8-2010

Term-End Exam Scheduling at United States Military Academy/West Point

Siqun Wang

Michael Bussieck

Monique Guignard
University of Pennsylvania

Alexander Meeraus

Term-End Exam Scheduling at United States Military Academy/West Point

Abstract
Scheduling term-end exams (*TEE*) at the United States Military Academy in West Point is unlike any other exam timetabling problem we know of. Exam timetabling normally produces a conflict-free timetable covering a reasonably long exam period, where every exam is scheduled exactly once for all the students enrolled in the corresponding class. The situation is quite different at West Point. There are hundreds of exams to schedule over such a short time period that there is simply no feasible solution. The challenge is then to allow something that is not even considered elsewhere, that is, creating multiple sessions of some exams, scheduled at different times within the exam period, to allow each student to take all exams he/she must take. The overall objective is to find a feasible exam schedule with a minimum number of such duplicate exams.

The paper describes a system that has been developed at GAMS Development Corp. in close cooperation with the scheduling staff at West Point, and that has been used successfully since 2001. It uses mathematical optimization in several modules, and some of the techniques proposed are new. It is fast and flexible, and allows for human interaction, such as adding initially unexpected constraints, coming for instance from instructors' preferences and dislikes, as well as their hierarchical rankings. It is robust and can be used by people familiar with the organization at West Point, without the need for them to be technically-trained. Overall, using the course and student information databases, it is an effective decision support system that calls optimization tools in an unobtrusive way.

Keywords
exam scheduling, timetabling, 0-1 integer programming

Term-End Exam Scheduling at United States Military Academy/West Point

Siqun Wang[*], Michael Bussieck[†], Monique Guignard[‡], Alexander Meeraus[†], Fred O'Brien[§]

Submitted in June 2008, Latest Revision on August 10, 2009

Abstract

Scheduling term-end exams (*TEE*) at the United States Military Academy in West Point is unlike any other exam timetabling problem we know of. Exam timetabling normally produces a conflict-free timetable covering a reasonably long exam period, where every exam is scheduled exactly once for all the students enrolled in the corresponding class. The situation is quite different at West Point. There are hundreds of exams to schedule over such a short time period that there is simply no feasible solution. The challenge is then to allow something that is not even considered elsewhere, that is, creating multiple sessions of some exams, scheduled at different times within the exam period, to allow each student to take all exams he/she must take. The overall objective is to find a feasible exam schedule with a minimum number of such duplicate exams.

The paper describes a system that has been developed at GAMS Development Corp. in close cooperation with the scheduling staff at West Point, and that has been used successfully since 2001. It uses mathematical optimization in several modules, and some of the techniques proposed are new. It is fast and flexible, and allows for human interaction, such as adding initially unexpected constraints, coming for instance from instructors' preferences and dislikes, as well as their hierarchical rankings. It is robust and can be used by people familiar with the organization at West Point, without the need for them to be technically-trained. Overall, using the course and student information databases, it is an effective decision support system that calls optimization tools in an unobtrusive way.

Keywords: exam scheduling, timetabling, 0-1 integer programming;

[*]LKC School of Business, Singapore Management University; Now with DemandTec, Inc.

[†]GAMS Development Corporation, Washington DC

[‡]Wharton School, University of Pennsylvania. Research partially supported by NSF Grants DMI-9900183 and DMI-0400155

[§]Major Fred O'Brien was in charge of the development of the United States Military Academy's Student Information System.

1 INTRODUCTION

The timetabling problem discussed in this paper was addressed in a joint project - the Term-End Exam (*TEE*) scheduling project - between the United States Military Academy (USMA) in West Point and GAMS Development Corp. Scheduling term-end exams at West Point is unlike any other exam timetabling problem we know of. Exam timetabling normally produces a timetable where every exam is scheduled exactly once for all the students enrolled in the corresponding class. In addition, the exams taken by a particular student are scheduled in non-overlapping time slots, with a maximum number of exams per day, over a reasonably long exam period. The situation is quite different at West Point. There are so many exams to schedule over a very short time period that there is simply no feasible solution. The challenge is then to allow something that is not even considered elsewhere, that is, creating multiple sessions of some exams, scheduled at different times during the exam period, to allow each student to take all exams he/she must take. In practice, for a given course, one of the exams will be designated as the main - called *primary* - exam, with as many students as possible assigned to it. Given the natural reluctance of instructors to create multiple problem sets, and the unavoidable unfairness resulting from different exam questions, one wants to keep to a minimum the number of such extra exams.

To be more specific, the USMA in West Point must schedule exams for roughly 250 courses and about 4,000 cadets, with each cadet taking five to eight courses, for a total of roughly 20,000 possible individual cadet-exam assignments. There are six exam days (usually from Monday through Saturday) with at most two exam time slots available per day. The unusually small number of time slots and the large number of exam-student combinations to accommodate in the schedule always create conflicts. To resolve these conflicts, one may have to schedule, in addition to a course's primary exam, a second and if necessary, a third exam called "*makeup(s)*". A feasible exam timetable must provide a conflict-free exam schedule for each cadet. The overall objective is to find feasible exam schedules with as few makeup exams as possible. While makeups are occasionally encountered in university exam schedules, their number is usually very small, a few units at most. What is unusual here is the large number of makeups necessary to achieve a conflict-free schedule.

What makes the exam timetabling problem at USMA particularly challenging - and interesting - is its high average *conflict density*[1] in addition to a variety of "real world" constraints and rules. The automated *TEE* scheduling system must determine for which courses to offer *makeups* as well as when to schedule primaries and makeups so that the resulting individual cadet timetables are conflict-free. We designed the automated *TEE* system as an optimization-based tool. Given the magnitude and complexity of the problem,

[1] The conflict density of an exam course is calculated as the total number of other exams that it conflicts with divided by the total number of exams.

it is unrealistic to expect finding an optimal, or even close to optimal, solution using a single mathematical programming model. We present here a specialized modeling and solution approach based on a sequence of mathematical optimization models. This approach was implemented in the Academy's Management System and together with a substantial amount of human expertise has been in use for over seven years.

The paper is organized as follows. In Section 2, we review the relevant literature, describe the *TEE* scheduling problem and present the notation system that was used in its design. In Section 3, we first concentrate on the definition of the decision variables, and introduce a 0-1 linear programming formulation. The details of this are provided in an appendix which delineates all *TEE* constraints in a precise mathematical fashion. Next, we present a reformulation, which yields a different, bilinear, 0-1 programming model. Our solution approach is presented in Section 4, and it is based on the bilinear model and comprises mainly of the following steps: (1) the generation of an initial solution, and (2) the solution improvement. We continue with a description of the data set and the experimental design in Section 5. In the last section, we conclude with computational results that demonstrate the validity and the potential of our approach.

2 DESCRIPTION OF THE *TEE* TIMETABLING PROBLEM

2.1 Literature overview

Examination timetabling involves assigning all exams to a certain number of time slots such that no student is required to take more than one examination at a time. The problem may be further constrained by limited resources (e.g. rooms, teaching aids, etc.) and other requirements. Even without extra constraints, the examination timetabling problem (ETTP) being equivalent to the general graph coloring problem is NP-complete. It is a very important part of school operation and support services, and as such has recently received a lot of attention from both researchers and practitioners. A working group on Automated Timetabling (WATT) was formed in 1996 to discuss, promote, and perform research in automated timetabling issues and methods, and an international conference on the Practice and Theory of Automated Timetabling (PATAT) has been held bi-annually since 1995 as a forum for both researchers and practitioners of timetabling to exchange ideas. There has recently been an international timetabling competition (ITC2007) (see http://www.cs.qub.ac.uk/itc2007/) that includes a track on exam timetabling with various new exam timetabling benchmarks.

As early as 1961, starting with Appleby *et al.* (1961), Bush *et al.* (1961) and Gotlieb (1962), a number of algorithms were developed and used for practical automated timetabling problems. Due to the magnitude of real problems, however, almost all effective solution approaches are heuristic for real-world applications (and so is ours!), and thus do not guarantee optimality. De Werra (1985) reviewed some basic elements

and models for timetabling problems based on graph and network methods. Carter (1986) presented a chronological survey on successful exam timetabling applications that uses various graph coloring heuristics which were popular at the time. He also provided a few guidelines for practitioners on selecting and/or designing a timetabling algorithm. No well-known heuristic has been proven to dominate the large number of techniques previously available, since each timetabling problem in practice is highly institution-specific and there are no standard data and comparison rules. In these heuristics, the consideration of getting a conflict-free timetable has much higher priority than any other secondary constraint. Schaerf (1999) conducted a more recent survey on various formulations, techniques and algorithms. Several possible future research directions were also discussed. This survey focused especially on metaheuristics (e.g., genetic algorithms, tabu search, simulated annealing, and constraint satisfaction). Other important recent timetabling surveys can be found, e.g. in Burke *et al.* (1996)(a), Carter *et al.* (1996), Burke and Petrovic (2002), and Qu *et al.* (2009). There is also a webpage[2] provided by the School of CSiT, University of Nottingham on exam timetabling bibliography since 1996. Mathematical programming methods only occupied a small portion of the surveys, and were only briefly mentioned in passing.

