invigilate find it very inconvenient and would prefer to have hired personnel to do the invigilation.

The examination timetabling problem at UiTM was presented and due to its number of branch campuses, number of students and number of faculties, the Center of Integration Information System was established to help in the integration of data. Personnel need to be trained and located at every branch campuses to deal with the collection of data and manage the rooms and invigilators assignment. The manual process of assigning invigilators and rooms may involve many man hours for all faculties and branch campuses. Thus, it is highly desirable for UiTM to have a computer-based system to be deployed over all campuses.

References

Burke E.K., Elliman D.G., Ford P.H. and Weare R.F. (1996), "Examination Timetabling in British Universities - A Survey",The Practice and Theory of Automated Timetabling (eds EK Burke and P Ross), Lecture Notes in Computer Science Vol. 1153, Springer,pages 76-92.

Carter M.W., Laporte G. (1997), "Recent Developments in Practical Examination Timetabling", The Practice and Theory of Automated Timetabling", E.K. Burke and P. Ross (eds), Springer-Verlag, Berlin, pp 3-21

Carter M.W. (1986), "A Survey of Practical Applications of Examination Timetabling Algorithms", Operations Research Society of America, Vol 34 No 2 March-April.

Comm C.L. and Mathaisel D.F.X (1988), "College Course Scheduling. A Market for Computer Software Support,", Journal of Research of Computing in Education, vol 21, ppp 187-195.

JTAP (1998), "Central Timetabling By Computer: A review of existing information" Report by JISC Technology Applications Programme, June 1998 url: http;//www.jtap.ac.uk

Schaerf A. (1999), "A survey of automated timetabling", Artificial Intelligence Review. n.13, pp 87-127.

Wan Ya and Baharudin N. 2001, Interview with Manager and System Analyst, Examination Unit, Center for Integrated Information System, University Technology MARA, August 2001

A Review of Existing Interfaces of Automated Examination and Lecture Scheduling Systems

Barry McCollum[1], Samad Ahmadi, Edmund Burke[2] Rossano Barone, Peter Cheng[3], Peter Cowling[4]

[1]School of Computer Science
Queen's University
Belfast
University Road
N.Ireland
BT7 1NN
b.mccollum@qub.ac.uk

[2]Automated Scheduling and planning Group
School of Computer Science and IT

[3]School of Psychology
University of Nottingham
Nottingham, UK

[4]Department of Computing
University of Bradford
Bradford BD7 1DP
UK

The development of automated examination and lecture scheduling systems currently available has tended to be concentrated upon the construction of a solution by automating the elimination of constraints by algorithmic means. The interface presented to the user often appears to be almost an afterthought. Little recognition is usually shown to the fact that significant work may still have to be done to ensure that the solution can be realized as a working timetable. This would usually involve a series of drop and drag operations. Often, for instance, certain constraints on particular events may have to be relaxed or hardened after the construction of a solution for sound institutional reasons [1,2,3]. Also, on the occasion when exams are unable to be scheduled by the automated process, little information is available on where they can be placed in the finished solution. As a substantial amount of time and effort within institutions are devoted to this manual intervention in the construction of a timetable [3,4] it is quite surprising that software producers in this area have not provided more tools available for the manipulation of the final results.

In the authors opinion there are four pieces of software which are the market leaders in this area: Optime, Facility CMIS, Syllabus Plus and Celcat. All four market leaders in the University Sector provide standard interfaces to the examination and lecture scheduling systems with the inclusion of colour coding or the use of icons to highlight clashes and other types of constraint violations. Similar techniques are employed to alert the user as to where events could be rescheduled by hand while violating the least number of constraints.

Optime employs a series of icons which alerts the user to whether an event has been scheduled without compromising the associated constraints[5]. If the event is to be moved to another timeslot, simply dragging the module over that timeslot will show what degree of difficulty the user will have in trying to reschedule.
Within Facility, level of difficulty can be viewed visually. That is, the background of the timetable will change colour depending on how difficult it would be to move the selected event to a specific timeslot. In addition, it is possible to select an event and tell the system to display all the alternative timeslots in which the event can be placed. If the automation process has been exhaustive, this situation will not arise leaving the user to decide how the timetable should be restructured to allow a particular event to be rescheduled.
A similar technique to the above is apparently used by Scientia's product [6], syllabus plus though a recent copy has not been evaluated. Knowledge in this case is gained by the authors first hand experience of the product during his tenure as timetabling officer at the University of Nottingham 1994-97.

Celcat is unique among the software as it has limited capability to automatically produce a solution, leaving the majority of the work in the creation of the timetable to the user from an early stage in the process.
With all four products, as with freeware available on the web, it is not transparent to the user how he/she can manipulate the presented solution once the scheduling engine has completed its work. Simply showing the difficulty of rescheduling an event is of limited use as it leaves the responsibility to the user for deciding which constraints to relax or harden. In the event where no compromise is possible, the only solution is to choose an area in the schedule and try and eliminate the constraint violations which would occur if rescheduling took place. This is fraught with difficulty as once again it is the responsibility of the user to remember which constraints effect which event. In addition, by moving another event to allow the current event under investigation to be rescheduled, there is always the possibility of creating new constraint violations elsewhere. This task quickly becomes impossible for all but the most experienced expert timetablers and even then limits exist.

The full paper will provide an overview of the visual capabilities of the software mentioned here in relation to producing a workable institutional timetable from the automatically produced solution. This will be compared with the interface developed under the project funded by the ESRC/EPSRC "People at the Center of Communication and Information Technologies Programme (PACCIT)" entitled Representational Design Principles to Huminise Automated Scheduling Systems.

1 EK Burke, DG Elliman, PH Ford and RF Weare. "Exam Timetabling in British Universities – A Survey" in the Practice and Theory of Automated Timetabling, e.d. EK Burke and P Ross, Springer-Verlag (Lecture Notes in Computer Science), 1996. Department of Computer Science, University of Nottingham, UK.

2 M.W.Carter, G.Laporte, "Recent Developments in Practical Examination Timetabling", Selected papers from the 1st International Conference on the Practice and Theory of Automated Timetabling. PATAT95, Springer Lecture Notes in Computer Science Vol 1153, pp 22-45. Burke & Ross, eds., 1996

3 M.W.Carter, G.Laporte, Recent Developments in Practical Course Timetabling, Selected papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling. PATAT97, Springer Lecture Notes in Computer Science Vol 1408, pp 3-19. Burke & Carter, eds., 1998

4 McCollum B.G.C.,The Implementation of a Centrally computerised timetabling system in a large British Civic University, Selected papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling. PATAT97, Springer Lecture Notes in Computer Science Vol 1408, ISBN 0-7727-6703-3, pp 205 –215.

5 B McCollum, J Newall, "Introducing Optime: Examination timetabling Software", Proceedings of the 3rd international conference on the Practice and Theory of automated timetabling August 16-18 2000, pp 485-490, ISBN 3-00-003866-3,

6 The Syllabus Plus Exam Scheduler: Design Overview and Initial Results Richard Barber and Geoffrey Forster, Scientia Ltd, Cambridge, UK, Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling. PATAT97.

Integrating human abilities and automated systems for timetabling: A competition using STARK and HuSSH Representations at the PATAT 2002 Conference

Samad Ahmadi[2,3], Rossano Barone[3,2], Edmund Burke[2], Peter Cheng[3],
Peter Cowling[1], Barry McCollum[4]
[1]Corresponding author. MOSAIC research group, Department of Computing,
University of Bradford, Bradford BD7 1DP, UK,
Peter.Cowling@scm.brad.ac.uk
[2]ASAP research group, School of Computer Science & IT, University of Nottingham,
Jubilee Campus, Nottingham NG8 1BB,UK,
{sza, ekb}@cs.nott.ac.uk
[3]CREDIT research group, School of Psychology, University of Nottingham,
Nottingham NG7 2RD, UK,
rb@psychology.nott.ac.uk, peter.cheng@nottingham.ac.uk
[4]School of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, UK,
b.mccollum@qub.ac.uk

Abstract. Creating effective examination timetables in practice is a difficult task, and while automation can bring enormous benefits, it is still essential that a human expert can modify timetables and have input into their creation, to handle constraints, objectives and trade-offs between them which are unmodellable or which it is not economically feasible to accurately model. In this paper we present the STARK (Semantically Transparent Approach to Representing Knowledge) representation for examination timetables, which uses principles of representational design, from the cognitive science research community, to present timetables in a way which minimises the cognitive load associated with understanding and editing them, and the HuSSH (Human Selection of Scheduling Heuristics) approach which allows a human user to work at a high level, selecting the heuristics which will be used to modify solutions. We will conduct a competition at the PATAT conference, with two prizes of £100 (about 140 Euros) for the PATAT delegates who can produce the best solutions using the STARK system alone, and using the STARK+HuSSH system. This will allow us to capture invaluable data as to the way in which timetablers and timetabling researchers use the system.

1 Introduction

Creating timetables and schedules which are effective in practice is often a very demanding task. In many situations, where there are a number of complex, conflicting constraints and objectives [3] it is impossible or not economically viable to accurately model all of the possible factors which may occur, and produce a total ordering across all possible solutions, which then allows us to apply computer heuristics and

algorithms to generate a solution. In cases such as these, when an adequate set of measures is available, multiobjective approaches may be applied [2], to generate a number of solutions for comparison. When such a set of measures is unavailable, or when the comparison itself is too difficult, we need to have some way of effectively representing candidate solutions so that the user can modify them and choose between different candidates.

The task of understanding and modifying candidate solutions is cognitively very demanding. A great body of research in the cognitive science community suggests that effectively representing problem data is critical [1]. Using an appropriate representation can make problems many times easier to solve [11]. A major current challenge for the cognitive and computer science communities is the design of effective representations for knowledge-rich domains. Effective representations of schedules should aim to consistently match information dimensions, such as time, space, and degree of constraint violation, with visual dimensions such as spatial arrangement, size and colour. Our work has brought together expertise in representation design from cognitive scientists, expertise in heuristic design and modelling from computer scientists and timetabling specialists to devise an effective representation for the complex problem of examination timetabling, the STARK diagram [6], which we discuss in more detail below. The STARK diagram aims to facilitate both qualitative evaluation and editing of timetables. [6] included a pilot study to compare user performance when using a STARK representation and a more conventional graphical representation. The study demonstrates that users are able to much more effectively edit solutions which are represented using STARK diagrams.

Commercial systems for examination timetabling now offer sophisticated graphical user interfaces. Systems such as Optime, Facility CMIS, Syllabus Plus and Celcat represent timetables using colour coding and a mix of iconic and numerical information, and provide tools such as drag and drop for editing solutions. However, these representations do not always consistently match information and spatial dimensions in a way which supports human cognition. A typical example where information and visual dimensions are not related would be when representing time using both x and y dimensions (e.g. days down the y axis and times across the x axis), whilst providing only a numerical representation of space (e.g. student seating capacity).

There has been much interest in recent times in the idea of general-purpose "good enough soon enough" approaches for complex optimisation problems arising in practice. One approach which shows promise is the hyperheuristic idea [8], where instead of generating solutions to a given problem directly, we model a problem using a range of simple, low-level heuristics, and one or more measures of solution quality. We then apply a hyperheuristic which chooses at each decision point which low level heuristic to apply based upon the characteristics of each low level heuristic and the characteristics of the region of the solution space currently under exploration. These ideas have been explored in [10] for a complex scheduling problem in poultry management and [7], [8], [9] for timetabling and personnel scheduling. The HuSSH representation allows the user to control the evolution of a hyperheuristic approach, giving the user a range of familiar buttons and sliders to control a range of heuristic

approaches which partially unschedule or reschedule the current solution, in addition to providing the STARK representation and editing tools.

At the PATAT conference, we are grateful to the UK's Economic and Social and Engineering and Physical Sciences Research Councils for sponsoring a demonstration of our work and a competition where expert timetablers and designers of timetabling models, heuristics and algorithms will be able to use the STARK and HuSSH representations to solve timetabling problems of different sizes. Our principal aim is to gain feedback from users as to their effectiveness, as well as gather data as to how they are used in practice. In addition, we will offer a cash prize of £100 (about 140 Euros)and a trophy, for the user who produces the best solution using each of the STARK and HuSSH representations. We aim to make the two representations, together with test problems, available on the web prior to the conference for interested delegates to provide initial feedback and to try them out.

2 Principles of representational design

Existing approaches to the design of visual interfaces in the field of human computer interaction tend to focus on the types or dimensions of information used in the task [12]. In timetabling this would include attributes of timetable slots and exams (e.g., capacity, duration, number of students). The general aim is to match the identified types of information with appropriate visual dimensions; for instance, using discrete visual features for different categories of things or continuous visual properties for variable quantities.

Our approach is similar in that it recommends that interface design focuses on making effective mappings between the domain to be represented and the display doing the representing. However, our approach differs as it claims that it is most appropriate to consider mappings of the structure of the knowledge of the domain, at a higher and semantically richer level than the basic informational categories or dimensions. Knowledge of a domain encompasses more than the primitive classes of information and includes, for example: basic relations amongst information classes and dimensions (e.g., spare capacity of a slot = size of room – sum of sizes of exams allocated to the room); higher order relations of relations (e.g., classes of constraint violations); prototypical and extreme cases (e.g., completely filled slot); alternative perspectives (e.g., available resources versus demands to be met). Our thesis is that by capturing the underlying structure of the knowledge of a domain in the inherent structure of a representation, that representation will more naturally support understanding, problem solving and learning of the domain.

This thesis was developed in our research on designing novel representations to improve cognition and instruction in conceptually demanding topics in science and mathematics, such as electricity and probability theory [4]. By inventing new representations and contrasting them with the use of extant notational systems (e.g., algebra), principles of the design of representations for knowledge rich tasks have been derived [6]. There are six principles in two groups, which address how the meaning of a domain should be made apparent in the structure of the representation (semantic transparency) and how the representation should provide efficient

procedures for problem solving (syntactic plasticity). The semantic transparency principles were particularly important in the design of the STARK diagram. These three principles posit that an effective representation should: (1) integrate the different levels of abstraction of the domain, by using the same representational structure to simultaneously encode concrete cases and the underlying laws or rules of the domain; (2) integrate different perspectives of the domain by support alternative interpretations within a single representational format; (3) provide a coherent interpretative scheme for global features of the domain that consistently makes local low level distinctions.

STARK diagrams will be described below. [6] considers how the structure of the representation allows it to satisfy the principles outlined above.

3 The examination timetabling problem

The examination timetabling problem is a complex and interesting problem for representational and heuristic design. Practical instances of this (NP-hard) problem cannot usually be solved to optimality, and a wide range of heuristics have been applied. We will not attempt to survey the very wide range of approaches which have been applied, and we are confident that there will be further novel approaches in these proceedings. For further information see, for example, the survey of [3]. The problem is complex and a wide range of constraints and objectives are considered in practice. However, the problem is relatively stable, with the data for the problem known to a high level of accuracy in advance of solution, the time available to generate a timetable is generally relatively high, and the problem has a range of models which possess certain common information dimensions which we may attempt to represent visually, as discussed in the next section.

4 The STARK representation

The STARK diagram's representation of timetables is illustrated in figure 1. We represent time using a single spatial dimension (the x-axis) and space using another (the y-axis). Empty timetable slots are represented naturally as a blue rectangle, a possible timeslot in a given room has both a duration and a capacity. A yellow rectangle placed within this blue rectangle represents a timetabled examination, with a particular duration and size. Hence we can immediately see if there is insufficient time or space, since the yellow rectangle will extend outside the blue rectangle, if it is too long or too large. Constraint violations involving the same students are denoted using straight lines which join the left hand side of two examination rectangles, with the number of students involved in the clash determining the position of the end point on each examination. The distance down each examination rectangle where the line commences corresponds to the number of students affected. Hence a clash which requires a student to be in two places at one time will be represented by a vertical line connecting two examinations in the same period, and one which requires a student to

sit consecutive exams will be represented by a sloping line joining examinations in adjacent periods. Constraint violations involving a particular time or room, or a preferred ordering of examinations, are represented by lines joining the centre of an examination and a particular time, room or another examination.

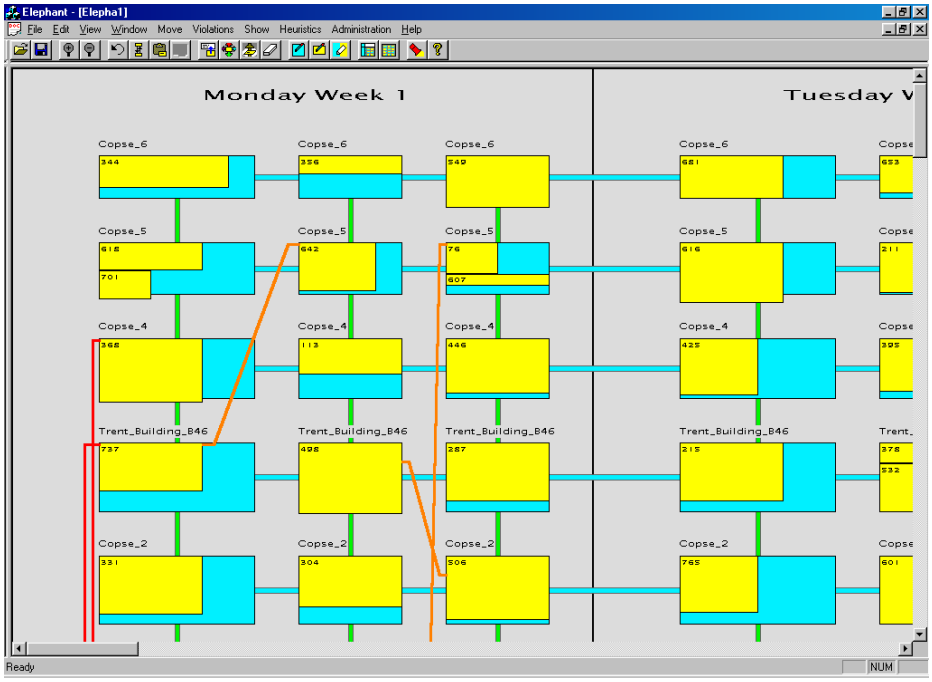

Fig. 1. STARK representation of timetables and constraint violations.

5 Integrating human abilities and automated systems

In [6] an experiment was conducted to compare the performance of a test group of subjects using STARK and using more conventional representations to modify an existing timetable and improve it according to fixed objective criteria. While the test group (of six subjects) was small, the results showed a marked superiority for those who used the STARK representations over those who did not, even for one test subject using the STARK representation who initially made the solution much worse before any improvement was made. The HuSSH system aims to allow user interaction at a higher level, by allowing the user to specify, via an easy-to-use graphical interface, which of a range of heuristics to use at each step. The user will be able to specify which of a range of constructive heuristics to use to augment a partial examination timetable, and which unscheduling heuristic should be used to remove a portion of poorly-scheduled examinations from the schedule, using an interface as shown in figs. 2 and 3. For example, when scheduling or unscheduling exams, the

user can select exams ordered at random, by decreasing penalty for student clashes, or by increasing number of available periods where they may be scheduled without penalty. When scheduling periods may be selected according to user-controlled weights assigned to each objective parameter, and rooms may be assigned either according to best fit, to the largest room first, or at random. When unscheduling the user can control the number of exams to be unscheduled by adjusting a parameter which determines by how much we wish to reduce the penalty.

Our objectives in giving the user a sophisticated interface for controlling the heuristics are two-fold. First we would like to see whether a typical user can use such an interface effectively, and here we consider that there may be a marked difference in the performance of heuristic designers and timetablers, since the former will have a greater degree of familiarity with the heuristics under user control. Secondly, we hope that analysis of creative use of the heuristic control interface will allow us to deduce new and effective strategies for using these heuristics, and that these approaches may be used within a hyperheuristic framework which increases the degree of automation in heuristic choice. In effect, we would like to design a hyperheuristic which imitates the better features of system use (and hence human heuristic design).

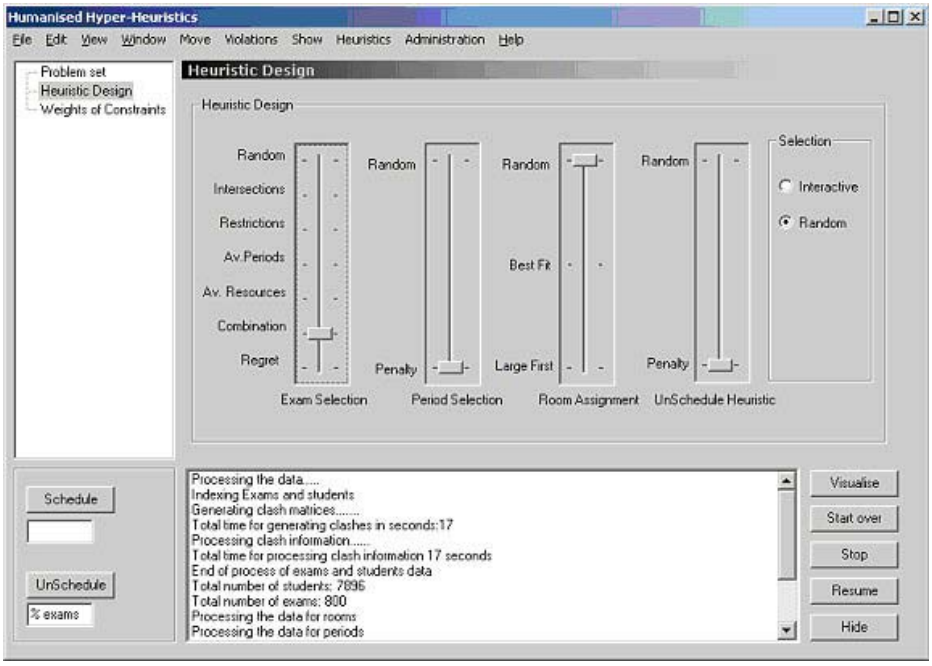

Fig. 2. The HuSSH heuristic choice interface.

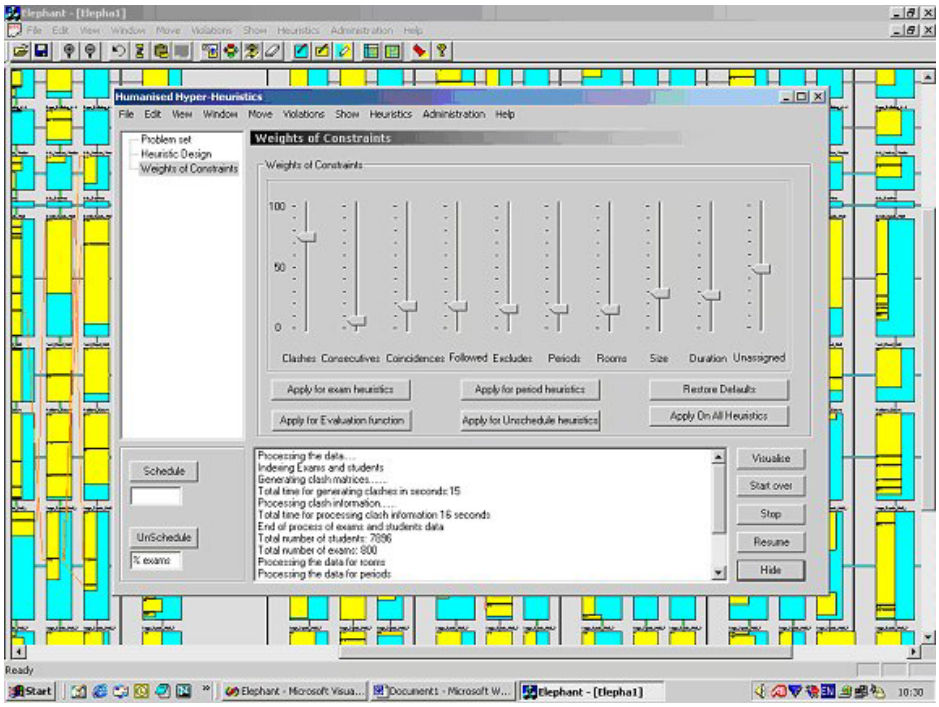

Fig. 3. The HuSSH heuristic parameters interface.

6 A competition to assess the effectiveness of our representations

At the PATAT conference, we will give delegates the opportunity to use the STARK and HuSSH representations and interfaces on problems of varying difficulty. A prize of £100 and a trophy will be given to the delegate who provides the best performance using each system (in terms of solution quality, measured by the objective function, and time taken). We will used fixed and known weightings of objective criteria to allow the user to direct the search. This abstracts the real situation, where the user would generally wish to resolve inaccuracies in the model, but will provide a criterion for PATAT delegates to evaluate the solutions which they create.

We aim to make each of the systems available in advance on the Internet for delegates to hone their skills. We will capture data concerning every mouse click and key press that delegates make, and cross reference this with a short anonymous questionnaire which will provide information about the background of the delegate who is using the system, and in particular the level of familiarity with the problem domain, and with the heuristic design process. This will give us a great deal of information about the types of user who can effectively use the system. We will also

capture feedback from delegates who use the system, which will be used to enhance the system further.

The experiment will allow us to capture data as to how the STARK and HuSSH tools are used, and whether they are too complex, or need to be modified in some way to make them more useful. We will capture a wealth of data on what timetablers and heuristic designers do when using a user interface, in general, and the STARK and HuSSH systems, in particular. In addition, we hope to disseminate our ideas for representational design and the integration of human cognitive abilities and automated timetabling approaches and obtain feedback from the research and user communities.

7 Conclusions

Our experiments so far, and informal feedback from timetablers and timetabling researchers suggest that the STARK and HuSSH systems are easy to use and effective in presenting solution data and refining solutions. We hope that the experiment at PATAT will provide feedback from a wider range of timetablers and timetabling researchers, and that the competitive element and the prize money will motivate PATAT delegates to try as hard as possible to generate good solutions, so that we can capture data on the way in which the system is used, and use that data to refine the systems, and learn about the processes of human timetablers.

Please come along to our stand in the conference foyer.

Acknowledgements

This research is funded by an Economic and Social Research Council (ESRC)/Engineering and Physical Sciences Research Council (EPSRC) People At the Centre of Communication and Information Technologies (PACCIT) grant number L328253012.

References

1. M. Anderson, P C-H Cheng, V. Haarslev (eds.) (2000) Theory and Application of Diagrams: First International conference, Springer LNAI vol. 1889.
2. E.K.Burke,Y.Bykov and S.Petrovic (2001) A Multi-Criteria Approach to Examination Timetabling. In E.K.Burke and W.Erben (eds.): The Practice and Theory of Automated Timetabling III: Selected papers (PATAT 2000). Springer LNCS vol. 2079, 118-131.
3. M.W. Carter and G. Laporte (1996) Recent developments in practical examination timetabling. In: E.K. Burke, P. Ross (eds.): The practice and theory of automated timetabling: Selected Papers (PATAT95). Springer LNCS vol. 1153, 3-21.
4. P. C-H. Cheng (1999). Unlocking conceptual learning in mathematics and science with effective representational systems. Computers in Education, 33(2-3), 109-130.

