

An Optimization-Based Decision Support System for a
University Timetabling Problem: An Integrated Constraint and
Binary Integer Programming Approach

Ömer S. Benli*

A. Reha Botsali

Department of Information Systems

Department of Industrial Engineering

California State University, Long Beach

Texas A & M University

obenli@csulb.edu

reha@neo.tamu.edu

July 2004

* *Corresponding author:* Ömer S. Benli, Ph.D. Associate Professor, Department of Information Systems, California State University, Long Beach, College of Business Administration, 1250 Bellflower Boulevard, Long Beach, CA 90840-8506, USA. ++(1)562-985-5918. Email: obenli@csulb.edu

An Optimization-Based Decision Support System for a University Timetabling Problem: An Integrated Constraint and Binary Integer Programming Approach

Abstract

Constraint programming is a relatively new approach for solving combinatorial optimization problems. This approach is especially effective for large scale scheduling problems with side conditions. University course scheduling problem is one of the hard problems in combinatorial optimization. Furthermore, the specific requirements of each institution make it very difficult to suggest a generalized model and a solution algorithm for this problem. The purpose of this study was to design a decision support system for scheduling courses at a university. This system utilizes both constraint programming and mathematical programming techniques. The problem is solved in three stages. The first two stages, in tandem, generate a course schedule using constraint programming; in the last stage, classrooms are assigned to courses based on solution of a binary integer programming model. The proposed system is validated by experimental runs using actual course offerings and classroom data from past semesters.

Keywords: Timetabling, course scheduling, constraint programming, integer programming, optimization, decision support systems.

1 Introduction

Timetabling is one of the computationally very difficult problems in scheduling. In timetabling, the aim is to find suitable time slots for a number of tasks that require limited resources. Depending on the nature of the problem, the constraints in the problem can vary and there may be many different objectives. For example, in some cases the objective may be to minimize the length of the total time period over which the tasks are to be scheduled; in other cases the objective may be to find a feasible solution, subject to a fixed total time period and several other constraints. Yet in others, the objective may be to find a solution in which the least number of constraints are violated.

Schaerf (1999) classifies timetabling problems at educational institutions as school timetabling, course timetabling, and exam timetabling problems. The significance of this problem is mainly due to the difficulty of constructing a feasible timetable that satisfies the preferences of the administration, the instructors, and the students. In certain cases, it may be extremely difficult to find even a single feasible solution.

Although there are studies, such as Grobner, Wilke & Buttcher (2003) and Reis & Oliveira (2001), attempting to establish a common structure representing the timetabling problems, to the best of our knowledge, there does not exist a general model that is applicable for all cases. This is mainly due to the fact that every educational institution has its own special constraints and objectives. For example, for a secondary school, there should not be any gap between the class meetings; on the other hand this is allowed, and in some cases encouraged, in a university.

There are two types of timetabling problems in a university: examination and course scheduling problems. The focus of this study is on the course scheduling problem. Burke & Petrovic (2002) classify the methodologies used for these problems as sequential methods, cluster methods, constraint-based methods, and meta-heuristic methods. It is also possible to classify the timetabling problems in terms of the specific solution techniques used. The common solution techniques used in timetabling research are *graph coloring heuristics*, *mathematical programming*, *simulated annealing / tabu search*, *genetic*

algorithms, network flow models, and constraint programming.

Among these techniques, **graph coloring heuristics** are the first that were used to attack timetabling problems. A survey of the graph coloring techniques used in timetabling can be found in Carter (1986). Another traditional solution approach for timetabling problem is **mathematical programming**, and there are extensive studies using mathematical programming models such as Birbas, Daskalaki & Housos (1997), Tripathy (1984), Tripathy (1992) and Daskalaki, Birbas & Housos (2004).

Simulated annealing (SA) and **tabu search** (TS) are iterative improvement procedures that are designed to search for the optimal solution without being trapped at a local optimum. Hertz (1992) proposed a model that uses TS to find a course schedule where the length of class meetings is not known in advance. Alvarez-Valdes, Crespo & Tamarit (2002) solve the course scheduling problem at a Spanish university by dividing the problem into three phases. In the first phase, their solution generates an initial timetable. In the second phase, a TS algorithm is used to improve the initial timetable, and in the last phase a heuristic algorithm is used to improve the course - classroom assignments. Johnson (1990) gives an MP model for examination timetabling and suggests an approach that uses simulated annealing algorithm instead of solving the MP model with a large number of binary variables. Elmohamed, Coddington & Fox (1997) compare the performance of the variations of the annealing algorithm on the course scheduling problem.