Compared to other technical approaches, integer programming techniques have not been widely reported in timetabling surveys. Before 1990, it seems that IP techniques have only been used for small scheduling problems. Most of them are reported for school/course timetabling, not for exam timetabling. Lawrie (1969) described an integer programming formulation for school timetabling, which was solved by an *ad hoc* procedure with rounding and enumeration strategy. Shih and Sullivan (1977) formulated a multi-term course scheduling problem for college faculty members via 0-1 programming, in which the objective was centered upon faculty preferences and other academic goals, rather than on minimizing the number of student conflicts. Limited by the computational capacity at the time, they only reported results for scheduling 9-10 courses by 5 instructors for up to two terms. Realizing that only small problems (up to a maximum of 20 students) could be solved by Branch and Bound, Frieze and Yadegar (1981) described a Lagrangean relaxation based algorithm on a 3-dimensional assignment problem with applications in scheduling a teaching practice, in which they used a bilinear decomposition method to transform a Lagrangean solution into a good feasible solution. Tripathy (1984) proposed Lagrangean relaxation formulations for course scheduling, solved by either a sequencing heuristic or a subgradient method. A grouping technique was used for students and rooms, so that the size of the problem could be substantially reduced before actually solving the problem. The special form might not be suitable for other practical timetabling problems. Ferland and Roy (1985) described a mathematical programming approach that involves solving two assignment subproblems sequentially, and for each subproblem, they tried to transform it to a relaxed version of an equivalent 0-1 quadratic assignment problem that could be solved by using a methodology similar to that proposed by Carlson and

[2] http://www.asap.cs.nott.ac.uk/resources/ETTPbibliography.shtml

Nemhauser (1966). Arani *et al.* (1988) presented a Lagrangean relaxation approach for an integer program with the single objective of minimizing the number of students with two or more exams per day at the State University of New York at Buffalo. Aubin and Ferland (1989) presented a nonlinear integer formulation with variables in independent linear constraints, therefore allowing the course timetabling problem to decompose into two separate components: a master timetabling subproblem and a student sectioning subproblem. The objective was to minimize the number of conflicts caused by simultaneously assigning lectures taken by the same students or offered by the same instructor. They proposed an iterative procedure based on an exchange process between the two subproblems; the subproblems, however, still had nonlinear objectives and were not easy to solve exactly. Birbas *et al.* (1997) presented a 0-1 integer programming model (with the objective of minimizing a cost function based on certain quality rules) for finding a weekly schedule in Greek high schools. The survey of Schaerf (1999) gave a general integer programming formulation (with an open objective function as actual objectives are highly variable in practice) of the exam timetabling problem by using cliques for the non-conflict student constraints. Dimopoulou and Miliotis (2001) described the design and implementation of a PC-based computer system to aid the construction of a combined university course-exam timetable, which used an integer programming (IP) model to assign courses to time slots and rooms. Daskalaki and Birbas (2005) provided an integer programming formulation of the class-teacher problem and solved it with a two-stage relaxation procedure, which provided significantly reduced computation times compared to a single stage approach. Avella and Vasilev (2005) studied the polyhedral structure of course timetabling problems to provide effective classes of cutting planes for using a set-packing based formulation that used clique and lifted odd-hole inequalities. Boland *et al.* (2008) focused on the course blocking and population problem for school timetabling, which demonstrated that integer linear programming approaches can solve the problem very quickly for a local high school.

2.2 Description of the real problem at USMA

There are about 4000 cadets at the USMA, West Point, with roughly 1000 students per class year. Each cadet's daily activities are a carefully regimented balance of academic, military, and physical development. Scheduling, as well as most other activities at USMA, follows a *strict* catalogue of business rules.

The academic program at West Point is uniquely designed around the requirements that all students must complete in four years, making up a total of eight academic semesters or terms. Each cadet enrolls in a set of five to eight academic courses during each of the eight terms. Classes are small, typically consisting of 12 to 18 cadets. All cadets take their exams for their various courses at the end of a term, which usually falls in the third week of May for the Spring term and the week before Christmas for the Fall term. The academic department indicates which courses will have term-end exams (*TEE*). In the *TEE* week, roughly 250 courses with a total of about 20,000 student-exams combinations have to be scheduled. There are six

exam days in all (usually from Monday through Saturday) with at most only two exam slots available per day - morning (7:35-11:05) and afternoon (15:00-18:30).

The exam timetabling problem at USMA involves the allocation of examinations to a limited number of available time slots so that each student is assigned to no more than one exam at a time and that additional specific constraints are satisfied. A feasible conflict-free solution does not usually exist. When conflicts are inevitable, the current policy is to offer additional exams for courses that have conflicts, scheduled within the same six-day time span. If an additional offering is placed prior to the main exam (also called primary), it is called a "*makeahead*", if placed after the primary it is called a "*makeup*'. Although academic departments prefer makeups to makeaheads, the scheduler does not distinguish between the two, and throughout the paper we will use the term "*makeup*" to refer to either type of additional exam offering.

Some course-timetabling applications have actually considered the issue of student sectioning [Muller and Murray (2008)][Beyrouthy *et al.* (2008)], in which students must be assigned to particular sections of courses they request. This is somewhat similar to our *TEE* problem, in the sense that one course can have multiple exam sessions (primary and makeup(s)). The *TEE* problem however differs from student sectioning in the literature, in that the number of exam sessions for each course is a priori unknown, and the enrolment for a primary session must be at least a certain percentage of the total enrolment for the course. In addition, the total number of exam sessions is the objective to be minimized, while in other exam timetabling applications the objective is either to minimize a weighted preference cost for students, or the number of time conflicts or the number of students taking more than one exam in the same exam time slot [Carter *et al.* (1996)], or to maximize the students' study time [Bullnheimer (1998)].

As far as we know, resolving conflicts by using makeups scheduled during the exam period has not been considered in the exam timetabling literature. To deal with this, instead of defining a timetable for each exam, the *TEE* system must offer a separate exam schedule for each cadet. The basic exam timetable must therefore be a three dimensional relationship, (cadet, exam, time slot), requiring a large number of binary variables because of the extra dimension corresponding to a large number of cadets. It is more complicated than an (exam, time slot) relationship, as in normal educational timetabling problems.

As previously indicated, good schedules are those that minimize the number of makeup exams. Also important but less critical are goals such as that of maximizing instructor preferences and cadet satisfaction which can be achieved, for example, by scheduling particular courses apart from each other, or by scheduling for each examinee no more than three exams in a row.

Prior to reviewing the details of the problem, we introduce some terminology: a "plebe" is a freshman at a military or naval academy. Second, third, and fourth-year students are called "yearling", "cow" and

"firsties" respectively. As with many other real-world timetabling applications, the *TEE* problem is enriched by complex constraints rooted in the vast catalogue of business rules that apply at USMA. The following list of requirements provides a general introduction to some of the terminology frequently used at USMA:

- Hard constraints: Constraints that must never be violated. (The terminology is taken from the traditional timetabling literature.)
  - Assignment: Every course should have an exam scheduled; and each cadet's exam request must be assigned to an exam offering (either primary or a makeup).
  - Non-conflict: No cadet can take more than one exam within the same time slot.
- Soft constraints: Quality schedule - according to the relative importance of preference.
  - Fixed-courses: The primaries of some courses should as much as possible be scheduled in particular time slots (e.g., special needs). This constraint allows the human scheduler to fix some part of the exam timetable.
  - Prohibited-courses: If possible, one should avoid scheduling some courses in some specific exam time slots.
  - Inclusive-cumulative: A group of particular course exams should preferably be scheduled together (e.g., for a basic level course and its more advanced counterpart, primary exams are preferably scheduled together).
  - Exclusive-disjunctive: Exams of certain courses should preferably not be scheduled together (e.g. courses taught by the same instructor who has to supervise the two exams and be available at both to answer questions).
  - Plebe constraint: A plebe should only take one exam per day, preferably in the morning.
  - Senior constraint: Senior courses should not be scheduled on the last day, i.e., seniors should finish all their exams by day 5 of the *TEE* week.
  - Makeup constraint:
    - No Makeups: Some exam courses should not offer a makeup (e.g. core courses with a large enrollment);
    - Primary Enrollment: A makeup exam should only be taken by a small percentage of the total number of students enrolled in a course, i.e., the enrollment of the makeup should be substantially lower than the enrollment in the primary.

Another rule prohibits a cadet from taking more than three exams in a row. Although it can easily be formulated (see Appendix), based on historic schedules and current operations, this constraint has never been

enforced at West Point and has been largely disregarded thus far. Finally note that exams are organized in such a way that room capacity is not an issue at USMA.

Again, it is important to remember that because of the small class sizes, a highly dynamic curriculum and strict business rules, the *conflict density* at USMA is comparatively high (see "Data Set and Experiments" in Section 5). This makes the course scheduling and exam timetabling particularly challenging. Before the automated *TEE* scheduling system was built, the exam timetabling work was mainly done by a human scheduler based on human expertise. The exam scheduling task typically took four weeks, and resulted in a large number of makeups (more than 90) necessary to resolve the multiple scheduling conflicts. In addition, only a partial schedule was typically derived, and this was rarely completely satisfactory. Popular commercial software was tried, but this failed to meet the special needs and strict rules of USMA. Clearly, a new approach was needed.

Seven years after its implementation, human expertise is still a critical component of the *TEE* scheduler. As described in the following sections, infeasibility is one of the problems the automated system must deal with. A violated constraint usually means that a degree of manual "post scheduling" will be required, following different rules to satisfy the constraint. For example, among the soft constraints, the first two, "fixed" and "prohibited", should allow the human scheduler to take limited or full control over the scheduling process. From the scheduler's point of view, this represents a safety mechanism of sorts. Practitioners do not really trust automated scheduling systems, partly because in the end, the responsibility for the schedule rests with them. In order to ease the introduction of an automated scheduler, the software/model approach has to allow for "overwrites and fixings" by the human schedulers. At the extreme, they can still build the schedule completely by hand. The experience with the *TEE* scheduler and other scheduling applications at USMA shows that these overwrites and fixes are used significantly at the beginning after the introduction of the system but tend to become less and less important as the trust in the scheduling system grows over time.

