# *UTTSExam*: A Campus-Wide University Exam-Timetabling System

**Andrew Lim, Juay-Chin Ang, Wee-Kit Ho, Wee-Chong Oon**

School of Computing, National University of Singapore
3 Science Drive 2
Singapore 117543, Singapore
{alim, angjc, howk, oonwc}@comp.nus.edu.sg

**Abstract**

*UTTSExam* is the exam-scheduling portion of the University Timetable Scheduler (*UTTS*) software, an automated university timetabling program developed in the National University of Singapore. It was successfully used to schedule the examination timetable for the first semester of the 2001/2002 academic year in NUS, a task involving 27,235 students taking 1,350 exams. The use of the software resulted in significant time savings in the scheduling of the timetable and a shortening of the examination period. This paper explains the development and design of *UTTSExam.*

## Introduction

The National University of Singapore (NUS)[1] introduced the modular academic course structure in 1993. This allowed students to choose the modules that they wished to study in order to complete their degree requirements. As a result of this added flexibility, the task of scheduling the examination timetables in NUS became much more complex, especially in view of the increasing number of cross-faculty modules (i.e. modules that can be taken by students from different faculties).

The task of scheduling examination timetables was previously done manually, an error-prone process that tends to take several weeks to complete. As a result, in 1999 the university sponsored the development of the University Timetable Scheduler (*UTTS*) software, an automated university timetable-scheduling program. When completed, the program is expected to automatically schedule both the course and examination timetables for all the faculties in the entire university that employ the modular academic course structure.

The course-scheduling portion of the program is currently still under development (Lim et al. 2000a). However, the exam-scheduling portion (Lim et al. 2000b), called *UTTSExam*, has reached the deployment stage and was used to generate the examination timetable for the first semester of the 2001/2002 academic year in NUS. This paper describes the steps involved in the development of *UTTSExam* from its conception to its completion, along with a breakdown of the inner workings of each portion of the software.

1 http://www.nus.edu.sg



## Problem Description

This examination-timetabling problem (ETTP) involves creating a schedule such that a set of examinations is allocated into venues with limited capacities within an examination period. ETTP is an instance of the Constraint Satisfaction Optimization Problem (CSOP) (Tsang 1993), which is a combination of two types of problems. The first is the Constraint Satisfaction Problem (CSP), which involves:

- A set of variables $X = \{x_1, \ldots, x_n\}$
- For each variable $x_i$, a finite set $D_i$ of possible values (its domain)
- A set of *critical* constraints restricting the values that sets of variables may take simultaneously

A solution to the CSP is an assignment of values to all variables such that every constraint is satisfied. General timetabling and scheduling problems, which are NP-Hard (Garey and Johnson 1979), may be modeled as a CSP. In particular, the ETTP can be modeled as a CSP by treating each examination as a variable, and the domain of each variable would be the available examination sessions. The ETTP has a further provision that the solution should take up as few sessions as possible. The two critical constraints for all ETTP are as follows:

- Two examinations that are taken by any particular student cannot be scheduled in the same session.
- The total number of candidates taking the papers scheduled in a particular venue must not exceed the venue's capacity.

Real-life ETTP problems also have another set of *non-critical* constraints, which may be violated while still retaining the feasibility of the solution. The *quality* of two feasible solutions may be measured by a weighted sum of its non-critical constraint violations. The best fulfillment

of these non-critical constraints, as measured by a weighted sum, is an optimization problem.

In the case of the first semester of the 2001/2002 academic year in NUS, this involves 27,235 students each taking one or more of 1,350 examinations (for a total of 100,599 student-examination tuples), to be scheduled in the following venues (Table 1):

| Alias | Venue | Capacity |
|---|---|---|
| STC | Suntec City Exhibition Hall | 1600 |
| GYM | Gymnasium | 312 |
| MPH1 | Multi-Purpose Sports Hall 1 | 750 |
| MPH2 | Multi-Purpose Sports Hall 2 | 850 |
| CH | Competition Hall | 396 |
| EH | Eusoff Hall | 175 |
| TH | Temasek Hall | 136 |
| LT8 | Lecture Theatre 8 | 117 |
| LT11 | Lecture Theatre 11 | 125 |
| LT13 | Lecture Theatre 13 | 81 |
| LT17 | Lecture Theatre 17 | 112 |

*Table 1: List of Examination Venues*

The examination period was from the 12th to 26th November 2001. There were 3 sessions a day (morning, afternoon and evening) for the 12 available days in this period, equaling 36 sessions in total. Suntec City is a commercial venue that was unavailable for the last 5 sessions of the examination period.