5. P. C-H. Cheng, R. Barone, P.I. Cowling, S. Ahmadi (2002). Opening the information bottleneck in complex scheduling problems with a novel representation: STARK diagrams. To appear in M. Hegarty, B. Meyer, & N. H. Narayanan (Eds.), Theory and Application of Diagrams: Second International Conference, Diagrams 2002, Springer LNAI.
6. P. C-H. Cheng, (2002). AVOW Diagrams Enhance Conceptual Understanding of Electricity: Principles of Effective Representational Systems. Forthcoming in Cognitive Science.
7. P. Cowling, L. Han, G. Kendall, (2002a) An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. To appear in Proceedings of CEC2002, IEEE Computer Society Press.
8. P. Cowling, G. Kendall, E. Soubeiga, (2001) A Hyperheuristic Approach to Scheduling a Sales Summit, In E.K.Burke and W.Erben (eds.): The Practice and Theory of Automated Timetabling III: Selected papers (PATAT 2000). Springer LNCS vol. 2079, 176-190.
9. P. Cowling, G. Kendall, E. Soubeiga, (2002b) Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation, to appear in proceedings of EvoCOP2002, Springer LNCS.
10. E. Hart, P. Ross, J. Nelson (1998) Solving a real-world problem using an evolving heuristically driven schedule. Evolutionary Computation, 6, 61-80.
11. K. Kotovsky, J.R. Hayes, H.A. Simon (1985). Why are some problems hard? Cognitive Psychology, 17, 248-294.
12. J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey (1994). Human-Computer Interaction. Addison-Wesley, Wokingham, UK.

Whose fault is it anyway?

R.C.Rankin

School of Computing
Napier University
10 Colinton Road
Edinburgh EH10 5DT
UK
rcr@dcs.napier.ac.uk

Abstract. Technical correctness is a necessary but not sufficient condition for the proper use of automatic timetabling software. Such software is used as part of a larger, often complex, system. Other errors in that system may be wrongly attributed to the software. This paper reports a study into sources of errors and user perceptions of their causes.

1 Introduction

Real timetables are seldom "perfect". They are the end product of a complex process extending over several months. The process operates on data which grow incrementally and usually change during that time. Errors may be introduced at all stages. Where an automatic timetable generator is used, it is only one part of this process.

This paper investigates the perceptions which users have of the quality of real timetables, the types of errors they find, and to what they attribute these errors.

Current findings, based on informal discussions with users, suggest that the automatic timetable generator is wrongly identified by users as the cause of errors which actually arise elsewhere, often from imperfections in the source data.

2 Background

Napier University is located on 9 campuses on the west of Edinburgh. It has approximately 12,500 enrolled attending 171 programmes. The timetable for one recent semester[1] contained 1608 events[2], 2037 student groups, 307 rooms and 639 lecturers.

[1] The winter semester of session 2001/2002
[2] These require to be timetabled by the software. There were a further 1,300 fixed events.

This timetable was generated using two internally developed software packages: "Neeps" for data collection, and timetable display and distribution, and "Tatties[3]" for automatic generation of timetables. This software has been in use for several years. Its initial development, use in a real department and extension for University wide use, has been described elsewhere, for example see [1], [2], [3], [4], [6].

Informal contact with the user community has revealed a variety of attitudes to the timetables issued. Examples of this are complaints that particular staff timetables were unacceptable. When investigated it was found that these timetables breached the hard constraints[4] of the software, and can be shown to have been the consequence of manual editing of timetables. However, the recipients of these timetables often believe they were caused by the timetabling software. This can have political and managerial consequences. An example is that improvements in the accuracy and timeliness of data collection, and in the care taken in post-generation editing [6], are unlikely while inaccurate evaluation and diagnosis of underlying causes persists.

It was decided that the extent and nature of errors, and the attitudes of users to them should be investigated.

3 The method

Three categories of "user" are in process of being surveyed, using questionnaires adapted for each group:

1. Managers who have no direct responsibility for producing timetables and whose daily activities are not directly influenced by a timetable.
2. Administrators who produce the data from which timetables are generated.
3. Academics whose daily activities are directed by timetables, or at least strongly influenced by them. There will be a few individuals who belong to both groups 2 and 3.

The purpose of the questionnaires will be to identify the types of errors which have been experienced by those individuals, what effect the errors have, and what is believed to be the source of them.

[3] **T**eachers' **A**id for **T**ime**t**abling using **I**nteractive, **E**volutionary **S**oftware. It is also a Scottish dialect word for "potatoes", and similarly "Neeps" means turnips. Hence "Turnips and Potatoes"

[4] For example, no break for lunch, or no inter-campus travel time.

4. Discussion

The "Neeps and Tatties" software has been used to produce working timetables since August 1994. The software was used by the author for three sessions to timetable only a single academic department, then in 1997 it was adopted by the University and operated centrally to produce class[5] timetables for the whole university. Its capabilities and features are well-understood, and it is accurate and reliable; i.e. it is relatively free of errors.

It has a number of fixed constraints: for example, it will not permit clashes[6], it automatically allows a lunch break, it automatically allows inter-campus travelling time.

From time to time training sessions in the use of this software are organised for both new staff and those newly assigned to timetabling activities. Discussions with such trainees about the nature of "good" timetables, and the effectiveness of the software, revealed an awareness of faults such as clashes of events, failure to provide agreed breaks, and overloading of lecturers and students.

The developers of the software can satisfy themselves that almost all cases are the consequence of errors or inaccuracies in the original data, or are caused by manual changes to a timetable after its production by the generator. In spite of this, end-users frequently assert that the software is the cause.

It is hoped that this study will permit a more accurate view of the real extent and nature of the errors and inaccuracies, so that these can be fed back into the system as a whole with a view to improving the quality of timetables.

References

1. Paechter B., Luchian H., Cumming A., and Petriuc M., "Two Solutions to the General Timetable Problem Using Evolutionary Methods", The Proceedings of the IEEE Conference of Evolutionary Computation, 1994
2. Rankin R.C., "Memetic Timetabling in Practice", Proceedings of the 2nd International Conference on the Theory and practice of Automatic Timetabling, Toronto, August 1995, pp336-9
3. Paechter, Ben, Rankin, R.C., Cumming, Andrew, "Improving a Lecture Timetabling System for University-Wide Use", Proceedings of the 2nd International Conference on the Theory and practice of Automatic Timetabling, Toronto, August 1997, pp336-9
4. Paechter, B., Cumming, A., Luchian, H., "The Use of Local Search Suggestion Lists for Improving the Solution of Timetable Problems with Evolutionary Algorithms", AISB Workshop in Evolutionary Computing, Sheffield, April 1995.

[5] That is, timetables for lecturers, student groups and rooms.

[6] A clash occurs when two or more events share a common resource at the same time.

5. Paechter, B., Rankin, R. C., Cumming, A, and Fogarty, T.C. “Timetabling the Classes of an Entire University with and Evolutionary Algorithm”, Parallel Problem Solving from Nature (PPSN) V, Springer LNCS 1498, 1998.
6. Cumming, Andrew., Paechter, Ben, Rankin, R.C., "Post-Publication Timetabling”, Proceedings of the 3nd International Conference on the Theory and practice of Automatic Timetabling", Toronto, August 1997, pp107-8

Timetabling at the University of Sheffield, UK - an incremental approach to timetable development

Simon Geller, Assistant Registrar, University of Sheffield,

285 Glossop Rd, Sheffield S10 2HB, UK
s.geller@sheffield.ac.uk

Abstract. The University of Sheffield adopted Small Project Management methodology to look at ways of enhancing the timetabling service it offers. The Project Group found that an incremental approach was needed to bring departmental timetablers into the system. There was a requirement to maintain departmental autonomy whilst also creating a pool of central held data which could be made available to all the stakeholders in the institution.

Traditionally, timetabling at Sheffield has been departmentally led. Departments have had a great deal of autonomy in creating their own timetables, and either a member of academic staff or a departmental administrator is allocated to deal with the timetable. Timetabling has therefore become something of a "black art", with individuals developing different methods of timetabling depending on their professional backgrounds, and concerns have arisen about the lack of a pooled knowledge base for timetabling.

With the introduction of modularisation the need arose for a central timetable, so that students could check times for a wide range of modules across departments.

At the same time, the introduction of new Oracle-based systems led to the need for an upgrade to the University Room Bookings system. Following a review of available options the decision was taken to purchase Facility CMIS, not least because of the "open system" approach taken by the developers which has enabled us to successfully integrate the timetabling software with other systems in use by the University. This package, supplied by CCM Software Services Ltd, is now the standard timetabling package for the University.

Our Service Level Agreement determines the centre's input to the timetabling process. We assemble the timetable for all teaching events that are part of a taught module, and take place in centrally pooled rooms.

It had become increasingly apparent that this arrangement does not fully cover the reporting requirements of the institution with regard to Management Information on teaching hours and room utilisation.

Also, because departmental timetablers did not have access to a central pool of information and the system did not hold critical information on student registrations, regulations and staff, it is possible for timetable clashes to arise.

Following a pilot scheme to determine whether departmental timetablers would find the Facility software useful, it was found if they were to start using it effectively a substantial amount of training and support, combined with a high-level directive that the software must be used for timetabling, would be needed.

We need to deliver complex course structures effectively, and this is an important factor in the overall learning experience in Sheffield for our students. If the aim of modularisation is to offer students a wider range of choices, then it is important that these choices can actually be delivered by the timetable. A course structure is meaningless without a schedule, and it is important that this schedule avoids such pitfalls as bunching events together, or forcing students to travel long distances between consecutive teaching events.

Our concerns were therefore that space was not being used efficiently, and that this affected the quality of the teaching timetable.

Although new space cannot be created by an automated timetabling system, it can help to avoid such problems through more efficient use of existing resources and by enabling an overview of timetabling so that under-used resources can be identified. However, the effective use of such a tool would require a change in the University's modus operandi for timetabling.

Options for change

We initiated a project using Small Project Management (SPM) techniques to look at options for developing the use of timetabling software. The following options were considered:

a) Set up a Central Timetabling Office.
b) Use of the Faculty structure.
c) Provide better access to timetabling functions via the Web
d) The "do-nothing" option.

Review of space utilisation

We looked at room utilisation figures to see whether these were a cause for concern, and if so make recommendations for improvement. We found that whilst overall room utilisation figures were poor, there was a high demand for certain types of rooms, most notable seminar rooms with a capacity of 20-60, and lecture theatres seating in excess of 500. A high level of discrepancy between room booking and room usage was also discovered.

Review of the available software.

We reviewed the main software options available to universities for timetabling. In the UK the three main products are Celcat, Facility CMIS and Syllabus Plus. The University had previously considered whether to write it's own timetabling software in-house, but had rejected this option on the grounds of resource availability.

Survey of Departmental Timetablers.

We conducted an on-line survey of our departmental timetablers, and the results may be summarised as follows:

Timetable complexity is a major problem.

The central room bookings office was blamed for the non-availability of rooms

There was a high level of interest in the use of automated timetabling software and enhanced Web services for timetablers.

Departmental timetablers did not have control over the processes used to create the timetable.

Results

As a result of the project we were able to make recommendations through the university's committee structure, which may be summarised as follows:

a) A formal review to be undertaken by Facilities Management of the use of departmental rooms.

b) Enhance timetabling Management Information.

c) Adopt an incremental approach to timetable development. Start with the largest departments; work closely with departmental timetablers to create a full departmental timetable. Ensure continued maintenance of the data. Monitor outcomes.

d) Monitor use of the software, and make provision to purchase further user licences should the need arise.

e) Purchase the latest version of the software to take advantage of advanced features for departmental timetabling.

f) Replace homegrown web applications with specialised proprietary software.

g) Issue of room notification request forms electronically.

Conclusion

To timetable successfully in a large institution an approach is needed that allows for departmental autonomy whilst also allowing the centre to collate and produce a central timetable. Software solutions should therefore be modelled on this approach, and implementation should be incremental, allowing for existing departmental timetabling procedures to be integrated with the central timetabling system.

Creating a new university timetable containing mixed structure types with (new) software

Theo de W Jooste

Potchefstroom University for Christian Higher Education,
Computer, Mathematical and Statistical Sciences, Private Bag X6001,
Potchefstroom, 2520 South Africa,
wsktdwj@puknet.puk.ac.za

This paper describes a case study in educational timetabling. It is based on work that had to be done to create a new teaching timetable for a university in South Africa, to enable a fundamental change in its teaching strategy. In this situation innovative steps had to be taken even though timetabling software were acquired. Knowledge of these measures could be of value to a wider audience as it underscores the utility of such software.

What I learned was that by suitably adapting the data necessary for student registration according to curricula specifications, it becomes possible to satisfy general as well as peculiar timetabling requirements with the aid of software that was designed to cater for basic automated timetabling. Developing a synergy between the reasoning power of the operator and the computing power of software and machine opens up amazing possibilities which suppliers of these software types would do well to support.

The fact that a newly developed package was used in this case meant that the developers were keen to learn from our implementation issues and were also willing to facilitate our data preparation and transformation processes, with mutual benefit arising from the cooperation. Their product, called **O!** *Automated educational timetable generation and optimisation software* was developed in South Africa around a kernel of optimisation software provided by the ILOG-group in France. They developed a compatibility interface with the ITS Administrative Software used by the majority of tertiary institutions in South Africa, but we use a different system. In our case they created a separate interface for importing data from files in Microsoft Excel format, which suited us well because our data came from completely restructured curricula with new module codes, and these were not available in the administrative system at the time.

Mixed degrees in which students major in more than one subject is at the order of the day in South African universities. In many cases our curricula consist of combinations of modules from a number of departments, even across faculty boundaries. Potchefstroom University is medium sized with about 8000 on campus students in 8 faculties. In addition to mixed degrees we also offer some specialized degrees, but even here the students have to take auxiliary courses from other faculties where they share lectures with students majoring in those subjects.

In response to a government directive towards restructuring of tertiary education, Potchefstroom University decided to redesign all its educational pro-

grammes to be explicitly outcomes based. From a timetable point of view this meant a drastic swing away from the previous static system which was based on modules of equal size (with respect to contact sessions per week). These modules were pre-grouped and the timetable itself consisted basically of time slots assigned to the groups, with venue allocation being done yearly on a desentralized basis.

On account of the magnitude of the changeover the new educational programmes have to be phased in, starting with the academic year in January 2002. (In South Africa academic and school years coincide with the calendar year.) This implied that two timetabling exercises had to be executed. Firstly, all of the newly designed curricula had to be evaluated to see whether they could be accommodated in a timetable of reasonable size, under some simplifying conditions. One significant outcome was that the timetable had to be enlarged from 40 to 48 sessions per week. The software was instrumental in determining this size since we could assume that its optimisation processes had packed in the modules as densely as possible.

Secondly, a real timetable for 2002 had to be constructed. This posed a challenge of another kind, because the new curricula would apply in this academic year only on first year level. The pre-grouped timetable structure had to be maintained for higher levels of the existing curricula. Moreover, there had to be synchronization between the two at a number of points.

The main contribution of this paper is a description of the data manipulation which enabled me to meet rather unusual timetabling requirements, and to simultaneously generate synchronized timetables for the two separate structures.

Given the fact that ours was its first large scale application the software proved itself to be of sound construction. There were some days when we had to search for the source of hiccups that occurred, but these mostly originated in the data. Small programming improvements were made to accomodate some of our requirements, especially with respect to locking and blocking actions. On the optimisation side, the effect of choosing mid-morning as the preferred scheduling time was clearly visible in the end product. Unless the various restrictions prevented that, a module was placed on or close to the mid-morning session. One very nice feature was that the timetable for the whole academic year could be generated in one go. The majority of modules run over a semester, but there are some quarter modules as well. The software kept everything apart, allowing clashes only where the type of module and curriculum specifications indicated that they would occur in separate quarters. However, one area where additional software aid would be welcome is in the analysis of the data when no solutions can be obtained in a timetable of preferred size.

In all instances the few problems that had to be solved by hand at the beginning of the first semester could be traced back to omissions or errors in the data. Solving these problems were also facilitated by the fact that combinations of the timetables for specific curricula could be selected from the complete timetable and viewed simultaneously on screen. The end product is a working timetable for our current academic year.

Complexity Issues

Flow Formulations for the Student Scheduling Problem

Eddie Cheng, Serge Kruk, and Marc Lipman

Department of Mathematics and Statistics
Oakland University, Rochester, Michigan, USA. 48309-4485
echeng@oakland.edu,sgkruk@acm.org,lipman@oakland.edu
http://www.oakland.edu/~kruk

Abstract. We discuss the student scheduling problem as it generally applies to high-schools in North-America. We show that the problem is NP-hard and discuss various variations to its formulation. We focus on multi-commodity flow problems because there has recently been much work and a number of interesting results on approximates solutions to such problems.

Key words: Educational Timetabling, Complexity Issues, Operational Research, Mathematical Programming, Scheduling, Multi-commodity flow.

1 Student Scheduling

The *Student Scheduling Problem*, (we use the terminology found in [3]) is the assignation of students to sections of courses offered at various times during the week. The objective is to fulfill student's request, providing them with a *conflict-free schedule* (no two assigned sections meeting at the same time), while respecting room capacities and possibly also balancing section sizes (or some other side constraint).

This problem is usually simple in a university setting because of the relatively sparse schedule of students. It may be, on the other hand, complex in a typical North-American high-school because students schedules are complete (every hour is accounted for) and because student requests for a given course are often binding. One of the authors has been involved for the past twenty years in the development of a commercial package for high-school scheduling. We are aware that the practical difficulty of the problem varies tremendously between schools or even between semesters in the same school. Under what may seem like similar conditions, the computing time may range from a few seconds to many hours using the same machine with the same software.

As reported in the survey paper [3], which includes only refereed articles describing algorithms that have been implemented, it seems flow models have not been used in this particular facet of the timetabling problem. There have been algorithms based on branch and bound followed by heuristic improvements

[12], some greedy approaches moderated by an intelligent ordering of the students [17], simulated annealing [4] and goal programming [16].

In view of the strides of approximation algorithms resulting from polyhedral theory during the past decade, it seems reasonable to revisit the problem and try to reformulate it with an eye towards such approximation algorithms [20]. We are thinking specifically of multi-commodity flow problems and variants [7, 2].

The input to the problem is the students's list of selected courses (as illustration, see Table 1) which we will refer to as the *selection*, and the *master schedule* of course offerings with their multiple sections, each with possibly a number of meeting times, rooms and instructors (see Table 2). The master schedule also

Table 1. Student course selection example.

Student	Name	Selections
1001	John Q. Student	101, 126, 134, 156
1002	Susan B. Bright	101, 126, 135, 158

contains course description, instructor names, rooms reserved, which may change for different meetings. It also contains the meeting times, which we show here encoded. They may be represented as days and times, or refer to some other table of equivalences as we illustrate here.

Table 2. Master schedule example.

Course	Description	Section	Meetings	Instructor	Room	Size
101	French	01	M8,W8,F8	Cheng	R101	15
		02	M8,W8,F8	Kruk	R201	15
		03	T9,R9,F9	Cheng	L101	15
126	Chinese	01	M8,W8	Lipman	L123	10

A few comments about constraints are in order. In this example, both students wish to take Chinese (126) along with French (101). The goal of the student scheduling problem will be to satisfy, if possible, both student selections by assigning them to non-conflicting sections, while maintaining the number of students within the maximum size prescribed (10 in the unique Chinese section and 15 in each of the French sections). The first constraint is a hard constraint: Assignments must not conflict in time (a student cannot be at two different places at once) and student requests must be satisfied if possible . The size constraint is a softer one: a valid solution minimizes the number of students above the prescribed limit but usually not at the expense of fulfilling selections, though

this can vary with schools. The system must usually assign as much of a student selection as possible, dropping some unsatisfiable selection if need be.

2 A Combinatorial Formulation of the Decision Problem

To make precise the problem described in the introduction, we now describe a combinatorial formulation. Table 3 establishes the notation.

Table 3. Notation for problem data.

Symbol	Interpretation
K,	the set of all students,
I,	the set of all courses,
C_k,	the set of courses selected by student $k \in K$,
S_i,	the set of sections of course $i \in I$,
T_{ij}	the number of meetings of course $i \in I$, section $j \in S_i$,
Z_{ij},	the maximum size of course $i \in I$, section $j \in S_i$.

We will also assume some preprocessing of the data to create a matrix of sections conflicting in time: For all $i \in I$, $j \in S_i$, $\bar{\imath} \in I$, $\bar{\jmath} \in S_{\bar{\imath}}$,

$$M_{ij\bar{\imath}\bar{\jmath}} = \begin{cases} 1, & \text{if course } i, \text{ section } j \text{ conflicts with course } \bar{\imath}, \text{ section } \bar{\jmath}; \\ 0, & \text{otherwise.} \end{cases}$$

This preprocessing is simple and done without loss of generality but could be avoided at the expense of a slightly more complex model. The decision variable is, for all $k \in K$, $i \in I$, $j \in S_i$,

$$y_{ijk} = \begin{cases} 1, & \text{if student } k \text{ is scheduled into course } i, \text{ section } j; \\ 0, & \text{otherwise.} \end{cases}$$

The first constraint indicates that to each course selected by a student corresponds exactly one section of that course.

$$\forall k \in K, \forall i \in C_k, \quad \sum_{j \in S_i} y_{ijk} = 1. \tag{1}$$

There are $\sum_{k \in K} |C_k|$ such constraints. We then need to enforce that there are no conflicts in the schedule of a student.

$$\forall k \in K, \forall i \in C_k, \forall j \in S_i, \quad \sum_{\bar{\imath} \in I, \bar{\jmath} \in S_{\bar{\imath}}} y_{\bar{\imath}\bar{\jmath}k} M_{\bar{\imath}\bar{\jmath}ij} \leq 1. \tag{2}$$

There are $\sum_{k \in K} |S_{C_k}|$ such constraints. Finally, we indicate that the maximum size per section must not be exceeded.

$$\forall i \in I, \forall j \in S_i, \quad \sum_{k \in K} y_{ijk} \leq Z_{ij}. \tag{3}$$

There are $\sum_{i \in I} |S_i|$ such constraints.

Now that we have a precise formulation of the problem, we give a simple proof of its complexity.

Theorem 1. *The student scheduling problem is NP-complete.*

Proof. We will consider the problem of finding a schedule for one student. The proof is by reduction to the *independent set problem* [6]: Given a graph $G = (V, E)$ and an integer n, is there is a set of nodes of size at least n, no two adjacent?

From an instance of the independent set problem, we construct a master timetable consisting of n different courses, each has $|V|$ sections (Say $s_1, \ldots, s_{|V|}$). Section i of all courses have the same meeting times which are constructed to satisfy the following condition: sections s_i and s_j have a meeting in common if and only if vertices v_i of G, are adjacent.

We assume one student with a selection comprising all n courses. Then we have a conflict-free schedule for the student if and only if G has an independent set of size at least n since a conflict-free schedule is a set of n sections (one per course), no two sharing a meeting time.

The construction of the meeting times can be done by a search over each node and its neighbours, hence is proportional to $|V||E|$. □

Notice that this establishes the difficulty of finding a conflict-free schedule for one student, with no room size constraints. The general problem for many students and size constraints is, perforce, not easier.

3 Flow Models of the Optimization Problem

The integer feasibility problem described above is interesting but of little practical value. One problem is that, even for a school of small size, the feasible set is empty, hence the problem, as stated, has no solution. There will almost always be students with unsatisfiable course selections. Therefore trying to solve the integer feasibility problem (1)–(3) yields no useful information on the schedules of all students that could see their selections satisfied. In practice the infeasibility of the problem should not detract from the (yet unstated) objective of providing as good a schedule as possible, for some measure of "goodness" for as many students as possible. In this section we present alternative models and show how various objectives can be accommodated.

The models are based on the integral multi-commodity flow problem. Each student represents the source (and sink) of a given commodity and the objective is to maximize the total flow. By Theorem 1, there must be additional constraints

since the problem is NP-hard for one student yet the integral flow problem with one source-sink pair is polynomially solvable. The additional difficulty is reflected in the following models either as non-integral capacities or as gains on a subset of arcs.

3.1 Fractional Capacities

We construct the first network in the following manner. (The labels set in `typewriter face` refer to Fig. 1 which illustrates the subnet of a single student). The first layer consists of a set of source nodes, one for each student (`student-s`), duplicated, in the last layer, as presink (`student-p`) and sink nodes (`student-t`). To the source is associated a supply value, equal to the number of courses selected by a student, with the corresponding demand at the student sink. These supply-demand pairs are of different commodities for each student. The objective is for a flow from student source i to go to student sink i and nowhere else. After duplication, there is a total of $3|K|$ of these nodes.

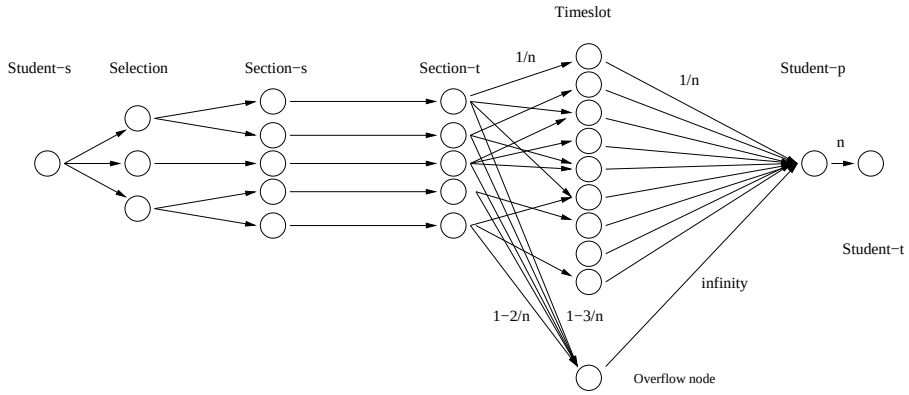

Fig. 1. Multi-commodity fractional flow model. Unlabeled arcs have unit capacity.

Concentrating now on the sub-network of one student, the first layer ends in a set of selected courses (`Selection`) with an arc of capacity one from the student source. There is such an arc and node if the student has selected the course. There are $\sum_{k \in K} |C_k|$ such nodes.

The second layer consists of the sections of each course duplicated, in the third layer, as source (`Section-s`) and sink (`Section-t`). The arc from a course to one of its section has capacity 1, while the capacity of a section-source to section-sink is equal to the maximum size of the section. Each of the sections of the selected courses of the each student are identified so that the flow from all students merge at this layer and diverge afterwards.

The section sinks have arcs to the next layer `Timeslot`, indicating the meeting times. If a section occupies k timeslots then the corresponding section sink node

has $k+1$ outgoing arcs. The first k have capacity $1/n$ where $n = |V_T|$, and the last has capacity $1 - k/n$ and ends at the overflow node. (Only two such arcs are labeled with their correct capacity to avoid clutter).

These timeslot nodes reach the student presink with capacity $1/n$, except for the arc coming out of the overflow node with capacity ∞. The presink reaches the sink with capacity n.

The resulting multi-commodity flow problem can be formalized in the following manner: Let $x_{ij}(k)$ be the measure of commodity k flowing on arc (i, j) where $k \in K$, the set of all students. Note that the various commodities are coupled only at the third layer. We therefore have a particularly well-structured multi-commodity flow problem. The notation of Table (4) will simplify the expressions.

Table 4. Subsets of the node set V.

Symbol	Set	Symbol	Set
V_K^+	`Student-source` nodes	V_K^{--}	`Student-presink` nodes
V_K^-	`Student-sink` nodes	V_S^+	`Section-source` nodes
V_S^-	`Section-sink` nodes	V_T	`Timeslots` nodes

Program (4) represents the formulation of a fractional multi-commodity flow problem.

$$\max \sum_{(i,j)\in A, i\in V_K^+} \sum_{k\in K} x_{ij}(k) \tag{4a}$$

$$\text{s. to} \sum_{(i,j)\in A} x_{ij}(k) - \sum_{(j,l)\in A} x_{jl}(k) = 0, \quad \forall k \in K, j \in V \setminus \{V_K^+ \cup V_K^-\} \tag{4b}$$

$$\sum_{k\in K} x_{ij}(k) \le Z_{ij}, \qquad \forall k \in K, (i,j) \in A, j \in V_S^- \tag{4c}$$

$$x_{ij}(k) \ge 0 \qquad \forall k \in K, (i,j) \in A \tag{4d}$$

Definition 1. *A flow represents a schedule in the sense that if the flow from some V_K^+ to V_K^- passes through a node V_S^+, the student is assigned to the corresponding section.*

Definition 2. *A flow is valid if it is feasible and if it is integral on all arcs with integral capacity.*

We are now is a position to state the useful properties of this model. For the rest of this section, *feasible* means feasibility with respect to program (4). The first result, stated without proof, is obtained from the construction of the model.