Use of **genetic algorithms** (GA) is yet another heuristic approach for producing acceptable timetables. There is a large number of studies that use GA to solve timetabling problems. Surveys of the work in this area can be found in Corne, Ross & Fang (1994) and in Burke & Petrovic (2002).

Another solution approach for timetabling problems is considering the problem as a **network flow model**. Chahal & de Werra (1989) use this idea to construct an interactive system for timetabling. They validate this system for a small size timetable problem of an adult education school in Geneva. Dinkel, Mote & Venkataramanan (1989) propose a decision support system for course scheduling based on network flow models. de Werra (1985) gives an extensive overview of research in this area up to mid 1980's.

Constraint programming (CP) is a relatively new technique that is effective in solving large scale combinatorial problems. In CP approach, the timetabling problem is modeled as *constraint satisfaction problem* (CSP). A CSP consists of three main elements: *variables*; set of values that each variable can take (*variable domain*); and *constraints* restricting the values that the variables can take simultaneously.

In timetabling problems solved by CP, in general class meetings are defined as the variables and time slots are the domains of the variables. The typical timetabling constraints form the constraint set. The problem is to find values for all the variables that satisfy the constraints. In its optimization version, the problem is to find the value assignment of variables that minimizes or maximizes an objective function.

There are various reasons for CP to be an attractive alternative approach for solving timetabling problems. For example, in these problems, there is a large number of constraints that restrict the values that two variables can take simultaneously. CP makes use of this fact in *constraint propagation*. Constraint propagation is a mechanism in the solution process that generates new constraints on the values that the variables can take depending on the current set of constraints. *Domain reduction* is another technique that makes CP powerful. When a variable is assigned to a value in its domain, by domain reduction, the value domains of the other variables are reduced.

Such features of CP provide pruning of the search space while assigning values to the variables. This is important for a timetabling problem, because large number of courses and time slots cause a very large search space which results in excessive computation. Besides pruning the search space, the ability to define a *search strategy* in CP makes it more attractive. Hooker (2000) presents an excellent overview of constraint programming and its relation to mathematical programming.

Frangouli, Harmandas & Stamatopoulos (1995) use one of the constraint logic programming languages, the *Eclipse Constraint Programming System* [<http://www.icparc.ic.ac.uk/eclipse/> accessed 1 November 2003], to construct a timetabling system called UTSE at University of Athens. They also provide a user interface that allows user to specify the features of the timetable such as distance of class meetings, room utilization, and other features. Henz & Würtz (1995) solve the timetabling problem of a German university by using *Oz* which is a concurrent constraint language providing for functional,

object oriented, and constraint programming. (For details on this programming language see Smolka (1995). Recently *The Mozart Programming System* [<http://www.mozart-oz.org> accessed 21 November 2003] has replaced Oz.) Azevedo & Barahona (1994) first give a MP model for the timetabling problem for the Faculty of Science and Technology of the New University of Lisbon and show that the problem has a large number of binary variables in such a formulation. Then, instead of MP model, they suggest a CP model that is solved by using system *DOMLOG*, which is a constraint logic programming system for finite domain variables. In another study Deris, Omatu, Ohta & Samat a timetabling problem involving 536 courses, 45 time slots and 21 classrooms. However, as different from other studies, they implement CP using their own C++ code instead of using a general CP software. Zervoudakis & Stamatopoulos (2001) study the course scheduling problem using an object oriented model. They use ILOG Solver's C++ library [<http://www.ilog.com/products/oplstudio/> accessed 21 November 2003] in their model as is done in our study. However, their problem is an optimization problem at a departmental level, whereas the problem considered in this study is finding a feasible course schedule on a university level. Since the size of our problem is larger, the solution approach we use is different.

In this study, a course scheduling system for course scheduling problem at Bilkent University is proposed that utilizes CP and MP techniques. In the next section, the course scheduling problem at Bilkent University is described. In the following section, the models used in this study are given. In the section that follows that, the proposed system for solving the course scheduling problem is discussed. The final section summarizes and presents the conclusions from this study.

2 Problem at Bilkent University

Bilkent is a large university in Ankara, Turkey, having two campuses, several schools and around ten thousand students. The course scheduling problem is becoming considerably more demanding each year and thus, there is a need to use advanced techniques for course timetabling.

At Bilkent, the classes can meet during the weekdays from 08:40 to 17:30. Each day there are nine

50 minute time slots in which a class meeting can be scheduled.

A course that requires two time slots will be referred to as a 2-hour course. In the similar manner, there are 3-, 4-, and 5-hour courses which requires three, four and five time slots, respectively. Other than the 2-hour courses, all other courses are to be scheduled on two separate days. Each course has at least one class meeting requiring two time slots and all class meetings are done on consecutive time slots. Moreover, if a course is to meet on two separate days, then its class meetings cannot be assigned to two consecutive days.