2.3 Notation

We will make use of the following indices, sets, and parameters:

Indices

c	cadets
$r, r_1, r_2, ...$	exam-course
$p, p_1, ...$	exam time slots
d	days
m	exam sessions (either primary exam, or first makeup, or second makeup[3], etc.)

[3] We allow a second makeup offering, or even more, since this makes the problem more general.

Sets

C	set of available cadet, $c \in C$
R	set of available exam-course, $r \in R$, i.e., those courses that require an exam, $r \in R$
P	set of active time slots for exams, $p \in P$
D	set of active days for exams, $d \in D$
CR	set of (c, r) such that cadet c must take an exam for course r, $(c, r) \in CR \subset C \times R$
$R(c)$	set of courses that require exams by cadet c, $R(c) \subset R$
$C(r)$	set of cadets that require an exam for course r, $C(r) \subset C$
C^{plebe}	set of plebe cadets, $C^{plebe} \subset C$
RP^f	set of (r, p) that exam-course r must be fixed in time slot p, $(r, p) \in RP^f \subset R \times P$
RP^{prohib}	set of (r, p) that are prohibited in the schedule, $(r, p) \in RP^{prohib} \subset R \times P$
$P(d)$	set of active time slots in the same day d
$EXCL$	set of exclusive exam-courses $(r_1, r_2) \subset R \times R$ which can not be scheduled at the same time
$INCL$	set of inclusive exam-courses $(r_1, r_2) \subset R \times R$ which must be scheduled at the same time
R^{no_makeup}	set of exam-courses that allow no makeup, $R^{no\_makeup} \subset R$

The system used previously at USMA did some pre-scheduling even if not all student/exam assignments, courses, etc., were entered in the system. Those that were already there were considered "available". "Active" time slots/days are those available for scheduling. During the period of system development, as we were able to reduce by more than 50% the number of makeup exams compared with the previous human-produced schedules, the USMA at one point tried to reduce the number of examination days from 6 to 5. Then they needed a way to specify which periods (and consequently which days) were "active" for scheduling. At the very least, they wanted the capability of seeing by how much the number of makeups would increase if the number of time slots was reduced from 12 to 10.

Parameters

$p^{last}(c)$	last available time slot for cadet c.
ε_r	total enrollment of course r
π_r	percentage requirement for primary exam of course r
K_{r_1,r_2}	number of students taking both courses r_1 and r_2

Let us explain some of the above notation. The time slots are ordered $(p^1, p^2, ...)$ and grouped in set $P(d)$ by each day $d \in D$. We must assign the exam for course $r \in R$ to an available time slot $p \in P$. Each cadet $c \in C$ must attend a certain set of exam courses given by $R(c)$, and must finish all his or her exams by the last available time slot $p^{last}(c)$. An exam for a course r can be offered several times to avoid time conflicts

for the cadets, but only one of these exam sessions can be designated as the primary exam. The primary exam session must have at least π_r percent (usually around 60%) of the total course enrollment ε_r.

3 *TEE* MODELS

We will first introduce two groups of decision variables that will be used throughout the paper. The first group consists of variables that come naturally to mind when trying to model the *TEE* timetabling problem. Let binary variable $Y(r,p)$ represent whether or not an exam course r is offered in time slot p. As explained before, exams for a given course r might be offered in multiple time slots to resolve conflicts, but only one of these exam sessions can be designated as the primary exam. Let binary variable $W(r,p)$ represent whether or not the primary exam for course r is scheduled in time slot p.

Finally, for each cadet c, there might be multiple exam-sessions offered for a course r that he/she is taking, so let binary variable $Z(c,r,p)$ represent whether or not cadet c is assigned to an exam for course r scheduled in time slot p.

Linear 0-1 programming model.

We initially formulated the *TEE* timetabling problem as a pure 0-1 integer linear programming problem in term of variables $Z(c,r,p)$, $Y(r,p)$ and $W(r,p)$, where one minimizes the total number of makeup exams. We will refer to it as the big $(\mathcal{L}-\mathcal{TEE})$ model through the paper (see Appendix for details.) The big $(\mathcal{L}-\mathcal{TEE})$ model is useful as a catalogue of constraints for the *TEE* problem, but we are not analyzing it further, as it is not an effective formulation for the problem. Due to the large enrollments, it has more than 200,000 binary variables. In addition, with all constraints enforced, it is likely to be infeasible.

Bilinear 0-1 programming model.

To overcome the complexity and magnitude of the big $(\mathcal{L}-\mathcal{TEE})$ model, we reformulate it with different decision variables. The advantage of this reformulation (which we later refer to "*variable bilinearization and decomposition*") is algorithmic. By introducing an extra dimension, namely the session number, one can use fewer decision variables that still contain the same information, but in a different, decomposed way.

First, for every course, we introduce a new index m to represent the m^{th} exam session, and define the set $M=\{1,2,3...\}$ of these session values. Then we introduce a second group of binary variables:

$X_1(c,r,m)$ binary variable; equals to 1 if cadet $c \in C$ is scheduled in exam $r \in R$ at its m-th session.

$X_2(r,m,p)$ binary variable; equals to 1 if the m-th session of exam $r \in R$ is scheduled in time slot $p \in P$

We can now decompose $Z(c,r,p)$ and replace it by the newly introduced variables via the following equation:

$$Z(c,r,p) = \sum_{m \in M} X_1(c,r,m) X_2(r,m,p), \qquad \forall (c,r) \in CR \subset C \times R, \forall p \in P \tag{1}$$

This implies that the original timetabling decision can be split into two main scheduling components: 1) Make a decision based on $X_1(c,r,m)$; that is, assign each individual cadet c who needs to take an exam for course r to one exam session (r,m); 2) Make a decision based on $X_2(r,m,p)$; that is, allocate each exam session (r,m) to an available time slot p. Notice that, in general, $|M|$ is much smaller than $|P|$ (where $|M|$ and $|P|$ represent the corresponding cardinality of set M and P). Therefore there will be fewer binary variables when using the new variable definition: indeed there will be roughly 9,000 variables $X_2(r,m,p)$ and fewer than 40,000 variables $X_1(c,r,m)$ vs. 200,000 variables $Z(c,r,p)$. These are actual numbers of variables, involving only meaningful index combinations, ignoring for instance exam sessions (r,m) for a cadet c that is not registered for course r. But we will see later that there is another algorithmic advantage to using the new variables.

From this point onwards, we shall use $|\cdot|$ to represent the cardinality of a set. For example, $|R|$ is the cardinality of set R, i.e., the number of total courses that require exams.

Using the decomposition equation (1), with the substitution of variable $Z(c,r,p)$ from the big $(\mathcal{L}-\mathcal{TEE})$ model (as in the appendix), we can reformulate the exam timetabling problem as a 0-1 programming model with fewer binary variables but with non-linear constraints:

$(\mathcal{NL}-\mathcal{TEE})$

$$Min \sum_{r \in R, p \in P} Y(r,p) - |R|$$

s.t.

$$\sum_{p \in P} \sum_{m \in M} X_1(c,r,m) X_2(r,m,p) = 1, \qquad \forall (c,r) \in CR \subset C \times R \tag{2}$$

$$\sum_{r \in R(c)} \sum_{m \in M} X_1(c,r,m) X_2(r,m,p) \leq 1, \qquad \forall p \in P, c \in C \tag{3}$$

$$\sum_{c \in C(r)} \sum_{m \in M} X_1(c,r,m) X_2(r,m,p) \leq \varepsilon_r Y(r,p), \qquad \forall r \in R, \forall p \in P \tag{4}$$

$$\sum_{c \in C(r)} \sum_{m \in M} X_1(c,r,m) X_2(r,m,p) \geq \pi_r \varepsilon_r W(r,p), \qquad \forall r \in R, p \in P \tag{5}$$

$$\sum_{p \in P} W(r,p) = 1, \qquad \forall r \in R \tag{6}$$

$$Y(r_1,p) + Y(r_2,p) \leq 1, \qquad \forall (r_1,r_2) \in EXCL \subset R \times R, \forall p \in P \tag{7}$$

$$W(r_1,p) = W(r_2,p), \qquad \forall (r_1,r_2) \in INCL \subset R \times R, , \forall p \in P \tag{8}$$

$$\sum_{p \in P} Y(r,p) = 1, \qquad \forall r \in R^{no\_makeup} \tag{9}$$

$$\sum_{p \in P(d)} \sum_{r \in R(c)} \sum_{m \in M} X_1(c,r,m) X_2(r,m,p) \leq 1, \qquad \forall d \in D, c \in C^{plebe} \tag{10}$$

$$W(r,p) = 1, \qquad \forall (r,p) \in RP^f \tag{11}$$

$$Y(r,p) = 0, \qquad \forall (r,p) \in RP^{prohib} \tag{12}$$

$$\sum_{m \in M} X_1(c,r,m) X_2(r,m,p) = 0, \qquad \forall (c,r) \in CR, p > p^{last}(c) \tag{13}$$

We now explain briefly the meaning of the various parts of the model. The objective is to minimize the number of makeup exams, since one knows that there must be at least as many exams as courses, i.e., $|R|$.