## Scheduling Strategy

In general, there are two approaches to the scheduling of university timetables, namely *centralized* or *de-centralized*. Both approaches have their own advantages and disadvantages.

Initially, we utilized the centralized approach, whereby a central authority uses the software to schedule the timetable for the entire university. In theory, this grants a global view of the problem domain, presenting all the information necessary to best create the timetable. Unfortunately, the sheer size of the problem proved to be too difficult, such that the scheduling program was unable to create a high-quality feasible timetable. Furthermore, co-ordination between the faculties and the central authority was difficult and error-prone.

Alternatively, the de-centralized approach lets each faculty schedule its own examination timetable using its own venue resources. However, this approach would rapidly become infeasible as more cross-faculty modules are introduced, since it works best if communication between faculties can be reduced to a minimum. There is also a problem in scheduling large examinations if the faculty does not have access to large venues.

We eventually adopted a hybrid centralized and de-centralized approach to scheduling (Ho, Lim and Oon 2002). Before student registration, the examinations' enrolment estimates were used to proportionally allocate a fixed number of *seats* for each faculty (called their *venue partition*). Each faculty would then schedule their exams according to their venue partitions, disregarding the actual venues and the effects of cross-faculty modules. These faculty timetables are then merged into a campus-wide tentative timetable.

The central authority would be able to obtain the finalized information for each exam after student registration. At this point, the tentative timetable is updated with the finalized information. Finally, the exams would then be allocated to their actual venues, verified by the individual faculties and published. This approach proved to create a much better timetable than the initial centralized approach.

## Application Description

The structure of the *UTTSExam* program is governed by the hybrid scheduling approach. Aside from the basic functions of data entry and scheduling, facilities must be provided for venue partitioning and transfer and communication of data. Minor side-applications include the creation of a candidate seating plan and the printing of reports.

### Hardware & Software

The system was coded in *Java 1.3* with *Swing*™ components using the *IBM VisualAge*™ *for Java 3.5* software and *Microsoft Access*™ *2000* databases. The central machine used was a *Pentium-800* PC with 256MB RAM.

### System Design

There are two versions of *UTTSExam.* The *Registrar Version* is the full-fledged program that has access to all the features of the program. This is controlled by the Registrar's Office (the central authority). Instead, the individual faculties operate the stripped-down *Faculty Version*, which they use to enter the enrolment estimates for their faculty's examinations, as well as schedule their faculty timetables according to their venue partitions.

The *UTTSExam* system design is based on the 3-Tier architecture that is commonly used when building Client/Server applications. It keeps distinct the GUI, object oriented and data storage portions of our program. By separating the system into 3 tiers, they can be worked on independently (Reese, 1997).

*UTTSExam* is divided into the following 3 tiers. The *View* tier involves the graphical user interface. The *Application* tier is composed of the modules in an object-oriented paradigm that manipulate the objects in the system. This includes the scheduling engine, the printing modules and the report generator. Finally, the *Persistence*

layer consists of the actual database access. Figure 1 shows the system design.

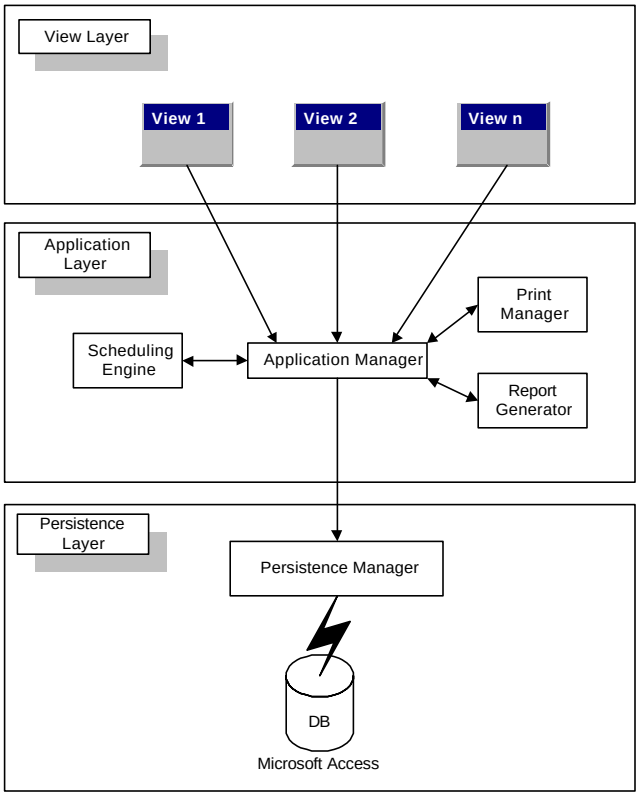


*Figure 1: UTTS System Architecture*

## Variable Definitions

*UTTSExam* contains several screens that enable the user to define all the variables of the timetabling problem, including:

- Examination information
- Venue information
- Candidate information
- Exam session particulars
- Constraint definitions

The Faculty version can only make changes and assign constraints to exams pertaining to that particular faculty.

## Constraint Definitions

Of particular interest are the constraint definitions. It is imperative that the program allows the users to define all the necessary constraints. *UTTSExam* allows the definition of a multitude of constraints:

- Separate all examinations with different duration.
- Spread all examinations of a student over the examination period as much as possible.
- Any 2 papers of a student should be placed minimally *x* sessions or *y* days apart.
- *Paper A* be placed *x* days away from *Paper B*
- *Paper A* be placed *x* sessions away from *Paper B*
- *Paper A* to be held before *Paper B*
- *Paper A* and *Paper B* to be held at the same time
- *Paper A* and *Paper B* are not to be held at the same time
- *Paper A* and *Paper B* to be held at the same time and same venue
- *Paper A* to be held as early/late as possible in the examination period
- *Paper A* is to be held in session *s*
- *Paper A* is not to be held in session *s*
- *Paper A* is to be held on/before/after date *d*
- *Paper A* is to be held within period *(d1, d2)*
- *Paper A* must not be held during period *(d1, d2)*
- *Paper A* is to be held in week *n*.
- *Paper A* is to be held at venue *v*.
- *Paper A* is to be held at a venue belonging to venue group *g*.

For ease of entry, *UTTSExam* also allows the definition of examination paper groups. In this way, the user can define both intra- and inter-group constraints, thereby allowing the definition of constraints between several papers at a time. In addition, the introduction of paper groups allows the following constraints:

- At most *x* papers from this group can be held in the same time slot
- Papers in this group must be held as far apart as possible

Figure 2 shows the Constraint Definition Screen.

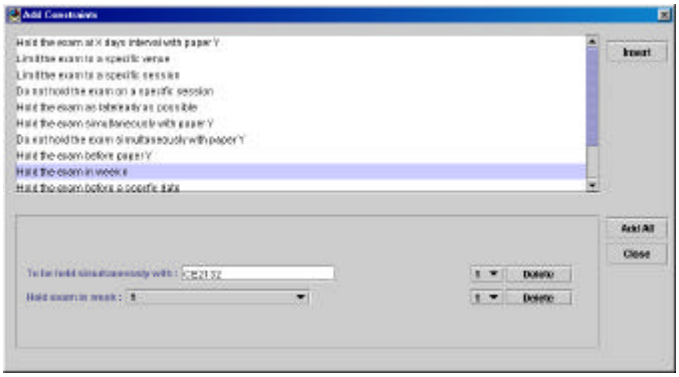

*Figure 2: Constraint Definition Screen*

## Venue Partitioning

When the faculties' enrolment estimates have been made, this information is used to proportionally assign venue partitions to each faculty in the form of the number of seats for each session. The heuristics used are given in the next section. *UTTSExam* provides a user interface for the manipulation and assignment of venue partitions to the faculties.

## Import/Export

Co-ordination between the faculties and the Registrar's Office is achieved via a central database. The Faculty Version allows the exporting of the enrolment estimates and examination information for retrieval by the Registrar's Office. With this information, the venue partitioning is done and uploaded to the database for retrieval by the faculties. The faculties then schedule their timetables and export the results. The merging of the various faculties' timetables is then done. This collated timetable is once more uploaded to the database for verification by the individual faculties. Once the changes are finalized, the timetable can be published.

## Output

Various reports can be created for reference by the program. These include the interim and finalized timetables; the constraints defined; examination, venue and candidate information; and the seating plan for each session. All of these reports are generated in html files, so that they can be displayed on the university webpages.

Figure 3 gives a section of the timetable generated for semester 1 of the 2001/2002 academic year.