Lemma 1. *Given a master schedule, any set of conflict-free student schedules not exceeding section sizes is representable by a valid flow.*

It is possible to add selections to each student schedule (lunch period, homeroom periods, "free" time, etc...) in such a manner that every hour of every day is accounted for. If this is done, then a full schedule will use all V_T nodes we obtain the following easy result which we state without proof.

Lemma 2. *A student schedule is complete (every hour is accounted for) if and only if the arc* $(i, j), i \in V_K^{--}, j \in V_K^{-}$ *corresponding to the student is saturated.*

More important for our purposes since we intend to develop algorithms to solve program (4), is the following:

Lemma 3. *Any valid flow represents a conflict-free schedule.*

Proof. Consider a node timeslot node $t \in V_T$. To show the schedule is conflict free, we need to show that any flow incoming to t traveled via a single section node. Note that there is only one outgoing arc from t, of capacity $1/|V_T|$. Say there is any flow coming from section-sink node $j \in V_S^-$. We claim this flow has exactly value $1/|V_T|$, hence saturates the only outgoing arc from t. It has that value because any nonzero valid flow has unit value on arc (i, j) and by conservation of flow constraint (4b), must split into flows of value $1/|V_T|$ towards each timeslot with excess going to the overflow node. Therefore no other section can send flow into node t. □

3.2 From Fractional Capacities to Gains

The first network is correct but can be modified to eliminate the fractional capacities. The first effect of this modification is to allow standard integer programming software to be used to solve the modified model but there are some other benefits as well. Though there are standard transformations we could apply to the first model to use standard integer programming software, they involve many more additional binary variables.

The modification consists in the elimination of the overflow nodes and all incident arcs and the addition of a gain amplifier to the V_S^- node that multiplies the flow into the node by an integer equal to the number of arcs out of the nodes. Since these arcs are incident to the timeslots nodes, the effect is the same as with the overflow nodes, namely, picking-up all the timeslots corresponding to a given section. The capacity of these last arcs is 1. Figure 2 illustrates for one student.

This variation has fewer arcs and all capacities are integral and we will show that both networks are equivalent, so that that results obtained with the previous network are still valid. We give the complete formulation of the modified problem in (5) where $N(j)$ means the downstream neighbours of node j,

$$N(j) := \{i \in V \mid (j, i) \in A\},$$

and the gains are given by

$$T_{ij} := \begin{cases} |N(j)| - 1 & j \in V_S^+, \\ 1 & \text{elsewhere.} \end{cases}$$

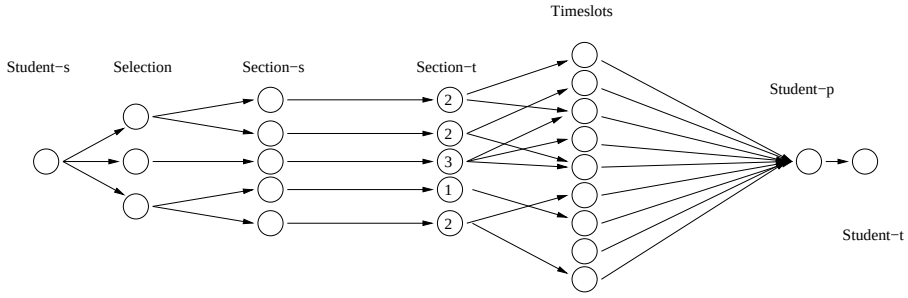

Fig. 2. Gain variation model. The node annotations represent the gains.

These gain values correspond to the fraction of the unit flow that previously was directed towards the overflow node in program (4).

$$\max \quad \sum_{(i,j)\in A, i\in V_K^+} \sum_{k\in K} x_{ij}(k) \tag{5a}$$

$$\text{s. to} \quad \sum_{(i,j)\in A} T_{ij}x_{ij}(k) - \sum_{(j,l)\in A} x_{jl}(k) = 0, \quad \forall k \in K, j \in V \setminus \{V_K^+ \cup V_K^-\} \tag{5b}$$

$$\sum_{k\in K} x_{ij}(k) \leq Z_{ij}, \qquad \forall k \in K, (i,j) \in A, j \in V_S^- \tag{5c}$$

$$x_{ij}(k) \geq 0 \qquad \forall k \in K, (i,j) \in A \tag{5d}$$

Lemma 4. *There is a one-to-one correspondence between a valid flow in program (4) (implicitly corresponding to the fractional capacitated flow model of Fig. 1) and a valid flow in program (5) (implicitly corresponding to the gain model of Fig. 2).*

Proof. Consider a node $j \in V_S^-$ with k neighbours in V_T. A incoming valid flow in the fractional model will have either value zero or unit value. Consider only the unit incoming flow. The k neighbours will each receive $1/|V_T|$ flow value with the excess $(1 - k/|V_T|)$ going to the overflow node. The corresponding valid flow in the gain model has the same unit flow incoming the node and the multiplier is k, ensuring that each of the neighbours gets a unit flow. □

Network with gains have been studied for a while [10, 13, 15, 8, 9], and some specialized solution techniques are known. One important property of this particular network is that there are no cycles, hence no endogenous flows.

Since we showed that both networks correctly model the student scheduling problem we have the following:

Theorem 2. *Let D be a directed network with both integral and non-integral capacities (or, equivalently, integral gains). Let s and t be two vertices in D. Let k be an integer. Then the question "does there exist a flow of integral value on all arcs of integral capacity, from s to t of value at least k" is NP-complete.*

3.3 Various Objectives

In both variations above, (4) and (5), we have written the objective function as

$$\max \sum_{(i,j)\in A, i\in V_K^+} \sum_{k\in K} x_{ij}(k).$$

This choice maximizes the number of courses assigned, a reasonable choice but not the only one. A slight modification yields

$$\max \sum_{(i,j)\in A, i\in V_K^-} \sum_{k\in K} x_{ij}(k),$$

which maximizes the number of timeslots assigned, hence the "occupation" level of the students.

Other possibilities are to minimize the number of students with incomplete schedules. This is also a reasonable objective since a student with an incomplete schedule, however incomplete will likely need to change his selections and therefore whatever was assigned will need to be de-assigned. This goal can be modeled with the following nonlinear objective function in the first network.

$$\max \sum_{(i,j)\in A, i\in V_K^-} \sum_{k\in K} f(x_{ij}(k)),$$

where

$$f(x) := (0-x)\left(\frac{1}{n}-x\right)\left(\frac{2}{n}-x\right)\ldots\left(\frac{n-1}{n}-x\right),$$

so that $f(x) = 0$ unless the flow in the sink layer is at capacity. This relies on the fact that the a maximum flow will be a multiple of $\frac{1}{n}$.

Another approach is to move the size constraints in the objective function. From the second network, eliminate constraint (5c) and let the objective be

$$\sum_{(i,j)\in A, j\in V_S^-} \left(\sum_{k\in K} x_{ij}(k) - Z_{ij}\right)^+ + \sum_{(i,j)\in A, i\in V_K^+} \sum_{k\in K} x_{ij}(k),$$

where the nonlinearity appears in the function

$$(x)^+ := \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

This formulation has the advantage of creating a separable multi-commodity flow problem, which is much more amenable to solutions via standard nonlinear convex programming techniques ([11, 14, 1, 18, 5, 19]).

Other variations could be developed to account for the various goals of distinct schools. For example, the distinction between a required course and an elective could be handled via a weight factor on each student selection.

4 Conclusion

Our intention in this paper has been to develop mathematically useful models for a difficult, important problem. In this context, difficult means NP-complete for even the simplest case. Important means that the problem occurs in its full complexity across North America thousands of times each year. Mathematically useful means that the flow models can be adapted to satisfy different optimization criteria (types of partial solutions) for a variety of applications, and further, that these problem formulations are amenable to attack by approximation methods. Since approximation algorithms for multicommodity flow problems are well-studied, and our network models are layered and coupled at only one layer, a sensible approach is to specialize existing algorithms to these models, taking advantage of the topology of the underlying networks. Experimentation has begun and will be reported in future work.

References

[1] Dimitri P. Bertsekas, Lazaros C. Polymenakos, and Paul Tseng. An ϵ-relaxation method for separable convex cost network flow problems. *SIAM J. Optim.*, 7(3):853–870, 1997.

[2] L. Brunetta, M. Conforti, and M. Fischetti. A polyhedral approach to an integer multicommodity flow problem. *Discrete Applied Math.*, 101:13–36, 2000.

[3] M. W. Carter and G. Laporte. Recent developmentsd in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, Lecture Notes in Computer Science, pages 3–19. Springer, 1998.

[4] L. Davis and L. Ritter. Schedule optimization with probabilistic search. In *Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications*, pages 231–236. IEEE, 1987.

[5] Renato De Leone, Robert R. Meyer, and Armand Zakarian. A partitioned ϵ-relaxation algorithm for separable convex network flow problems. *Comput. Optim. Appl.*, 12(1-3):107–126, 1999. Computational optimization—a tribute to Olvi Mangasarian, Part I.

[6] Michael R. Garey and David S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[7] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. of ACM*, 42:1115–1145, 1995.

[8] Richard C. Grinold. Calculating maximal flows in a network with positive gains. *Operations Res.*, 21:528–541, 1973.

[9] P. A. Jensen and Gora Bhaumik. A flow augmentation approach to the network with gains minimum cost flow problem. *Management Sci.*, 23(6):631–643, 1976/77.

[10] William S. Jewell. Optimal flow through networks with gains. *Operations Res.*, 10:476–499, 1962.

[11] John G. Klincewicz. A Newton method for convex separable network flow problems. *Networks*, 13(3):427–442, 1983.

[12] G. Laporte and S. Desroches. The problem of assigning students to course sections in a large engineering school. *Computational & Operational Research*, 13(4):387–394, 1986.

[13] M. Malek-Zavarei and J. K. Aggarwal. Optimal flow in networks with gains and costs. *Networks*, 1:355–365, 1971/72.

[14] Ángel Marín Gracia. Optimization of nonlinear, convex and separable, networks. In *Proceedings of the first international seminar on operational research of the Basque Provinces (Zarauz, 1986)*, pages 173–203, Bilbao, 1986. Univ. Pais Vasco.

[15] Jean François Maurras. Optimization of the flow through networks with gains. *Math. Programming*, 3:135–144, 1972.

[16] I. Miyaji, K. Ohno, and H. Mine. Solution method for partitioning students into groups. *European Journal of Operations Research*, 33(1):82–90, 1981.

[17] G.C.W. Sabin and G.K Winter. The impact of automated timetabling on universities - a case study. *Journal of the Operational Research Society*, 37(7):689–693, 1986.

[18] J. Sun and H. Kuo. Applying a Newton method to strictly convex separable network quadratic programs. *SIAM J. Optim.*, 8(3):728–745 (electronic), 1998.

[19] Paul Tseng and Dimitri P. Bertsekas. An ϵ-relaxation method for separable convex cost generalized network flow problems. *Math. Program.*, 88(1, Ser. A):85–104, 2000.

[20] V.V. Vazirani. *Approximation Algorithms*. Springer, 2001.

Sport Timetabling

Characterizing Feasible Pattern Sets with a Minimum Number of Breaks

Ryuhei Miyashiro[1], Hideya Iwasaki[2], and Tomomi Matsui[1]

[1] Department of Mathematical Informatics,
Graduate School of Information Science and Technology,
The University of Tokyo, Tokyo 113-8656, Japan
{miya, tomomi}@misojiro.t.u-tokyo.ac.jp
[2] Department of Computer Science,
The University of Electro-Communications, Tokyo 182-8585, Japan
iwasaki@cs.uec.ac.jp

Abstract. In sports timetabling, finding feasible pattern sets for a round robin tournament is a significant problem. Though this problem has been tackled in several ways, simple characterization of feasible pattern sets is not known yet. In this paper, we characterize of feasible pattern sets with a minimum number of breaks for up to 26 teams and propose a polynomial time algorithm for judging the feasibility of the pattern set.

1 Introduction

Constructing a timetable for a sports competition is an important task for organizers of the competition because the timetable affects not only the results of games but also the revenue of the competition. Since creating an appropriate timetable by hand is difficult, demand for automated timetabling have been increasing.

In this paper, we consider a single round robin tournament consisting of $2n$ teams and $2n-1$ slots. Each team plays against every other team once, and each game is held at the home of one of the teams playing. A game between teams i and j played at the home of i is called a *home game* for i and an *away game* for j.

A *pattern set* is a table indexed by teams and slots showing whether a team plays a home game or an away game in each slot. In a round robin tournament, the place where a game is held (home or away) can greatly affect the outcome of the game. Thus, the organizers often fix a pattern set first and then assign opponents to the pattern set to obtain a timetable [1][2]. Unfortunately, not all pattern sets can generate a timetable (Fig. 1).[3] A pattern set that can produce a timetable is called *feasible*, while one that cannot is called *infeasible*.

A *pattern set feasibility problem* determines whether a given pattern set is feasible or infeasible. Though we can solve this problem with integer programming [1], we know of neither a polynomial time algorithm that can solve this

[3] In Fig. 1, holding the three games among teams b, c and d in slots 1 and 2 is impossible.

	1	2	3	4	5
a :	A	H	A	H	A
b :	A	A	H	A	H
c :	A	H	H	A	H
d :	H	A	H	A	H
e :	H	H	A	H	A
f :	H	A	A	H	A

Fig. 1. An infeasible pattern set for 6 teams. 'H' means home, 'A' means away

problem nor simple characterization of feasible pattern sets. In the next section, we discuss the characterization of feasible pattern sets belonging to a particular class.

2 An Approach to Characterization of Feasible Pattern Sets with a Minimum Number of Breaks

In the remainder of this paper, we focus on a *pattern set with a minimum number of breaks* (*PSMB* for short). When a team plays consecutive away games or home games in slots s and $s+1$ in a pattern set, we say that the pattern set has a *break* at slot $s+1$. In general, organizers prefer a pattern set with fewer breaks because it improves the quality of the timetable.

A PSMB has $n-1$ slots at which a break occurs, and there are $\binom{2n-2}{n-1}$ PSMB, where $2n$ is the number of teams. In the following, we consider the feasibility of a PSMB. (Note that the following argument can be applied to an *equitable pattern set* described in [3], because any equitable pattern set is obtained by a cyclic permutation of slots in a PSMB.)

We represent a PSMB as a 0-1 sequence of length $2n-1$; the first element is 1, and the i-th element is 1 when the PSMB has a break at the i-th slot, otherwise 0 ($i = 2, \ldots, 2n-1$). We call this 0-1 sequence a *pattern set sequence*. This representation is based on the fact that each PSMB is uniquely characterized by the slots where breaks occur [3][4]. This sequence expression simplifies analysis of a PSMB.

There is a subsequence that makes a pattern set sequence infeasible. For example, it is easy to see that any pattern set sequence that contains the subsequence "111" is infeasible (Fig. 1).[4] We analyzed for such subsequences to judge the infeasibility of a PSMB.

We proved that if a given pattern set sequence contains a subsequence $\{q_i\}$ of length L that violates the following inequality, the sequence is infeasible.

$$\sum_{j=1}^{L} \min\left\{ \sum_{i=1}^{j} q_i\,,\ \sum_{i=1}^{L} q_i - \sum_{i=1}^{j} q_i \right\} \geq \binom{\sum_{i=1}^{L} q_i}{2}. \qquad (1)$$

[4] The pattern set sequence of Fig. 1 is "11100".

It is a necessary condition for a feasible PSMB that any subsequence of the corresponding pattern set sequence satisfies inequality (1). It takes $O(n^3)$ steps to check whether all subsequences of a given pattern set sequence satisfy inequality (1); in fact, it took less than 0.1 second to examine a pattern set sequence for up to 100 teams with an ordinary PC.

For each PSMB satisfying the above necessary condition, we formulated the pattern set feasibility problem as an integer programming problem and solved it using ILOG CPLEX 6.0.

Computational experiments showed that when the number of teams is less than or equal to 26, all PSMB satisfying the necessary condition are feasible. This means that the proposed condition is a necessary and sufficient condition for a feasible PSMB for up to 26 teams. For more than 26 teams, we did not perform computational experiments using integer programming because it would take too long to solve the problems.

We conjecture that, for any number of teams, our condition is a necessary and sufficient condition for a feasible PSMB. Though formal proof of this conjecture has not yet been obtained, our result is practical enough in terms of the number of teams because a round robin tournament with more than 26 teams is rarely held.

3 Conclusion

We considered the pattern set feasibility problem for a pattern set with a minimum number of breaks (PSMB). We proposed a necessary condition for a feasible PSMB, which can be checked in polynomial time. Computational experiments showed that the proposed condition is a necessary and sufficient condition for a feasible PSMB for up to 26 teams. We conjecture that, for any number of teams, the condition characterizes feasible PSMB. Proof of our conjecture remains for future work.

References

1. Nemhauser, G.L., Trick, M.A.: Scheduling a Major College Basketball Conference. Operations Research **46** (1998) 1–8
2. Schreuder, J.A.M.: Combinatorial Aspects of Construction of Competition Dutch Professional Football Leagues. Discrete Applied Mathematics **35** (1992) 301–312
3. de Werra, D.: Geography, Games and Graphs. Discrete Applied Mathematics **2** (1980) 327–337
4. de Werra, D.: Some Models of Graphs for Scheduling Sports Competitions. Discrete Applied Mathematics **21** (1988) 47–65

A Generate-and-Test Heuristic Inspired by Ant Colony Optimization for the Traveling Tournament Problem

Herman Crauwels[1] and Dirk Van Oudheusden[2]

[1] Hogeschool voor Wetenschap & Kunst, De Nayer instituut,
J. De Nayerlaan 5, B-2860 Sint-Katelijne-Waver, Belgium
hcr@denayer.wenk.be
[2] K.U.Leuven, Centre for Industrial Management,
Celestijnenlaan 300A, B-3001 Heverlee, Belgium
Dirk.VanOudheusden@cib.kuleuven.ac.be

In a sport competition, n given teams play against each other over a period of time according to a certain scheme. An example of such a scheme is double round robin. It determines that every team t plays twice against every other team: a *home* match when team t uses its own facilities and an *away* match when the match takes place at the facility of the opponent. When the number of teams, n, is even, and every team plays at most one match per date, the minimal number of dates over which the $n(n-1)$ matches are distributed, is equal to $2n-2$. In a classical tournament, one tries to minimize the number of *breaks*, i.e. the fact that a team plays two consecutive matches in the same location, where the term location denotes either home or away. Because of the attractiveness of the tournament, the situation that two teams play their two matches against each other on two consecutive dates should be avoided. Such a situation is called a *repeater*.

In this paper, we consider the traveling tournament problem, introduced by M.A. Trick (see http://mat.gsia.cmu.edu/TOURN/). It is a double round robin tournament for n teams, with n even and with no repeaters. Breaks are allowed because in a large country (e.g. USA), where the distances between the home cities can be very large, it is advantageous to organize a *tour* for a number of consecutive away matches. However, no more than three consecutive away games and also no more than three consecutive home games are allowed for any team. The objective is to minimize the sum of the length of the tours traveled by the teams, where we assume that all teams start and end in their home city and that the distance matrix is symmetric.

A general technique to solve such a highly constrained problem is backtracking. It is a trial-and-error process that gradually builds up and scans (prunes) a tree of subtasks. Because the search tree grows very rapidly in the traveling tournament problem, there is a need for some heuristic in order to reduce computation to tolerable bounds. For this heuristic, we investigate in this paper the application of techniques introduced by ant colony optimization.

Ant algorithms are inspired by the observation of real ant colonies. Real ants searching for food are capable to find the shortest path between their nest and a food source by exchanging information via pheromones. This substance is

deposited by ants on the ground while walking from the nest to a food source and vice versa. In this way a pheromone trail is formed. As other ants observe the pheromone trail and are attracted to it, the path is marked again and will attract even more ants to follow the trail. In ant colony optimization algorithms, a finite sized colony of artificial ants collectively searches for good quality solutions to the optimization problem under consideration. Each ant builds a solution in a finite number of steps whereby in each step the partial solution is extended with one additional element. The selection of this additional element is influenced through problem-specific heuristic information, called visibility, as well as pheromone trails, an indication of how good the choice of that element was in former runs. After an artificial ant has constructed a feasible solution, the pheromone trails are updated depending on the objective function value of the constructed solution. This update will influence the selection process of the next ants.

In our heuristic for the traveling tournament problem, we use a colony of n ants. Each ant constructs a timetable starting from a different team. At each step in the search tree, a list of candidates of teams for the next match is constructed. In the first iteration this list is ordered according to the distances from the team under consideration to all its opponents. The first candidate of the list is selected and the procedure continues analogously with the next match for the team until some constraint cannot be satisfied. Then, backtracking occurs and the next element from the candidate list is selected. This process continues until a first complete timetable is built. When each ant has constructed its timetable, the pheromone trails are laid. The pheromone trail between team i and team j is incremented with a value depending on the total distance traveled when the timetable contains a direct route from team i to team j. There are three situations: (1) team i starts a tour in its home city and its first away match is against team j; (2) team i and j are two consecutive away opponents of a third team; and (3) team j ends a tour in its home city and its last away match is against team i. In the succeeding iterations, the order in the candidate list is not only based on the distances but also on the pheromone trails. In this way the construction of a timetable is not only influenced by "nearest neighbours" but also by subsequences of matches that are part of good solutions.

Computational experiments on the instances of the Challenge Traveling Tournament web page at http://mat.gsia.cmu.edu/TOURN/ are reported with different settings for the ant colony optimization parameters: parameters to regulate the influence of the visibility and the pheromone trail; the trail persistence parameter, the elitist strategy and ranking; and the effect of hybridization by the integration of local search.

References

1. Easton, K., G. Nemhauser and M. Trick: The Traveling Tournament Problem. Description and Benchmarks. INFORMS, National Meeting, Salt Lake City (2000). [http://mat.gsia.cmu.edu/trick/ttp_final.ps].

Generating Fair and Attractive Football Timetables

Thomas Bartsch[1], Andreas Drexl[2], and Stefan Kröger[3]

[1] SAP Portals Europe GmbH, Neurottstraße 16, 69190 Walldorf, Germany
thomas.bartsch@sapportals.com
[2] Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, 24118 Kiel, Germany
drexl@bwl.uni-kiel.de
[3] Schott Glas, Hattenbergstr. 10, 55122 Mainz, Germany
stefan.kroeger@schott.com

Generating a regular season timetable is a demanding task for any sports league. In Europe, the creation of a suitable timetable for every national top football league not only has to address numerous conflicting inner-league requirements and preferences. Additionally, the games of the European Cup matches (Champions League, UEFA Cup, National Cup Winners) have to be taken into account. In this paper we consider the case of Germany and Austria, that is the planning problem the "Deutsche Fußball-Bund" (DFB) and the "Österreichische Fußball-Bund" (ÖFB) are confronted with.

In Germany the DFB is responsible for the development of the schedule for the top men soccer league, called "1. Fußballbundesliga". The league consists of 18 teams, which play against each other twice per year, once in the home arena and once in the arena of the opponent. Each team has to play 34 games and, hence, in total $(18 \cdot 34)/2 = 306$ games have to be scheduled. There are two half-seasons covering 17 games each. Each of the 34 games takes place on a particular day of one of 34 rounds. The rounds are disjunct and ordered. There are two types of days where games might be scheduled, that is, the weekend days Friday, Saturday and Sunday and the non-weekend days Tuesday and Wednesday. (On Monday and Thursday no game must be scheduled.) Each of the 34 rounds covers these five days. Games on Saturday must take place in the day-time and in the evening on the other days. A minimum and maximum number of games is required to be scheduled on every day because of television, radio network and lottery requests.

In addition to these characteristics a couple of requirements are important. For the DFB, for instance, these requirements can be sketched out as follows:

Organisational requirements cover a set of rules which have to guarantee that all the games can be scheduled according to the regulations imposed by the DFB. O1: Arena availability. O2: The home arenas of some teams are located in specific regions. In order to avoid heavy traffic and too much demand for local safety staff only a maximum number of home games must be scheduled on each of the days within each of the regions. O3: Some games, for instance, those between teams having well-known hooligans, must be scheduled at certain hours. Security aspects require such games to take place on day-time, that is on Saturday.

Attractiveness requirements focus on what arena visitors, television spectators and the players expect from the sequence of 34 games, that is, a varied, eventful, and exciting season. A1: The number of breaks has to be minimized, that is, two teams have no break while the other teams have one break per half-season. To minimize the number of breaks gives arena visitors the opportunity to see a home game every two weeks. A2: The overall competition is divided into two rounds, a first one and a second one and every team plays against each other team once in the first round and once in the second round with a complementary home away pattern. A3: Teams want to have a home game on certain days, because a large number of visitors is expected due to, for instance, a public festival (like the "Oktoberfest" in Munic). Furthermore, on the first day of the season newcomers should have a home game while the other teams should start with a home game if the starter in the previous season was an away game and vice versa. A4: Broadcasting corporations demand, that attractive games are evenly distributed over the year. Usually, games are attractive if both teams obtained a high score in the previous season. Also, a local derby may be attractive. A5: The competition should be thrilling as long as possible. If, for instance, the champion of the previous season is scheduled against the second team at the end of the second round then it is very likely that the winner of the current season is not known beforehand. "Fixing" pairings in this regard does not mean that a specific game has to be scheduled in the last round, rather than in one of the last three rounds.

Fairness requirements have to guarantee that no team is handicapped or favored in comparision with the competitors. F1: At least two days without a game have to be scheduled between every two games, taking into account not only the games of the league under concern but also other games (like UEFA-cup, champions league) which are contained in a frame desk diary. F2: Tough and weak opponents have to be distributed evenly over the whole season for every team. What 'tough' and 'weak' means is measured in terms of the final score of the previous season. The six teams with the highest score form one group, the next six teams the second group and the remainig six teams make up the third group. The schedule then has to take care of the opponent strength in such a way that no two teams belonging to the same group are scheduled against each other in two consecutive rounds. F3: In the second round and in the last round no breaks should occur in order to avoid distortions of competition.

In this paper we develop a model covering these requirements; furthermore, algorithms which yield reasonable timetables quickly are presented (for details see [1]). The models borrow their expressive power from so-called partially renewable resources. Partially renewable resources are obtained by assuming for each resource a capacity on subsets of periods (see [2]).

For the solution of the models three algorithms have been developed. A semi-greedy algorithm which constructs a couple of (feasible) solutions quickly. A truncated branch-and-bound algorithm, which searches the solution space more completely and, finally, an exact branch-and-bound algorithm, which searches the solution space exhaustively. The algorithms are evaluated experimentally.

The algorithms have been embedded in an interactive decision suppport system based on Paradox. Our approach generates timetables which have been accepted for play by the DFB once and five times by the ÖFB. In these applications, partially renewable resources are a core tool for defining requirements in terms of constraints. It is strongly conjectured that football scheduling problems of other (European) countries can be solved adopting the approach presented in this paper.