The interpretation of the constraints is as follows. (2) the "cadet assignment" constraint: a cadet must have one exam for each course he/she is taking. (3) the "no-conflict" constraint: there can be no more than one exam for a cadet in one time slot. (4) the "course opening" constraint: one can only assign a cadet to an existing (i.e., scheduled) exam. (5) the "primary enrollment" constraint: if the primary exam for course r is scheduled in time slot p, then at least a certain proportion of the total number of students enrolled for that course should take that exam. (6) the "one-primary" constraint: there is exactly one primary exam scheduled per course. (7) the "exclusiveness" constraint: for some *exclusive* exam pairs (r_1, r_2), only one of each pair should be scheduled in a given time slot. (8) the "inclusiveness" constraint: for some *inclusive* exam pairs (r_1, r_2), the primary exams should be scheduled in the same time slot. (9) the "no makeup" constraint: for certain courses, there should be no makeup exam. (10) the "plebe" constraint: a plebe should take at most one exam per day. (11) the "fixed exam" constraint: certain exams have a known imposed schedule. (12) the "prohibited exam" constraint: certain exams should not be scheduled in certain time slots. (13) the "completion" constraint: each cadet should finish all his or her exams by his/her last available time slot $p^{last}(c)$. In addition, all variables must be 0-1.

The proposed model ($\mathcal{NL} - \mathcal{TEE}$) is not a "directly" useful choice at first glance, as it is a nonlinear integer model. Its objective function, however, is linear and all the constraints are either bilinear or linear. In addition, the term based on $Z(c,r,p) = \sum_{m \in M} X_1(c,r,m) X_2(r,m,p)$ is bilinear. This implies that if either the X_1 or X_2 variables are fixed at binary values, the resulting sub-problem will be linear. This is the main advantage of the reformulation, aside from a reduced number of 0-1 variables. Indeed it is not difficult to generate an initial exam timetable using some greedy heuristic method. Based on this, we can generate a feasible solution for either X_1 or X_2. An iterative solution improvement heuristic can then be applied based on the two resulting *linear* sub-models derived from ($\mathcal{NL} - \mathcal{TEE}$). We will discuss the details of this approach in the following section.

4 THE *TEE* SCHEDULER SOLUTION APPROACH

The *TEE* Scheduler algorithm consists of four phases:

1. Data Reading and Error Checking
2. Creating an Initial Solution
3. Improving the Solution
4. Output and Report Generation

The second and third phases of the overall algorithm represent the core of the scheduling algorithm and will be described in detail. Though no sophisticated algorithmic techniques are deployed in the first and fourth phase, the design and proper implementation of the two phases are critical for the success of the overall method.

Although application and model developers may have agreed on a specific interface, experience shows that data with inconsistencies will often be passed on to the algorithm. There are simple inconsistencies (e.g. negative numbers where positive numbers are expected). However, what ought to be of greater concern are inconsistencies that result from combining different data blocks and which are usually difficult to detect (e.g. a case where we have two courses which are not allowed to have make-up exams and yet are required to be fixed in the same time slot and have to be taken by common cadets). The error checking phase of the *TEE* scheduler consists of a number of data queries (accumulated over time) to detect logical problems in the data. In cases where these queries detect a problem, the problem is reported back to the user. Otherwise the core algorithm will cope with the data set that is passed to produce a solution.

In the second phase, a greedy heuristic places the primaries of all exam courses into available time slots. The resulting conflicts are resolved by first placing makeups. If conflicts remain and adding makeups fails to improve the number of conflicts, higher order makeups are added to the schedule. The algorithm will try to satisfy all given constraints and business rules but is allowed to violate them at user specified cost. The clear emphasis in the construction of the first schedule is the minimization of "soft constraint" violations.

The third phase uses the results from the second phase as input. The initial timetable together with the set of violated constraints is passed on to the improver module. In this module we cannot violate constraint at the cost of satisfying other constraints or reducing the number of makeups. The reduction of of makeup exams and of violated constraints is solely based on rearranging primaries and makeups subject to all previously satisfied constraints.

In the fourth phase, the reporting module takes a schedule and creates exception reports for violation of business rules and constraints. Besides reporting and output generation, this module provides a tool for analyzing and comparing schedules from different sources.

4.1 Finding an initial solution (greedy heuristic)

The *TEE scheduler* builds an initial solution mainly in two sub-procedures: 1) placing the primaries, and 2) resolving conflicts by adding makeups. Dealing only with primaries is advantageous in that the individual cadet schedule is determined by the schedule of the courses. Moreover, for any two courses, the number of conflicts resulting from placing them into one time slot is the number of cadets who take both courses.

Placing the primaries. First, the *TEE scheduler* tries to reduce the complexity of the problem by building clusters or buckets of courses. To process this, courses will be classified by graduate year, i.e. a course will be assigned to the year with the largest enrollment (e.g. CH385 has 59 yearlings, 24 cows, and 5 firsties enrolled, so the *TEE scheduler* classifies it as a yearling course.). We create buckets of courses of the same year that do not have cadets in common and are hence likely to be scheduled together. At the same time, the inclusive and exclusive constraints are obeyed (for courses of the same year). The *TEE scheduler* does not build more buckets per year than the number of exam time slots. For the plebes, the scheduler even builds as few as half as many buckets as there are time slots, because of the "one exam a day" constraint. Courses with fixed primaries are not assigned to buckets. Due to the limited number of buckets, the *TEE scheduler* might have to leave some courses unassigned. With the exception of conflicts due to the inclusive constraint, the courses in a bucket would not have cadets in common. Hence, instead of scheduling primaries of individual courses, we can schedule buckets at the cost of a potentially larger number of conflicts when placing two buckets together.

Second, the courses with fixed primaries are then placed in their assigned time slots and next the *TEE scheduler* assigns the grouped buckets and the remaining unassigned courses into the exam time slots by deploying optimization models. In this research, we formulate a 0-1 programming model with the single objective of minimizing the resulting cadet conflicts. Such type of timetabling formulation is common in the timetabling literature [Carter *et al.* (1996)]. Without going into great detail, we still want to present the main constituents of the model. The formulation has two types of binary variables: $w(r, p)$, which determines the decision of whether or not to place a course (or a bucket of courses) r into time slot p; and $v(r_1, r_2)$, which represents whether two courses (r_1, r_2) are placed together. The relationship between the two types of variables can be set in the following constraints: $v(r_1, r_2) \geq w(r_1, p) + w(r_1, p) - 1(\forall p)$. Notice that by only dealing with the primaries, the cadet schedules can be determined by the placement of the courses. Hence, for any two unplaced courses (or buckets of courses) (r_1, r_2), we can derive the number of resulting

conflicts K_{r_1,r_2} if we place the courses together in any time slot. The objective function $\sum_{r_1,r_2} K_{r_1,r_2} v(r_1, r_2)$ is therefore linear.

A model that simultaneously places the buckets and courses of all graduate years and minimizes the number of conflicts subject to the constraints discussed earlier would be desirable. Experiments have shown, however, that such a model cannot be solved reliably and efficiently. Hence in real application, the *TEE scheduler* schedules the buckets and courses year by year, and each sub-problem can be solved quickly via 0-1 programming.

Resolving conflicts by adding makeups. After placing all primaries, the *TEE scheduler* tries to eliminate the conflicts by placing makeups. For each time slot, given the set of previously-scheduled primaries, the minimum set of courses that must offer a makeup is determined by solving a vertex cover problem, i.e., by determining a minimal set of nodes that are incident with all edges in the following graph: the nodes are exam courses (primaries), there is an edge between any pair of exam nodes if and only if they are in conflict (having at least one student enrolled in both courses). As such, the problem can be built into a 0-1 linear programming model which defines a group of binary variables to represent whether a scheduled primary exam should offer a makeup, and the objective is to minimize the total number of such makeups subject to a group of set-covering constraints to cover all the conflicts. Unfortunately, these makeups have to be placed in the 6-day exam period and might create additional new conflicts with the courses scheduled in other time slots. However, placing the makeups following a simple greedy approach and repeating the process of identifying and placing makeups produces an initial schedule without conflicts. While the primaries are fixed to their original position in this iterative process, the makeups will be repositioned to their best positions in each iteration.

Throughout the phase of constructing an initial solution to the *TEE* problem, the algorithm is allowed to relax constraints at given costs. For example, placing two exclusive courses into one time slot creates an additional penalty besides the conflicts. Penalizing constraint violations usually places the burden of ranking constraints on the user. Unlike other scheduling applications, most of the *TEE* constraints are treated as hard constraints by the user and the algorithm does not have to choose between violations of different constraints. A violated constraint in the final schedule triggers an exception report and usually results in a rerun with modified data or a human post-scheduling in order to overcome the violation.

More specifically, the emphasis when finding an initial solution is to look for a schedule that fulfills as many of the "soft" constraints as possible. There are some switches for turning off some "soft" constraints. For example, in instance USMA(3) (see "Data Set and Experiments"in Section 5) there was no emphasis on satisfying the "one exam a day for plebes" constraint. Similarly, we never enforced the "no more than three exams in a row" "soft" constraint. In next phase, the improver module never relaxed a satisfied "soft"

constraint, so the satisfied "soft" constraints all became hard constraints for the improver. The only trade-off in the improver was between satisfying a violated constraint and reducing the number of make-ups. Since the violated constraints after finding an initial schedule were usually few and reviewed by a human, the emphasis in the improver lies clearly in minimizing the number of makeups.

