S. Pac. J. Nat. Sci., (2002), Vol. 20, pp. 64 – 67
Section B: Short Communication

A Heuristic Approach to Constraint Optimization in Timetabling

Atish Chand
Department of Mathematics and Computing Science, University of the South Pacific, Suva, FIJI
E-mail: atish.chand@usp.ac.fj

Abstract

Timetabling is a difficult (NP-complete) problem and belongs to a general class of problems known as scheduling. Due to a variety of constraints typical in different timetabling environments, it has been difficult to develop a generic solution for timetabling. This paper is an attempt to define a generic computational model for examination timetabling for predefined constraints found in the problem, and proposes a heuristic method of developing an acceptable solution. The declarative nature of the developed constraints language (based on the structured query language) is utilized to construct constraints and specify the timetabling problem as a constraint satisfaction problem. A university examination timetabling problem is used to illustrate and test the model.

1 General Introduction

This paper proposes a model for representing various constraints in a generic format that can be easily implemented in the differing timetable institutes. One reason why timetabling is a difficult problem is that there are many conflicting constraints that need to be resolved. Research done so far in this field has suggested various methods of building timetables that are context specific, as different timetabling institutes have many different types of constraints and data formats. This paper shows how the various constraints, in essence, are similar and that the seemingly different data have a generic format. A heuristic algorithm is applied to the model to generate a timetable. A prototype of the model was tested successfully with the University of the South Pacific's examination timetabling problem.

2 Introduction

Timetabling belongs to the general category of scheduling problems. A scheduling problem is one in which the task is to allocate resources to certain processes or objects in a certain sequence. Examples of general scheduling problems are making a duty roster, project schedules, bus scheduling, airflight schedules, etc including the well-known Travelling Salesman Problem (Lawler, E., *et al*., 1987). In this problem the task is to visit a number of cities in the shortest route possible.

If there are 40 cities in a location, then starting from the first city, there will be another 39 choices available for the second city, 38 for the third and so on. Thus the number of possible routes is 40 factorial which is approximately 8 * 10 ^47.

One could do a simple comparison of each route to determine which is the shortest. However the enormous number of possible routes makes this simple method very time-consuming and impractical. Assuming a computer could examine one billion routes in one second (that is if a 1GHz computer could examine one route per CPU cycle) then it would be possible to examine all routes in 8.15915E *10^38 seconds (9.44346E+33 days). Such a simple method is not viable when time is limited. Other strategies are needed.

Similarly, in examination timetabling, there are many choices available for possible exam time and venue allocations. Timetabling has been shown to be an NP-complete problem (Cooper, T. B., Kingston, J. H., 1995).

Wren (Wren, A., 1995) describes timetabling as "the allocation, subject to constraints, have given resources to objects being placed in space-time in such a way as to satisfy as nearly as possible a set of desirable objectives".

The timetable can be represented as a sequence of bins (of predefined fixed duration) arranged against time and each of the bins contains objects and resources (figure 1). A solution to a timetabling problem is such a sequence of bins that satisfy a given set of hard and soft constraints (where hard constraints are those constraints that have to be satisfied, and soft constraints are those which can be relaxed).

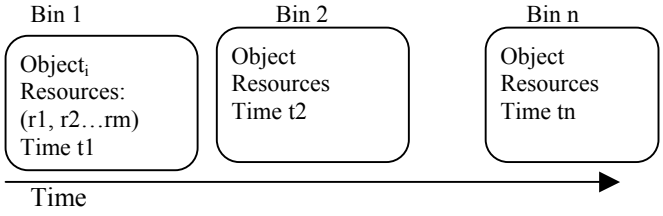

Figure 1 Grouping of Objects and Resources

In examination timetabling, a time slot and a venue need to be allocated to courses such that a given set of constraints are satisfied. The constraints can be of two types, *hard* and *soft*. A hard constraint is one that must be satisfied, for example, no student should be required to sit for two exams at the same time. A soft constraint is one that is preferably satisfied but not necessarily; for example, no student should be required to sit for two exams on the same day. The degree of *hardness* (or *softness*) of constraints define a hierarchical constraint structure that is exploited in a systematic search for solutions.