References

1. Bartsch, Th., Drexl, A., Kröger, S.: Scheduling European Soccer Leagues: Models, Methods, and Applications. Working Paper No. 557, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Germany 2002
2. Böttcher, J., Drexl, A., Kolisch, R., Salewski, F.: Project Scheduling Under Partially Renewable Resource Constraints. Management Science **45** (1999) 543–559

Solving Sports Scheduling Problems Using Network Structure

Ayami Suzuka[1], Yasufumi Saruwatari[2], and Akiko Yoshise[3]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, Ibaraki, Japan,
asuzuka@sk.tsukuba.ac.jp
[2] Institute of Policy and Planning Sciences, University of Tsukuba, Tokyo, Japan,
saru@gssm.otsuka.tsukuba.ac.jp
[3] Institute of Policy and Planning Sciences, University of Tsukuba, Ibaraki, Japan,
yoshise@sk.tsukuba.ac.jp

1 Introduction

Sports schedules have been made manually due to many requirements to meet, such as organizational conditions and commercial interests. A systematic method for the scheduling is desirable. In this context, the sports scheduling problems, abbreviated by SSP, have widely been studied. De Werra [2, 3] introduced a characterization and graph-theoretical properties of some particular schedules, which are exploited in a schedule for Dutch major soccer league [6]. An interactive decision support system was developed by Ferland and Fleurent [4]. Campbell and Chen [1] and Nemhauser and Trick [5] treated scheduling problems for basketball league as case studies. These studies were based on integer programming and enumeration techniques.

In this paper, we extract some general conditions of SSP, which will be stated in Section 2 as C.1–C.6 and formulate a problem *SSP in general form*, abbreviated by SSPGF. We assume that SSPGF satisfies the round-robin tournament, which requires each team to match every other team a fixed number of times. We propose a branch-and-bound algorithm, in which the subproblems can be reduced to network problems.

2 Definition of SSPGF

We consider a league of $2n$ teams and denote it by P. Each pair of teams, say i and $j \in P$, is required to have matches $2r$ times. A season, denoted by T, consists of m terms, i.e. $T = \{1, 2, \cdots, m\}$. We say that a match between i and j is i's home-game if the match is held at i's home. The attendance for a match may depend on opponents, terms and homes, and it is estimated statistically by data of the matches of past seasons.

SSPGF is a problem of finding a schedule which maximizes the total attendance under the conditions that:

C.1 Each team must play exactly one match in a term.
C.2 Every match between i and j must be held at either i's or j's home.
C.3 The number of matches between i and j must be $2r$ for any i, j.
C.4 Among the matches between i and j the number of matches at i's (resp. j's) home must be r for any i, j.
C.5 At most one match between i and j can be held in three consecutive terms for any i, j.
C.6 No team is allowed to play at its home in more than three consecutive terms.

The above conditions C.1 and C.3 claim that the number of terms m should be $m = 2r(2n-1)$.

3 Branch-and-bound based algorithm

In the following, we assume that the competition is a double round-robin tournament, i.e. each team in the league matches every other team exactly twice.

We formulate SSPGF into an integer programming and define our subproblems by relaxing some constraints in order to use branch-and-bound methodology. A feature of our algorithm is that subproblem has network structure.

3.1 Subproblems

Here we state the subproblem for obtaining a lower bound of SSPGF.

For each pair of teams i and $j \in P$, denote by $V_{ij} = \{v_{ij}^t \mid t \in T\}$ a set of vertices, each of which corresponds to a match between i and j held in term t at i's home. Taking the condition C.5 into consideration, let $\bar{G_{ij}} = (V_{ij} \cup V_{ji}, \bar{A_{ij}})$ be a bipartite graph with the arc set $\bar{A_{ij}}$ where $\bar{A_{ij}} = (V_{ij} \times V_{ji}) \setminus \{(v_{ij}^{t_1}, v_{ji}^{t_2}) \mid |t_1 - t_2| \leq 2\}$.

By adding a pair of dummy vertices denoted by v_{ij}^{o} and v_{ij}^{e} as well as $\bar{A_{ij}^{\mathrm{o}}} = \{(v_{ij}^{\mathrm{o}}, v_{ij}^t) \mid t \in T\}$ and $\bar{A_{ij}^{\mathrm{e}}} = \{(v_{ji}^t, v_{ij}^{\mathrm{e}}) \mid t \in T\}$, to $\bar{G_{ij}}$, we obtain $G_{ij} = (W_{ij}, A_{ij})$ with $W_{ij} = \{V_{ij} \cup V_{ji} \cup \{v_{ij}^{\mathrm{o}}, v_{ij}^{\mathrm{e}}\}\}$ and $A_{ij} = \{\bar{A_{ij}} \cup \bar{A_{ij}^{\mathrm{o}}} \cup \bar{A_{ij}^{\mathrm{e}}}\}$, shown in Fig. 1(A). With each arc $(p, q) \in A_{ij}$, cost c_{pq}, upper bound u_{pq} and lower bound ℓ_{pq} of capacity are associated. If $q = v_{ij}^t$, c_{pq} is set to the estimated attendance for the match corresponding to v_{ij}^t. Otherwise we set $c_{pq} = 0$. For every arc $(p, q) \in A_{ij}$, $u_{pq} = 1$ and $\ell_{pq} = 0$.

By the above construction of G_{ij}, a schedule between i and j which satisfies C.2–C.5 is obtained as a maximum cost unit flow on G_{ij}.

Tight lower bounds are obtained by constructing a graph $G = (V, A)$ shown in Fig. 1(B). Namely we identify v_{ij}^{e} in each G_{ij} with a single vertex, say v^{e}, and for each t we also identify the vertices $v_{ji}^t \in V_{ji}$ for all $i, j \in P$ with a single vertex v^t. Then $V = \bigcup_{i,j \in P}(W_{ij} \setminus (V_{ji} \cup \{v_{ij}^{\mathrm{e}}\})) \cup \{v^t \mid t \in T\} \cup \{v^{\mathrm{e}}\}$ and $A = \bigcup_{i,j \in P}(A_{ij} \setminus \bar{A_{ij}^{\mathrm{e}}}) \cup \{(v^t, v^{\mathrm{e}}) \mid t \in T\}$. By setting $c_{pq} = 0, u_{pq} = n$ and $\ell_{pq} = 0$ for $(p, q) \in \{(v^t, v^{\mathrm{e}}) \mid t \in T\}$, a flow with value r originating from each v_{ij}^{o} on G satisfies C.2–C.5. A lower bound can be obtained by solving the maximum cost flow problem (MCFP) where x_{pq} is a flow of arc (p, q):

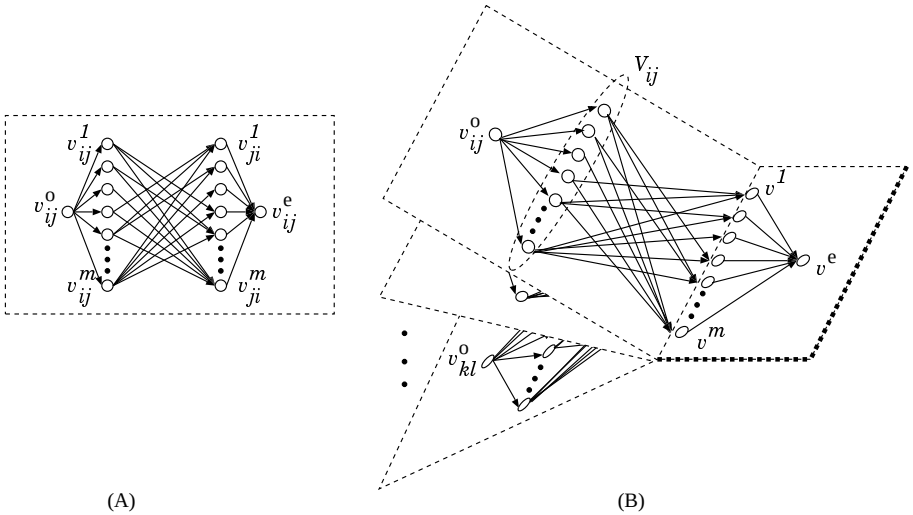

Fig. 1. The graph G which represents C.2–C.5.

MCFP

$$\max \sum_{(p,q)\in A} c_{pq}x_{pq}$$

s. t.

$$\sum_{q:(p,q)\in A} x_{pq} - \sum_{q:(q,p)\in A} x_{qp} = \begin{cases} 1, & p = v^{\mathrm{o}}_{ij},\ \forall i,j \in P,\ i<j, \\ -n(2n-1), & p = v^{\mathrm{e}}, \\ 0, & p \in V \setminus \{v^{\mathrm{o}}_{ij}, v^{\mathrm{e}} \mid i,j \in P,\ i<j\}, \end{cases}$$

$$\ell_{pq} \le x_{pq} \le u_{pq},\ \forall (p,q) \in A.$$

Since the total number of matches at j's home does not exceed n for each term t, the bound obtained by MCFP becomes tight.

3.2 Branch-and-bound algorithm

We employ the bounding rule stated below so that the resulting flow satisfies the condition C.1.

R.1 Find a term $t \in T$ and a team $i \in P$ such that i matches more than one team in t.
R.2 Find a team k who plays against i in t with a maximum attendance.
R.3 Construct a subproblem with side-condition that i match k in t, and the other subproblem where i never plays against k in t.

When a match between i and k is fixed in t, we check the feasibility of C.6. We developed some extra rules to tighten our lower bounds.

4 Conclusion

For sports scheduling problems, we have proposed an algorithm based on the branch-and-bound method, where the subproblems, being the maximum cost flow problems, are solved efficiently by use of network structure. Computational results support the validity of our algorithm.

References

1. R.T. Campbell and D.S.Chen, "A minimum distance basketball scheduling problem," Management Science in Sports, pp.15–25, North-Holland, Amsterdam, 1976.
2. D. De Werra, "Scheduling in Sports," O.R.Working Paper 45, Ecole Polytechnique Federale de Lousanne, 1979.
3. D. De Werra, "Geography, games and graphs," *Discrete Applied Mathematics,* Vol.2, pp.327–337, 1980.
4. J.A. Ferland and C. Fleurent, "Computer aided scheduling for a sport league," *INFOR,* Vol.29, pp.14–25, 1991.
5. G. Nemhauser and M. Trick, "Scheduling a major college basketball conference," *Operations Research,* Vol.46, pp.1–8, 1997.
6. J.A.M. Schreuder, "Constructing timetables for sports competitions," *Mathematical Programming Study,* No.13, pp.58–67, 1980.

Examination Timetabling

Enhancing Timetable Solutions with Local Search Methods

E. K. Burke and J. P. Newall

School of Computer Science and Information Technology, University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{jpn|ekb}@cs.nott.ac.uk

Abstract. It is well known that domain specific heuristics can produce good quality solutions for timetabling problems in a short amount of time. However they often lack the ability to do any thorough optimisation. In this paper we will study the effects of applying local search techniques to improve good quality initial solutions generated using a heuristic construction method. While the same rules should apply to any heuristic construction, we use here an adaptive approach to timetabling problems. The focus of the experiments is how parameters to the local search methods affect quality when started on already good solutions. We present experimental results which show that this combined approach produces the best published results on several benchmark problems and we briefly discuss the implications for future work in the area.

1 Introduction

1.1 The Examination Timetabling Problem

The examination timetabling problem consists of allocating a number of exams to a limited number of periods or timeslots subject to certain constraints. These constraints may relate to operational limitations (such as no student having to attend two exams at the same time, limitations on seating, etc.) which cannot be overcome, or they may be regarded as being sufficiently important that they should always be observed. In either case we call these *hard constraints*. We also call a timetable that obeys all hard constraints a *feasible* timetable. The remaining constraints are those that are considered to be desirable to satisfy, but not essential, such as allowing students study time between exams. These constraints are termed *soft constraints*. The level of satisfaction of these soft constraints can be considered to be a measure of the *quality* of a timetable. More information on the various constraints that can exist for the problem can be found in a survey of UK universities [1]. Surveys of practical applications of examination timetabling can be found in [2, 3].

1.2 Local Search Methods

It is possible to think of local search techniques as methods which typically function by iteratively applying simple moves to a solution. Before a move is applied the effects

it has are evaluated and a decision is made to accept or reject the move based on some criteria. The criteria for accepting moves is generally what distinguishes the various varieties of local search.

The simplest form of local search are descent based methods (or Hill-Climbing, depending on your point of view). Here a move is only accepted if it produces a solution at least as good as the current solution. For mainly comparative purposes we consider this simple form of local search in this paper.

A more sophisticated approach, that we consider in this paper, is Simulated Annealing[4]. This differs from descent methods in that it has a stochastic method whereby worsening moves can be accepted. The probability p of accepting a worsening move is given in EQ 1, where Δ represents the change in quality and T represents the current temperature of the system, this being derived from thermodynamics' Boltzmann's distribution. The temperature of the system is periodically lowered according to a *cooling schedule*. This results in worsening moves being less likely to be accepted as the process continues. The performance of simulated annealing is generally regarded to be highly dependant on the choice of parameters such as starting temperature, terminating temperature and the cooling schedule.

$$p = e^{-\frac{\Delta}{T}} \tag{1}$$

An alternative, that is also investigated in this paper and that has been shown to work particularly well on the examination timetabling problem, is the Degraded Ceiling method proposed by Burke et al.[5]. This approach functions in a similar fashion to simulated annealing and its variant Threshold acceptance[6]. However, worsening moves are accepted or rejected according to whether solution quality falls below or above a specified ceiling, which is reduced linearly throughout the process. The method was successfully tested on timetabling problems, proving to be as good as a well configured simulated annealing process. The main advantage though over simulated annealing is the lower number of parameters, and that those required can easily be guessed or worked out from a parameter describing the desired run time.

2 Applying Local Search Techniques to Good Initial Solutions

When applying local search techniques such as degraded ceiling or simulated annealing to good quality initial solutions there are factors we need to take into account. Firstly these methods usually perform a far ranging search of the solution space before terminating, which may not be desirable when starting from a good initial point as we could quickly lose any benefit of the initial quality. Fortunately we can control how far ranging a search we wish to do through parameters for two of the three local search methods that we investigate. It should be noted that the actual method of generating initial solutions is irrelevant here as we are concerned exclusively with establishing with how much, and under what conditions, solutions can be enhanced using local search methods.

2.1 Adaptive Initial Solution Generation

An adaptive heuristic approach is used to generate initial solutions. This basically functions by repeatedly trying to construct a timetable based on an ordering, while promoting any exams that cannot be scheduled to an acceptable level. This in itself produces very competitive results relatively quickly. As the experiments here are largely concerned with the improvements attainable with local search we will not discuss the generation method at length here. For a detailed explanation of the method and experimental results see [7].

As mentioned above we are not concerned with the actual construction method used. It is likely the same techniques could be applied just as effectively to other heuristic algorithms such as those presented by Carter et al. [8]. We will now briefly discuss the three local search approaches, and how, where possible, they can be configured to take account of already good solutions.

2.2 Hill-Climbing

As Hill-Climbing by its nature will not accept worsening moves it can only explore a very limited portion of the search space. It is however very fast compared to the other methods and will never produce a solution that is worse than the original. For these reasons Hill-Climbing provides a good baseline for comparison with the other more thorough search methods.

For these experiments a simple randomised Hill-Climbing method was used, where moves are generated at random (in the same way as for the other methods) and accepted if they do not lead to a worse solution. The process is aborted after 1,000,000 unsuccessful successive attempts at generating a move. Figure 1 gives the Hill-Climb-

```
sinceImprovemnt := 0
WHILE sinceImprovement < 1,000,000  DO
        choose exam e and period t at random s.t. t != period(e)
        IF   penalty(e, t) ≤ penalty(e, period(e))   THEN
                move exam e to period t
                sinceImprovement := 0
        ELSE
                sinceImprovement += 1
        ENDIF
DONE
```

Fig. 1. Pseudo-code describing the Hill-Climbing procedure

ing procedure used in the tests, where *period(e)* gives the current period of exam *e*, and *penalty(e,t)* gives the penalty arising from scheduling exam *e* in period *t*.

2.3 Simulated Annealing

The inherent controlling factor in Simulated Annealing is the temperature. A higher initial temperature will result in increased acceptance of worsening moves, and therefore the search tends to move further away from the starting point. Alternatively a lower temperature will search more in the vicinity of the initial solution, at the expense that it may not be able to reach other better, though still relatively near, areas of the search space. Figure 2 shows the process used for Simulated Annealing, where T_0 is

```
T := T0
WHILE T > Tmin DO
        choose exam e and period t at random s.t. t != period(e)
        IF  penalty(e, t) ≤ penalty(e, period(e))  THEN
                move exam e to period t
        ELSE
         Give probability exp ( (penalty(e,period(p)) - (penalty(e, t)) / T) of
        being  accepted
        ENDIF

        T := α T
DONE
```

Fig. 2. Pseudo-code for the Simulated Annealing Process

the initial temperature, T_{min} is the minimum temperature and α represents the cooling schedule.

As selecting a suitable initial temperature T_o can be a problem for simulated annealing we use an approach whereby we specify a desired average probability of accepting worsening moves. A temperature is then calculated by generating a number of random moves and evaluating the average probability of acceptance. If a temperature of 1 does not provide at least the desired average probability it is doubled and the process repeated. This continues until the generated average probability matches or exceeds the desired average probability. This allows us to specify the temperature indirectly in a less problem specific way.

Values for T_{min} and α are somewhat easier to determine. As our penalty function has a granularity of 1 a uniform value of $T_{min} = 0.05$ is used throughout to guarantee a Hill-Climbing like phase at the end of the process. Instead of specifying α as a parameter, we give the desired number of moves N as a parameter and calculate α according to equation 2.

$$\alpha = 1 - \frac{\ln(T_0) - \ln(T_{min})}{N} \quad (2)$$

2.4 Degraded Ceiling

The degraded ceiling method of local search provides a much simpler mechanism than simulated annealing and is known to be effective on exam timetabling problems [5]. Here the controlling factor is the value of the initial ceiling, where with a higher ceiling the search is less restricted. We can easily calculate an initial ceiling by multiplying the penalty of the initial solution by some chosen factor.

Figure 3 shows the degraded ceiling procedure where *b* is the amount we wish to reduce the ceiling by each iteration, and is calculated based on the user specified desired number of iterations *N*. As it is likely that the ceiling will fall below the current penalty as the solution nears a local optimal 1,000,000 iterations without improvement are allowed in order to accommodate a Hill-Climbing type phase towards the end of the process.

```
P := initial penalty of solution
B := P * user provided factor
b := B / N
sinceImprovement := 0
WHILE (B > 0 AND  P < B) OR sinceImprovement < 1,000,000 DO
        delta := penalty(e, t) - penalty(e, period(p))
        IF  P + delta < B  OR delta < 0 THEN
                move exam e to period p
                P := P + delta
                sinceImprovement := 0
        ELSE
                sinceImprovement += 1
        ENDIF

        B := B - b
DONE
```

Fig. 3. Pseudo-code for the Degraded Ceiling Approach

3 Results

3.1 Experimental Setup

Experiments were performed on a range of real world examination timetabling problems, shown in table 1, all of which are available from:

`ftp://ftp.mie.utoronto.ca/pub/carter/testprob/`

Table 1. The benchmark problems used

Data	Institution	Periods	Number of Exams	Number of Students	Density of Conflict Matrix
car-f-92	Carleton University, Ottawa	32	543	18,419	0.14
car-s-91	Carleton University, Ottawa	35	682	16,925	0.13
ear-f-83	Earl Haig Collegiate Institute, Toronto	24	190	1,125	0.29
hec-s-93	Ecole des Hautes Etudes Commerciales, Montreal	18	81	2823	0.20
kfu-s-93	King Fahd University, Dharan	20	461	5349	0.06
lse-f-91	London School of Economics	18	381	2726	0.06
sta-f-83	St Andrew's Junior High School, Toronto	13	139	611	0.14
tre-s-92	Trent University, Peterborough, Ontario	23	261	4360	0.18
uta-s-93	Faculty of Arts and Sciences, University of Toronto	35	622	21,267	0.13
ute-s-92	Faculty of Engineering, University of Toronto	10	184	2,750	0.08
yor-f-83	York Mills Collegiate Institute, Toronto	21	181	941	0.27

The objectives of these problems are to firstly create a conflict free timetable, and secondly to minimise the number of cases where a student has exams *s* periods apart. The weight w_s for a student taking two exams *s* periods apart is given by: $w_1=16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, $w_5 = 1$. These are summed up and divided by the number of students in the problem to give a measure of the average conflicts per student.

Degraded ceiling was applied to the heuristically generated solutions with various initial ceiling values (given in the form of a factor of the initial quality). Similarly sim-

ulated annealing was applied to the same problems with varying desired initial average probabilities (as described above). Both algorithms were given a desired number of iterations of 20,000,000 and 200,000,000. Each individual experiment was run 5 times with varying random number seeds. An index of improvement was calculated by summing the average percentage improvement for each of the problems.

Often in local search methods it is prudent to keep a copy of the best solution found so far, as the final resting point is not necessarily the best solution encountered. In this case however it seems to be more revealing to take the final resting point of the search, as we can then perceive actual reductions in solution quality.

3.2 Results when using Degraded Ceiling Local Search

The degraded ceiling algorithm was run on the generated solution with the initial ceiling set at various values based on the original quality. Figure 4 shows the total improvement index, with the horizontal bar indicating the improvements found by basic stochastic Hill-Climbing. It is clear that we need to give the search procedure

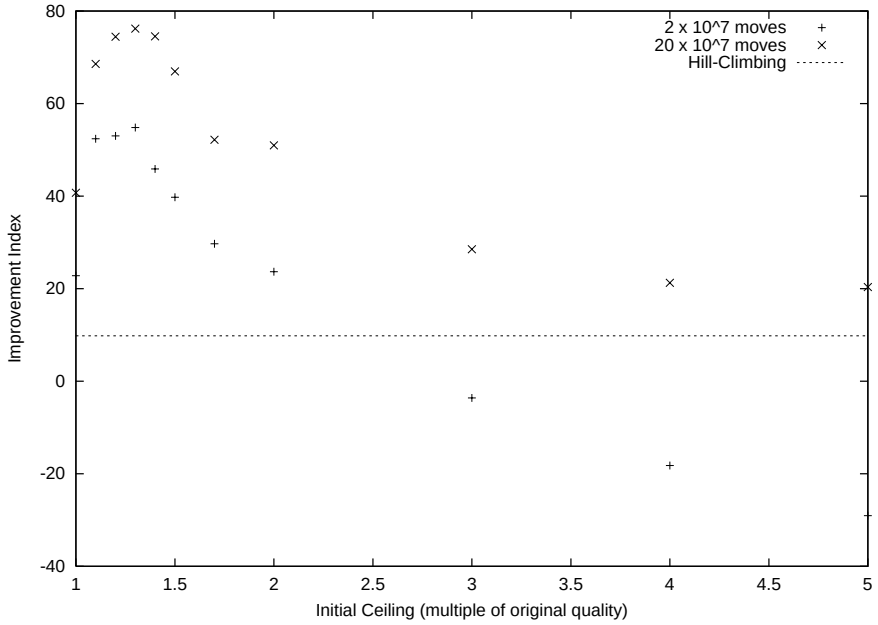

Fig. 4. Results when using degraded ceiling to improve solutions

some extra scope to manoeuvre, though too much scope results in overall degradation of the results. It is even the case that too much scope (a ceiling factor of 5) produces

results noticeably worse than the original unless given a sufficient amount of time. This said it seems sensible to set the initial ceiling at the quality of the initial solution, multiplied by a factor of 1.3. Launching the algorithm for 10 times the number of moves results in slightly better improvements, however the same rules on the initial ceiling still apply. It is worth noting that all reasonable configurations perform considerably better than stochastic Hill-Climbing, yielding nearly 8 times the improvement of Hill-Climbing in the best case.

3.3 Results When Using Simulated Annealing

When experimenting with simulated annealing we test a range of temperatures, based on the average probability of accepting worsening moves. The probability is varied between 0.1 and 0.9 and shows the variation in improvement when using different starting temperatures. The results of this are shown in Table 5. Simulated annealing

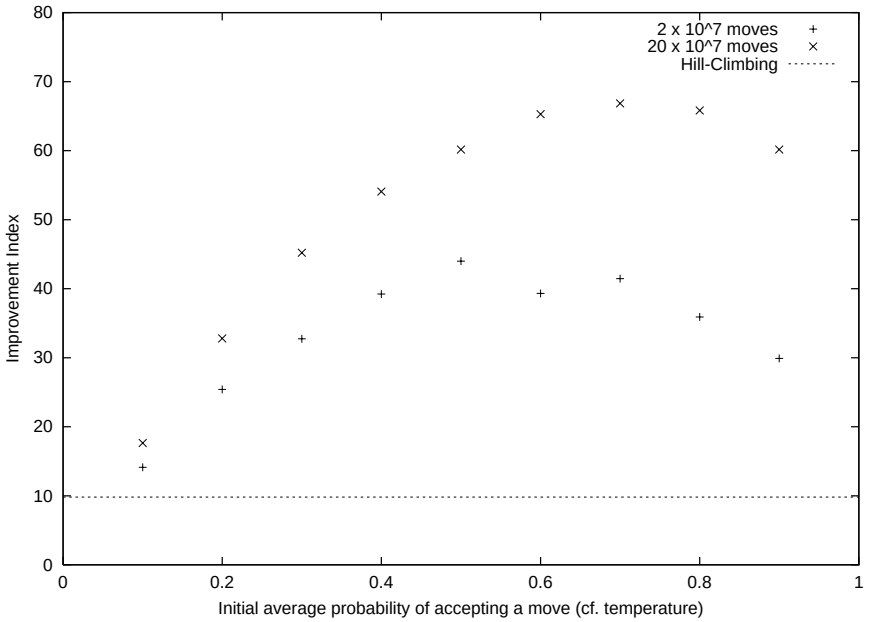

Fig. 5. Results when using Simulated Annealing to improve solutions

exhibits a similar behaviour to the degraded ceiling approach in that there is a “sweet spot” that balances out the need for a far ranging search against the desire to preserve the quality of the original solution. The differences here though are less pronounced, with none of the tested scenarios producing solutions worse than the originals or, for

that matter, than those produced with Hill-Climbing. We would however expect this to change as the probability approaches 1 and becomes more like a non-seeded application of simulated annealing.