4.2 Solution improvement (*Improver*)

According to our experiments, the initial timetabling solution generated from the previous greedy heuristics is still likely to have a large number of makeups (see Section 6 Table 3), and the automated *TEE* scheduler might end up with some non local-optimal solutions or some "silly" solutions which can be easily improved by a human scheduler. These are indeed undesirable. Therefore an automated solution-improving procedure, an "*improver*", is needed after producing the initial solution. The *improver* module needs to rearrange primaries and makeups in order to improve the solution in terms of reducing the number of makeup exams and violated constraints. Rather than using the traditional graph coloring technique and recent popular local improvement heuristic methods such as genetic algorithm, Tabu search, etc., we deploy integer programming techniques to improve the timetabling solutions.

It should be mentioned that we even tried to restart from some feasible solution in the hope that the big $(\mathcal{L}-\mathcal{TEE})$ model might be able to improve the solution quality. Unfortunately, the big $(\mathcal{L}-\mathcal{TEE})$ model is also too weak to be used as a trial "improver" (see Section 6). Therefore we chose to adopt another iterative solution improving strategy based on the proposed bilinear model $(\mathcal{NL}-\mathcal{TEE})$ in Section 3.

In model $(\mathcal{NL}-\mathcal{TEE})$, if either the X_1 or X_2 variables are fixed at 0 or 1, the resulting sub-problem is linear. Given an initial timetabling solution, we could utilize the information to fix X_1 or X_2 and break down the original problem into the following two manageable stages:

- Part(i): a cadet-exam session scheduling with solution $X_1(c,r,m)$.
- Part(ii): an exam session timetabling with solution $X_2(r,m,p)$;

In the *TEE* application, Part (i) usually results in fewer than 40,000 binary variables, and Part (ii) fewer than 50,000. Variables $Y(r,p)$ and $W(r,p)$ are binary in both Part (i) and Part (ii), and this leads to an iterative solution approach, much more economical and potentially algorithmically effective.

The decomposition is non-hierarchical, i.e., it works both ways. If we know the exam-session timetable (i.e. we know the values of $X_2(r,m,p)$), we solve a linear integer programming model for assigning optimally the cadets to exam sessions ($X_1(c,r,m)$). Conversely, if we know the cadet-exam session schedule (i.e. we

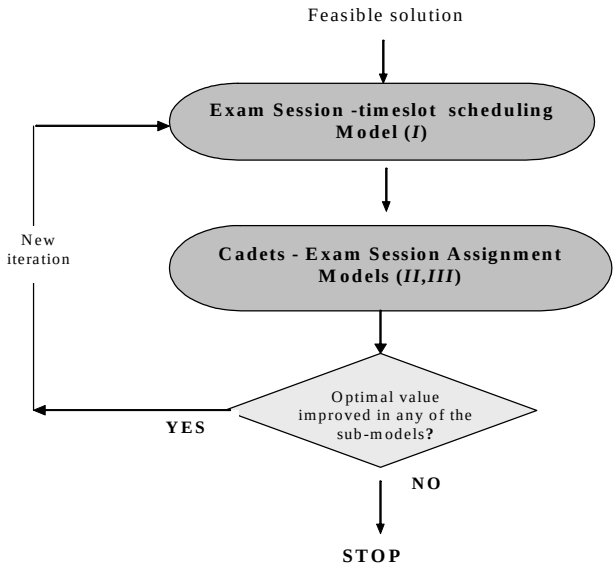

Figure 1: Simplified layout of the improving procedure

know the values of $X_1(c, r, m)$), we solve a linear integer programming model to determine the optimal timetable for courses ($X_2(r, m, p)$).

Note that this bilinear-structured formulation and the non-hierarchical property not only allow us to decompose the original problem into manageable stages, but also enable us to solve the subproblems of Part (i) and (ii) iteratively until no further improvement can be obtained. The solutions, if optimal in the sub-problems, will be at least locally optimal for the original problem. The local property of solutions, although unimportant from a global perspective, is a critical credibility factor in the actual implementation. The human expert will indeed be quick to dismiss good solutions if there are local flaws since they tend to judge the quality of a solution by its local characteristics.

The whole improving procedure is illustrated in Figure 1 and a more detailed description can be found in Box 1. Notice that there are some differences between the algorithm used in the actual application and the earlier discussion about the model reformulation. In our earlier discussion, the algorithm had two parts with one sub-model for each. In the actual implementation, although we still keep these two main parts, denoted I, II in Box 1, we introduce a third step part (or sub-model, denoted III on Box 1), to potentially allow further improvement at the next iteration. Indeed, in addition to minimizing the total number of makeups, we also maximize the number of cadets in primary exams, i.e., we try to move cadets out of makeups, and this may have a better chance to produce an improved timetable when we come back to part I. As the formulation of these three sub-models can be easily derived from the above ($\mathcal{NL} - \mathcal{TEE}$) model based on

the earlier discussion of the decomposition scheme, we are skipping the details.

More importantly, all the sub-models (I, II, III) given in the solution improving procedure (Figure 1 and Box 1) can be quickly solved to optimality by using current integer programming software (see "Computational Results" in Section 6). This allows us to effectively use an iterative procedure to reach the final solution when each single part is optimal, given knowledge about the other parts. In addition, notice that each of the sub-models is still relatively large with ten of thousands of binary decision variables. Therefore, the advantage of using mathematical programming can be fully realized and local optimality can be guaranteed within a large range when the final solution is obtained. The solutions are "good" global solutions and are locally "optimal", and this makes further improvements by human inspection essentially impossible.

We now summarize the procedure of our *variable bilinearization and decomposition* technique. The first step consists of disaggregating the decision variables, with the effect of introducing some nonlinearity in the model. The second step consists of splitting the resulting problem into several sub-problems, each taking care of a subset of the decisions in the following order: allocating time slots to exams, then assigning cadets to exams and finally moving cadets away from makeup exams to reduce the number of makeups. The solution process is iterative, i.e., one solves these sub-problems repeatedly as long as time permits, or until no further improvement can be achieved, and we summarize it in the following Box 1.

Box 1: Improvement Algorithm

- Step 0: Initialization
  - Read the initial solution (including $X_1(c, r, m)$)
  - Relax constraints that are violated in the initial solution
- Step 1: Sub-model (I)
  - Build sub-model (I) minimizing the total number of makeup sessions:
    - Objective: minimize $(\sum_{r \in R, p \in P} Y(r, p) - |R|)$
    - Constraints: equivalent to Model $(\mathcal{NL} - \mathcal{TEE})$ by fixing $X_1(c, r, m))$
  - Solve sub-model (I), get solution (including $X_2(r, m, p)$), update the exam-session timetable
  - Re-apply the relaxed constraints.
- Step 2: Sub-model (II)
  - Build sub-model (II) minimizing number of makeups assigning cadets to existing timetable::
    - Objective: minimize $(\sum_{r \in R, p \in P} Y(r, p) - |R|)$
    - Constraints: equivalent to Model $(\mathcal{NL} - \mathcal{TEE})$ by fixing $X_2(r, m, p)$
  - Solve sub-model (II)
  - If the total number of makeup sessions reduced
    - Update the exam-session timetable set;
    - For any course r with multiple exam sessions, assign a number $m = 1, 2, 3...$to each exam session in descending order of number of cadets assigned; update schedule;
    - Update $X_2(r, m, p)$.
- Step 3: Sub-model (III)
  - Build sub-model (II) minimizing the number of cadets in makeups:
    - Objective: minimize $\sum_{(c,r) \in CR, m > 1} X_1(c, r, m)$
    - Constraints: equivalent to Model $(\mathcal{NL} - \mathcal{TEE})$ by fixing $X_2(r, m, p)$ and $Y(r, p)$
  - Solve sub-model (III) and obtain new solution $X_1(c, r, m)$
  - If any of the optimal values of the three models has improved, go to Step 1.
- Step 4: Export solution. STOP

5 DATA SET AND EXPERIMENTAL DESIGN

The *TEE* scheduler was completed in 5 months, April-August 2001. We worked with 3 data sets that we have obtained from the USMA. The first one is from the spring term 2001. We refer to this as USMA(1). USMA(1) consists of 18937 individual exams in 226 courses. In late August 2001, we obtained an early instance of the fall term, referred to as USMA(2), that includes 18512 individual exams in 213 courses, and the finalized data of the same term USMA(3), where we are faced with 21176 individual exams in 253 courses and a reduced number of time slots of just 11, rather than 12.

Compared with other real-world applications, the West Point data instances are really not that large. There are about 4000 students and 250 courses involved. What makes the scheduling problem really hard is the very small number of available time slots for the 20,000 individual-exams (only around 11-12), and the high ***conflict densities***. The average conflict density is around 16%, and, for some particular courses, conflict densities could even be as high as 60-80%.

We will illustrate the above statement by analyzing the datasets from USMA together with other real-world application instances as follows. Assume $\overline{d}$ is the average conflict density, N is the number of exam courses, and T is the total number of time slots available. Suppose we randomly put all these exams into T time slots, then the probability of putting any two conflict courses into the same slot is $\frac{1}{T}$. Since, for every exam course, the average number of courses in conflict with it is $\overline{d} \times N$, the probability of having a conflict for each particular exam-course could be calculated as follows:

$$\begin{aligned} Probability(\text{conflict}) &= 1 - Probability(\text{no conflict with any other course}) \\ &\approx 1 - (1 - \frac{1}{T})^{\overline{d} \times N} \end{aligned} \tag{14}$$

which is a function of $\overline{d}, N$ and T. We consider it a reasonable measure of the difficulty of the scheduling problem, since the resultant timetable must be conflict-free.