Each section of a course may have different class sizes ranging from 10 to 65 students. The variety in section sizes requires scheduling of class meetings in an appropriate classroom which has sufficient capacity to accommodate all students. Currently, the classrooms can be categorized into four types based on their capacity as *type-1*, *type-2*, *type-3* and *type-4* classrooms with maximum capacities of 25, 30, 40 and 65 students respectively.

Although in some other institutions instructor and course section assignments may be done by timetabling programs (for example in Badri, Davis, Davis & Hollingsworth (1998), the number of sections for each course and the instructor of each course section are known in advance at Bilkent. Furthermore, each course section is reserved for a specific student group. That is, for each course section, the students who can enroll are known in advance. This makes it possible to compute the number of required sections for each course prior to the course scheduling process at each semester.

Summing up, following ten constraints are needed to comply with the conditions in and requirements of Bilkent University.

1. All class meetings will be assigned to a day.
2. All class meetings will be assigned to a feasible time slot.
3. Two class meetings belonging to the same course section cannot be scheduled on two consecutive days.
4. There is a limited daily classroom capacity.

5. There are limited number of classrooms in each classroom type.
6. The course sections assigned for specific student groups cannot overlap.
7. Sections of the courses that are taught by the same instructor cannot overlap.
8. An instructor cannot teach more than a fixed number of class meeting hours per day.
9. A student can attend at most eight hours of class meetings per day.
10. All class meetings should be assigned to an appropriate size classroom.

3 Models

Since course scheduling problem is NP-complete, the increase in the size of the problem makes it computationally very difficult to find a solution in a reasonable time. To overcome this problem, the constraint set is decomposed into subsets. By this way, the overall course scheduling problem can be represented as three integrated subproblems (see Figure 1) that handle with allocation of class meetings to days, construction of course schedule for each day, and assignment of classrooms to class meetings.

Our aim is to provide an interactive decision support tool for the scheduler to generate and compare feasible course schedules. In its current version, there is no objective function to optimize, hence the above sequential approach does not cause any degradation in the quality of the resulting solution. The course schedule is generated in three stages. In the first two stages, the respective problems are modeled as a constraint programs. In the third stage, the problem of class meeting - classroom assignment is formulated as a binary integer program.

The problems in the first two stages can also be formulated as binary integer programs. (An example of possible formulation is given in Appendix.) However, the sizes of the resulting programs make them computationally prohibitive to be used in actual applications. Furthermore, CP formulation provides considerable flexibility for inclusion of additional constraints, as future conditions make it necessary,

with minimal increase in the computational requirements. The notation used in the models is given in the next subsection.

3.1 Nomenclature

The following definitions and notation are needed to describe the models.

Group is a set of students. For each student group, there is a set of course sections. A student from a student group can attend course sections in that group's set of course sections. For this reason, the class meetings belonging to the course sections from the same set cannot be scheduled at the same time slots and total hours of such class meetings cannot exceed eight in a day, since a student cannot attend more than eight class hours.

Resource The classroom types are the resources. In this problem there are four types of resources $\{r_1, \dots, r_4\}$ corresponding to the classroom types $\{type-1, type-2, type-3, type-4\}$.

Sets

S Set of class meetings, $\{m_1, \dots, m_n\}$

I Set of instructors, $\{I_1, \dots, I_m\}$

S_{I_i} Set of class meetings given by instructor I_i

G Set of student groups, $\{g_1, \dots, g_k\}$

S_{g_i} Set of class meetings belonging to student group g_i

R Set of resources, $\{r_1, \dots, r_4\}$

C_{r_k} Set of classrooms (classroom type k) belonging to resource r_k ($r_k \in R$)

S_{r_k} Set of class meetings that require resource r_k ($r_k \in R$)

P Set of class meeting pairs m_j, m_k that belong to the same course section

H_i Set of class meetings that are i hours long, $i \in \{1, 2, 3\}$

Parameters and Functions

h_{m_j} Hours of class meeting m_j

d_{r_k} Maximum capacity of resource r_k per day ($r_k \in R$)

e_{r_k} Maximum capacity of resource r_k per hour ($r_k \in R$)

p_{g_i} Maximum daily class hours student group g_i can attend ($g_i \in G$)

s_{I_i} Maximum daily class hours instructor I_i can teach ($I_i \in I$)

$$f(x, y) = \begin{cases} 1, & \text{if } x = y; \\ 0, & \text{otherwise.} \end{cases}$$