3.4 Analysis

The results indicate a slight advantage in favour of the degraded ceiling method when working on improving solutions. The simpler mechanism of degraded ceiling also makes the method more attractive than simulated annealing. The experiments show that for both experiments the value of the parameters can have a dramatic effect on solution quality. Tables 2 and 3 show a comparison between simulated annealing (SA) and degraded ceiling (DC) when started with and without good solutions, for 20,000,000 and 200,000,000 iterations. It should be noted that the initial solution for the basic approach was also generated using a similar heuristic method except that soft constraints were not considered. This was necessary as both simulated annealing and degraded ceiling have great difficulty finding feasible solutions for some of these problems. For longer runs of 200,000,000 iterations in all but one case the solution produced by the basic approach is worse that the solution produced using a well configured combination approach. In general degraded ceiling emerges as the better method, obtaining the best results on 7 of the 11 problems. As we would expect though the benefits of initial solution quality are much more noticeable when dealing with shorter 20,000,000 iteration runs. Table 4 shows the average individual results for

Table 2. Comparison of both methods with and without heuristic initialisation (20,000,000 iterations)

Data	SA (basic) init prob=0.999		SA (heuristic) init prob=0.7		DC (basic) init ceil=1.0		DC (heuristic) init ceil=1.3	
	cost	time (s)	cost	time (s)	cost	time (s)	cost	time (s)
car-f-92	5.27	31	4.19	61	4.92	90	4.14	81
car-s-91	6.48	35	4.92	81	5.48	108	4.84	109
ear-f-83	41.18	16	37.62	19	39.43	46	37.39	36
hec-s-93	12.43	11	12.02	11	12.34	94	11.94	51
kfu-s-93	16.00	49	15.31	74	15.95	76	15.51	64
lse-f-91	16.16	40	10.77	56	15.80	55	10.75	50
sta-f-83	176.34	14	165.70	16	176.30	32	166.47	42
tre-s-92	9.05	20	8.44	26	8.95	45	8.42	38
uta-s-93	5.10	36	3.29	75	4.79	74	3.26	90
ute-s-92	28.30	23	26.30	26	27.98	52	26.74	37
yor-f-83	40.94	13	39.44	15	39.13	72	39.31	71

Table 3. Comparison of both methods with and without heuristic initialisation (200,000,000 iterations)

Data	SA (basic) init prob=0.999		SA (heuristic) init prob=0.7		DC (basic) init ceil=1.0		DC (heuristic) init ceil=1.3	
	cost	time (s)	cost	time (s)	cost	time (s)	cost	time (s)
car-f-92	4.86	307	4.15	340	4.52	916	**4.10**	416
car-s-91	5.23	353	4.73	409	4.87	1156	**4.65**	681
ear-f-83	39.69	163	**36.57**	167	38.45	451	37.05	377
hec-s-93	11.81	108	11.71	98	11.65	966	**11.54**	516
kfu-s-93	15.18	486	14.34	506	15.17	749	**13.90**	449
lse-f-91	13.53	396	10.90	411	12.91	643	**10.82**	341
sta-f-83	176.10	141	**168.37**	132	176.44	311	168.73	418
tre-s-92	8.67	196	8.52	192	8.54	473	**8.35**	304
uta-s-93	4.60	351	**3.19**	385	4.00	881	**3.20**	517
ute-s-92	26.46	231	25.88	226	26.06	544	**25.83**	324
yor-f-83	40.00	125	37.82	121	**37.16**	786	37.28	695

Table 4. Comparison of results

Data	Tabu Solver (avg.) Di Gaspero & Schaerf	Carter et al.	Adaptive with degraded ceiling (1.3) (avg.)	
	Cost	Cost	Cost	Time (s)
car-f-92	5.6	6.2-7.6	**4.10**	416
car-s-91	6.5	7.1-7.9	**4.65**	681
ear-f-83	46.7	**36.4**-46.5	37.05	377
hec-s-93	12.6	**10.8**-15.9	11.54	516
kfu-s-93	19.5	14.0-20.8	**13.90**	449
lse-f-91	15.9	**10.5**-13.1	10.82	341
sta-f-83	166.8	**161.5**-165.7	168.73	418
tre-s-92	10.5	9.6-11.0	**8.35**	304
uta-s-93	4.5	3.5-4.5	**3.20**	517
ute-s-92	31.3	**25.8**-38.3	25.83	324
yor-f-83	42.1	41.7-49.9	**37.28**	695

degraded ceiling when given a ceiling factor of 1.3 and allowed 200,000,000 moves, and also compares them with the results of two other methods on the same problem,

namely the tabu search method proposed by di Gaspero and Schaerf[9] and the heuristic backtracking approach used by Carter et al. [8]. The times given are in CPU seconds on a 900Mhz Athlon.

On some problems there is substantial improvement on the previously reported results with the others being comparable. The exception is the **sta-f-83** problems, on which out proposded method performs the worst (though not by a large margin) of all the techniques. It is also worth noting that the method also outperforms the tabu search approach proposed by White and Xie[10] on the two problems they used for testing.

It would be impractical to display all the produced results here, however the keen reader can obtain all individual results, together with produced solutions from:

http://www.asap.cs.nott.ac.uk/misc/jpn/patat2002/

4 Conclusions

It is clear that under the right circumstances both simulated annealing and the degraded ceiling method can be used to improve on already good quality solutions. While obtaining maximal improvement requires a delicate balance of the parameters, using guideline values for these parameters should provide near-maximal improvement. This combination of heuristic construction and a subsequent phase of local search produces very competitive results on the benchmark problems, introducing a new upper bound for some of the problems.

A possible issue that was not clearly reflected in the experiments is the relationship between the amount of time the search is allowed and how much the search process is limited. It seems logical that if the search process is given more time, the limitation on the search space can be relaxed a little. There is a suggestion of this in figure 5, where the best desired average probability is a little higher when run for 10 times the number of moves. Determining this relationship more formally however will require substantial experimentation and will be a focus of further work.

Another area in which this work could have a serious impact is the development of hyper-heuristics (heuristics to choose heuristics). As an aspect of hyper-heuristics deals with what methods are best to use, given a set time frame in which to work, this approach could form part of a larger hyper-heuristic framework, where the framework decides what parameters to use and for how long to perform local searching.

Acknowledgments

The research described in this paper was supported by EPSRC grant GR/N36837/01.

The authors would also like to acknowledge helpful advice given by Yuri Bykov.

References

[1] E. K. Burke, D. G. Elliman, P. H. Ford, and R. F. Weare. Examination timetabling in british universities - a survey. In E. Burke and P. Ross, editors, *The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference, Lecture Notes in Computer Science 1153*, pages 76–90. Springer-Verlag, Berlin, 1996.

[2] M. W. Carter. A survey of practical applications of examination timetabling. *Operations Research*, 34:193–202, 1986.

[3] M. W. Carter and G. Laporte. Recent developments in practical examination timetabling. In E. Burke and P. Ross, editors, *The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference, Lecture Notes in Computer Science 1153*, pages 3–21. Springer-Verlag, Berlin, 1996.

[4] S. Kirkpatrick, C. D. G. Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[5] E. K. Burke, Y. Bykov, J. P. Newall, and S. Petrovic. An investigation of time predefined search for the examination timetabling problem. Technical Report NOTTCS-TR-2002-1, University of Nottingham, UK, School of Computer Science & IT, 2002.

[6] G. Dueck and T. Scheuer. Threshold accepting: A general purpose algorithm appearing superior to simulated annealing, 1990.

[7] E. K. Burke and J. P. Newall. A new adaptive heuristic framework for examination timetabling problems. Technical Report NOTTCS-TR-2001-5 (submitted to Annals of Operations Research), University of Nottingham, UK, School of Computer Science & IT, 2002.

[8] M. W. Carter, G. Laporte, and S. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 47(3):373–383, 1996.

[9] 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 (Lecture Notes in Computer Science 2079)*, pages 104–117. Springer, Berlin, 2001.

[10] G. M. White and B. S. Xie. Examination timetables and tabu search with longer-term memory. In E. K. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III (Lecture Notes in Computer Science 2079)*, pages 85–103. Springer, Berlin, 2001.

A Hybrid Algorithm for the Examination Timetabling Problem

Liam T.G. Merlot[1], Natashia Boland[1], Barry D. Hughes[1], and Peter J. Stuckey[2]

[1] Department of Mathematics and Statistics,
The University of Melbourne, Victoria 3010, Australia
merlot@ms.unimelb.edu.au, natashia@ms.unimelb.edu.au,
hughes@ms.unimelb.edu.au
[2] Department of Computer Science and Software Engineering,
The University of Melbourne, Victoria 3010, Australia
pjs@cs.mu.oz.au

Abstract. Examination timetabling is a well-studied combinatorial optimization problem. We present a new hybrid algorithm for examination timetabling, consisting of three phases: a constraint programming phase to develop an initial solution, a simulated annealing phase to improve the quality of solution, and a hill climbing phase for further improvement. The examination timetabling problem at the University of Melbourne is introduced, and the hybrid method is proved to be superior to the current method employed by the University. Finally the hybrid method is compared to established methods on the publicly available data sets, and found to perform well in comparison.

1 Introduction

The difficulty of developing appropriate examination timetables for tertiary education institutions is increasing. Institutions are enrolling more students into a wider variety of courses including an increasing number of combined degree courses. For example, at the University of Melbourne, approximately 20 000 students have to be fitted into about 650 exams over a two and a half week period. For these 20 000 students, there exist approximately 8 000 different individual examination timetables. Consequently examination timetabling is a difficult combinatorial optimization problem and too complex an issue to be resolved by manual means. Appropriate algorithms are required to provide adequate examination timetables for universities.

The development of an examination timetable requires the institution to schedule a number of examinations ('exams') in a given set of exam sessions ('time slots', or simply 'sessions'), so as to satisfy a given set of constraints. A common constraint for universities is that no student may have two exams scheduled at the same time. However, some universities allow a student to have two examinations scheduled at the same time (a 'clash'), as long as an appropriate arrangement can be made (such as 'quarantining' students between exams). This

is the situation at the University of Melbourne, where every semester students are scheduled with two exams in the same session. As quarantining is expensive and inconvenient, we propose a different examination timetabling method for the University of Melbourne that avoids these 'clashes'.

This paper discusses key features of examination timetabling problems and reviews existing methods for publicly available and other data sets in Section 2. A new hybrid exam scheduling method is presented in Section 3. This hybrid method seeks good quality schedules, but attempts to avoid unnecessary clashes. The new method is a combination of constraint programming, simulated annealing and hill climbing (local search). Details of the problem for the University of Melbourne are given in Section 4, while benchmark problems in the literature are discussed in Section 5. The hybrid method is demonstrated in Section 4 to be superior to the current timetabling system used by the University of Melbourne, and in Section 5 to be superior or comparable to well-known existing methods, measured against established benchmarks.

2 Previous Work on Examination Timetabling

2.1 Examination Timetabling Problems

The primary form of the exam timetabling problem faced by educational institutions is to allocate a session and a room to every exam, so as to satisfy a given set of constraints. The result is a feasible exam timetable. However, each institution will have some unique combination of constraints, as policies differ from institution to institution. Furthermore, institutions may take different views on what constitutes the *quality* of an exam timetable. In some cases, any feasible timetable will do, while in other cases, timetables exhibiting desirable features are sought. This makes it difficult to give a universal definition of exam timetabling, but, although the exact nature of the constraints and quality measures tends to be unique to individual institutions, they tend to take on only a limited number of forms.

The most common forms of constraint are:

1. **Clashing:** no student may have two exams in the same session.
2. **Capacity:** the total number of students sitting in all exams in the same session in the same room must be less than the capacity of the room.
3. **Total Capacity:** the total number of students sitting in all exams in the same session must be less than the total capacity for that session.
4. **Exam Capacity:** the total number of exams in the same session must be less than some specified number.
5. **Exam Availability:** some exams are preassigned to specific sessions or can only be held in a limited set of sessions.
6. **Room Availability:** some rooms are only available in specific sessions.
7. **Pairwise Exam Constraints:** some pairs of exams must satisfy pairwise scheduling constraints (e.g., one must be held before the other).
8. **Exam/Room Compatability:** some exams may require specific rooms.

9. **Student Restrictions:** there may be restrictions on students' individual examination timetables (e.g., no student can have two exams scheduled in three consecutive sessions).
10. **Large Exams:** large exams should be held earlier in the exam period (e.g., exams with more than 500 students must be held in the first 10 sessions).

Each institution will apply some or all of these constraints. The exact form will be dependent on the institution, and some may be treated as soft constraints (constraints that hold where possible, but can be violated). For example the University of Melbourne treats 'Clashing' and 'Large Exams' as soft constraints. Quality measures (or objectives) of a solution are usually derived from soft constraints, most frequently from 'Student Restrictions'. For example, the number of clashes (instances of a student with two exams scheduled in the same session) is a quality measure for the University of Melbourne, as is the number of instances of a student with an exam scheduled in both the morning and afternoon sessions of the same day. If several different quality measures are used simultaneously, the objective is a linear combination of these measures, with relative weights that reflect their perceived importance.

For some institutions (including the University of Melbourne), the allocation of rooms to the exams in a given session is a secondary problem: exam rooms may be large, or exams easily split between rooms. In these cases, the assignment of sessions to exams has only to respect the total capacity constraint for each session, and the assignment of exams to specific rooms can be done later as a separate activity. Not all institutions are so fortunate.

2.2 Previous Methods

In this section, we review some influential and recent methods for solving exam timetabling problems. Often, different methods have addressed somewhat different versions of the exam timetabling problem, with different constraints and quality measures. Quality measures are often combined to form a mathematical *objective* for the problem, and methods which optimize with respect to that objective developed. We discuss variations of the problem encountered in the literature in more detail in Section 5; here we focus on the methods that have been applied.

Surveys of different methods for exam timetabling by Burke et al. [2] and Carter et al. [8], classify the different approaches as cluster methods, sequential construction heuristics, constraint programming, and local search (genetic algorithms, memetic algorithms, simulated annealing and tabu search). In recent years, Carter[3] and Burke[4] have made data sets for exam timetabling publicly available via the internet. Only three different approaches, that we are aware of, have been applied to this publicly available data.

Sequential construction heuristics have been applied to the publicly available data in a variety of forms by Burke et al. [6], Carter et al. [9, 10] and Caramia

[3] ftp://ftp.mie.utoronto.ca/pub/carter/testprob/
[4] ftp://ftp.cs.nott.ac.uk/ttp/Data/

et al. [7]. Sequential construction heuristics order the exams in some way (for example, largest exam first), and attempt to allocate each exam to a session in order, while satisfying all the constraints. The different heuristics feature different orders. They also have other differences: Carter et al. [10] allow limited backtracking (deallocation of exams), Burke et al. [6] select exams from a randomly chosen subset of all exams, and Caramia, Dell'Olmo and Italiano [7] include an optimization step after each exam allocation.

Burke, Newall and Weare use memetic algorithms for exam timetabling [3, 4]. In Burke et al. [4] an initial pool of timetables is generated via a random technique, which attempts to group together exams with similar sets of conflicting exams. Then timetables are randomly selected from the pool, weighted by their objective value, and mutations are applied by rescheduling randomly chosen exams, or all exams in a randomly chosen session. Finally hill climbing (local search) is applied to the mutated timetable to improve its quality. The process continues with the new pool of timetables. Burke and Newall [3] improve upon their earlier work by applying the memetic algorithm only to the first k exams as defined by a sequential construction method ordering. After the best timetable for the first k exams is found, the exams are fixed in place, and the memetic algorithm applied for the next k exams, until all are fixed.

White and Xie [19] and Di Gaspero and Schaerf [13] use tabu search methods. White and Xie keep two tabu lists, the usual short-term tabu list, and a long-term tabu list which keeps track of the most moved exams. Di Gaspero and Schaerf [13] use a single tabu list, but when exams are added to this list it is for a randomly determined number of iterations. They also modify the objective function as the algorithm progresses.

There is a considerable body of work on exam and other timetabling problems, which has not been applied to the publicly available data sets. The most closely related to our work appear to be the constraint programming approach used by Boizumault et al. [1] and the simulated annealing approaches explored by Dowsland and Thompson [11, 12, 16–18]. The principal innovation in our work, compared to these others, is the sequential use of these two methods as the first two stages of a total strategy. A similar sequential approach has been taken in work on other problems: White and Zhang [20] use constraint programming to find a starting point for tabu search in solving course timetabling problems, and for high school timetabling Yoshikawa et al. [21] test several combinations of two stage algorithms, including a greedy algorithm followed by simulated annealing and a constraint programming phase followed by a randomized greedy hill climbing algorithm (which is deemed to be the best combination of those used). In a similar vein, Burke, Newell and Weare [5] use their work on sequential construction heuristics [6] to generate initial solutions for their memetic algorithm [4]. We use a simpler constraint programming approach than Boizumault et al. [1], but it is only used as our initialization step. Dowsland and Thompson initialize by simulated annealing; the later stage of their simulated annealing algorithm is similar to ours. Following the simulated annealing stage, our hybrid method uses a third hill climbing stage. The approach used by Schaerf [15] for high school

timetabling also combines a metaheuristic with a hill climbing method: a tabu search algorithm, based on a small neighbourhood, employs a (randomized) hill climbing method,[5] based on a larger neighbourhood, whenever it detects a local minimum (with respect to its smaller neighbourhood).

3 A Three Stage Method

We consider the exam timetabling problem in which a session must be allocated to each exam. We apply three of the constraints defined in Section 2.1: Clashing, Exam Availability and Total Capacity. We also treat the Large Exams constraint as a hard constraint, modelled via the Exam Availability constraint. Room allocation is not critical for the application of most interest to us (see Section 4), so we neglect it here. The approach we take to the exam timetabling problem consists of three stages, yielding a hybrid method:

1. **Constraint Programming:** to obtain a feasible timetable.
2. **Simulated Annealing:** to improve the quality of the timetable.
3. **Hill Climbing:** for further refinement of the timetable.

The first stage of the hybrid method is used primarily to obtain an initial timetable satisfying all the constraints. As we shall see below, our approach aims to achieve this using as few sessions as possible, and although all constraints will be satisfied by the resulting timetable, in some cases, some exams may remain unscheduled.

The second and third stages aim to improve the quality of the timetable, and to schedule as many exams as possible. The methods used in both these stages are optimization methods, which will seek to optimize a given objective function. For this purpose, we formulate an objective function which takes into account both aims. The measures of quality, and hence the objective function formulated, differ according to the particular data set used, but typically the objective function is some combination of penalties applied for proximity of exams (time-wise) in students' timetables. We discuss the objective functions, which we call *objective scores*, used in detail in the context of the data sets considered, in Sections 4 and 5.

It is possible (although rare for the data sets we tested) that some exams remain unscheduled after all three stages. In this case, we employ a simple greedy heuristic to schedule the remaining exams. We discuss this heuristic in Section 3.4.

3.1 Constraint Programming

Constraint programming is used to find a first feasible timetable in our hybrid method. Our constraint programming model is defined as follows. We use the notation below:

[5] A neighbour is generated at random. It is accepted if it is at least as good as the current solution.

- $E = \{1, ..., n\}$ denotes the given set of n exams,
- s_i is the number of students in exam i, for all $i \in E$,
- $T = \{1, ..., v\}$ denotes the set of v sessions,
- $R \subset E \times T$ represents the set of given exam-session restrictions, so that $(a, b) \in R$ indicates that session b cannot be allocated to exam a,
- C_t is the total capacity of session t, that is, the total number of students who can sit exams in session t, for all $t \in T$,
- D_{ij} is the number of students enrolled in both exams i and j, for all $i, j \in E$, and
- variable $x_i \in T$ indicates the session allocated to exam i, for all $i \in E$.

Note that the sessions are assumed to be time ordered, so $t_1 < t_2$ for $t_1, t_2 \in T$ if and only if session t_1 occurs before session t_2. We also define the *domain* of each variable x_i to be the set of all sessions that can feasibly be allocated to exam i. Initially, the domain of x_i is the set of sessions $t \in T$ such that $(i, t) \notin R$. During a constraint programming search, the domain of variables may be reduced by the removal of sessions, so as to ensure some form of consistency with respect to the constraints. The initialization of the variable domains ensures that the Exam Availability constraint is satisfied. The Clashing constraint is modelled in our constraint program as follows:

$$x_i \neq x_j \text{ for all exams } i, j \in E \text{ with } i \neq j \text{ and } D_{ij} > 0.$$

The Total Capacity constraint is modelled as

$$\sum_{i \in E} s_i(x_i = t) \leq C_t \text{ for all sessions } t \in T.$$

Here $(x_i = t)$ is a Boolean switch, taking on value 1 if $x_i = t$, and 0 otherwise.

A typical constraint programming method, applied to the above exam timetabling model, will operate as follows. An exam is chosen, and a session in its domain is allocated to it, that is, all but one session is removed from its domain. In this case we say the exam has been *scheduled*; otherwise it is *unscheduled*. Consistency of the domains of all variables with respect to the constraints is then checked. For example, any exam which clashes with the chosen exam should have the session allocated to the chosen exam removed from its domain. After consistency is checked, another exam is chosen, and is allocated a session from its domain. This process is repeated until either all exams have been allocated a session, or until infeasibility has been detected, usually as a result of the domain of some variable becoming empty. In this case, the method backtracks and re-allocates sessions to some exams. Clearly there is a lot of flexibility in this method: which exam is chosen at each step, and which session in its domain is chosen to be allocated to it? The precise form of these choices determines the *search strategy* of the constraint programming method.

Our experiments revealed that the best search strategy for our exam timetabling problem is to choose the unscheduled exam with the smallest current domain size, that is, the exam with the smallest domain size greater than one,

and to allocate it the earliest session in its domain. Other search strategies explored were: choosing exams based on the number of students in each exam, choosing exams based on the number of conflicting exams, and choosing sessions based on the total space remaining given the current allocations (most or least).[6]

This constraint programming model and search strategy were implemented using the ILOG package OPL [14]. As the sessions were chosen in order (exams were scheduled as early as possible), the timetables found tended to use a number of sessions close to the minimum number required (and often less than the maximum number allowed). In order to take advantage of this result, and consequently speed up the overall algorithm, additional 'dummy' sessions are included (with a capacity equal to the maximum used for all other sessions), which are feasible for all exams, and occur 'after' the final session v. Any exams allocated these dummy sessions by OPL are interpreted as unscheduled. A component of the objective in the simulated annealing phase of the algorithm encourages scheduling of unscheduled exams. In Section 5.3, 12 different data sets are used, and for these, two data sets required one dummy session and one required two dummy sessions, but after the simulated annealing phase there were no exams remaining in these dummy sessions.

Our constraint programming approach thus generates timetables satisfying the given constraints, with the proviso that in some cases, some exams may remain unscheduled (although the search strategy we use ensures this is rare).

As the timetables produced by this process are not subject to any quality measures, they are usually of very poor quality. Optimization of exam timetables using constraint programming proved impractical: our experiments showed that the large search space and weak pruning of the minimization constraints made it difficult for a constraint programming method to improve significantly timetable quality. However, as we shall see later, a simulated annealing method *is* able to make substantial improvements.

3.2 Simulated Annealing

The timetable produced by the constraint programming algorithm is used as the starting point for the simulated annealing phase of the hybrid method. This phase is used to improve the quality of the timetable.

In what follows, we refer to a candidate timetable, together with any unscheduled exams (exams scheduled in dummy sessions), as a *solution*. In all cases, all scheduled exams in any solution will satisfy the Clashing, Total Capacity, and Exam Availability constraints, so all exams allocated a given session will not conflict with each other, the total number of students sitting exams allocated the same session will not exceed the capacity of the session, and no exam will be allocated a session for which it is not available.

[6] Boizumault et al.[1] found for their data the best search strategy was to choose the exam with the largest number of students and the session with the most available space.

A key component of any simulated annealing method is the *neighbourhood* of a solution. The neighbourhood we use is a slight variant on the Kempe chains neighbourhood used by Thompson and Dowsland [18]. A Kempe chain is determined by an exam i currently allocated session t, and another session $t' \neq t$. Let G be the set of all exams allocated session t, and G' be the set of exams allocated session t'. Note that our definition of a solution ensures that both G and G' are conflict-free sets of exams. The Kempe chain can be thought of as the (unique) minimal pair of sets of exams, $F \subseteq G$ and $F' \subseteq G'$, such that $i \in F$ and both $(G \setminus F) \cup F'$ and $(G' \setminus F') \cup F$ are conflict-free sets of exams. For a given exam i in session t and other session t', the Kempe chain (F and F') can easily be constructed by a simple iterative procedure. We call the timetable obtained by (re-)allocating session t to all exams in F' and (re-)allocating session t' to all exams in F a *neighbour* of the solution. The *neighbourhood* of a solution is defined to be the set of all such neighbours.

In our simulated annealing algorithm, a current solution is maintained, and a neighbour of the current solution chosen at random. We choose a neighbour by selecting an exam i at random from the set of all exams, and selecting a session t' at random from the set of sessions available for i, i.e. with $(i, t') \notin R$, such that $t' \neq t$, where t is the session allocated to exam i in the current solution. Together i, t and t' induce a Kempe chain, and hence a neighbour of the current solution. (Thompson [18] selects two sessions at random, and randomly chooses an exam allocated the first session.)

Our simulated annealing method is quite standard. Once a neighbour has been generated (each generation of a neighbour is defined as an iteration), it is tested for feasibility (of Total Capacity and Exam Availability), and if feasible, the objective function value, or *score*, for this neighbour is calculated. If this neighbour is superior in quality to (has a lower score than) the current solution then the current solution is replaced by the neighbour. Otherwise the current solution may or may not be replaced by the neighbour, depending on the difference in scores and the current *temperature* (as is standard in simulated annealing). Let $o(p)$ denote the objective score for a solution p. Let x be the current solution and let q be the neighbouring solution to x selected at random at the current iteration. Let u be the current temperature. Then if $o(q) > o(x)$, the probability that q will replace x as the current solution is $e^{(o(x)-o(q))/u}$.

The score used varied by problem class (different problem classes have different objectives), so we define the scores used in Sections 4 and 5, where we document each problem class tested.

Thompson and Dowsland [16] discuss different cooling schedules for simulated annealing processes applied to exam scheduling problems. They find that slow cooling schedules are generally more effective, but that no cooling schedule is markedly better than any other. We use a geometric cooling schedule, in which at every a iterations, the temperature, u, is multiplied by α, where a and α are given parameters of the algorithm. Initial experimentation revealed that the parameter settings of a starting temperature of 30 000, $\alpha = 0.999$, $a = 10$ and

a stopping temperature of 10^{-11}, giving approximately 350 000 iterations, were appropriate for all problems discussed in this paper.

Throughout the simulated annealing phase, the algorithm keeps the best solution found so far, and yields as output this best solution, at the conclusion of the algorithm. As with most heuristics, simulated annealing solutions will vary in quality depending on how long the algorithm is permitted to run. It was found that solutions using the standard parameters, generated in under a minute of CPU time on an Alphastation XP900 (466MHz), were of good quality for the University of Melbourne's examination timetabling problem.

We use simulated annealing to improve the solution determined by the constraint programming phase, and to schedule any remaining unscheduled exams. We could have used simulated annealing starting from a random timetable to satisfy all the constraints. Experiments with a purely simulated annealing approach performed poorly, since finding a feasible solution was in many cases quite difficult, and this overwhelms the search for a good solution. Indeed for one of our data sets there was no solution to the hard constraints as given, and we would never determine this using simulated annealing alone.

3.3 Hill Climbing

The hill climbing algorithm starts with an initial solution, x, called the 'current' solution, with objective score $o(x)$. The algorithm processes each exam in some order, determined a priori: we chose the exam subject code order. For each exam, e, in this order, the algorithm considers every neighbour of x (using the Kempe chain definition of neighbour given in Section 3.2) in which e is allocated a different session to the session allocated to it in x. For all such neighbours feasible with respect to the Exam Availability and Total Capacity constraints, the objective score is calculated and the neighbour q with minimum objective score is found. If there are several choices for q, select q to move e to the earliest time. If $o(q) \leq o(x)$, the current solution x is replaced by q, that is, the algorithm sets $x := q$; otherwise x is unchanged. In either case, this completes the processing of e, and the algorithm moves on to the next exam in the given order.

The hill climbing algorithm does not necessarily stop once all exams have been processed: at this point we say that one 'iteration' of the hill climbing algorithm has been completed. The hill climbing algorithm may well go on to again process all exams in the given order, several times over; it may perform many iterations. The hill climbing algorithm stops when either all neighbours generated during the last iteration had strictly larger objective scores than the current solution, or after a specified number of iterations. On the data sets we experimented with, there was usually no improvement in the objective score of the current solution after around 7 iterations, and consequently 10 hill climbing iterations were deemed to be sufficient; we specificied a limit of 10 iterations for the hill climbing algorithm.