The last column of Table 1[4] shows that $Probability$(conflict) for the 3 data sets of USMA (West Point) are the highest, being in the vicinity of 95%. The *TEE* project at USMA distinguishes itself from other similar projects by the difficulty of the conflicts it must handle. As far as we know, the idea of offering makeups to resolve conflicts, which have to be scheduled during the same exam time span, has not been considered in the literature.

[4]Most data information was obtained from Carter's papers [Carter (1986)] [Carter et al.(1996)]. (*) means the information is either not available or is just some estimated numbers.

URL (USMA data): http://www.gamsworld.org/minlp/apps/tee

URL (Carter's bechmark data): ftp://ftp.mie.utoronto.ca/pub/carter/testprob/

Table 1: Characteristics of the USMA data vs. other data in real-life applications

Instance	Number of Exam-courses N	Students	Enrollment	Available time slots T	Conflict Density $\overline{d}$	$\overline{d}\times N$	Probability (conflict) $1-(1-\frac{1}{T})^{\overline{d}\times N}$
USMA(1)	226	4020	18937	12	0.153	34.58	0.951
USMA(2)	213	3963	18512	12	0.162	34.24	0.949
USMA(3)	253	4166	21176	11	0.161	40.65	0.979
Car-S-91 (Carleton Univ., Ottawa)	682	18,419	55,552	51	0.13	88.7	0.83
Car-F-92 (Carleton Univ., Ottawa)	543	16,925	56,877	36	0.14	76	0.88
KFU-S-93 (King Fahd Univ., Dharan)	486	5,349	25,118	21	0.06	29.6	0.76
Nott (Nottingham Univ., UK)	800	7,896	34,265	23	0.03	24	0.66
Pur-S-93 (Purdue Univ., Indiana)	2419	30,032	120,690	30	0.03	72.6	0.91
TRE-S-92 (Trent Univ., Ontario)	261	4,360	1490	46	0.18	47	0.64
Univ. of Manchester, Eng 1968	1000*	6000*	NA*	30	0.015	15	0.399
Uta-S-92 (Univ. of Toronto)	622	21,267	58,981	37	0.13	80.9	0.89
Univ. Waterloo 1981	552	17000*	NA*	36	0.05	27.6	0.54
L'Ecole PolyTech de Montreal 1978	160	800*	12000*	24	0.23	36	0.78
Cedar Crest college 1981	84	750	NA*	12	0.29	24.7	0.88

6 COMPUTATIONAL RESULTS

6.1 Running the big model ($\mathcal{L}-\mathcal{TEE}$)

Two tests have been run for the big ($\mathcal{L}-\mathcal{TEE}$) model with GAMS/CPLEX 7.1 on a 1GHz Linux system in year 2001 when we first implemented our *TEE* scheduler approach. The first test ran from scratch for more than 24 hours, but we did not obtain any feasible solution. The second test ran for about 9 hours, starting from one of the best feasible solutions obtained by our *TEE* heuristic approach, but we obtained neither improvements nor good lower bounds.

In spite of great progress in integer programming technology in the last 7 years, the big ($\mathcal{L}-\mathcal{TEE}$) model still cannot be solved exactly. To verify this, we recently redid the computational experiment on the big ($\mathcal{L}-\mathcal{TEE}$) model with GAMS/CPLEX 11.1.1 on a 2GHz Windows system. We dropped some soft constraints that were not satisfied after applying the greedy algorithm (for example, the enrollment limit for primaries, etc.) and penalized violations of the "plebe constraint" with a small penalty in the objective. We did three runs for each of our three instances.

Run 1 (referred to as "CPLEX default"): run CPLEX with default option for 3 hours;

Run 2 (referred to as "CPLEX tuned"): run CPLEX with the options "varsel 4" (turn on inexpensive pseudo cost initialization), "cuts no" (disable cuts) and "mipemphasis 1" (emphasize integer feasibility) for

3 hours;

Run 3 (referred to as "CPLEX start"): run "CPLEX tuned" again as an "improver" for one hour starting from our current best *TEE* solutions.

Here is the summary of the new results. First, with CPLEX defaults, no solution was found for data USMA(1)&(3) and one solution was found with 488 makeups for USMA(2) (see column 2 in Table 2) and it is still producing useless cuts and initializing pseudocosts for branching (very similar to what happened seven years ago). With the tuned Cplex options, CPLEX produces solutions, in contrast to the runs which we did seven years ago. The results (see column 3 in Table 2) are significantly worse than what we got seven years ago using our TEE scheduler approach (see last column in Table 2), and not just by a few percent. For USMA(1) and USMA(3) CPLEX has twice as many makeups. For USMA(2) CPLEX still has 1.5 times as many makeups as we have. Finally, when CPLEX restarts from our current best solution, it does not improve much after one hour(see column 3 in Table 2) (although CPLEX eliminates one more makeup in USMA(1) when started with our current solution).

Table 2: Runs on the big (TEE) model in Year 2008

	Best Solution (number of *makeups* achieved)			
instances	CPLEX default (CPU=3×3600s)	CPLEX tuned (CPU=3×3600s)	CPLEX start (CPU=3600s)	*TEE* Scheduler Approach
USMA(1)	-	82	37	38
USMA(2)	488	76	49	49
USMA(3)	-	131	56	56

The result in the last column of Table 2 required less than 30 minutes in year 2001 at the time we implement our method as described in the following section. Thanks to MIP solver improvements and better hardware we produce such results nowadays in less than 5 minutes.

The literature overview provides a significant number of potential methods that could be tried on the *TEE* problem. We got permission to publish[5] an anonymized version of the three instances. The data instances and the big ($\mathcal{L} - \mathcal{TEE}$) are coded in the GAMS language but can be easily transferred in other environments.

6.2 Running the *TEE* Scheduler solution approach

Three instances may sound few, but it was extremely difficult to get hold of real data and we had to test the first versions with made-up data. Limited performance testing also means that certain performance issues

[5] http://www.gamsworld.org/minlp/apps/tee

might arise after deployment of the application. For that reason alone, the implementation of the algorithm requires, even after the prototype phase, a flexible environment such as a modeling system.

In order to test for reliability of the *TEE* scheduler and to provide information about the impact of the various constraints, we created variations of these 3 instances by relaxing particular constraints. One can easily guess the type of variation from the name of an instance. The number before the underscore indicates the specific data set from which the variant instance in question is derived. The following letter coding has been applied:

F **F**ixing requirement has been relaxed
E **E**xclusiveness requirement has been relaxed
I **I**nclusiveness requirement has been relaxed
P **P**rohibited requirement has been relaxed
M No-**M**akeup requirement has been relaxed

For example, the instance 1_FIP has been created from the original instance USMA(**1**) by relaxing the **f**ixing, **i**nclusiveness and **p**rohibited constraints. The following instances have been created and tested:

1, 1_F, 1_MP, 1_MPIE, 1_MPIEF

2, 2_F, 2_I, 2_IE, 2_IEP, 2_IEPM

3, 3_F, 3_FE, 3_FEI, 3_FEIM, 3_FI

The solution improving approach described in the previous section has been implemented in GAMS distribution 20.0 with an updated GAMS/CPLEX version 7.1. The computing platform was a Windows 2000 Server with an 800 MHz single CPU and 512 MB of core memory. The *TEE* scheduler was executed in default mode, i.e., the algorithm runs to create the initial solution and continues for another 10 minutes to improve on it. The test problem characteristics, running time (in seconds), number of makeups, and solution improvements are shown in Table 3.

There are several observations to be made about Table 3:

- The improver module could improve the initial solutions in the range of 20%-55%, by 11-73 makeups;
- The final solution still depends on the quality of the initial solution.

Figure 2 shows the typical behavior of the reduction in number of makeup sessions for the 3 USMA instances (two variants for each) within the 600 seconds time limit. Letting the improver heuristic run for longer than the default 10 minutes can help significantly, as shown in Figure 3, where the number of makeups gets reduced by more than 60. Figure 3 also shows the total number of makeups remaining after a certain amount of elapsed time. The illustrated test was done on the 3_F instance.

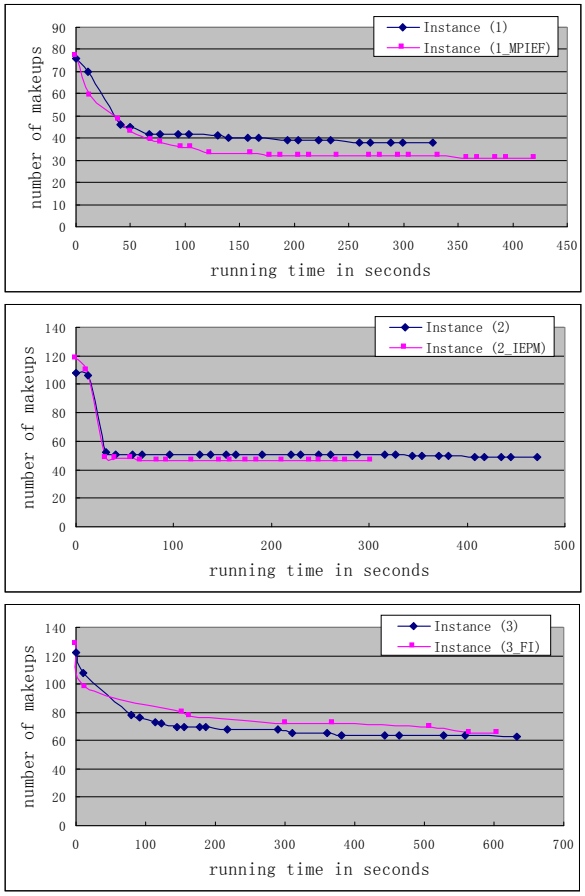

Figure 2: Running time of Improver on some USMA instances

Table 3: Comparison of the results from the greedy step and the improver step

	Initial Solution		Improver Procedure		%
Instance	CPU time	makeups	CPU time	makeups	Improved
1	276	57	327	38	33.33
1_F	359	78	621	36	53.85
1_M	238	55	418	34	38.18
1_MP	238	55	606	33	40.00
1_MPIE	198	49	478	32	34.69
1_MPIEF	170	68	420	31	54.41
2	223	60	472	49	18.33
2_F	190	69	303	31	55.07
2_I	165	61	399	45	26.23
2_IE	122	65	218	47	27.69
2_IEP	119	65	360	48	26.15
2_IEPM	179	57	302	46	19.30
3	634	104	632	63	39.42
3_F	508	122	622	67	45.08
3_FE	485	130	620	61	53.08
3_FEI	651	131	619	58	55.73
3_FEIM	556	128	626	61	52.34
3_FI	521	124	605	65	47.58