Variables

D_{m_j} Assigned day of class meeting m_j

T_{m_j} Start time of class meeting m_j

$$X_i(m_j, k, l) = \begin{cases} 1, & \text{if class meeting } m_j \text{ is assigned to} \\ & k^{\text{th}} \text{ element of } C_{r_i} \text{ at time slot } l \text{ (} r_i \in R \text{);} \\ 0, & \text{otherwise.} \end{cases}$$

3.2 Allocation of Class Meetings to Days

In this stage, the class meetings are allocated to the weekdays with respect to the constraints 1, 3, 4, 8, 9 given in section 2. The objective is to find a solution satisfying the following constraint representations:

$$(1) \quad D_{m_j} \in \{0, \dots, 4\}, \quad \forall m_j \in S$$

$$(3) \quad D_{m_j} - D_{m_k} \geq 2 \vee D_{m_k} - D_{m_j} \geq 2, \quad \forall (m_j, m_k) \in P$$

$$(4) \quad \sum_{m_j \in S_{r_k}} f(D_{m_j}, t) \times h_{m_j} \leq d_{r_k}, \quad \forall r_k \in R, t \in \{0, \dots, 4\}$$

$$(8) \quad \sum_{m_j \in S_{I_i}} f(D_{m_j}, t) \times h_{m_j} \leq s_{I_i}, \quad \forall I_i \in I, t \in \{0, \dots, 4\}$$

$$(9) \quad \sum_{m_j \in S_{g_i}} f(D_{m_j}, t) \times h_{m_j} \leq p_{g_i}, \quad \forall g_i \in G, t \in \{0, \dots, 4\}.$$

As stated in the previous section, each class meeting should be assigned to a day and two class meetings of a course section cannot be scheduled on consecutive days. These two constraints are forced by constraint sets 1 and 2.

Each class meeting requires a classroom of a specific type; the classroom types are considered as resources which are used by class meetings. The daily resource capacities (d_{r_k}) depend on the number of classrooms belonging to the classroom type represented by that resource. For example, having 25 *type-4* classrooms means that the daily class hour capacity of *type-4* resource is $25 \times 9 = 225$ (number of classrooms \times number of time slots in a day). At a day where a class meeting is allocated, this class meeting requires a classroom of its type for x number of hours where x is the length of the class meeting. The resource capacity constraint is represented by constraint set 3.

The total hours of the class meetings that a student can attend per day are at most assumed be eight (one out of nine time slots should be left for lunch break). Similarly, it is assumed that each instructor can at most teach five hours per day. It is possible to change this limit for an instructor upon her request. These two constraints are enforced by constraint sets 7 and 6, respectively.

If all the constraints are considered, it will be seen that this problem is a constraint satisfaction problem (CSP) where class meetings are variables and days are the domains of the variables. This problem is formulated as a CP model.

When this problem is solved, the output will indicate which class meeting will take place on which day. The constraints ensure that the daily classroom capacities, daily class hour limits of the instructors and the students are not exceeded. In the next stage of the solution process, daily course schedules are constructed based on the resulting day-class meeting assignment of the initial model.

3.3 Construction of Daily Course Schedules

This model assigns class meetings (that are assigned to a specific day by the previous stage) to a specific time slot of the day, subject to the constraints 2, 5, 6, 7 defined in Section 2. The constraint sets specifying the feasible solution space are given below:

$$(2) \quad T_{m_j} \in \{0, \dots, 9 - h_{m_j}\}, \quad \forall m_j \in S$$

$$(5) \quad \sum_{m_j \in S_{r_k}} \sum_{i=0}^{h_{m_j}-1} f(T_{m_j}, t - i) \leq e_{r_k}, \quad \forall r_k \in R, t \in \{0, \dots, 8\}$$

$$(6) \quad \sum_{m_j \in S_{g_i}} \sum_{i=0}^{h_{m_j}-1} f(T_{m_j}, t - i) \leq 1, \quad \forall g_i \in G, t \in \{0, \dots, 8\}.$$

$$(7) \quad \sum_{m_j \in S_{I_i}} \sum_{i=0}^{h_{m_j}-1} f(T_{m_j}, t - i) \leq 1, \quad \forall I_i \in I, t \in \{0, \dots, 8\}$$

The constraint set 1 ensures that if a class meeting is allocated to a day by first stage, then it should be scheduled on a time slot on that day. The classroom types are, again, considered as resources with maximum hourly resource capacities equal to the number of classrooms in that resource type. For any time slot, the class meetings using a specific classroom type cannot exceed the number of classrooms in that classroom type which is stated by constraint set 3.

The last two constraint sets ensure that the students cannot attend more than one class meeting, and the instructors cannot teach more than one class meeting at a time slot.