Normally the hill climbing stage does not significantly improve the quality of solution produced by the simulated annealing phase. However, in approximately 5-10% of cases, the simulated annealing algorithm had not sufficiently

explored the neighbourhood of its best solution. For example, the best solution encountered may have been found early when the temperature was high, and moves that increase the objective are more frequently accepted. The algorithm may move away and settle in an inferior local minimum with a higher objective as the temperature is reduced. When this happenened, the hill climbing stage produced a significant improvement on the simulated annealing solution. The overall effect of the hill climbing phase of the algorithm is to make the results more consistent between runs of the three stage algorithm by improving the less satisfactory simulated annealing solutions.

3.4 A Greedy Heuristic for Scheduling Unscheduled Exams

The constraint programming stage may allocate dummy sessions to some exams (the exams remain unscheduled). Normally the simulated annealing stage will re-allocate normal sessions to these exams. However it is possible for unscheduled exams to remain after all three stages of the algorithm.

In all tests we have performed, the only cases where this occurred were in the University of Melbourne problem and then only in those data sets where a clash free solution is impossible. Although the University of Melbourne does allow clashes, when clashes occur, another constraint, known as the 'Three-in-a-Day' constraint, must be satisfied; we discuss that constraint in detail below. For the purpose of describing our heuristic, we simply assume that there is some set of *essential constraints*, that must be satisfied even when clashes are allowed.

In the event that exams remain unscheduled at the conclusion of the three stage method, we apply a greedy heuristic to schedule these exams. This heuristic will, of necessity, introduce clashes into the timetable, however it attempts to minimize these. Furthermore, it will ensure the timetable satisfies the essential constraints, and will leave an exam unscheduled rather than violate these constraints.

The heuristic proceeds as follows, where the unscheduled exams are considered in order of subject code. For each exam, e, left in a dummy session (that is, unscheduled), a normal session $t \in \{1, \ldots, v\}$ is chosen so that if exam e was scheduled in session t the essential constraints would hold, and so that the number of clashes introduced by scheduling exam e in session t is minimized over all such sessions. If there are no such sessions, then the exam remains unscheduled (though this never occurred in our tests).

4 The University of Melbourne Problem

At the University of Melbourne, there are 600-700 exams to be scheduled for each of two teaching semesters. The June exam period at the end of semester 1 normally has about 600 exams, to be scheduled in 26–30 sessions (13–15 days with morning and afternoon sessions). In the November exam period at the end of semester 2 there are more exams (at least 650), and these have to be scheduled in 30–34 sessions (15–17 days). There are five rooms in which exams can be held,

two of these on campus (with capacities 135 and 405) and three at an off campus venue (with capacities 540, 774 and 1170).

The constraints that the university imposes can be defined as Capacity, Total Capacity, Exam Availability, Room Availability and Large Exams. It is a matter of university policy that Large Exams is a soft constraint to be respected if possible, but in practice the university's current procedure takes Large Exams as a hard constraint. The Large Exam constraint can thus be treated as an Exam Availability constraint. The university also has some Pairwise Exam Constraints, in that some pairs (or sets) of exams have common content and are required to be held at the same time. We simply combine such exams into a single exam.

As there are only five available rooms, and splitting exams between the off-campus rooms does not present any difficulty, room allocation is not considered a serious issue at the University of Melbourne. Consequently, the University of Melbourne's current software, as well as our algorithm, initially only attempts to allocate a session to every exam, and does not allocate exams to rooms. Thus we only use the Capacity and Room Availability constraints to determine the total capacity for each session, needed for the Total Capacity constraint.

Unlike most educational institutions, the University of Melbourne does not enforce the Clashing constraint. Instead the university enforces a Student Restrictions constraint, called *Three-in-a-Day*, that students cannot have three exams scheduled in the same day (recall there are two sessions per day). By appropriate quarantining, students with two exams scheduled concurrently can be accommodated. This does, however, incur inconvenience and expense. The authors believed that allowing clashes and resolving them by quarantining students was unnecessary, and implemented the tighter Clashing constraint that no student may have two exams scheduled at the same time (as is required by our hybrid algorithm). If, at the end of the three stages of the hybrid algorithm, some exams remain unscheduled, we use the greedy heuristic described in Section 3.4, with essential constraints Total Capacity, Exam Availability and Three-in-a-Day, to allocate sessions to these exams. In all our test cases, the greedy heuristic successfully scheduled all remaining exams.

Data from two different semesters were examined. These were for semesters 1 and 2 of the 2001 academic year at the University of Melbourne (mel01s1 and mel01s2). The semester 1 data set (mel01s1) required 609 exams to be scheduled into 28 sessions (30 sessions, with two sessions lost in the middle to a public holiday), and the semester 2 data set (mel01s2) required 657 exams to be scheduled into 31 sessions. To cope with the Pairwise Exam constraint that some exams had to be held at the same time, such exams were combined into a single exam. This left 521 exams in mel01s1 and 562 exams in mel01s2.

The current University of Melbourne software (which we refer to for brevity as 'UM') is an unnamed and scantily documented VMS executable code specially written for the university about 10 years ago. Little is known about how the UM software works, but the authors conjecture from running it that it may use some form of simulated annealing algorithm.

The objective score used for University of Melbourne problems in our simulated annealing and hill climbing stages is calculated as follows. Recall that these stages maintain clash-free solutions, and that there are two sessions on each day, so no student can be sitting more than two exams allocated sessions on the same day. Using the notation of Section 3.1, we have normal exam sessions $T = \{1, \ldots, v\}$ ordered chronologically, and introduce dummy sessions $T' = \{v+1, \ldots, v'\}$. For a normal session t, we use boolean $w(t)$ true if t is the last session before a weekend or public holiday. For a given solution x, we calculate the objective score $o(x)$ as follows, where U is the penalty per student for unscheduled exams, w_{sd} is the penalty per student for two exams scheduled on the same day, w_{am} is the penalty per student for an exam scheduled in the afternoon followed by one scheduled in the morning of the next day, w_{g2} is the penalty per student for two exams scheduled with one session between them, and w_{ma} is the penalty per student for an exam scheduled in the morning of one day and an exam scheduled in the afternoon of the next day.

initialize $o(x) := \sum_{\{i \in E \,:\, x_i \in T'\}} U(s_i + 1)$
for each exam $i \in E$ with $x_i \in T$ and $w(x_i)$ not true **do**
  for each exam $j \in E$ with $x_j \in T$, $x_i < x_j \leq x_i + 3$ and $D_{ij} > 0$ **do**
    if $x_j = x_i + 1$ **then**
      if x_i is a morning session **then** $o(x) := o(x) + w_{sd} D_{ij}$
      else (x_i must be an afternoon session) $o(x) := o(x) + w_{am} D_{ij}$
    else if $w(x_i + 1)$ not true **then**
      if $x_j = x_i + 2$ **then** $o(x) := o(x) + w_{g2} D_{ij}$
      else (x_j must equal $x_i + 3$)
        if $w(x_i + 2)$ not true **then** $o(x) := o(x) + w_{ma} D_{ij}$
      endif
    endif
  endfor
endfor

In all tests on University of Melbourne data we used $U = 10\,000$, and after trying a number of different values for the penalty parameters, we found those that gave best results were $w_{sd} = 2$, $w_{am} = 1$, $w_{g2} = 0$, and $w_{ma} = 0$. So in fact we found that a much simpler score would have sufficed for this data.

In addition to the exam and student data sets, the university also provided the data for Exam Availability and Large Exams constraints. As the Large Exams constraint is treated as a hard constraint by the university's timetabling software, this was combined with the Exam Availability constraint in order to restrict the large exams to earlier times. Upon initial examination, it was discovered that due to the Large Exams being treated as a hard constraint, it was impossible to avoid clashes in the timetable for the mel01s2 data set. For example, exams with more than 500 students were required to be held in the first 14 exam sessions, and because of the number of exams in this category, it was not possible to find a clash-free solution. However, if these exams were allowed to be

Table 1. *Comparison between the University of Melbourne current software (UM) and our hybrid algorithm.* The best timetable was deemed to be the one with the fewest clashes, with ties resolved by choosing the lowest objective score. The corresponding score for the best timetable can exceed the average. Note that it was not possible to compare directly to the UM program for the mel01s1 data set, and consequently the hybrid method was only compared to the timetable produced by the university for that semester. This timetable had been manipulated afterwards to satisfy constraints not known when the timetable was produced, and was not minimizing the same objective.

Data Set			Hybrid	UM
mel01s1	521 Exams	Best Clashes	**0**	8
	28 Sessions	Corresponding Score	**1210**	2085
		Average Clashes	0	—
		Average Score	1503.6	—
		Average Time (s)	56	—
mel01s2a	562 Exams	Best Clashes	2	**1**
	31 Sessions	Corresponding Score	**1835**	2384
		Average Clashes	5.4	1.4
		Average Score	1514	2298
		Average Time (s)	74	1008
mel01s2b	562 Exams	Best Clashes	**0**	**0**
	31 Sessions	Corresponding Score	**1115**	3373
		Average Clashes	0	1
		Average Score	1300	2632.2
		Average Time (s)	73	1008

scheduled in the first 15 exam sessions, (one more than previously allowed), it was possible to produce a clash-free solution.

In order to test the hybrid algorithm under the assumption that clash-free solutions were possible, two different data sets for this semester were used: one used the Exam Availability and Large Exams data sets that the university used for that semester, which forced clashes due to the Large Exams constraint (mel01s2a), and the second modified the Large Exams constraint slightly so that clash-free solutions were possible (mel01s2b). Five different runs were made by the hybrid algorithm on all data sets (with the standard simulated annealing parameters and 10 hill climbing iterations) and five different runs were made by the UM program on the mel01s2a and mel01s2b data sets using the university's parameters. The results are shown in Table 1.

It can be seen that the hybrid method is superior in terms of the objective and time. The algorithms were run on different machines, the hybrid algorithm on an XP900 Alphastation (466Mhz CPU) and the UM algorithm on a DEC Alphastation 8200 (dual 300Mhz CPU), but the time difference is still significant. The hybrid method was not designed to cope with clashes, and thus it does not prove as good at minimizing the number of clashes compared to the university's program when clashes are unavoidable. Conversely, in the potentially clash-free data sets, the university's program does not always provide a clash-free solution.

Table 2. *Papers on examination timetabling for publicly available data sets.* **B1:** Burke et al. [4], **B2:** Burke et al. [6], **B3:** Burke and Newall [3], **Ca:** Caramia et al. [7], **C:** Carter et al. [10], **D:** Di Gaspero and Schaerf [13] and **W:** White and Xie [19]. *Note that Burke et al. [6] (B2) use a slightly different version of the KFU-S-93 data set to that of the other authors; they use a different number of sessions.

Data Set	Exams	P1	P2	P3	P4
CAR-F-92	543	C,Ca	C,Ca,D,W	B1,D,Ca	B2,B3,D
CAR-S-91	682	C,Ca	C, Ca,D	B1,D,Ca	—
EAR-F-83	189	C,Ca	C,Ca,D	—	—
HEC-S-92	80	C,Ca	C,Ca,D	—	—
KFU-S-93	461	C,Ca	C,Ca,D	B1,D,Ca	B2*,B3,D
LSE-F-91	381	C,Ca	C,Ca,D	—	—
PUR-S-93	2419	C,Ca	C,Ca	—	B3,D
RYE-F-92	487	C,Ca	C,Ca	—	—
STA-F-83	138	C,Ca	C,Ca,D	—	—
TRE-S-92	261	C,Ca	C,Ca,D	B1,D,Ca	—
UTA-S-92	638	C,Ca	C,Ca,D,W	B1,D,Ca	B2
UTE-S-92	184	C,Ca	C,Ca,D	—	—
YOR-F-83	180	C,Ca	C,Ca,D	—	—
NOTT	800	—	—	B1,D,Ca	B3,D

5 Benchmarks

5.1 Publicly Available Data

As stated in Section 2.2, Carter and Burke have made data sets for exam timetabling publicly available. For each of these data sets, attempts have been made to solve up to four different problems. The solutions to these problems provide benchmarks to compare different exam timetabling methods. The problem definitions, and consequently the initial benchmarks for these problems, are presented in the papers of Carter et al. [10] (problems P1 and P2), Burke et al. [4] (problem P3), and Burke and Newall [3] (problem P4). Below we discuss in detail each of these problems and the results of solving them with our method.

Four other papers compared alternative methods to the original benchmarks established by Burke et al. [3, 4] and Carter et al. [10]: Burke et al. [6], Di Gaspero and Schaerf [13], Caramia et al. [7], and White and Xie [19]. In Table 2 the publicly available data sets are listed, along with the papers that have tested methods on each problem. It should be noted that the computing resources used differ. We have not been able to convert the reported run times in seconds to equivalent run times for a common standard, so all tabulated times reported below should be taken as indicative only.

5.2 Problem P1: Graph Colouring Benchmarks

Carter, Laporte and Lee [10] look at several of the publicly available data sets, and attempt to find the minimum number of sessions required for a feasible

timetable subject only to the Clashing constraint. They set no Capacity or Total Capacity constraints, and thus the problem reduces to a graph colouring problem. The sequential construction heuristic of Carter et al. [10] produces a variety of solutions, depending on the order in which the the exams were processed. Using a different sequential construction heuristic, Caramia et al. [7] managed to produce equal (and superior in one case) results to Carter et al. [10].

The constraint programming stage of the hybrid method produces solutions that tend to use a relatively small number of sessions, as we demonstrate in Table 3. Here we compare the number of sessions used in the solution produced by the constraint programming stage with the results of Carter et al. (Table 3 in [10]) and Caramia et al. (Table 1 in [7]). The results of the comparison can be seen in Table 3.

The constraint programming stage of the hybrid method only produced one solution with as few sessions as the best of the Carter et al. [10] and Caramia et al. [7] methods. However, in all but one data set, the constraint programming phase produced a solution with only one or two more sessions than the minimum attained by the other methods, and never used more than four additional sessions. These results confirm that the constraint programming stage produces timetables with close to the minimum number of sessions.

5.3 Problem P2: Uncapacitated Benchmarks

In addition to their work on graph colouring benchmarks, Carter, Laporte and Lee [10] developed a second uncapacitated version of the examination timetabling problem. They set a maximum number of sessions, and devise an objective function designed to favour timetables which space out students' exams. The objective function applies a penalty w_t to a timetable whenever a student has to sit two exams scheduled t periods apart, with $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$ and $w_5 = 1$. The total penalty is divided by the number of students to get an average penalty per student; this is the value of the objective function for the given timetable. No account was taken of weekends and there was no differentiation between consecutive exam periods within the same day, versus overnight.

For this problem class, our hybrid algorithm uses an objective score identical to the objective function defined by Carter et al. [10], calculated as follows.

initialize $o(x) := \sum_{\{i \in E \,:\, x_i \in T'\}} U(s_i + 1)$
for each exam $i \in E$ with $x_i \in T$ **do**
  for each exam $j \in E$ with $x_j \in T$, $x_i < x_j \leq x_i + 5$ and $D_{ij} > 0$ **do**
    $o(x) := o(x) + w_{(x_j - x_i)} D_{ij}$
  endfor
endfor

Caramia et al. [7], Di Gaspero and Schaerf [13] and White and Xie [19] compared their algorithms to that used by Carter et al. [10], using the same problem definition, on some or all of the P2 data sets. The method of Caramia

Table 3. *Problem P1: graph colouring benchmarks.* The constraint programming stage of the hybrid algorithm yields timetables with close to the minimum number of sessions. The time reported for the methods of Carter et al. [10] and Caramia et al. [7] is the time for the run producing the best result. The time for the hybrid algorithm is the average time per run. Reported times have not been converted to account for different computing resources.

Data Set	No. of Exams		Hybrid Stage 1	Carter et al.	Caramia et al.
CAR-F-92	543	Best	31	**28**	**28**
		Range	—	28–32	28–32
		Time (sec)	5.37	227.2	559.2
CAR-S-91	682	Best	30	**28**	**28**
		Range	—	28–35	28–32
		Time (sec)	7.34	75.1	86.3
EAR-F-83	189	Best	24	**22**	**22**
		Range	—	22–24	22–23
		Time (sec)	1.2	8.7	86.3
HEC-S-92	80	Best	18	**17**	**17**
		Range	—	17–18	17–18
		Time (sec)	0.36	0.5	10.6
KFU-S-93	461	Best	21	**19**	**19**
		Range	—	19–20	19–20
		Time (sec)	3.4	97.2	159.6
LSE-F-91	381	Best	18	**17**	**17**
		Range	—	17–18	17–18
		Time (sec)	2.45	78.0	9.6
RYE-F-92	487	Best	22	**21**	**21**
		Range	—	21–23	21–23
		Time (sec)	3.95	343.8	225.9
STA-F-83	138	Best	**13**	**13**	**13**
		Range	—	13–13	13–13
		Time (sec)	0.62	2.7	10.2
TRE-S-92	261	Best	21	**20**	**20**
		Range	—	20–23	20–23
		Time (sec)	1.03	32.8	214.7
UTA-S-92	638	Best	32	32	**30**
		Range	—	32–35	30–34
		Time (sec)	6.39	272.3	1023.5
UTE-S-92	184	Best	11	**10**	**10**
		Range	—	10–10	10–10
		Time (sec)	0.75	1.6	24.3
YOR-F-83	180	Best	23	**19**	**19**
		Range	—	19–21	19–21
		Time (sec)	1.05	190.4	226.2

Table 4. *Problem P2: uncapacitated benchmarks.* The number of sessions is restricted, but there are no capacity constraints. Scores listed under 'best' and 'average' are per student scores. For the hybrid method, this is obtained by dividing the final score by the number of students. Times reported are average times per run for the hybrid method, and times of the best run for Carter et al. [10] and Caramia et al. [7]. Reported times have not been converted to account for different computing resources.

Data Set		Hybrid	Carter et al.	Caramia et al.	Di Gaspero & Schaerf	White & Xie
CAR-F-92	Best	**4.2**	6.2	6.0	5.2	—
Exams 543	Average	4.3	7.04	—	5.6	4.7
Sessions 32	Time (sec)	171	47	142.7	—	—
CAR-S-91	Best	**5.1**	7.1	6.6	6.2	—
Exams 682	Average	5.2	8.38	—	6.5	—
Sessions 35	Time (sec)	296	20.7	34.7	—	—
EAR-F-83	Best	34.7	36.4	**29.3**	45.7	—
Exams 189	Average	36.9	40.92	—	46.7	—
Sessions 24	Time (sec)	26	24.7	29.3	—	—
HEC-S-92	Best	10.5	10.8	**9.2**	12.4	—
Exams 80	Average	10.8	15.04	—	12.6	—
Sessions 18	Time (sec)	5.4	7.4	11.0	—	—
KFU-S-93	Best	13.9	14.0	**13.8**	18.0	—
Exams 461	Average	14.1	18.76	—	19.5	—
Sessions 20	Time (sec)	40	120.2	112.8	—	—
LSE-F-91	Best	11.0	10.5	**9.6**	15.5	—
Exams 381	Average	11.1	12.36	—	15.9	—
Sessions 18	Time (sec)	35	48.0	92.8	—	—
RYE-F-92	Best	8.8	7.3	**6.8**	—	—
Exams 487	Average	8.9	8.68	—	—	—
Sessions 23	Time (sec)	70	507.2	89.4	—	—
STA-F-83	Best	**157.3**	161.5	158.2	160.8	—
Exams 138	Average	157.4	167.14	—	166.8	—
Sessions 13	Time (sec)	5.1	5.7	6.5	—	—
TRE-S-92	Best	**8.5**	9.6	9.4	10.0	—
Exams 261	Average	8.6	10.78	—	10.5	—
Sessions 18	Time (sec)	39	107.4	102.8	—	—
UTA-S-92	Best	**3.5**	**3.5**	**3.5**	4.2	—
Exams 638	Average	3.6	4.8	—	4.5	4.0
Sessions 35	Time (sec)	233	664.3	589.4	—	—
UTE-S-92	Best	25.2	25.8	**24.4**	29.0	—
Exams 184	Average	25.5	30.78	—	31.3	—
Sessions 10	Time (sec)	8.6	9.1	5.0	—	—
YOR-F-83	Best	37.2	41.7	**36.2**	41.0	—
Exams 180	Average	38.0	45.6	—	42.1	—
Sessions 21	Time (sec)	30	271.4	125.4	—	—

et al. [7] proved to be superior on 10 of the 13 data sets, equal to Carter et al. [10] on the UTA data set, with Di Gaspero and Schaerf [13] superior on the other two data sets.

The hybrid method described in this paper was run on 12 of the data sets, and the results are summarized in Table 4 (compared to results from Table 5 in Carter et al. [10], Table 3 in Caramia et al. [7], Table 1 in Di Gaspero and Schaerf [13], and Table 9 in White and Xie [19]). Note that for all these data sets, the hybrid algorithm produced a clash-free timetable within the maximum allowed number of sessions, without recourse to the fourth greedy heuristic stage. The hybrid method is superior to that of Di Gaspero and Schaerf [13] and to that of White and Xie [19], and better than Carter et al. on 9 of 12 data sets (with one tie). However, the method of Caramia et al. [7] produces the best results with superior results to the hybrid method in 7 of the data sets (again with a tie on the UTA-S-92 data set).

5.4 Problem P3: Capacitated Benchmarks (Set 1)

Burke, Newall and Weare [4] created a new class of capacitated problem for the publicly available data sets, having three sessions per weekday with a morning session on Saturday. It was assumed that the exam period starts on a Monday. They set a maximum number of exam sessions and imposed the Clashing and Total Capacity constraints. For the Nottingham data sets (NOTT), an Exam Availability constraint was also applied: exams over two hours in length had to be held in the first session of the day. The objective was to minimize the number of instances of a student having two exams in a row on the same day. For this problem class, our hybrid algorithm uses an identical objective score, calculated as follows.

initialize $o(x) := \sum_{\{i \in E \,:\, x_i \in T'\}} U(s_i + 1)$

for each exam $i \in E$ with $x_i \in T$ and x_i not the last session of the day **do**
  for each exam $j \in E$ with $x_j = x_i + 1$ and $D_{ij} > 0$ **do**
    $o(x) := o(x) + D_{ij}$
  endfor
endfor

Di Gaspero and Schaerf [13] and Caramia et al. [7] applied their methods to this problem class. The results of Burke et al. [4] were bested by either Di Gaspero and Schaerf [13] or Caramia et al. [7] in every case, each with approximately half the best results on the data sets examined. Our hybrid method, with the objective score calculated as above, was tested on these data sets: it was run five different times for each problem instance. Note that for all these data sets, the hybrid method produced a clash-free timetable within the maximum allowed number of sessions, without recourse to the fourth greedy heuristic stage. The results can be seen in Table 5 (compared to results in Tables 1 and 5 in Burke et al. [4], Table 4 in Caramia et al. [7] and Table 2 in Di Gaspero and Schaerf [13]).

Table 5. *Problem P3: capacitated benchmarks, set 1.* The objective to be minimized is the number of students with two consecutive exams on the same day. Times reported are average times per run for the hybrid method, times of the best run for Caramia et al. [7] and approximate run times for Burke et al. [4]. Reported times have not been converted to account for different computing resources.

Data Set			Hybrid	Burke et al.	Caramia et al.	Di Gaspero & Schaerf
CAR-F-92						
Exams	543	Best	**158**	331	268	424
Sessions	31	Average	212.8	—	—	443
Capacity	2000	Time (sec)	96	24240	80.4	—
CAR-S-91						
Exams	682	Best	**31**	81	74	88
Sessions	51	Average	47	—	—	98
Capacity	1550	Time (sec)	125	21120	31.4	—
KFU-S-93						
Exams	461	Best	**237**	974	912	512
Sessions	20	Average	290.6	—	—	597
Capacity	1995	Time (sec)	45	24240	118.2	—
TRE-S-92						
Exams	261	Best	**0**	3	2	4
Sessions	35	Average	0.4	—	—	5
Capacity	655	Time (sec)	16	10800	222.4	—
NOTT						
Exams	800	Best	**2**	53	44	11
Sessions	26	Average	15.6	—	—	13
Capacity	1550	Time (sec)	44	24240	359.1	—
NOTT						
Exams	800	Best	**83**	269	—	123
Sessions	23	Average	105	—	—	134
Capacity	1550	Time (sec)	42	18000	—	—
UTA-S-92						
Exams	638	Best	**334**	772	680	554
Sessions	38	Average	393.4	—	—	625
Capacity	2800	Time (sec)	173	24000	265.1	—

Clearly the hybrid method is the superior method when applied to these capacitated versions of the data sets, providing the best solution in all instances. The method consistently provides lower scores than Di Gaspero and Schaerf [13], which we believe is due to the choice of Kempe chain neighbourhood. The results provided by Burke et al. [4] are no longer competitive. However it is important to note that on the Nottingham data sets, Burke et al. [4] set a Capacity constraint and allocated rooms to the exams. The hybrid method did not do this, whilst Di Gaspero and Schaerf [13] and Caramia et al. [7] do not state whether they do this or not. The extra constraints applied by Burke et al. [4] may be part of the reason for their relatively poor results on these data sets.

The major difference between the quality of the results produced in the uncapacitated version and the capacitated versions of these problems is between our hybrid method and the algorithm of Caramia et al. [7]. In the uncapacitated version, the algorithm of Caramia et al. [7] is superior in seven of the data sets, and the hybrid method in four, but with the capacitated data sets, the hybrid method is superior in all six instances (Caramia et al. [7] do not attempt the Nottingham data set with 23 sessions). Of these six instances, three were data sets that the hybrid method was superior for the uncapacitated version and two had almost equal results (the Nottingham data set was not attempted as uncapacitated). Therefore, one would expect that the hybrid method would fare better with this comparison. However, the improvement cannot be explained by simply favourable data sets alone, as there is a significant difference in this problem for data sets that were comparable for problem P2.

5.5 Problem P4: Capacitated Benchmarks (Set 2)

Burke and Newall [3] build on previous work undertaken in Burke et al. [4], using a modified version of their memetic algorithm. They looked at some of the publicly available data sets, and optimized these with a different objective function. They consider a situation with three exam sessions per day on weekdays (morning, lunchtime and afternoon) and one exam session on Saturday morning. The objective function considers only students with two exams in two consecutive sessions. They give a penalty of three per student for two exams in a row in the same day, and one per student for two exams in a row overnight. For this problem class, our hybrid algorithm uses an identical objective score, calculated as follows.

initialize $o(x) := \sum_{\{i \in E \,:\, x_i \in T'\}} U(s_i + 1)$
for each exam $i \in E$ with $x_i \in T$ and x_i not on a Saturday **do**
  for each exam $j \in E$ with $x_j = x_i + 1$ and $D_{ij} > 0$ **do**
    if x_i is the last session of the weekday **then**
      $o(x) := o(x) + D_{ij}$
    else (x_i must be a morning or lunchtime session of a weekday)
      $o(x) := o(x) + 3D_{ij}$
    endif
  endfor
endfor

The Nottingham data set has the Exam Availability constraint that any exam over 2 hours in length must be held in a morning session.

Di Gaspero and Schaerf [13] compared their algorithm to the results produced by Burke and Newall [3]. Burke et al. [6], independent of their work on memetic algorithms, wrote another paper on sequential construction heuristics using the publicly available data. This method was was not compared directly to any other work, though, as it was run on the CAR-F-92 data set with the same problem

Table 6. *Problem P4: capacitated benchmarks, set 2.* The objective function is based on students with exams in consecutive sessions. Times reported are average times per run. Reported times have not been converted to account for different computing resources.