This paper shows how hard and soft constraints can be captured and easily specified in a constraint language and how heuristics can be applied to use the constraints in reducing the time needed to search the solution space for an acceptable solution. Comparisons will be made between results obtained from automated and manually built

timetables. The examination timetabling problem from the University of the South Pacific (Suva, Fiji) is studied and the computational model developed on it for the illustration of the proposed approach and testing.

3 Representation

The timetabling environment is modelled in the ERD model below.

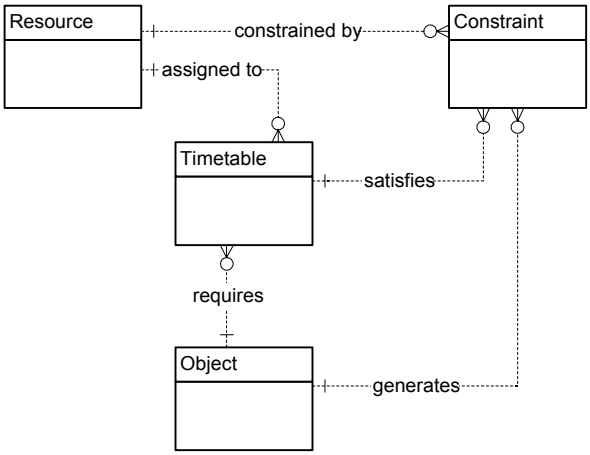

Figure 2 Model of the general timetabling environment

Time is treated as a resource. The above model also depicts the general class of scheduling problem when the entity Timetable is replaced with Schedule.

Various researchers (Reis 2000, Burke et al, 1997) have stated that one of the difficulties in researching timetabling systems is the wide variety of constraints and data format that exist in and differ from one environment to another.

The above model enables constraints that appear to be different on the surface, to be specified in the same format as discussed below.

The above model is further abstracted in the following model.

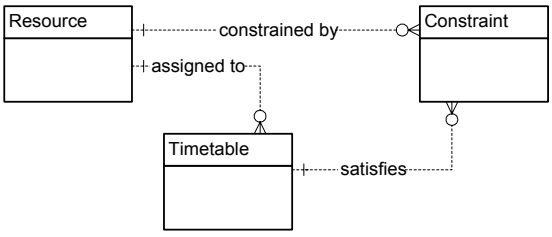

Figure 3 Higher abstracted model of the general timetabling environment

In this case the timetabling objects are treated as resources. Both these high levels of abstraction allow focus to remain on the core problem of timetabling.

The time slot (consisting of a day and an hour of the day), and the room are the resources that have to be allocated to the objects. The constraints are expressions specifying relationships between the resources.

Some constraints limit the exams to certain days. For example, "all exams have to be held on a week day" or "schedule exam EX100 on a Monday". Similarly other constraint may limit exams to certain hour of the day and certain rooms. The constraints can be categorised as *resource* constraints. In general these constraints is specified as

Resource_constraint (exam_list, resource_list, Weight),

where Resource_constraint is the name of the constraint, *exam_list* is a list of exams that are limited to the resources in the *resource_list* and *Weight* is a numeric measure of hard or soft the constraint is. A value of 0 means that it is soft and a value of 1 means it is hard. Alternatively a range of values could be used to denote various degree of constraint hardness. (This paper discusses hard constraints only, and thus weight can be ignored as all constraints are treated hard.)

The constraint "all exams have to be held within the ten day period" is specified as

Day_Constraint((EX1, EX2 , … EXN), (1,2,3,4,5,6,7,8,9,10))

where Day_Constraint is the name of the resources, (EX1,..., EXN) is the list of exams and (1,2,3,4,5) are day numbers given that the exams are scheduled over the weekdays of two weeks..

Similarly, "schedule exam EX100 on a Monday" is specified as

Day_Constraint((EX100), (1,6)) where the first and sixth days of the exam periods are Mondays.