Comparison with the old manual approach. Note that a manually-scheduled timetable was built by a human scheduler for the test data USMA(1). The resulting schedule was however only partial and there were in fact a large number of makeups (around 90). Our *TEE* scheduler actually produced a complete timetable with only 31 makeups (see instance 1_MPIEF in Table 3), which is indeed a vast improvement.

Effectiveness of the improver module. From Table 3, we can see that the *TEE* improver module is quite effective in solution improvement, reducing the number of makeups by around 40% on average, given the default 10 minutes running time limit. Notice that although the exam timetabling problem is NP-hard, it is not difficult to obtain a timetabling solution by bending some soft constraints using greedy algorithms or other heuristic methods. If the initial greedy solution is not locally optimal, it can always be passed on to the improver module together with a set of violated soft constraints. This solution improvement approach may prove useful for other difficult scheduling tasks, where a hierarchy of decisions may lead to similar bilinear remodeling and separation of the problem into individually and sequentially solvable sub-problems.

The tests are reproducible on a similar type of machine. Slower or faster machines might give slightly different solutions, as some of the internal decisions made by the *TEE scheduler* are time-dependent (basically

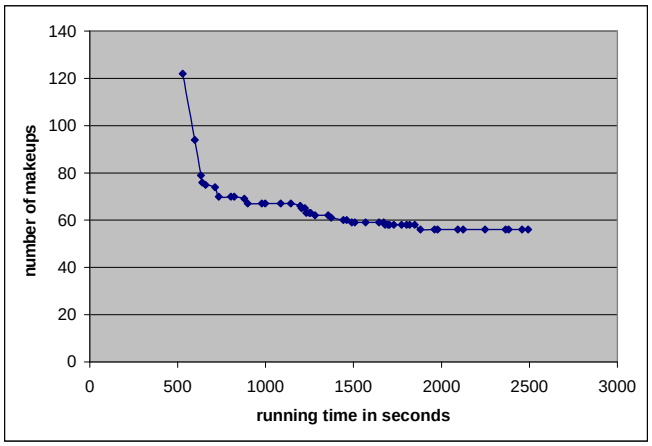

Figure 3: A typical instance run of *TEE scheduler*

the maximum time an individual optimization model is allowed to run), which might lead to a different solution path. However, results are expected to be in the same range as the results reported here.

From 2002 on, the *TEE* timetabling support system has proved to be very valuable in aiding human schedulers or more junior and less experienced staff in scheduling term end exams at West Point.

7 CONCLUSION

In year 2002, the United States Military Academy (USMA) in West Point began using integer-programming models to schedule the exam timetables for their cadets. Prior to that, scheduling was done manually for each academic term and usually required several weeks of work. In this paper, we present the *TEE* scheduling as a special exam timetabling problem, with the basic challenge of scheduling examinations and *their makeup*s over a very limited number of time periods, so as to avoid conflicts and to satisfy a number of flexible requirements. The problem is complex because of the many operating rules that have to be satisfied in an acceptable solution and of the dynamic nature of the scheduling environment. Solving directly a 0-1 linear integer programming formulation turned out to be impracticable because of its large size. We nevertheless managed to get satisfactory answers to this practical exam timetabling problem by designing an automated system that relies on heuristics and optimization models. The system first creates schedules by greedy heuristics that are then improved by a *variable bilinearization and decomposition* technique, taking advantage of the decomposability of the problem. The newly introduced concept of the *variable bilinearization and decomposition* technique allows the *improver* module to easily adapt and to improve the initial timetabling solutions. In addition, 0-1 programming can be used to optimize the final solution in a

relatively large neighborhood. The solutions obtained may not be globally optimal, yet they are satisfactory in the sense that each single part can be achieved to optimality given the knowledge of the other parts, and cannot be easily improved by inspection.

There are some salient features that distinguish our approach from those reported in the literature:

(a) Instead of maximizing instructors/students preferences, the objective function here attempts to minimize the total number of makeup exams. As far as we know, this has not been reported in the literature.

(b) In contrast with many other schools, colleges and universities, the exam timetabling task at USMA must schedule all the exams and makeups within a very short fixed length timetable (with only a dozen time slots available for about 20,000 individual exams), and not use any additional period. The resulting mixed-integer programming models are therefore very large and so far cannot be solved by the best available commercial software packages.

(c) We propose a reformulation that enables us to greatly reduce the problem size and to decompose the original large problem into more manageable subproblems. Large problems can now be handled by solving smaller subproblems using standard linear integer programming software.

(d) In practice, a schedule may need to be updated because of changes. An advantage of an integer linear programming approach is its flexibility. By using mathematical programming methods, it is easy to incorporate the change in the constraints. The automated system can thus adapt easily to the dynamic nature of the scheduling environment.

We have developed *TEE Scheduler*, an automated examination scheduling system suited to the needs of USMA in West Point. *TEE Scheduler* runs on a PC and is fast and user-friendly. It provides a choice of feasible schedules and solutions in which examinations are well spread out for most students. It handles requirements regarding the proximity of a student's exams, room or time availability, and other factors. The *TEE Scheduler* has been used with success for the last seven years.

We would also like to remind the reader that the optimization model described in this paper is just one of many components needed in the Academy's Management System. User interfaces that blend well with the history and culture of an organization, comprehensive and reliable databases and associated applications that collect and distribute related information, and a responsive support environment, are essential and critical to the success of our modeling work.

The automated system also relies on the knowledge and judgment of human schedulers. Interaction between the automated scheduler and the human schedulers is critical to the success of the *TEE* project. The computer-based system must allow the human scheduler to take limited or full control over the scheduling process. From the human schedulers' point of view, this acts as some kind of safety mechanism and helps to increase acceptance for the new application. Besides, it is very difficult to arrive at a perfect solution that

satisfies all rules and requests at West Point using only the automated scheduler. A violated constraint will typically involve some manual post-adjustments according to specific rules to eliminate constraint violations. The human scheduler will evaluate the resultant schedule based on the number of constraint violations and the number of makeups required to produce a final conflict-free schedule. It is our resulting strongly-held opinion that without the support and expertise of the human scheduler, most automated scheduling algorithms are likely to fail.

Acknowledgement 1 *We are very grateful to the anonymous referees and the editors for their insightful comments, which led to improvement of this paper.*

References

[1] Abramson, D. (1991). Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management Science, 37, 98-113.

[2] Abramson, D.A., Dang, H., and Krisnamoorthy, M. (1999). Simulated annealing cooling schedules for the school timetabling problem. Asia-Pacific Journal of Operational Research, 16, 1-22.

[3] Appleby, J.S., Blake, D.V., and Newman, E.A. (1961). Techniques for producing school timetables on a computer and their application to other scheduling problems. The Computer Journal, 3, 237-245.

[4] Arani, T., Karwan, M., and Lofti, V. (1988). A Lagrangian relaxation approach to solve the second phase of the exam scheduling problem. European Journal of Operational Research, 34, 372-383.

[5] Aubin, J., and Ferland, J.A. (1989). A large scale timetabling problem. Computers and Operations Research, 16(1), 67-77.

[6] Avella, P., and Vasilev, I. (2005). A computational study of a cutting plane algorithm for university course timetabling. Journal of Scheduling, 8, 497-514.

[7] Azevedo, F., and Barahona, P. (1994). Timetabling in constraint logic programming. Proceedings of 2nd World Congress on Expert Systems.

[8] Beyrouthy, C., Burke, E.K., Landa-Silva, D., McCollum, B., McMullan, P., and Parkes, A.J. (2008). Conflict inheritance in sectioning and space planning. Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22 August 2008.

[9] Birbas, T., Daskalaki, S., and Housos, E. (1997). Timetabling for Greek high schools. The Journal of the Operational Research Society, 48, 1191-1200.

[10] Boizumault, P., Delon, Y., and L.Peridy, L. (1996). Constraint logic programming for examination timetabling. Journal of Logic Programming, 26, 217-233.

[11] Boland, N., Hughes, B.D., Merlot, L.T.G., and Stuckey, P.J. (2008). New integer linear programming approaches for course timetabling. Computers and Operations Research, 35, 2209 – 2233.

[12] Bullnheimer, B. (1998). An examination scheduling model to maximize students' study time. Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science, 1408, 78-91 (Springer Berlin / Heidelberg).