Data Set			Hybrid	Burke & Newall	Burke et al.	Carter et al.	Di Gaspero & Schaerf
CAR-F-92							
Exams	543	Best	2188	**1665**	2555	2915	3048
Sessions	36	Average	2267.6	1765	—	—	3377
Capacity	2000	Time (sec)	106	186	—	—	—
KFU-S-93							
Exams	461	Best	**1337**	1388	—	2700	1733
Sessions	21	Average	1487.8	1608	—	—	1845
Capacity	1995	Time (sec)	39	105	—	—	—
NOTT							
Exams	800	Best	720	**498**	—	918	751
Sessions	23	Average	784.8	544	—	—	820
Capacity	1550	Time (sec)	44	467	—	—	—

definition, it can be compared for this problem instance. Unfortunately the other data this paper used was the KFU-S-93 set with a different number of sessions, and the UTA-S-92 data set, which have not been tested by other authors.

The hybrid method was run on the data sets used in Burke and Newall [3]. The results, together with the results of Burke and Newall [3], Burke et al.[6] (for the CAR-F-92 problem only), Carter et al. [10] and Di Gaspero and Schaerf [13], can be seen in Table 6 (using data from Tables 2, 3 and 4 in Burke and Newall [3] and Table 3 in Di Gaspero and Schaerf [13]).

Clearly the algorithm of Burke and Newall [3] is superior: for two of the data sets their results are clearly the best, and for the third they are only just behind our hybrid algorithm. It should be noted that the values for the memetic algorithm for Burke and Newall [3] in this table have been chosen as the best from 5 different runs with 18 different parameter settings. This is a total of 90 different runs for each data set, compared to 5 for the hybrid method. However, if the hybrid method is run many times, the scores do not improve significantly, as the method is tied to the solution produced deterministically by the constraint programming stage, which does not change with multiple runs. Instead the hybrid method requires more simulated annealing and hill climbing iterations, to allow it to move further away from the initial solution, to improve the solution. To test this theory a series of longer runs were undertaken on these data sets. In addition to the standard run (approximately 350 000 simulated annealing iterations and 10 hill climbing iterations[7]), a medium length run (approximately 3 500 000 simulated annealing iterations and 30 hill climbing iterations[8]) and a long run (approximately 35 000 000 simulated annealing iterations and 100 hill

[7] $\alpha = 0.999, a = 10$
[8] $\alpha = 0.999, a = 100$

Table 7. *Problem P4: longer hybrid runs.* Our experiments and those of Burke and Newall [3] are on similar but not identical machines.

Data Set			Hybrid Standard	Hybrid Medium	Hybrid Long	Burke & Newall
CAR-F-92						
Exams	543	Best	2188	1809	1744	**1665**
Sessions	36	Average	2267.6	1970.6	1801.4	1765
Capacity	2000	Time (sec)	106	828	6671	186
KFU-S-93						
Exams	461	Best	1337	1182	**1082**	1388
Sessions	21	Average	1487.8	1255.6	1214.4	1608
Capacity	1995	Time (sec)	39	348	3202	105
NOTT						
Exams	800	Best	720	492	**371**	498
Sessions	23	Average	784.8	576.2	425.4	544
Capacity	1550	Time (sec)	44	317	2818	467

climbing iterations[9]) were performed on the data sets. The results can be seen in Table 7.

It is quite clear that the longer the hybrid algorithm is allowed to run, the better the quality of the solution. However, with the CAR-F-92 data set, even though the hybrid algorithm is allowed to run about 50 times longer than the memetic algorithm of Burke and Newall [3], the solution remains slightly inferior (less than 5% worse). While it looks plausible that if the hybrid algorithm was allowed to perform an even longer run it would provide a superior solution, it is clearly an inferior method for this data set. However, for the other two data sets, the hybrid method does provide superior solutions in less time (a short run for KFU-S-93, and a medium run for NOTT). Unfortunately, with only three different comparisons between the methods, it is difficult to conclude definitively which method is superior.

6 Conclusions

The hybrid method for examination timetabling described in the present paper is superior to the method currently used by the University of Melbourne, and performs well in comparison to other, well known methods that have been applied on the publicly available data sets. However, too few of these data sets have had benchmarks established on them (problem P4 has only been developed for 5 of the 14 data sets), and only three comparisons have been made between the hybrid method and the Burke and Newall [3] memetic algorithm with sequential construction. It is therefore not possible to make a definitive assessment of the relative quality of solutions found by the hybrid method and

[9] $\alpha = 0.999, a = 1000$

by existing methodologies. In addition, it is desirable to perform comparisons of the algorithms using the same computer resources.

In spite of the shortcomings of the comparisons, the hybrid method is still proven as a worthwhile algorithm, among the best currently in use for examination timetabling. The constraint programming stage provides a fast route to a first feasible solution. This solution is improved by the simulated annealing stage, with the Kempe chain neighbourhoods proving effective at diversifying solutions (though occasionally more time is needed). The hill climbing stage further improves the solutions, and reduces the effect of unfavourable fluctuations in the simulated annealing stage.

We suggest that the dominant methods of the future for the examination timetabling problem will combine solution construction with local search. The stages of the hybrid method may be integrated more fully, to produce a still more powerful algorithm.

Acknowledgements

The authors would like to thank Aiden Tran and Gerry Barretto for providing the University of Melbourne data, and Michael Carter, Luca Di Gaspero and James Newall for help they gave in understanding their previous work.

References

1. P. Boizumault, Y. Delon, and L. Peridy. Constraint logic programming for examination timetabling. *Journal of Logic Programming*, 26:217–233, 1996.
2. E.K. Burke, K. Jackson, J. Kingston, and R.F. Weare. Automated university timetabling: the state of the art. *The Computer Journal*, 40:565–571, 1997.
3. E.K. Burke and J. Newall. A multistage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3:63–74, 1999.
4. E.K. Burke, J. Newall, and R.F. Weare. A memetic algorithm for university exam timetabling. In: Burke, E.; Ross, P. (eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August/September 1995. Selected Papers. *Lecture Notes in Computer Science* 1153, Springer-Verlag, Berlin Heidelberg New York, 1996, 241–250.
5. E.K. Burke, J. Newall, and R.F. Weare. Initialisation strategies and diversity in evolutionary timetabling, *Evolutionary Computation Journal (special issue on Scheduling)*, vol 6.1, pp. 81–103, 1998.
6. E.K. Burke, J. Newall, and R.F. Weare. A simple heuristically guided search for the timetable problem. *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems*, pp. 574–579, 1998.
7. M. Caramia, P. Dell'Olmo, and G.F. Italiano. New algorithms for examination timetabling. In: Nher, S.; Wagner, D., (eds.): Algorithm Engineering 4th International Workshop, WAE 2000, Saarbrücken, Germany, September 2000. Proceedings. *Lecture Notes in Computer Science* 1982, Springer-Verlag, Berlin Heidelberg New York, 2001, 230–241.

8. M. Carter and G. Laporte. Recent developments in practical examination timetabling. In: Burke, E.; Ross, P. (eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August/September 1995, Selected Papers. *Lecture Notes in Computer Science* 1153, Springer-Verlag, Berlin Heidelberg New York, 1996, 373–383.
9. M. Carter, G. Laporte, and J. Chinneck. A general examination scheduling system. *Interfaces*, 24:109–120, 1994.
10. M. Carter, G. Laporte, and S.T. Lee. Examination timetabling: algorithmic strategies and applications. *Journal of the Operational Research Society*, 47:373–383, 1996.
11. K. Dowsland. Using simulated annealing for efficient allocation of students to practical classes. In: Vidal, R.V.V. (ed.): Applied Simulated Annealing. *Lecture Notes in Economics and Mathematical Systems* 396, Springer-Verlag, Berlin New York, 1993, 125–150.
12. K. Dowsland. Off-the peg or made-to-measure? Timetabling and scheduling with sa and ts. In: Burke, E.; Carter, M. (eds.): Practice and Theory of Automated Timetabling II Second International Conference, PATAT'97, Toronto, Canada, August 1997. Selected Papers. *Lecture Notes in Computer Science* 1408, Springer-Verlag, Berlin Heidelberg New York, 1998, 37–52.
13. L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In: Burke E.; Erben W. (eds.): Practice and Theory of Automated Timetabling III Third International Conference, PATAT 2000, Konstanz, Germany, August 16–18, 2000. Selected Papers. *Lecture Notes in Computer Science* 2079, Springer-Verlag, Berlin Heidelberg New York, 2001, 104–117.
14. P. Van Hentenryck. *The OPL Optimization Programming Language*. MIT Press, Cambridge, Massachusetts, 1999.
15. A. Schaerf. Tabu search techniques for large high-school timetabling problems. *Proceedings of the National Conference of the American Association for Artificial Intelligence*, AAAI Press, Menlo Park Cambridge London, 1996, 363–368.
16. J. Thompson and K. Dowsland. General cooling schedules for a simulated annealing timetabling system. In: Burke, E.; Ross, P. (eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August/September 1995. Selected Papers. *Lecture Notes in Computer Science* 1153, Springer-Verlag, Berlin Heidelberg New York, 1996, 345–363.
17. J. Thompson and K. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, 63:105–128, 1996.
18. J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Computers and Operations Research*, 25:637–648, 1998.
19. G.M. White and B.S. Xie. Examination timetables and tabu search with longer-term memory. In: Burke E.; Erben W. (eds.): Practice and Theory of Automated Timetabling III Third International Conference, PATAT 2000, Konstanz, Germany, August 16-18, 2000. Selected Papers. *Lecture Notes in Computer Science* 2079 Springer-Verlag, Berlin Heidelberg New York, 2001, 85–103.
20. G.M. White and J. Zhang. Generating complete university timetables by combining tabu search with constraint logic. Practice and Theory of Automated Timetabling II Second International Conference, PATAT'97, Toronto, Canada, August 1997. Selected Papers. *Lecture Notes in Computer Science* 1408, Springer-Verlag, Berlin Heidelberg New York, 1998, 187–198.
21. M. Yoshikawa, K. Kaneko, Y. Nomura and M. Watanabe. A constraint-based approach to high-school timetabling problems: a case study. *Proceedings of the National Conference of the American Association for Artificial Intelligence*, AAAI Press, Menlo Park Cambridge London, 1994, 1111–1116.

An Evolutionary Approach For The Examination Timetabling Problems

Kaveh Sheibani

Tadbir Institute for Operations Research
and the Systems Design, Tehran, Iran
ksheibani@inforsd.com

Abstract. One of the most significant problems in the training centers is presenting an exam timetabling due to enrolled subjects for each student with maximum spread between exams. In this paper, a special mathematical programming model is presented which its purpose is to minimize interference of the time period of each exam for each student. It is too difficult and even impossible to optimally solve the above problem in a reasonable computational time. Thus, an evolutionary computing, for example genetic algorithm, is used as one of the stochastic search methods for solving an exam timetabling problem. The problem is considered as the assignment of facilities (i. e., exam subject) to the specific locations (i. e., positions). In other word, the exam subjects are assigned within time periods to its optimal position.

1 Introduction

Planning and scheduling problems have a sensitive and special complexity. All of the people are often faced up to the planning scheduling problems in their daily affairs. In this case, we can point to the transportation centers, railways, bus lines, airlines, ship lines, conferences and so on, and specially in the training centers and institutions [1]. In this paper, an exam scheduling problem is considered and an appropriate problem-solving method is proposed. Generally, the schedulers are faced up to many limitations. For instance, suppose the students enrolled in different subjects cannot obviously give exams at the same time. The limitations on the numbers and capacity of salons for classes or an exam and the preference of instructors to choose a specific class are also the problems that have effect on choosing the subjects by students [2]. Without proper scheduling, we face low utilization in the overall system. In this way, a computer program based on the efficient algorithm is needed to provide an appropriate course scheduling in order to response the system's requirement [3]. In the last decade, a number of applied researches have been carried out and the last international conference on this subject was held in Germany in August 2000.

Graph coloring [4], quadratic assignment problems (QAP) [5], traveling salesman problems (TSP) [6] and tabu search [7] are used to solve the above problems by considering the mentioned limitations. A genetic algorithm [8 and 9] is one of the efficient and robust methods to solve exam scheduling problems. In this

paper, a special mathematical model is proposed to solve such problem based on genetic algorithms. The output shows the assignment of exam subjects as facilities in the best possible status of the given interval.

2 Examination Scheduling Problems

There is usually a final exam schedule in any training center so that there are some limited numbers of classes in a given period of time. By considering the common limitations, the problem will be discussed:

a) The exams are set up for two days
b) Each day is divided into two intervals of two hours

The aim is to assign the exam subjects in the given period in such a way the time period between exam subjects which have been chosen by each student be maximized and or if possible they not have two exams in one day. In other words, the aim is to establish the courses in the optimum assignment of exam subject in the timetable by considering the close relationship between the exam subjects.

3 Mathematical Model

The exam schedule problem is supposed as a facilities layout problem. In Table 1, the exam subjects known as facilities are assigned to the locations in a considered level. In other words, n exam subjects must be assigned in n places considering the optimality condition. This problem is a class of combinatorial optimization problems known as a NP-hard [10 and 11]. Thus, a number of heuristic algorithms have been proposed to solve the above problem [12 and 13]. In this paper, a special mathematical model is presented and solved by genetic algorithms in order to allocate the best possible period in an examination timetable. The objective function is defined as maximizing the intervals in the courses chosen by a student in a semester by considering the relationship between them. This objective function leads to the maximum value in the global space problem accurately.

$$\text{Max } Z = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(|x_i - x_j| \right) \quad \forall i, j \tag{2}$$

$$x_i \geq 0, \quad \text{integer} \quad \forall i \tag{3}$$

s.t.:

where,
$|x_i - x_j|$ = distance between S_i and S_j

w_{ij} = weight between S_i and S_j
n = number of classes/ locations
S_i = exam subject i

Equation 1 indicates the objective function of the exam schedule problem, which maximizes the interval between exam subjects with the minimum closeness relationship. First constraint given in Eq. 2 ensures that distance variable between exam subjects is always a positive integer number and equal or smaller than total numbers period terms or make span. Second constraint given in Eq. 3 shows the location of exam subjects in the string called chromosome, which its gene's values, is always integer numbers.

Table 1-An examination timetabling

	Period 1		Period 2	
Day1	S_i			
Day2				S_j

4 Genetic Algorithms

Genetic algorithms (GAs) [14 and 15] are a class of stochastic search techniques that use a series of rules controlled by genetic operators. They are one of the promising and robust algorithms to first generate different course timetabling at random. GAs try to improve the solution quality in a proper way by using a selection mechanism and genetic operators in order to find the best solution in the space problem searched so far [14 and 16]. GAs have been applied to different problems such as combinatorial optimization, reliability, job shop scheduling, transportation, facilities layout and location, sequencing, aggregate production planning, nonlinear programming, minimal spanning tree, traveling salesman, and quadratic assignment problems [16]. The main components of genetic algorithms follow as genetic coding and representation, initialization, fitness function, selection scheme, reproduction, crossover and mutation operators. Figure 1 shows a simple structure of GAs [18 and 19].

1) Selecting a genetic coding system (chromosome),
2) Defining an appropriate fitness function,
3) Finding a good selection mechanism,
4) Choosing and designing the genetic operators such as crossover and mutation,
5) Establishing the genetic parameters for controlling the algorithm,
6) Terminating the algorithm when the conditions are satisfied, and
7) Reporting the statistic output and the best solution searched so far.

Children

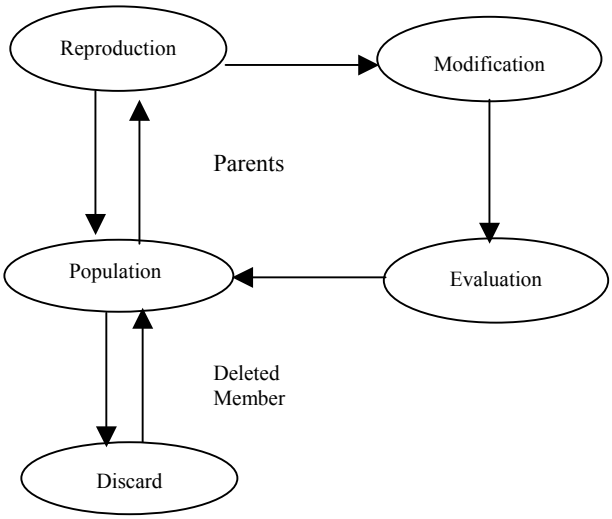

Fig. 1. Flow chart of a genetic algorithm

5 Close relationships between exam subjects

In this paper, the method is proposed for relating the exam subjects and assigning the quantitative values in such a way that a user is able to change and modified the method in possible limitations. It should be noted that the relationship between subjects must be realizable by computer in terms of numerical values. The university courses are generally prerequisite to some other courses that are the main factors to choose a subject in a semester. The subjects chosen by a student are guessed. One of the best and efficient tools to do this is an activity-on-arrow network. It guesses which subjects that a student for each course must take in a given semester. By using this network, the subjects are classified into a specific set. The set 1 represents the permissible subjects that can be chosen in Semester 1 and the set 2 represents the permissible subjects chosen in Semester 2, and so on. In other word, the member of a set i is prerequisite to set i+1. It is possible that a student may not pass the exam, then another constraint is introduced to the left and right of the set i, i.e. sets i+1 and i. A characteristic string is defined for every subject. Figure 2 shows this string where the cell 0 represents the subject code; the cell 1 represents the sets taken by the subjects.

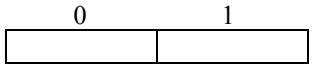

Fig. 2. A characteristic string

The proposed algorithm for assigning the quantitative values between subjects follows as:

Step 1) every two subjects are compared with each other.
Step 2) two subjects chosen in step 1 are compared in cell 1. If they are in the same set, the rank will be zero. Otherwise, an integer number between one and ten is assigned in respect to the close relationship between the sets.
Step 3) the ranks added in the above steps represent the close relationship between exam subjects.

6 Computational Results

Figure 3 illustrates a chromosome representing an examination timetabling as shown in Table 1. The final selected chromosome is a desirable timetable obtained by the problem-solving process of genetic algorithms. In order to investigate the above process, exam subjects are considered according to the Table 2. First, a closeness relationship between exam subjects must be defined, as shown in Figure 4. Sets of exam subjects are established by using an activity-on-arrow network. After using the proposed algorithm, the closeness relationships are obtained, as shown in Figure 5. The solving process is studied with attention to the process of the general genetic algorithms.

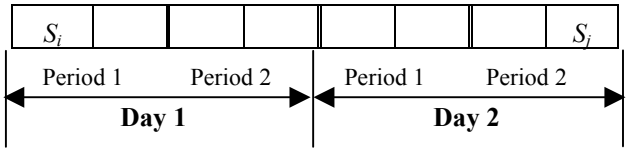

Fig. 3. A chromosome of a studied problem

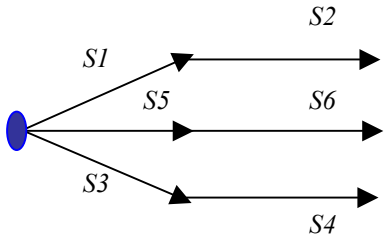

Fig. 4. Activity of arrow for subjects

	S1	*S2*	*S3*	*S4*	*S5*	*S6*	*S7*	*S8*
S1	0	10	1	15	1	16	2	2
S2		0	10	1	20	1	2	2
S3			0	10	1	16	2	2
S4				0	20	1	12	12
S5					0	1	2	2
S6						0	8	8
S7							0	6
S8								0

Fig. 5. Matrix of closeness relationship between exam subjects

The first step for solving the problem is to translate the decision variables of the given problem to the useful codes for genetic algorithms. With attention to the dedicated codes, an initial population of four chromosomes is generated, as shown in Figure 6. The fitness value for each chromosome is calculated according to the product of weight and distance between two exam subjects. Maximum selected value is on the basis of privilege of each chromosome. It means that the values of the results advance to the maximum sum of the product of weight and distance between two exam subjects. New chromosomes called offspring are generated by the use of the genetic operators. One of the most popular genetic operators is crossover rolled as a backbone of genetic algorithms. In the process of crossover, two randomly selected parents are combined and mated. For each parent by the length of L, the gene value of one parent is intervallic exchanged by the gene value of another parent in the same location, as shown in Figure 7. At the end of this process, the feasible and valid chromosome (genotype) is illustrated in Figure 8. This genotype is then translated to the desirable solution (phenotype), as summarized in Table 3. Figure 9 shows a comparison of two average fitness vales of consequence populations.

In this paper, the results generated by GAs are compared with another method. This method is based on the minimal spanning tree of the graph obtained from the problem studied in this paper [20]. The comparison of these two methods is

graphically illustrated in Figure 9. Table 4 also shows the final solution obtained by the graph theory.

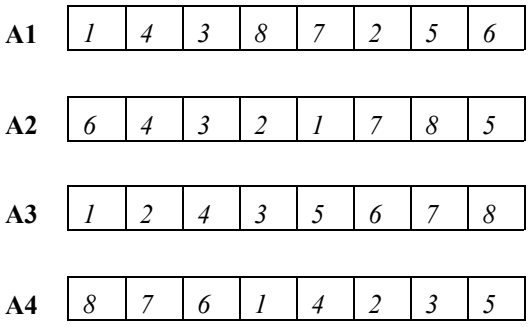

Fig. 6. An initial population

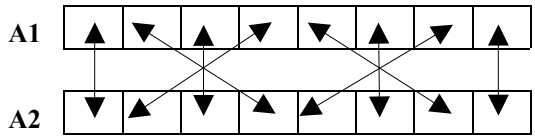

Fig. 7. Crossover operator

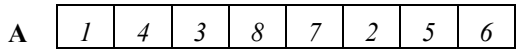

Fig. 8. Final chromosome

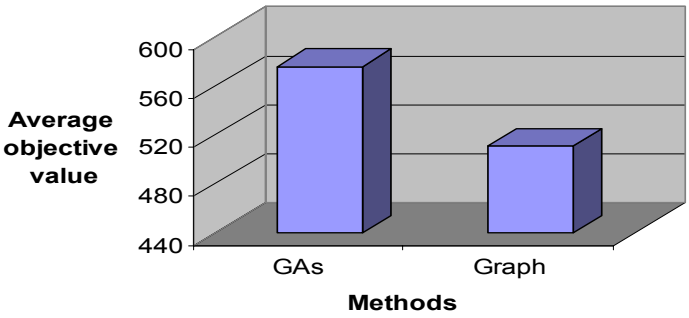

Fig. 9. Comparison of average objective values

Table 2. A typical exam subjects presented

Item	Exam Subject	Subject code	Number of subjects set
1	*S1*	1	1
2	*S2*	2	2
3	*S3*	3	1
4	*S4*	4	2
5	*S5*	5	1
6	*S6*	6	2
7	*S7*	7	1
8	*S8*	8	1

Table 3. An examination timetabling by genetic algorithms

	Period 1		Period 2	
Day1	S_1	S_4	S_3	S_8
Day2	S_7	S_2	S_5	S_6

Table 4. An examination timetabling by the graph theory

	Period 1		Period 2	
Day1	S_1	S_4	S_2	S_5
Day2	S_7	S_8	S_3	S_6

7 Conclusion

In this paper, a mathematical programming of examination timetabling has been presented. In order to solve such model, a special design of the genetic algorithm's structure and genetic operators has been introduced. Then an examination timetabling scheduling including all constraints has been generated and reported. The efficiency of the proposed algorithm has been studied for a number of test problems with different sizes. The model presented in this paper has not had any limitations in terms of subjects and courses.

References

1. Carter, M. W.: A Survey of Practical Applications of Examination Timetabling Algorithms. Vol. 34. Operations Research, (1986) 193-202
2. Carter, M. W., Laport, G.: Recent Developments in Practical Examination Timetabling. Proc., The First Int. Conf. on The Practice and Theory of Automated Timetabling, Edinburgh, U.K. (1995) 3-21
3. Bardadym, V. A.: Computer-Aided School and University Timetabling: The new wave. Proc., The First Int. Conf. on The Practice and Theory of Automated Timetabling, Edinburgh, U.K. (1995) 22-45
4. De Werra, D.: Some Combinatorial Models for Course Scheduling. Proc., The First Int. Conf. on The Practice and Theory of Automated Timetabling, Edinburgh, U.K. (1995) 296-308
5. Leong, T. Y., Yeong, W. T.: Examination Scheduling: A Quadratic Assignment Perspective. Proc. Int. Conf. on Optimization: Techniques and Applications, Singapore (1987) 550-558
6. Lotfi, V., Cerveny, R.: A Final-Exam-Scheduling Package. Vol. 42. J. Operations Research, (1991) 205-216
7. Gaspero, D., Schaerf, A.: Tabu Search Techniques for Examination Timetabling. Proc. The 3th int. conf. On The Practice and Theory of Automated Timetabling, Constance, Germany (2000)
8. Erben, W., Keppler, J.: A Genetic Algorithm Solving a Weekly Course-Timetabling Problem. Proc., The First Int. Conf. on The Practice and Theory of Automated Timetabling, Edinburgh, U.K. (1995) 198-211
9. Edmund, K. B., Elliman, D. G., Weare, R. F.: A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. Proc., The 6th Int. Conf. On Genetic Algorithms, edited by Eshelman, L. J., Laboratories, Morgan Kaufmann Publishers, Inc. (1995) 605-610
10. Junginger, W.: Course Scheduling by Genetic Algorithms in Evolutionary Algorithms in Management Applications. Edited by Biethahn, J., Nissen, V., Springer-Verlag (1995) 354-368
11. Reeves, C. R. (editor): Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, Osney Mead, Oxford (1993)
12. White, G. M., Xie, B. S.: Examination Timetables and Tabu Search with Longer Term Memory. Proc. The 3th int. conf. On The Practice and Theory of Automated Timetabling, Constance, Germany (2000)
13. Sheibani, K., Tavakkoli-Moghaddam, R.: The Use of an Evolutionary Computing For Solving Course Scheduling Problems. Proc., The First Int. Conf. on The Autonomous Intelligent Systems, Geelong, Australia (2002)
14. Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing, Co. (1989)
15. Holland, J. H.: Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, 2nd ed., Cambridge (1992)
16. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Design. John Wiley & Sons, Inc. (1997)
17. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
18. Tavakkoli-Moghaddam, R.: Programming Models for Solving Unequal-Sized Facilities Layout Problems - A Generic Search Method. Vol. 12, No. 4,International Journal of Engineering (1999) 233-245

19. Ladd, S. R.: Genetic Algorithms in C++, MIS Press (1996)
20. Sheibani, K., Tavakkoli-Moghaddam, R.: A Graph-Theoretic Approach for Solving Course Timetabling. Proc., The 6th Student Conf. on Industrial Engineering, Tehran, Iran (1999)

Recolour, Shake and Kick: a recipe for the Examination Timetabling Problem

Luca Di Gaspero

Dipartimento di Matematica e Informatica
Università di Udine
via delle Scienze 206, I-33100, Udine, Italy
digasper@dimi.uniud.it

1 The Examination Timetabling problem

The Examination Timetabling problem (see, e.g. [1]) is a combinatorial optimisation problem that commonly arises in universities and other academic institutions. Basically one has to schedule a certain number of exam proofs in a given number of time periods so that no student is involved in more than one examination at a time.

Usually, there also are some other constraints to take into account depending on the examination rules of the specific institution. However, in order to compare our results with previous literature, we decided to adopt a common formulation of the problem proposed by Carter *et al.* [2], for which a set of benchmark instances is available. In brief, the aim of this formulation is to minimise students' workload by penalising the fact that a student has to take two exams in a short time (the so-called *second-* and *higher-order* conflicts).