This problem is a CSP and to find a daily course schedule, it is possible to construct a CP model for each day. This model schedules the class meetings of each day resulting in the weekly course schedule. In case there is no feasible solution for a particular day, the scheduler runs the first stage model again with extra constraints until there exist feasible daily course schedules for all days. The resulting feasible schedules contain only course and time information, but no classroom assignments are made. The next

model assigns classrooms to the scheduled class meetings.

3.4 Assignment of Class Meetings to Classrooms

The problem is to find appropriate size classrooms for all class meetings scheduled on that day. In this stage, it is allowed to assign a larger classroom type to a smaller class. By defining the parameter $s(m_j)$ as the start time of class meeting m_j which is assigned in the second stage, the following constraints are required to be satisfied:

$$(1) \quad \sum_{n=i}^4 \sum_{k \in C_{r_n}} X_i(m_j, k, s(m_j)) = 1, \quad \forall i \in \{1 \dots 4\}, \forall m_j \in S_{r_i}$$

$$(2) \quad \sum_{m_j \in (\cup_{n=1}^i S_{r_n})} X_i(m_j, k, l) \leq 1, \quad \forall i \in \{1 \dots 4\}, \forall k \in C_{r_i}, \\ l \in \{0, \dots, 8\}$$

$$(3) \quad X_i(m_j, k, s(m_j)) = X_i(m_j, k, s(m_j) + 1), \quad \forall i \in \{1 \dots 4\}, \forall k \in C_{r_i}, \\ \forall m_j \in ((\cup_{n=1}^i S_{r_n}) \cap H_2)$$

$$(4) \quad X_i(m_j, k, s(m_j)) = X_i(m_j, k, s(m_j) + 1), \quad \forall i \in \{1 \dots 4\}, \forall k \in C_{r_i}, \\ \forall m_j \in ((\cup_{n=1}^i S_{r_n}) \cap H_3)$$

$$X_i(m_j, k, s(m_j)) = X_i(m_j, k, s(m_j) + 2),$$

$$(5) \quad X_4, X_3, X_2, X_1 \in \{0, 1\}.$$

In the above formulation, constraint set 1 ensures that each class meeting is assigned to an appropriate size classroom. Constraint set 2 states that there can be at most one class meeting in a classroom during a time slot. The last two set of constraint sets require that all the hours of the 2-hour and 3-hour class meetings should be assigned to the same classroom respectively.

The objective function can be formulated according to the scheduler's preferences, such as maximizing classroom utilization or minimizing the distance traveled by the students, among other possibilities.

4 Discussion

The course scheduling system proposed in this study is an optimization-based decision support system (DSS). As discussed in Geoffrion & Marutana (1995), such systems integrate optimizing capability with machinery for preparing the necessary data as well as providing a good user interface. As seen in Figure 2, by appending and removing constraints from the CP models of the first two stages, the scheduler can obtain different course schedules and compare them with respect to different criteria. Moreover, by series of appending constraints, the scheduler directs the series schedules generated by the system towards a final course schedule.

We should stress that in this system the first and second stage models play the most crucial role. Since the scheduler makes the decision analysis by adding and removing constraints, using CP models meets the requirements of these two stages best. In addition to this, the modularity provided by the division of the problem into stages allows the scheduler to make changes on specific parts of the course schedule without affecting the overall schedule. Whenever, the scheduler agrees upon a class meeting - time assignment, all she needs to do is to run an MP model for which always a feasible solution is guaranteed by the CP models.

The three main components of a DSS are a model base, a database, and an interactive software system to linking the user to each of these. The interrelationship of models and the database in our system is depicted in Figure 3. In this system the interaction between the models and the database is provided by the software ILOG OPL Studio 2.1.3. OPL Studio [<http://www.ilog.com/products/oplstudio/> accessed 21 November 2003] which is an integrated development environment for combinatorial optimization applications. It can be effectively used for constructing and solving linear programming, integer programming, and constraint programming models. Its inputs are the model codes written in opti-

mization programming language (OPL) developed by van Hentenryck (1999). The database connection capability available in OPL Studio enables the data to be easily retrieved and stored before and after running the models.

OPL Studio has several tools including CPLEX (mathematical programming solver), SOLVER (solver for constraint programming) and SCHEDULER (a tool developed for constraint based scheduling). Once a model is constructed, after compilation, OPL Studio automatically detects the problem type and determines the most convenient solver to solve it. Since in the proposed system, the solution is obtained by utilizing both constraint programming and mathematical programming, OPL Studio is very convenient single system to use. In addition, its user-friendly graphical environment and graphical representation of the solutions make it more attractive.