Apart from the restriction to certain resources, exams also have to be scheduled such that clashes are minimised and exams are spread over the time period so that a student will have exams evenly spread. Exams for large classes should also be preferably held earlier during the examination week to allow more time for marking.

The requirements that exams should not clash or be spread over a period of time are actually cases of the same constraint as shown below. These constraints are instances of

Spread_Constraint*(exam_list, spread_list)* where Spread_Constraint is the name of the constraint and *spread_list* is the list of number of time slots by which the exams in the exam list must differ in their schedules.

The constraint "Exams EX1 and EX2 must be on different days" is specified as

DaySpread((EX1, EX2), (1,2,3,4,5,6,7,8,9)

where the difference of the day numbers of EX1 and EX2 schedules is equal to a number in the *spread_list*. The above expression states that the day difference has to be between 1 and 9 inclusive, which implies the exams cannot be held on the same day. Thus clash constraints can be stated as spread constraints.

The exam clash is a special case of the spread constraint, where the difference cannot be zero. If exams EX4 and EX5 are not to clash, the constraint is specified as

TimeSpread((EX4, EX5), (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19))

Given that the exam is being held over a ten day period with two exams per day, the slots would be numbered from 1 to 20. Thus the difference between the slot numbers of EX1 and EX2 can not be 0 but can be any value from 1 to 19, as stated above.

4 Algorithm

In the manual examination timetabling one of the major problems is dealing with clashes and finding clash free slots. Making a change requires that one has to undo previous

exam allocation and look for a new allocation for a new allocations. This creates a series of backtracks which are difficult to resolve.

Also, in the manual system, it has been observed that exams that have fewer constraints imposed upon them are easier to shift around in the timetable then more heavily constrained ones. Resources that are less constrained are also more easily available. The heuristic logic is that if most constrained exams were allocated the least constrained resources first, then the need to back track would be minimised. Based on this, the following heuristic algorithm was developed to avoid the problem of having to undo previously generated partial timetable allocations. The generated exam timetables are ordered in descending order according to how constrained they are with respect to the given constraints. Similarly, the resources are ordered in ascending order according to how constraining they are.

Algorithm

```
Input (exams, resources, constraints.)
/*create blank timetable */
Timetable = new(exams,resources, constraints)
OrderedExamList =OrderExams(Exams, Resources,
constraints, Descending)
OrderedResources = OrderResources(Resources,
Exams, Constraints, Ascending)
/* Populate Timetable */
 For all exams in OrderedExamList do
        Clash_list = CreateClash(CurrentExam,
        Exams, Resources, Constraints)
        Resources_Available =
        CreateResAvail(CurrentExam, Exams,
        Resources, Constraints)
        While  not(resource_found) and resource left
        do
                CurrentResource =
                pop(ResourcesAvailable)
                If NoClash(CurrentResources,
                Clashlist, Timetable)  then
                ResourceFound = true
                AssignToTT(CurrentResource
                CurrentExam, TimeTable)
                end if.
        Loop While
Next Exam.
Return Timetable
```

The above algorithm was implemented in MsAccess. The exams, resources and constraints were constructed as tables while queries were used to filter and order the resources and exams.

5 Results

The automated system was used to generate an examination timetable for semester one, 2000 and was compared with its manual generation. The automated system considered only the important hard clash constraints. It did not consider soft constraints such as lecturer requests for early exams or for certain days.

The automated timetable was analysed for the number of students that have exam clashes at the same time and on the same day. These figures were compared with the timetable that was prepared manually for the same semester.

The automated system produced an initial timetable that considered most of the soft constraints. The automated system did not consider cases where classes were too large to fit into one venue and had to be split into two rooms. It assumed that no class was larger than the maximum capacity of an available room.

The manual system used the initial timetable to make further adjustments that catered for the over-sized classes.
In another experiment, Day Spread and Time Spread constraint satisfaction were analysed for the automated and manual examination timetable for semester one 2000 and 2001. The results are shown in Table 1. The number of examination scripts is the total number of enrolments for each exam. For example, ten students taking 4 exams each results in a total of 40 exam scripts.