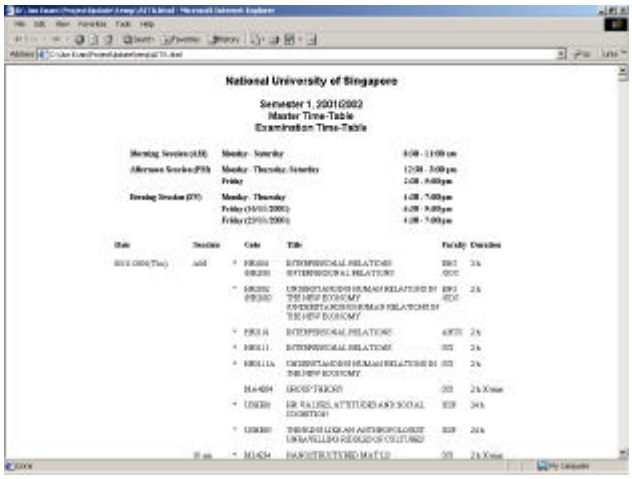


*Figure 3: Scheduled timetable output*

## Scheduling Engine

The heart of the program is the scheduling engine, including the merging process. The system allows manual intervention both before and after the automated scheduling process. The timetabling administrator might wish to manually insert some examinations into specific slots before invoking the scheduling engine. He can also tweak the generated timetable after the scheduling is done.

The details of the scheduling algorithm are given in the next section. Figure 4 shows the Scheduling Screen.

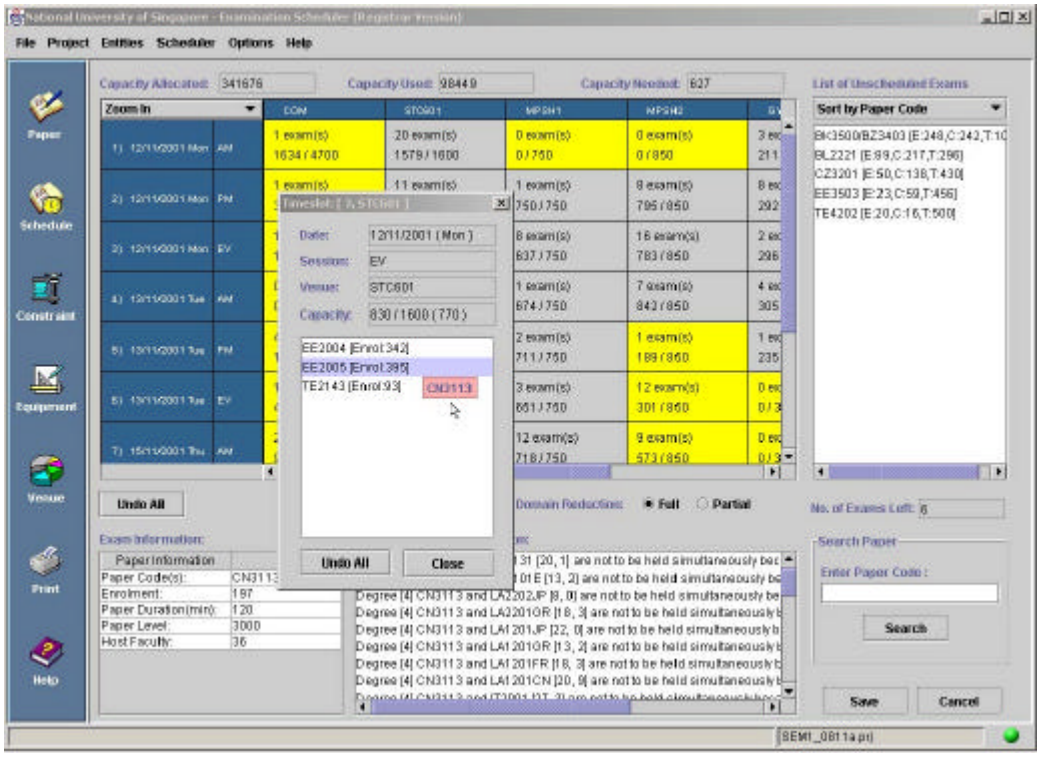


*Figure 4: Scheduling Screen*

## Uses of AI Technology

The main aim of *UTTSExam* is to create the examination timetable for NUS, and to that end there are three separate processes to examine, namely the venue partitioning process, the scheduling of the faculty timetables and the merging of the faculty timetables into the collated whole.