A special tool in OPL Studio, *scheduler*, contains efficient algorithms for solving the constraints of resource constrained scheduling problems. This option makes it possible to define and solve the constraints related to the resources and the activities very easily. Without using complicated mathematical expressions, it is possible to define constraints for resources, resource capacities, activities, resource requirements of activities as simple as writing a sentence in an ordinary language. Since the course scheduling problem is a type of resource constrained scheduling problem, in this study, special resource constraints of ILOG OPL Studio provide great ease in the solution process. The classroom types are represented as resources with available capacities and the class meetings are the activities to be scheduled whose durations equal to length of class meetings. In the second stage, in addition to the classroom resources, the student groups and instructors are defined as resources with capacity of one, since each class meeting requires a particular instructor and one or more student groups.

In order to validate this system, course schedules for Fall 99 and Spring 00 semesters were generated using data provided by *STARS (Student Academic Information Registration System)* [<http://stars.bilkent.edu.tr/> accessed 21 November 2003] at Bilkent. The input data were prepared for five colleges in the university involving eighteen departments. There were more than 670 course sections in both semesters. For fall semester 112 student groups and for spring semester 107 student groups were formed.

In both semesters, total number of students was more than 4,000. The models were run on a SunOS 5.5 - SPARCserver 1000E computer for basic constraints. The required courses of all the students were successfully scheduled in less than a hour. The solution time of the model runs and model statistics are displayed on Tables 1 through 3. In the constraint programming model statistics, the columns *choice points* and *failures* represent the branching and backtracking statistics during the search process. In our experiments, we have observed that if good search strategies are defined to explore the search space, the number of failures can be greatly reduced. In general, first fail principle in variable ordering is a good way to construct search strategy. To apply first fail principle, the variables with the fewest number of values in their domains are chosen for value assignment. At the same time another criteria for defining variable ordering in search strategy is first assigning values to the variables that are involved in the constraints which are hard to satisfy. This is a kind of first fail principle. If a variable is involved in a constraint that is difficult to satisfy, then there is higher probability that the values assigned to this variable will fail. The timetabler should determine the most effective search strategy for CP models in first and second stage by investigating the course data s/he deals with.

5 Summary and Conclusions

In this study, a university timetabling problem is analyzed and a system is proposed to provide the decision support for the scheduler for solving the course scheduling problem at Bilkent University.

The system consists of three stages. In the first stage, the class meetings are allocated to days and in the second stage daily course schedules are constructed. In both of these stages, constraint programming models are used. In the final stage, the scheduled class meetings are assigned to specific classrooms by using binary integer programming.

The system is validated for Bilkent University's course offerings and classroom data from previous semesters. The course schedule generated in this study involves only the required courses, but the results can easily be extended to a full course schedule involving the elective courses and laboratory sessions by

defining appropriate variables in the models.

A APPENDIX

A.1 Binary Integer Program for Allocation of Class Meetings to Days

Define a binary variable $x(m_j, t)$ as:

$$x(t, m_j) = \begin{cases} 1, & \text{if class meeting } m_j \text{ is assigned to day } t; \\ 0, & \text{otherwise.} \end{cases}$$

Using the same notation given in subsection 3.1, the constraints 1, 3, 4, 8 and 9 of section 2 can be represented in a mathematical programming model as below:

$$(1) \quad \sum_{t=0}^4 x(t, m_j) = 1, \quad \forall m_j \in S$$

$$(3) \quad \sum_{t=0}^3 x(t, m_j) + x(t+1, m_j) + x(t, m_k) + x(t+1, m_k) \leq 1, \quad \forall (m_j, m_k) \in P$$

$$(4) \quad \sum_{m_j \in S_{r_k}} x(t, m_j) \times h_{m_j} \leq d_{r_k}, \quad \forall r_k \in R, t \in \{0, \dots, 4\}$$

$$(8) \quad \sum_{m_k \in S_{I_i}} x(t, m_k) \times h_{m_k} \leq s_{I_i}, \quad \forall I_i \in I, t \in \{0, \dots, 4\}$$

$$(9) \quad \sum_{m_k \in S_{g_i}} x(t, m_k) \times h_{m_k} \leq p_{g_i}, \quad \forall g_i \in G, t \in \{0, \dots, 4\}$$

$$x(t, m_j) \in \{0, 1\} \quad \forall m_j \in S, t \in \{0, \dots, 4\}$$

A.2 Binary Integer Program for Construction of Daily Course Schedules

Define a binary variable $y(m_j, t)$ as:

$$y(t, m_j) = \begin{cases} 1, & \text{if class meeting } m_j \text{ starts at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