It is easy to recognise that this formulation of the problem underlies a graph colouring encoding. Each node of the graph represents an examination and there is a weighted edge between two nodes ex_1 and ex_2 if one or more student are enrolled in both examinations. The problem is then to assign a period to each examination in such a way that adjacent (i.e., conflicting) exams are not assigned the same period. Furthermore, the additional constraints can be included in a *objective function* f which evaluates the assignment according to the students' workload. To this aim, as in [2], we employ a proximity cost w_i whenever a student has to attend two examinations ex_1 and ex_2 scheduled within i periods. The costs decrease from 16 to 1 as follows: $w_1 = 16, w_2 = 8, w_3 = 4, w_4 = 2, w_5 = 1$ and are multiplied by the number of students involved in both examinations. Finally, the overall sum of these contributions is divided by the total number of students in order to obtain a measure that is independent from the problem size.

In a previous work, we presented the development of a family of Tabu Search algorithms for the solution of the Examination Timetabling problem [3] based on a simple move. In this paper, we describe an ongoing work which improves those results, by applying a novel multi-neighbuorhood local search algorithm based on a combination of two kinds of moves.

2 Multi-Neighbourhood Search

A recent trend in local search regards the exploitation of several different neighbourhood functions so as to increase the ability of the algorithm to navigate the search space (see, e.g., [5,7]). For practical problems, in fact, there is more than one neighbourhood structure that is sufficiently natural to deserve systematic investigation.

The main reason for considering combinations of different neighbourhoods is related to the diversification of search needed to escape from local minima. In fact a solution that is a local minimum for a given definition, is not necessarily a local minimum for another one, and thus an algorithm that uses both has more chances to move toward better solutions.

In this work, we combine different neighbourhoods using the *neighbourhood composition* and the *token-ring search*. The former combination considers as atomic moves, chains of moves belonging to different neighbourhoods. The token-ring search, instead, combines a set of algorithms based on different neighbourhood functions. This strategy makes circularly a run of each algorithm, always starting from the best solution found by the previous one.

In addition to the usual local search algorithms that can be built upon the given neighbourhood definitions, we consider also a a form of perturbation, that we call *kick*, in terms of neighbourhood compositions. A *kicker* is an algorithm that makes just one single move, and uses a neighbourhood composition of a relatively long length.

A kicker can perform either a random kick, i.e. a random sequence of moves (which approximately corresponds to the concept of random walk [6]), or a best kick, which means an exhaustive exploration of the composite neighbourhood looking for the best sequence.

In general, searching for an arbitrary long sequence of move is computationally infeasible. To reduce this cost, the kickers can be defined to explore only kicks composed by certain combinations of moves. In details, a kicker searches for a chain of moves that are related to each other, that we call *synergic* moves.

3 Recolour, Shake and Kick

In order to solve the Examination Timetabling problem by local search, first we have to define the search space and the cost function. Our search space is composed of all the colouring of the graph, including the infeasible ones. The cost function is a weighted sum of the number of hard violations plus the objective function f which accounts for second- and higher-order conflicts. Hard violations are assigned weight 1000.

We define two neighbourhoods for this problem. The first one, called recolour, considers two states as neighbours if they differ for the period assigned to a single course. At each state, only the courses involved in at least one violation (either hard or soft) are considered, in order to focus only on the exams which contribute to the cost function. This neighbourhood has been successfully applied to this

problem in combination with Tabu Search with an adaptive penalty mechanism (see [3] for more details).

The second neighbourhood, called shake, is defined in terms of *macro* moves which exchange the periods of two whole groups of courses at once. In detail, if we denote with G_i and G_j the set of courses which are assigned to period i and j respectively, this move swaps the two sets G_i and G_j thus assigning the period j to all the courses which were assigned to period i and vice versa.

The intuition behind this neighbourhood is that a move of this kind *shakes* the current solution searching for the best permutation of colours. In fact, the value of the objective function depends only on the distance between the periods, therefore this move can spread more evenly the workload of the students. Moreover, since the shake move changes several features of the current solution, it gives a new good starting point for the local search algorithms based on the recolour move.

Given these neighbourhoods, we define two Tabu Search (TS) algorithms equipped with the recolour and the shake move respectively, and a kicker that performs chains of recolour moves. Our solver implements a token-ring strategy which makes a run of the two TS algorithms in sequence until no improvement in the cost function is found anymore, and at the end it tries to optimise further the solution by means of best kicks of length 2. We call this solver *Recolour, Shake and Kick*.

The contribution of the three algorithms in the search is different. The recolour TS aims at wiping all the first-order conflicts and optimising the objective function, while the shake TS is meant to perturb the current solution in order to give new good starting point for the search. Finally, the kicker phase tries to obtain some additional improvement.

The described multi-phase strategy differs from the two-phase approach employed by Thompson and Dowsland [8] in that we apply it repeatedly until no improvement can be found by any of the algorithms. Conversely, in [8] the authors apply the two phases only once. Furthermore, the idea of employing the shake move is new and, up to our knowledge, it has not been used previously.

4 Preliminary results

The code for the solver has been written in C++ exploiting the EasyLocal++ framework [4], which generates automatically the code for exploration of complex neighbourhoods starting from the code for the basic ones.

At present, we have tested the proposed strategy on seven of the twelve benchmark instances proposed by Carter and co-workers [2]. In this experiments, we ran our solver for a reasonable amount of time (300 secs) and we recorded the best solution found up to that time. All the experiments were performed on an AMD Athlon 650MHz PC running Linux, equipped with 384Mb of memory.

We compare our solver with previous results obtained by the Tabu Search algorithm with the recolour move only [3], and with the results obained by a set of constructive solvers developed by Carter *et al.* [2].

Data set	exams	periods	R, S & K results		TS results		Carter's results		Impr.
			best	average	best	average	min	max	
CAR-S-91	682	35	**5.68**	5.79	6.2	6.5	7.1	7.9	-8.45%
EAR-F-83	189	24	39.36	43.92	45.7	46.7	**36.4**	46.5	-13.87%
HEC-S-92	80	18	10.91	11.41	12.4	12.6	**10.8**	15.9	-12.02%
LSE-F-91	381	18	12.55	12.95	15.5	15.9	**10.5**	13.1	-19.07%
STA-F-91	138	13	**157.43**	157.72	160.8	166.8	161.5	165.7	-2.10%
UTA-S-92	638	35	4.12	4.31	4.2	4.5	**3.5**	4.5	-1.90%
YOR-F-83	180	21	**39.68**	40.57	41	42.1	41.7	49.9	-3.21%

Table 1. Comparison among Recolour, Shake and Kick, plain Tabu Search [3] and Carter *et al.* solvers [2]

Table 1 summarises the performance of our Recolour, Search and Kick solver with respect to the plain TS and Carter's results. The best cost values found on each instance are highlighted in bold face.

The table shows that the multi-neighbourhood local search algorithm outperforms the plain Tabu Search algorithms on all the instances (in the last column it is reported the percentage of improvement over the TS algorithm). Concerning the comparison with Carter's results, instead, we obtain best results in three out of seven instances.

We consider these preliminary results quite promising, and we plan to extend this work by a thorough investigation of the proposed strategy on the whole set of benchmark instances and with respect to different formulations of the problem.

References

1. M. W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193–202, 1986.
2. M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Op. Research Society*, 74:373–383, 1996.
3. Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetabling. In E. Burke and W. Erben, editors, *Proc. of the 3rd Int. Conf. on the Practice and Theory of Automated Timetabling*, number 2079 in Lecture Notes in Computer Science, pages 104–117. Springer-Verlag, Berlin-Heidelberg, 2001.
4. Luca Di Gaspero and Andrea Schaerf. Writing local search algorithms using EASYLOCAL++. In Stefan Voß and David L. Woodruff, editors, *Optimization Software Class Libraries*, OR/CS. Kluwer Academic Publishers, Boston, 2001.
5. P. Hansen and N. Mladenović. An introduction to variable neighbourhood search. In S. Voß, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer Academic Publishers, 1999.
6. Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, pages 337–343, 1994.
7. Thomas Stützle. Applying iterated local search to the permutation flow shop problem. Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt, 1998.
8. J. Thomson and K. Dowsland. General cooling schedules for simulated annealing-based timetabling system. In *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling*, pages 345–363, 1995.

A Case Based Heuristic Selection investigation of Hill Climbing, Simulated Annealing and Tabu Search for Exam Timetabling Problems.

E. K. Burke, A. J. Eckersley, B. McCollum*, S. Petrovic, R. Qu,

Automated Scheduling, Optimisation and Planning Research Group,
Department of Computer Science & Information Technology,
University of Nottingham.

1 Abstract

The examination timetabling problem has been studied in many different forms and using a variety of techniques over the years [1], [3], [4], [6]. In its simplest form, it is required that a set of exams are scheduled into a set of time periods in such a way that no two exams with students in common are scheduled at the same time. In practice there are a wide range of other constraints which must be satisfied or minimized in order to achieve what is thought of as a "good" timetable. These constraints are discussed and analysed in some detail in [2] and [5]. Among the many techniques applied to exam timetabling problems, heuristic methods such as Hill Climbing, Tabu Search and Simulated Annealing are some of the simplest and easiest to implement [4] whilst still providing good results. Many Researchers have developed the simple idea behind these algorithms further to produce more complex and effective algorithms. A number of such Tabu Search techniques are discussed and compared in [6] also with Simulated Annealing and Hill Climbing (i.e. Memetic Algorithms).

The aim of this paper will be to develop a Case based reasoning heuristic and meta-heuristic selection system for examination timetabling problems which will include a wide variety of different algorithms. The main motivation for this is the goal of developing exam timetabling systems that are fundamentally more general. We are starting with a very simple set of algorithms and problems which will be progressively built on and modified with the aim of producing a much more complex system capable of providing competitive results when supplied with any exam timetabling problem. The ultimate goal is to develop a system which can obtain results which are at least comparative to the results

*Queen's University of Belfast

obtained by the range of problem specific approaches, but on a wide range of exam timetabling problems.

Case based reasoning has been applied to many different areas of research very successfully over the last 10-15 years, but as yet, very little research has been done on applying it to timetabling problems. The basic idea behind case based reasoning (explained in more detail in [7]) is to use past experience to solve a new problem which is in some way similar to one which has previously been encountered. A real life example of this would be when a doctor diagnoses a patient: symptoms are examined and matched with those seen previously in order to give an accurate diagnosis. The case base contains indexed information on a number of previous situations which can be matched up with any new problem and the solution of the old problem can be adapted to solve the new problem.

When applied to timetabling, this can be done either directly working on the problem or working at the level of a hyper-heuristic (i.e. a heuristic to choose a heuristic). Burke and Petrovic [4] give an overview of how Case based reasoning can be applied directly to timetabling problems and briefly discuss the role of hyper-heuristics. The project which we are working on uses Case based reasoning at a higher level to select the heuristic to be used to solve a new problem. This will be done by creating a case base containing cases consisting of problem data, algorithms used to successfully solve these problems and the results they produced. When given a new problem, the system will search the case base to match this problem with a previous problem using some measure of similarity. Once a match is found, the algorithm(s) used to produce results for the old problem will be retrieved and used after possible adaptation to solve the new problem. The theory behind this is that "similar" problems will be solved equally well by the same type of algorithm.

To form an initial case base, three simple heuristic methods: Hill Climbing, Simulated Annealing and Tabu Search have been implemented in order to produce results from a set of test cases. Initially the results produced by these algorithms will not be very competitive in comparison to specially produced algorithms aimed at solving a specific problem, but the idea is to build our case base from the very simplest cases. In doing this we will be performing a number of experiments with different parameters and different data sets in order to examine the effect that small changes to an algorithm will have on the results it produces. Once we have an initial case base, we will begin to develop and include more complex algorithms and data sets with more complicated constraints so that eventually we should have a wide variety of both problems and algorithms to choose from when presented with a new problem to solve.

This throws up a number of research issues which we will be covering in the near future in order to create a successful system. Most notably, the concept of "similarity" between two exam timetabling problems will need to be considered in some detail so that the underlying theory of similar problems being solved by the same heuristic with equally good results will hold. This will depend a lot on how different methods respond to particular parts of a timetabling problem. In this case, the tests we are performing using simple algorithms should give us

an idea of how much variance there is in solution quality as data sets become less similar.

Also, the issue of how to store and index the three key parts of each case (problem data, algorithm and result) needs to be considered so that the case base can be efficiently searched and problems retrieved quickly. Amongst other issues to consider will also be how or if the retrieved algorithm should be adapted and applied to the new problem and whether numerous different heuristics should be applied to the same problem using the case base to decide which heuristic to apply when.

References

[1] E. Burke and M. Carter (Eds). *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*

[2] E. Burke, D. Elliman, P. Ford, R. Weare. *Exam Timetabling in British Universities - A Survey*, in *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*

[3] E. Burke and W. Erben (Eds). *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*

[4] E. Burke and S. Petrovic. *Recent Research Directions in Automated Timetabling*. To appear in EJOR 2002.

[5] M. Carter, G. Laporte. *Recent Developments in Practical Examination Timetabling*, in *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*

[6] L. Di Gaspero and A. Schaerf. *Tabu Search Techniques for Examination Timetabling*, in [3], p104-117

[7] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc. San Mateo.

A Multiobjective Optimisation Technique for Exam Timetabling Problems Based on the Defined Trajectory

Sanja Petrovic, Yuri Bykov

University of Nottingham, School of Computer Science & IT, Wollaton Road
Nottingham NG8 1BB, UK
{sxp, yxb}@cs.nott.ac.uk

University examination timetabling problems comprise arranging exams in the given number of timeslots through the examination sessions. The primary objective of this process is avoiding of students' clashes (i.e. no one student should take two exams simultaneously). This requirement is generally considered as a hard constraint and should be obligatory satisfied in a feasible timetable. However, the number of other restrictions and regulations, which depend on a particular institution, should be also taken into account while solving examination timetabling problems. Some of them are also considered as hard constraints, while other constraints are soft, i.e. they cannot be completely satisfied, and therefore their violations should be minimised. The soft constraints usually have different importance for the timetable officer (decision maker). They are generally incompatible and often conflicting with each other.

Timetabling problems with various soft constraints typically have no single supreme solution. Formally, a number of solutions can be considered as optimal. The decision maker's goal is to obtain a desirable one in accordance to his/her preferences. This can be done by grouping the soft constraints into several objectives, where objectives measure the number of violations of the soft constraints, and applying an appropriate multiobjective optimisation technique. Those techniques show different performance and require different strategies for their application.

The multiobjective techniques are traditionally divided into two groups. The first group is known as "Decide-then-Search" (a priori). In these methods the decision maker specifies his/her preferences regarding the solution before starting a search. Generally, such approach involves the aggregation of problem's objectives into one objective function in order to apply some single-objective searching algorithm. The examples of application of this approach to examination timetabling can be found in [1], [2]. The second group of the approaches is called "Search-then-Decide" (a posteriori). Those methods produce a number of solutions among which the decision maker chooses the preferable one. To our knowledge, there are no publications about the use of these methods for exam timetabling. However several authors (ex. [4], [5]) have applied a-posteriori approach to the class/teacher timetabling, i.e. to the problem, which is similar to examination timetabling.

In this paper we introduce a new a priori multiobjective algorithm, which drives a search procedure through a predefined trajectory. The proposed algorithm is based on the local search algorithm, which we named *degraded ceiling algorithm* [3]. In the degraded ceiling algorithm a new candidate solution is accepted if its objective function is either better than a current one or does not exceed a value of the ceiling. The value of the ceiling reduces gradually throughout the search taking into consideration a predefined searching time of the algorithm. This algorithm was initially tested with a weighted sum as its objective function. It was shown that a longer search usually yielded a better final result. The algorithm can produce the results which outperform other techniques applied to exam timetabling problems at the price of relatively longer search period (time is usually not an issue in the exam timetabling problems).

In the developed multiobjective approach to degraded ceiling algorithm, in order to direct the search to a predefined trajectory, the weights of the objectives are changed dynamically, one at a time.

The proposed algorithm was experimentally checked on the real-world large scale examination timetabling data. The experiments confirmed the ability of the algorithm to drive the search close to the chosen trajectory. The developed approach is compared with other multiobjective optimisation techniques.

References

[1] E. K. Burke J. P. Newall, "A Multi-Stage Evolutionary Algorithm for the Timetabling Problem". *The IEEE Transactions of Evolutionary Computation* 3.1, 1999, 63-74.

[2] E. K. Burke, Y. Bykov, S. Petrovic. "A Multicriteria Approach to Examination Timetabling". E. Burke, W. Erben, eds. *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000). Lecture Notes in Computer Science* 2079. Springer-Verlag, Berlin Heidelberg, New York, 2001. 118-131.

[3] E. K. Burke, Y. Bykov, J. P. Newall, S. Petrovic "A Time-Predefined Local Search Approach to Exam Timetabling Problems", Computer Science Technical Report No. NOTTCS-TR-2001-6, University of Nottingham, 2001.

[4] M. P. Carrasco, M. V. Pato, "A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem", E. Burke, W. Erben, eds. *The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000). Lecture Notes in Computer Science* 2079. Springer-Verlag, Berlin Heidelberg, New York, 2001. 3-17.

[5] M. Tanaka, S. Adachi, "Request-based timetabling by Genetic Algorithm with Tabu Search", The Third International Workshop on Frontiers in Evolutionary Algorithms, 999-1002, 2000.

Decision Support without magic or mind reading for assigning magistrates to sessions of the Amsterdam Criminal Court

Jan A.M. Schreuder[1]

[1] University of Twente – TWRC-A205, Postbus 217, 7500 AE Enschede, Netherlands
j.a.m.schreuder@math.utwente.nl, http://www.math.utwente.nl/schreudr

Abstract

In the criminal court (Arrondissements rechtbank, sector strafrecht) of Amsterdam the assignment of magistrates to sessions needed to handle the cases presented to this court, has become a problem last years. A well known cause is the increase in the number of cases and their complexity which can hardly be followed by installing more magistrates, but certainly also the appearance of so called megasessions. 'Normal' sessions can be planned a number a day, but megasessions need several days and it is hardly to predict how many days.

The assignment takes a period of 4 weeks at a time. Each week about 60 magistrates and 80 sessions should be scheduled. The management of the Criminal Court asked the University of Twente to develop a Decision Support System for their scheduler. The objectives were to develop a system with which the scheduler could make the assignments of at least the same quality in a shorter timeperiod and more reliable easy to handle datastreams. In cooperation with WisseQ, a software consultancy and in order to reach for an optimal mix of support and computing power we used EXCEL for the more administration-data-representation-report parts and FORTRAN for the combinatorial assignment parts.

At first sight, this seems to be an easy to solve problem when it is restricted to persons who are qualified to carry out certain tasks and are always available. Such a problem is known as Employee Timetabling or scheduling and is solvable with approaches for the General Assignment Problem. Apart from the usual capacity and time constraints which could be rather nasty in itself, a typical complicating factor is that there are sessions which need three magistrates in stead of one: each of them representing a different function. A further complication is that magistrates are member of one of three possible groups and that the members of a team of three magistrates assigned to one session should belong to the same group. All in all, the magistrates ar characterized by function, experience, group, number of working hours, availabilities and are specialist in specific sessions. Such a problem can be formulated as an integer linear programming problem, but can due to overdetermind constraints only be solved in practice by an heuristic approach. There are two main steps of which the first one searches for each individual magistrate for his/her possible combinations of assignments to sessions per week taking into account as

many constraints as possible. The second step constructs the actual teams of magistrates.

One of the basic problems while encountered implementing the support system was that the management thinks that Fortran can perform magic (like Harry Potter) and the scheduler handles the package as if a computer can read his mind. How to cope with those fairy tales and directing them into a more realistic way of thinking will be the subject of this presentation. A specific example is how to consider an important management issue like percentage of judges against sessions: with or without breaking the rules, with or without adding/omitting sessions, with or without replacements for judges outside the groups, etc. We had to adjust the Fortran and EXCEL parts taking this way of thinking into account. Of course, changing to another package like Visual Objects was also a possibility, but there are costs! A very basic discussion about the rules and exceptions to take into account with the use of representative tests is absolutely necessary.

Keywords

Employee Timetabling, Implementations, Heuristic Search, Integer Programming.

References

1. Lagerholm, Martin, Peterson, Carsten and Söderberg, Bo: Airline crew scheduling using Potts mean field techniques. EUROPEAN JOURNAL OF OPERATIONAL RESEARCH 120 (2000) 81-96.
2. Schreuder, Jan A.M.: Practical advances in using models for staff timetabling. Presentations University of Twente. Enschede (1998).
3. Wilson, J.M.: The Scheduling of Magistrates to Courts. J.OPL.RES.SOC., Vol. 32 (1981) 121-124.

The cost of flexibility in vehicle routing and scheduling

Wout Dullaert[1], Bjarne Johannessen[2], Olli Bräysy[2], and Tore Dahl[2]

[1] Ufsia-Ruca Faculty of Applied Economics, University of Antwerp
Prinsstraat 13, 2000 Antwerp, Belgium
wout.dullaert@ua.ac.be
[2] SINTEF Applied Mathematics, Department of Optimisation,
P.O. Box 124 Blindern, N-0314 Oslo, Norway
{bjarne.johannessen, olli.braysy, Tore.Dahl}@math.sintef.no

1 Introduction

Recent research [1, 2] demonstrates the importance of scheduling flexibility for less-than-truckload (LTL) and full-truckload (FTL) routing and scheduling. Scheduling flexibility refers to the flexibility that customers give to the dispatcher to design routes. The more flexibility they give, the more cost efficient the routes the dispatcher can design. In practise, the cost-driving effect of (a lack of) scheduling flexibility has been recognized for a longer time. Both LTL and FTL haulage companies charge different rates depending on the scheduling flexibility of a load, often without having good cost estimates.

Correct estimates of the incremental cost of customers or customer types are essential for pricing routing services. The additional cost of routing a customer acts as a price-floor: the freight rate should at least cover the additional cost of servicing that customer. This information remains valuable, even if prices are not cost-based. If prices are aimed at capturing customers' willingness to pay (e.g. perceived value pricing) or if prices are based on competitors' price levels (e.g. going-rate pricing), the incremental cost of servicing a customer is vital for determining the contribution or profitability of servicing each customer.

In this paper, several methods for determining the incremental cost of a customer and estimating the cost of scheduling flexibility are examined.

2 Approximations for incremental costs in routing

To determine the exact incremental cost of routing a customer two VRPTWs have to be solved to optimality: once with and once without the customer involved. The difference yields the additional cost generated by that customer. Exact solution algorithms for routing and scheduling are only capable of solving relatively small problem instances [3]. For real-life applications, one must resort to heuristic procedures to approximate the optimal solution (without quality guarantee!) within a reasonable amount of computation time. For the determination of a customer's incremental cost, these computation time considerations are even more an issue.

Re-solving an entire VRPTW for each customer is a computationally expensive procedure to determine the incremental routing costs. For a small VRPTW of 100 customers, 100 VRPTWs of 99 customers have to be solved to determine the incremental cost of each customer. In real-life applications, the number of customers to be routed daily is a lot larger. For strategic decisions, company activity often needs to be simulated of a large number of days. The number of VRPTW that needs to be solved to obtain the incremental cost of different categories of customers soon becomes unmanageable, even for the fast, high-quality metaheuristics available today [4, 5]. Therefore more efficient ways of estimating the incremental routing cost of customers are suggested in this paper.

In this paper, we evaluate different approaches for determining the incremental cost. Full and partial re-optimization approaches are compared to naive re-optimization procedures that simply remove a customer from the route without changing the sequence of the other customers in the route.

Seven local search operators (Insert, Remove, Relocate, Cross, Exchange, 2-opt, Tour-deplete) are combined in a Variable Neighbourhood Search (VNS) framework [6, 7] and embedded in a Guided Local Search [8, 9] metaheuristic. *Insert* tries to add unserviced orders. *Remove* takes an order out of the current plan. *Relocate* moves an order from one tour to another, or, to a different position within the current tour. *Cross* exchanges the tails of two tours. In a more general fashion, *Exchange* swaps two consecutive segments of orders between tours. The *Tour-deplete* operator tries to reduce the number of tours. Moreover, a strategy similar to Large Neighbourhood Search [10, 11] is used when the VNS has not shown any improvement for some time.

Partial re-optimization procedures, reschedule only a part of the original VRPTW. Clearly, the route containing the customer under consideration needs to be rescheduled as the customer is temporarily removed. But also the customers of a number of adjacent routes are should be included in the analysis. Once the number of routes to be rescheduled is determined, local search heuristics can be used to re-optimize the routes. The re-optimization procedure is the same as for the full re-optimization process. The cost determining strategies are evaluated on real-life data from DFDS Tollpost-Globe, a large Norwegian LTL distribution company. The evaluation is performed using the SPIDER VRP Solver (http://www.oslo.sintef.no/spider) developed by SINTEF.

3 Analysis and conclusions

Knowing the incremental cost of its customers, allows a transport manager to evaluate their contribution margins. This information can be valuable for pricing decisions for individual customers. It is, however, more interesting to process the incremental cost data to reveal the cost driving factors of routing customers and determine their importance. We focus on three possible cost drivers: capacity, distance and driving time. If a customer has a high demand, he takes up a large proportion of a vehicle's capacity. This can prevent the vehicle from servicing other nearby customers and require them to be serviced in another or even a

new route. If a customer is located far from the depot, it can be harder to route efficiently. It's incremental cost can be expected to be high. Finally, a customer's service time window, can also affect the incremental cost. Tight time windows can force a dispatcher to service geographically clustered customers in separate routes or in a single route with high waiting times and/or additional distance to be travelled.

Regressing the incremental cost of each customer to its demand, this distance to the depot, and its scheduling flexibility measured by the width of this service time window, yields estimates of the importance of these cost driving factors. The information is crucial for designing or redesigning multi-dimensional tariffs for routing services. More specifics on the cost determining strategies and results on the data sets and the regression analysis will be presented at the conference.

Acknowledgement

This work was partially supported by the TOP program funded by the Research Council of Norway under contract 140689/420. The authors would also like to thank DFDS TollPost-Globe for providing real-life routing data and the company GreenTrip AS for access to SPIDER Designer, a route design product built on the SPIDER VRP solver.

References

1. Doerner, K., Gronalt, M., Hartl, R.F., Riemann, M.: Time constrained full truckload transportation: Optimizing fleet size and vehicle movements. Technical Report 2000/002, Department of Production and Operations Management, Institute of Management Science, University of Vienna (2000)
2. Dullaert, W.: Scheduling flexibility and the contribution maximizing vehicle routing problem with time windows. In: 9th World Conference on Transport Research, July 22-27, Seoul, South Korea (2001) 19
3. Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society **52** (2001) 928–936
4. Bräysy, O., Gendreau, M.: Metaheuristics for the vehicle routing problem with time windows. Technical Report STF42 A01025, SINTEF Applied Mathematics, Department of Optimisation, Norway (2001)
5. Bräysy, O., Hasle, G., Dullaert, W.: A fast local search algorithm for the vehicle routing problem with time windows. Paper presented at the Optimization Days, Montreal, Canada, May 6-8th (2002)
6. Mladenović, N., Hansen, P.: Variable neighbourhood decomposition search. Computers and Operations Research **24** (1997) 1097–1100
7. Hansen, P., Mladenovic, N.: Variable neighborhood search. In Pardalos, P., Resende, M., eds.: Handbook of Applied Optimization. Oxford University Press, New York (2002) 221–234
8. Voudouris, C.: Guided Local Search for Combinatorial Problems. PhD thesis, University of Essex, Colchester, UK (1997)

9. Voudouris, C., Tsang, E.: Guided local search. European Journal of Operations Research **113** (1998) 80–119
10. Shaw, P.: A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Glasgow, Scotland (1997)
11. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M., Puget, J.F., eds.: Principles and Practice of Constraint Programming - CP98, Lecture Notes in Computer Science. Springer-Verlag, New York (1998) 417–431