Venue partitioning is a complex problem. In order to decide the size and composition of the venue partitions to be allocated to each faculty, the following criteria must be taken into account:

- Obviously, the number of seats allocated should be proportional to the total candidacy of the faculty.
- The number of large papers for that faculty determines the number of large venue partitions given.
- Each faculty should be given extra seats so that they have some room to maneuver when scheduling the papers. However, the number of extra seats should *not* be strictly proportional to the total candidacy of the faculty. If too few extra seats are given, they are superfluous. Also, faculties with large candidacies tend to not need many extra seats since they have many small papers.

Currently, the process of venue partitioning is done manually. Although an automated venue-partitioning tool implementing the above heuristics is planned, manual intervention is probably still necessary. This is because it is difficult to specify all the pertinent constraints for a program to use, but relatively easy for a human to keep track of them. For example, past experience may reveal that one faculty tends to have many heavily constrained papers, and therefore needs larger partitions. This information is hard to express to an automated program.

When the venue partitions are obtained, the faculty timetables are scheduled using the Combined Method for solving CSOP (Ho and Lim, 2001). This involves first employing a stochastic search technique to find a high-quality solution that may violate some hard constraints. This solution is then used to guide a selection algorithm with consistency checking to create a feasible timetable with minimal changes to the initial solution. *UTTSExam* uses the Genetic Algorithm (Marin 1998) with Tabu Search post-optimization (Rayward-Smith et al. 1996), which is used to guide the Variable Ordering Method with AC-3 (Mackworth 1977) consistency checking.

To merge the faculty timetables, *UTTSExam* once again makes use of the Variable Ordering Method with AC-3. As each examination is considered, the program tries to schedule it into the slot depicted by the faculty timetable. If this causes a conflict, the examination is set aside. When all the examinations have been considered, the program attempts to insert the remaining exams into the schedule by a trial and error process: the program chooses a slot, then inserts the offending examination into it by removing the conflicting exams that were originally in the slot. This process is repeated several times until no such improving exchanges can be found.

If there are examinations left that still cannot be scheduled, there is no recourse but to contact the relevant faculties to request a loosening of the constraints. For the first semester of 2001/2002, there were 12 examinations left over at the end of the process. These were eventually inserted into the schedule after discussion with the relevant faculties.

This process produced a much better timetable than making use of the Combined Method alone to schedule the entire timetable. In fact, over 1000 examinations could not be scheduled using the Combined Method alone.

## Application Use and Payoff

*UTTSExam* was used to create the examination timetable for the recently completed semester 1 of the 2001/2002 academic year. The semester 2 examination timetable has already been scheduled based on the enrolment estimates. At the time of writing, student registration is under way. Upon its completion, the merging process could then begin.

Even though the program has only been used for one semester so far, the benefits of automating examination timetabling is obvious:

- In this initial implementation of the program, data entry and constraint specification was the most time-consuming part of the process, taking up to 2 weeks. However, subsequent semesters require much less data entry since the examination information remains largely the same. The actual scheduling, once the data entry was completed, takes less than 5 minutes.
- Assuming that the constraints entered are accurate, *UTTSExam* ensures that the produced timetable is conflict-free. This is especially important when taking into account the added complexity that cross-faculty modules bring to the task.
- Last-minute changes can be quickly catered for.
- Since each faculty is using the same system, all the output formats have been standardized.
- The speed of the system allows experimentation with different parameters and policies. Previously, the NUS examination period usually spanned a month, and there were only 2 sessions a day. It is only with the automated system that the new policy of 3-session days over 2 weeks could be implemented.
- The shortening of the examination period means that the renting of commercial venues (like Suntec City) is minimized. This translates to a substantial monetary saving.

## Application Development and Deployment

The *UTTSExam* program started development in NUS in mid-1999 when it became obvious that the introduction of cross-faculty modules would make the scheduling of course and examination timetables an increasingly difficult process. The existing manual timetabling process became exceedingly time-consuming, and cases of overlooked conflicts became more and more frequent. NUS therefore provided funding for the development of the University Timetable Scheduler program. This provided for two full-time application programmers and finances for research into timetable scheduling algorithms.

To facilitate the development of the program, a timetabling committee was set up containing two representatives from the administrative sections of each of the seven involved faculties, along with representatives from the Registrar's Office. Weekly meetings were held between the committee and the development team to discuss design issues, required features and other important points. Frequent email correspondence helped to keep both parties up to date with developments.