Mathematical programming representation of the constraints 2, 5, 6, 7 of section 2 are given below:

$$(2) \quad \sum_{t=0}^{9-i} y(t, m_j) = 1, \quad \forall i \in \{1 \dots 3\}, \forall m_j \in H_i$$

$$(5) \quad \sum_{m_j \in S_{r_k}} y(0, m_j) \leq e_{r_k}, \quad \forall r_k \in R$$

$$\begin{aligned} & \sum_{m_j \in (S_{r_k} \cap H1)} y(1, m_j) + \\ & \sum_{m_j \in (S_{r_k} \cap H2)} (y(1, m_j) + y(0, m_j)) + \\ & \sum_{m_j \in (S_{r_k} \cap H3)} (y(1, m_j) + y(0, m_j)) \leq c_{r_k}, \quad \forall r_k \in R \end{aligned}$$

$$\begin{aligned} & \sum_{m_j \in (S_{r_k} \cap H1)} y(t, m_j) + \\ & \sum_{m_j \in (S_{r_k} \cap H2)} (y(t, m_j) + y(t-1, m_j)) + \\ & \sum_{m_j \in (S_{r_k} \cap H3)} (y(t, m_j) + y(t-1, m_j)) + \\ & y(t-2, m_j) \leq c_{r_k}, \quad \forall r_k \in R, \\ & \quad \quad \quad t \in \{2, \dots, 8\} \end{aligned}$$

$$(6) \quad \sum_{m_j \in S_{g_i}} y(0, m_j) \leq 1, \quad \forall g_i \in G$$

$$\begin{aligned} & \sum_{m_j \in (S_{g_i} \cap H1)} y(1, m_j) + \\ & \sum_{m_j \in (S_{g_i} \cap H2)} (y(1, m_j) + y(0, m_j)) + \\ & \sum_{m_j \in (S_{g_i} \cap H3)} (y(1, m_j) + y(0, m_j)) \leq 1, \quad \forall g_i \in G \end{aligned}$$

$$\sum_{m_j \in (S_{g_i} \cap H1)} y(t, m_j) +$$

$$\sum_{m_j \in (S_{g_i} \cap H2)} (y(t, m_j) + y(t-1, m_j)) +$$

$$\sum_{m_j \in (S_{g_i} \cap H3)} (y(t, m_j) + y(t-1, m_j) + y(t-2, m_j)) \leq 1, \forall g_i \in G, t \in \{2, \dots, 8\}$$

$$(7) \quad \sum_{m_j \in S_{I_i}} y(0, m_j) \leq 1, \quad \forall I_i \in I$$

$$\sum_{m_j \in (S_{I_i} \cap H1)} y(1, m_j) +$$

$$\sum_{m_j \in (S_{I_i} \cap H2)} (y(1, m_j) + y(0, m_j)) +$$

$$\sum_{m_j \in (S_{I_i} \cap H3)} (y(1, m_j) + y(0, m_j)) \leq 1, \quad \forall I_i \in I$$

$$\sum_{m_j \in (S_{I_i} \cap H1)} y(t, m_j) +$$

$$\sum_{m_j \in (S_{I_i} \cap H2)} (y(t, m_j) + y(t-1, m_j)) +$$

$$\sum_{m_j \in (S_{I_i} \cap H3)} (y(t, m_j) + y(t-1, m_j) + y(t-2, m_j)) \leq 1, \forall I_i \in I, t \in \{2, \dots, 8\}$$

$$y(t, m_j) \in \{0, 1\} \quad \forall m_j \in S, t \in \{0, \dots, 8\}.$$

References

- [1] Schaerf A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, **13**:303–316.
- [2] Grobner M., Wilke P., & Buttcher S. (2003). A standard framework for timetabling problems. In E. Burke and P. De Causemacker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pp 24–38. Springer-Verlag.
- [3] Reis L.P. & Oliveira E. (2001). A language for specifying complete timetabling problems. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pp 322–341. Springer-Verlag.

- [4] Burke E.K., & Petrovic S. (2002). Recent research directions in automated timetabling. *Eur J Opnl Res*, **140**:266–280.
- [5] Carter M.W. (1986). A survey of practical applications of examination timetabling problems. *Opns Res*, **24**:193–201.
- [6] Birbas T., Daskalaki S., & Housos E. (1997). Timetabling for Greek high schools. *J Opl Res Soc*, **48**:1191–1200.
- [7] Tripathy A. (1984). School timetabling – a case in large binary integer linear programming. *Mngt Sci*, **30**:1473–1489.
- [8] Tripathy A. (1992). Computerised decision aid for timetabling – a case analysis. *Discrete Applied Mathematics*, **35**:313–323.
- [9] Daskalaki S., Birbas T., & Housos E. (2004). An integer programming formulation for a case study in university timetabling. *Eur J Opnl Res*, **153**:117–135.
- [10] Hertz A. (1992). Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics*, **35**:255–270.
- [11] Alvarez-Valdes R., Crespo E., & Tamarit J.M. (2002). Design and implementation of a course scheduling system using tabu search. *Eur J Opnl Res*, **137**:512–523.
- [12] Johnson D. (1990). Timetabling university examinations. *J Opl Res Soc*, **41**:39–47.
- [13] Elmohamed M.A.S., Coddington P., & Fox G. (1997). A comparison of annealing techniques for academic course scheduling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pp 92–112. Springer-Verlag.
- [14] Corne D., Ross P., & Fang H-L. (1994). Evolutionary timetabling: Practice, prospects and work in progress. In *UK Planning and Scheduling Workshop*.