Table 1 Comparison of timetables produced by manual and automated means for semester 1 of the years 2000 and 2001 for the University of the South Pacific**.**

	Sem1 2000		Sem 1 2001	
	Manual	Automated	Manual Adjustment	Automated
Time clashes	548	0	48	0
Day clashes (including same time clashes)	1170	399	1001	954
Day clashes (excluding same time clashes)	622	399	953	954
Number of examination scripts	12242	12242	11810	11810

6 Discussion

The results show that there was a dramatic drop in the number of same time clashes. In the manually prepared timetable there were 622 instances of students having to sit for two or more exams on the same day while in the automated timetable there were 399 instances only.

An important consideration in examination timetabling is to allocate exams to morning sessions when students are more refreshed. The automated
timetable allocates 82.0 percent of exams to morning sessions while the manual timetable allocates 62.7 % percent.

It is also desirable that exams are scheduled as soon as possible to allow more time for marking the scripts while allowing for exams to be spread more evenly. In the automated timetable 70 % of the exams are scheduled in the first week compared to the 65% of the manual timetable.

The chart shows the above two trends clearly and also shows that the automated timetable has a more uniform and even distribution.

The number of same time clashes has reduced from 548 in year 2000 to 48 in 2001 representing 1041% improvement. For semester one 2001, the same day clashes (excluding same time clashes) have increased from 622 to 953 showing an increase in the number of clashes by 53%.

This is possibly due to the fact that in semester one 2000, the manual timetabling system was not able to determine that by keeping some same day clashes, it would reduce dramatically the same time clashes. If the figures for same day clashes including same time clashes are compared, the number of clashes drops from 1170 to 1001 (17%) for the manual system. The automated system had increased the number of same day clashes from 399 to 954 (58%). This is due to the effect of the additional soft constraints that were considered for the 2001 timetable but not for the 2000 timetable. In satisfying these soft constraints, the other soft constraint " avoid students having two exams on the same day" became harder to satisfy. As noted by Burke (Burke, E.K., Newall, J.P., Weare, R.F., 1995), it is harder to obtain increased performance from a more pertinent heuristics then those that are more generic.

7 Conclusion and Recommendations

Constraints for the University of the South Pacific examination timetabling can be adequately specified in a generic format and represented in a relational database. It has been shown that the difficulty of dealing with the wide variety of constraints and data formats could be overcome by representing the constraints and data in a generic format that can be easily implemented in common databases. It is suggested that different methods could then access that data and convert it to their own native form. For further work, a data standardiser can be build that translates data from one native form to another. In this way, current timetabling methods could be compared for performance.

Heuristics can be applied to the constraints, resources and other examination timetabling objects to build a better optimised timetable, utilizing the SQL engine of a database as the search mechanism for constraint satisfaction.

This paper discussed resolving the main hard constraint in examination timetabling. Further work will involve using a varying weight for different constraints to reduce constrain violation and to also consider other soft constraints such as preferred exam times, room allocations, and splitting large exams into smaller groups.

Methods such as meta-heuristics (heuristics that control other heuristics) and hyper heuristics (heuristics that select which heuristic to use) can be incorporated to further fine tune the timetabling process. These issues are being considered for improving the current system.

8 Acknowledgements

I wish to thank Dr Dharmendra Sharma for his guidance and valuable suggestions, and Mr. Timothy Tuivaga, of the Academic Office, University of the South Pacific, for providing test data.

References

Cooper, T. B., Kingston, J. H., 1995. The Complexity of Timetable Construction Problems. *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, 4-7.

Lawler, E., Lenstra, J. K., Kan, A., and Shmoys, D. B., 1987 *The Travelling Salesman Problem*.

Wren, A., 1995 Scheduling, Timetabling and Rostering - A Special Relationship? *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, *46-75*.

Burke, E.K., Newall, J.P., Weare, R.F., 1995 A Simple Heuristically Guide Search for The Timetable Problem. *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95).*

Resi, L. P., Oliveira, E., A Language for Specifying Complete Timetabling Problems. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '00).*

Burke, E.K., Kingston, J., and Pepper P. A., A Standard Data Format for Timetabling Instances *Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '97).*