The initial meetings were concerned largely with formulating a data representation scheme that was sufficient to handle the requirements of the problem. Once that was done, the development team began converting the existing data from the university's central database into the *UTTSExam* format, while the faculty representatives worked out their respective constraints. To facilitate the entry of these constraints into the system, the Faculty Versions of *UTTSExam* was developed. Meanwhile, research was being done on various scheduling methods.

When all the data was obtained, preliminary experiments on scheduling the timetables for the entire university produced discouraging results. Due to the enormity of the problem, the scheduling engine was unable to create a feasible timetable at all (over 1000 examinations could not be placed within the examination period). We then decided to try the distributed approach, making use of the Faculty Versions (which were originally intended for data entry purposes only) by allowing each faculty to perform their own scheduling based on venue partitions.

Interestingly, even though the Faculty Versions made use of the same scheduling algorithm, this partition-schedule-merge approach managed to produce a timetable that was able to fit in all but 12 exams. After negotiation with the individual faculties, these exams were incorporated with the minimum of hassle.

As with all initial deployments of new software, there were a few teething problems. Up until the deadline for the publication of the timetable, numerous last-minute changes had to be made as a result of occurrences like late registrations and invigilator unavailability. There were also a few data entry errors that went unnoticed until very late in the process, which is always a potential problem on initial deployments. The automated system was able to cater to all of these changes quickly while maintaining the feasibility of the entire timetable, something that would be immensely difficult to do manually.

## Maintenance

Currently, the role of the Registrar's Office central authority is being played by the development team as alterations to the program are made as required. Control of the program will be handed over to Registrar's Office personnel in a few months, once some final issues are ironed out. Since the program is simple and instinctive to use, minimal training will be required (although an understanding of scheduling and constraints would aid its use). Data entry will also be minimal, as most modules will have much the same information and constraints across academic years.

*UTTSExam* can already unabashedly claim to be able to generate an examination timetable based on the more commonly encountered constraints. However, there are some rare special cases that need to be addressed. One such case involves an examination that had to be divided into two or more venues within the same session, since it was offered to students from different faculties who must be given different examinations (of the same module).

We are confident that the *UTTSExam* program will be able to produce examination timetables for NUS quickly and efficiently for years to come.

## Conclusion

This paper takes a look at what is required in the development of the examination-scheduling portion of the *UTTS* automated university timetable-scheduling program, entitled *UTTSExam.* We described the size and complexity of the problem of scheduling the exam timetables for a large university like NUS that employs a modular course structure with several cross-faculty modules, along with the strategy employed to best overcome these difficulties. The program's design, features and scheduling approach was also described.

Even though *UTTSExam* has only been deployed for one semester, the advantages of an automated examination-scheduling program are obvious and significant. Despite the difficulties encountered in its developmental process, all agree that it has been well worth the effort. We hope that our work will inspire other universities to consider automated timetable scheduling as well.

## References

Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.

Ho, W. K. and Lim, A. *A Hybrid-Based Framework for Constraint Satisfaction Optimization Problems*, in International Conference on Information Systems (ICIS) 2001, pg. 65-76.

Ho, W. K.; Lim, A.; and Oon, W. C. *UTTSExam: A University Examination Timetabling System*, submitted to IEEE Intelligent Systems 2002.

Lim, A.; Oon, W. C.; Ang, J. C.; and Ho, W. K *Development of a Campus-wide University Course Timetabling Application*, in 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT) 2000, pg. 71-77.

Lim, A.; Ang, J. C.; Ho, W. K.; and Oon, W. C. *A Campus-Wide University Examination Timetabling Application*, in Innovative Applications in Artificial Intelligence (AAAI/IAAI) 2000, pg. 1020-1025

Mackworth, A. K. *Consistency in Networks of Relations*, in Artificial Intelligence 8 (1977): 88-119

Marin, H. T. "Combinations of GA and CSP Strategies for Solving the Examination Timetabling Problem", *Ph.D. thesis, Instiuto Technologico y de Estudios Superiores de Menterrey*, 1998.

Rayward-Smith, V. J.; Osman, I. H.; Reeves, C. R.; and Smith, G. D. *Modern Heuristic Search Methods*, 1996.

Reese, G. *Database Programming with JDBC and Java*, O'Reilly 1997.

Tsang, E. *Foundations of Constraint Satisfaction*, 1993.