- [15] Chahal N. & de Werra D. (1989). An interactive system for constructing timetables on a PC. *Eur J Opnl Res*, **40**:32–37.
- [16] Dinkel J.J., Mote J., & Venkataramanan M.A. (1989). An efficient decision support system for academic course scheduling. *Opns Res*, **37**:853–864.
- [17] de Werra D. (1985). An introduction to timetabling. *Eur J Opnl Res*, **19**:151–162.
- [18] Hooker J. (2000). *Logic Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons.
- [19] Frangouli H., Harmandas V., & Stamatopoulos P. (1995). UTSE: Construction of optimum timetables for university courses - a CLP based approach. In *PAP 95*.
- [20] Henz M. & Wurtz J. (1995). Using Oz for college timetabling. In *Proceedings of the 1995 International Conference on the Practice and Theory of Automated Timetabling*.
- [21] Smolka G. (1995). The Oz Programming Model. *Lecture Notes in Computer Science*, **1000**:324–343.
- [22] Azevedo F. & Barahona P. (1994). Timetabling in constraint logic programming. In *Proceedings of the World Congress on Expert Systems'94*.
- [23] Deris S.B., Omatu S., Ohta H., & Samat P.A.B.D. (1997). University timetabling by constraint-based reasoning: A case study. *J Opl Res Soc*, **48**:1178–1190.
- [24] Zervoudakis K. & Stamatopoulos P. (2001). A generic object-oriented constraint-based model for university course timetabling. In Burke E & Erben W, eds, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pp 28–47. Springer-Verlag.
- [25] Badri M.A., Davis D.L., Davis D.F., & Hollingsworth J. (1998). A multi-objective course scheduling model: Combining faculty preferences for courses and times. *Comput Opl Res*, **25**:303–316.

- [26] Geoffrion A. & Marutana S. (1995). Generating optimization-based decision support systems. In *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, volume III, IEEE Computer Society Press, pp 439–448.
- [27] van Hentenryck P. (1999). *The OPL Optimization Programming Language*. MIT Press.

FIGURES

Figure 1: Stages of the Solution Process

Figure 2: Interaction of Models

Figure 3: Information Flow in the System

TABLES

Semester	Constraints (C)	Variables (V)	Choice Points (CP)	Failures (F)	Solution Time (ST)
Fall 99	3,910	7,525	2,294	10,054	50.12 sec
Spring 00	3,931	7,693	1,329	76	38.15 sec

Table 1: Model I Statistics for Fall 99 and Spring 00 Data

Day	Fall 99					Spring 00				
	C	V	CP	F	ST	C	V	CP	F	ST
Monday	666	1,246	243	0	3.55 sec	688	1,358	309	4	3.90 sec
Tuesday	828	1,617	289	10	5.26 sec	847	1,743	384	1	6.95 sec
Wednesday	875	1,743	291	1	5.74 sec	850	1,743	295	44	6.36 sec
Thursday	738	1,491	317	2	4.10 sec	745	1,533	394	4	5.11 sec
Friday	680	1,428	244	0	4.41 sec	622	1,316	317	0	3.90 sec

Table 2: Model II Statistics for Fall 99 and Spring 00 Data

	Fall 99			Spring 00		
Day	C	V	ST	C	V	ST
Monday	23,728	106,542	83.88 sec	25,680	115,128	153.53 sec
Tuesday	19,660	108,234	99.54 sec	25,970	123,183	265.25 sec
Wednesday	19,348	127,890	170.42 sec	23,924	130,941	181.27 sec
Thursday	15,630	92,772	72.30 sec	16,838	106,200	178.25 sec
Friday	16,362	84,861	59.15 sec	17,024	82,971	54.31 sec

Table 3: Model III Statistics for Fall 99 and Spring 00 Data

Captions for Figures and Tables

Figure 1: Stages of the Solution Process

Figure 2: Interaction of Models

Figure 3: Information Flow in the System

Table 1: Model I Statistics for Fall 99 and Spring 00 Data

Table 2: Model II Statistics for Fall 99 and Spring 00 Data

Table 3: Model III Statistics for Fall 99 and Spring 00 Data