[13] Burke, E.K., Elliman, D.G., P.H., F., and Weare, R.F. (1996). Examination timetabling in British universities - a survey. Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, 1153, 76-92 (Springer Berlin / Heidelberg).

[14] Burke, E.K., and Newall, J.P. (1999). A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation, 3(1), 63-74.

[15] Burke, E.K., Newall, J.P., and Weare, R.F. (1996). A memetic algorithm for university exam timetabling. Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, 1153, 241-250 (Springer Berlin / Heidelberg).

[16] Burke, E.K., and Petrovic, S. (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140, 266-280.

[17] Bush, R.N., Caffrey, J.G., Oakford, R.V., and Allen, D.W. (1961). Using machines to make the high school schedule. The School Review, 69(1), 48-59.

[18] Bussieck, M., Guignard, M., Meeraus, A., O'Brien, F., and Wang, S. (2001). Term end exam scheduling at United States Military Academy / West Point. OPIM Department Report 01-10-01, University of Pennsylvania.

[19] Carlson, R.C., and Nemhauser, G.L. (1966). Scheduling to minimize interaction cost. Operations Research., 14, 52-58.

[20] Carter, M.W. (1986). A survey of practical applications of examination timetabling algorithms. Operations Research, 34, 193-202.

[21] [Carter, Laporte and Chinneck (1994)] Carter, M.W., Laporte, G., and Chinneck, J.W. (1994). A general examination scheduling system. INTERFACES, 24(3), 109-120.

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

[23] Carter, M.W., and Tovey, C.A. (1992). When is the classroom assignment problem hard? Operations Research, 40 Supp. No.1, S28-S39.

[24] Colorni A., M. Dorigo & V. Maniezzo (1990). Genetic Algorithms: A New Approach to the Time-Table Problem. NATO ASI Series, Vol.F 82, Combinatorial Optimization, 235-239. (Springer-Verlag)

[25] Corne, D., Ross, P., and Fang, H.-L. (1994). Fast practical evolutionary timetabling. In T. Fogarty(Ed.), Evolutionary Computing: AISB Workshop 1994, Selected Papers, Lecture Notes in Computer Science, 865, 250-263 (Springer Verlag).

[26] [Csima and Gotlieb (1964)] Csima, J., and Gotlieb, C.C. (1964). Tests on a computer method for constructing school timetables. Communications of the ACM, 7(3), 160-163.

[27] Daskalaki, S., and Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. European Journal of Operational Research, 160, 106-120.

[28] Daskalaki, S., Birbas, T., and Housos, E. (2004). An integer programming formulation for a case study in university timetabling. European Journal of Operations Research, 153, 117-135.

[29] de Werra, D. (1985). An introduction to timetabling. European Journal of Operational Research, 19, 151-162.

[30] de Werra, D. (1997). The combinatorics of timetabling. European Journal of Operational Research, 96, 504-513.

[31] Dimopoulou, M., and Miliotis, P. (2001). Implementation of a university course and examination timetabling system. European Journal of Operational Research, 130, 202-213.

[32] Ferland, J.A., and Roy, S. (1985). Timetabling problem for university as assignment of activities to resources. Computers and Operations Research, 12(2), 207-218.

[33] Frieze, A.M., and Yadegar, J. (1981). An algorithm for solving 3-dimensional assignment problems with applications to scheduling a teaching practice. Journal of Operational Research Society, 32, 989-995.

[34] Gaspero, L.D., and Schaerf, A. (2001). Tabu search techniques for examination timetabling. In Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science, 2079, 104-117 (Springer Berlin / Heidelberg).

[35] Gotlieb, C.C. (1962). The construction of class-teacher timetables. Proceeding of IFIP Congress Munchen 1962, C.M. Popplewell (ed.), 73-77 (North Holland, Amsterdam).

[36] Hertz, A. (1991). Tabu search for large scale timetabling problems. European Journal of Operational Research, 54, 39-47.

[37] Lawrie, N.L. (1969). An integer linear programming model of a school timetabling problem. The Computer Journal, 12, 307-316.

[38] Merlot, L.T.G., Boland, N., Hughes, B.D., and Stuckey, P.J. (2003). A hybrid algorithm for the examination timetabling problem. Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science, 2740, 207-231 (Springer Berlin / Heidelberg).

[39] Műller, T., and Murray, K. (2008). Comprehensive approach to student Sectioning. PATAT 2008 - Proceedings of the 7th international conference on the Practice And Theory of Automated Timetabling.

[40] Mooney, E.L., Rardin, R.L., and Parmenter, W.J. (1996). Large scale classroom scheduling. IIE Transactions, 28(5), 369-378.

[41] Qu, R., Burke, E., McCollum, B., Merlot, L., and Lee, S. (2009). A survey of search methodologies and automated system development for examination timetabling. Journal of Scheduling, 12(1), 55-89.

[42] Ribeiro Filho, G., and Lorena, L.A.N. (2001). A Constructive evolutionary approach to school timetabling. Applications of Evolutionary Computing, Lecture Notes in Computer Science, 2037, 130-139 (Springer Berlin / Heidelberg).

[43] Schaerf, A. (1999). A survey of automated timetabling. Artificial Intelligence Review, 13(2), 87-127.

[44] Shih, W., and Sullivan, J.A. (1977). Dynamic course scheduling for college faculty via zero-one programming. Decision Sciences, 8, 711-721.

[45] Thompson, J.M., and Dowsland, K.A. (1998). A robust simulated annealing-based examination timetabling system. Computers and Operations Research, 25, 637-648.

[46] Tripathy, A. (1984). School timetabling - a case in large binary integer linear programming. Management Science, 30, 1473-1489.

[47] Tripathy, A. (1992). Computerized decision aid for timetabling - a case analysis. Discrete Applied Mathematics. 35(3), 1992, 313-323.

[48] Zervoudakis, K., and Stamatopoulos, P. (2000). A generic object-oriented constraint based model for university course timetabling. Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science, 2079, 28-47 (Springer Berlin / Heidelberg).

8 Appendix: A 0-1 linear programming formulation of *TEE* problem

We introduce this big $(\mathcal{L}-\mathcal{TEE})$ model here to present all the timetabling constraints in a precise mathematical fashion.

$(\mathcal{L}-\mathcal{TEE})$

$$Min \sum_{r\in R, p\in P} Y(r,p) - |R|$$

st.

$$\sum_{p\in P} Z(c,r,p) = 1, \qquad \forall (c,r) \in CR \subset C \times R \tag{15}$$

$$\sum_{r\in R(c)} Z(c,r,p) \leq 1, \qquad \forall p \in P, c \in C \tag{16}$$

$$\sum_{c\in C(r)} Z(c,r,p) \leq \varepsilon_r Y(r,p), \qquad \forall r \in R, \forall p \in P \tag{17}$$

$$\sum_{c\in C(r)} Z(c,r,p) \geq \pi_r \varepsilon_r W(r,p), \qquad \forall r \in R, p \in P \tag{18}$$

$$\sum_{p\in P} W(r,p) = 1, \qquad \forall r \in R \tag{19}$$

$$Y(r_1,p) + Y(r_2,p) \leq 1, \qquad \forall (r_1,r_2) \in EXCL \subset R \times R, \forall p \in P \tag{20}$$

$$W(r_1,p) = W(r_2,p), \qquad \forall (r_1,r_2) \in INCL \subset R \times R, \forall p \in P \tag{21}$$

$$\sum_{p_1=p}^{p+3} \sum_{r\in R(c)} Z(c,r,p) \leq 3, \qquad \forall p \in P, c \in C \tag{22}$$

$$\sum_{p\in P} Y(r,p) = 1, \qquad \forall r \in R^{no\_makeup} \tag{23}$$

$$\sum_{p\in P(d)} \sum_{r\in R(c)} Z(c,r,p) \leq 1, \qquad \forall d \in D, c \in C^{plebe} \tag{24}$$

$$W(r,p) = 1, \qquad \forall (r,p) \in RP^f \tag{25}$$

$$Y(r,p) = 0, \qquad \forall (r,p) \in RP^{prohib} \tag{26}$$

$$Z(c,r,p) = 0, \qquad \forall (c,r) \in CR, p > p^{last}(c) \tag{27}$$

The interpretation of the constraints is as follows. (15) the "cadet assignment" constraint: a cadet must have one exam for each course he or she is taking. (16) the "no-conflict" constraint: there can be no more than one exam for a cadet in one time slot. (17) the "course opening" constraint: one can only assign a cadet to an existing (i.e., scheduled) exam. (18) the "primary enrollment" constraint: if the primary exam for course r is scheduled in time slot p, then at least a certain proportion of the total number of students

enrolled for that course should take the exam at that time. (19) the "one-primary" constraint: there is exactly one primary exam scheduled per course. (20) the "exclusiveness" constraint: for some *exclusive* exam pairs (r_1, r_2), only one of each pair can be scheduled in a given time slot. (21) the "inclusiveness" constraint: for some *inclusive* exam pairs (r_1, r_2), the primary exams should be scheduled in the same time slot. (22) the "consecutiveness" constraint: no cadet can be assigned to more than three exams in a row. (23) the "no makeup" constraint: for certain courses, there cannot be a makeup exam. (24) the "plebe" constraint: a plebe can take at most one exam per day. (25) the "fixed exam" constraint: certain exams have a known imposed schedule. (26) the "prohibited exam" constraint: certain exams cannot be scheduled in certain time slots. (27) the "completion" constraint: each cadet should finish all his or her exams by the last available time slot $p^{last}(c)$. In addition, all variables must be 0-1.