A SET UNION BASED FORMULATION FOR COURSE SCHEDULING AND TIMETABLING

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Industrial Engineering

by
Jesse Paul Bukenberger
June 2014

COMMITTEE MEMBERSHIP

TITLE:	A Set Union Based Formulation for Course Scheduling and Timetabling
AUTHOR:	Jesse Paul Bukenberger
DATE SUBMITTED:	June 2014
COMMITTEE CHAIR:	Tali Freed, PhD Professor of Industrial Engineering
COMMITTEE MEMBER:	Lizabeth Schlemer, PhD Professor of Industrial Engineering
COMMITTEE MEMBER:	Beth Chance, PhD Professor of Statistics

ABSTRACT

A Set Union Based Formulation for Course Scheduling and Timetabling

Jesse Paul Bukenberger

The Course Timetabling Problem is a widely studied optimization problem where a number of sections are scheduled in concert with the assignment of students to sections in order to maximize the desirability of the resulting schedule for all stakeholders. This problem is commonly solved as a linear program with variables for each student or group of students with identical schedules. In this paper we explore an alternative formulation that aggregates binary student variables into integer variables denoting the number of students enrolled in a course. Our solution method assumes decomposition of the general schedule into time blocks, and applies a unique set theory based, integer linear programming formulation that seeks to maximize the total number of students enrolled in their desired sections across the time blocks. Once the problem has been solved, the simpler problem of disaggregating the solution is resolved. This approach can be used to find exact solutions, given sufficient computing power, or simplified to quickly find solutions within calculable bounds of optimality. Case studies with a local elementary school and a local high school show that the new formulation is significantly faster and can be made to be reasonably accurate.

Keywords: Operations Research, Integer linear programming, Scheduling, Couse timetabling, Blocking

ACKNOWLEDGMENTS

Matthew Davis Tyler Nuss	Assistance formatting data and building models
Kibbe Rubin	Providing data from Atascadero Fine Arts Academy
Andrew Cherry Maria Carroll	Providing data from Templeton High School
Tali Freed	Assistance procuring student data Providing a motivation for an improvement in this area
Cal Poly Industrial and Manufacturing Engineering Department	Funding for travel to conferences

TABLE OF CONTENTS

LIST OF TABLES vii
LIST OF FIGURES viii
I. Introduction 1
II. Literature Review 4
  Overview 4
  Graph Theoretical Applications 6
  Mixed Integer Programming Models 7
  Heuristic Search Strategies 9
  Our Contribution 11
III. Problem Definition and Solution Approaches 13
  Terminology 13
  Problem Definition 14
  Complete Binary Programming Model 17
  Aggregation of Student Variables into Sets 17
  Disaggregation into a Complete Solution 20
IV. Formulation 21
  Variable Definition 21
  Objective Function 22
  Constraints 23
V. Example 27
  Overview 27
  Complete Binary Programming Model 28
  Aggregated Student Model 31
VI. Application to Local Schools 36
  Atascadero Fine Arts Academy 36
  Templeton High School 37
VII. Conclusions 40
  Analysis of Results 40
  Scalability 41
  Potential for Future Research 42
List of References 44
Appendix 47

LIST OF TABLES

Table 1: Table of Symbols 22
Table 2: Example Course Details 27
Table 3: Example Student Preferences 27
Table 4: Complete Binary Programming Model Example Variables 28
Table 5: Complete Binary Programming Model Example Solution 31
Table 6: Aggregated Student Model Example Variables 32
Table 7: Aggregated Student Example Solution 35
Table 8: Atascadero Fine Arts Academy Comparison of CBPM vs. ASM 37
Table 9: Templeton High School Comparison of CBPM vs. ASM........ 38

LIST OF FIGURES

Figure 1: Enrollment when conflicting courses are given in the same block 19
Figure 2: Aggregation Constraint Examples 33

I. Introduction

Course Timetabling is a difficult problem faced by many academic institutions. The problem requires a scheduler to assign course sections to times, and students to sections, to maximize the desirability of the resulting schedule for all stakeholders. What makes a schedule desirable is a complex issue; it is often defined as a schedule that maximizes the number of students enrolled in the courses they have requested while meeting as many system constraints as possible. Other definitions of desirability focus on easing the burden on teachers and administrators, but these models are less common and often less complex. In this paper, we will use focus on student enrolment as the main contributor to desirability.

The problem is solvable when it is formulated into a linear program with binary variables that correspond to the assignment of resources to a time, an objective function that relates to the desirability of the schedule, and constraints that represent resource limitations. Many constraints are common to nearly all Course Timetabling Problems; for example, students are prevented from enrolling in two courses at the same time in all problems that track individual students. We will discuss the details of the problem more in the body of this paper.

When creating a timetable, the problem is commonly decomposed into blocks. Blocks partition the time available for courses to be offered into discrete periods, so a course offered in a given block will only conflict with other courses offered in that same block. Blocking greatly reduces the complexity of the problem from the scheduler's perspective, and because the majority of secondary schools require the schedule be partitioned into periods, blocking will not result in an inferior schedule at these institutions (Boland et al., 2006).

Despite the simplification afforded by blocking, the traditional model still takes far too long to reach optimal solutions to be useful for schedulers, particularly at large institutions. These institutions are forced to rely on manual or computerized heuristics to find solutions that are acceptable. Thus, there is a need for formulations that can reach optimal solutions quickly, or at least formulations that can reach more accurate solutions than the available heuristics in a similar timeframe (Boland et al., 2006).

In this paper we will explore a new and unique formulation that, instead of using binary variables for each student-course-block combination, aggregates student data from courses with one section into integer variables that represent the number of students taking a course at a specific time. The formulation uses set union principles to constrain the problem to the desired degree; the problem must be far more constrained to find exact solutions, but it may be more practical to relax some constraints to find a balance between speed and accuracy. We will make the assumption of blocking in all models and we will define desirability as the total number of students that can be enrolled in their desired courses across all time blocks without violating any hard constraints.

The structure of this paper is as follows: in section II we will review the existing literature on this topic and outline the contribution this work will make to the field, in section III we will explain the Course Timetabling Problem in detail and discuss our proposed formulation, in section IV we will fully explain our formulation in mathematical notation, section V we will create a small example to better illustrate the formulations, in section VI we will compare our new formulation to the pure binary programming formulation with two case studies at local schools, and in section VII we will make our conclusions and make suggestions for future research.

II. Literature Review

Overview

There are many approaches to solving the Course Timetabling Problem, and there are several problems that are similar enough to warrant mentioning here; we will discuss several relevant surveys of the literature and comment on many specific publications that are applicable to this problem. MirHassani & Habibi (2013) provide a survey of the recent work on Course and Examination Timetabling; they provide a comprehensive list of common hard and soft constraints, as well as common objective functions. The hard constraints generally focus on preventing the misappropriation of resources such as students taking two courses at the same time or two courses being offered in the same room while the soft constraints focus on superfluous benefits like balancing the number of students in each class or not having sections scheduled early in the morning. They mention student preferences as one of the more common objective functions that appears in the literature, but there are many others that are frequently used as well; however, many of these objective functions, such as making the timetable as compact as possible, are only applicable at institutions where each student is already guaranteed a full schedule and it is therefore a hard constraint in the model. Because we are dealing with a large number of non-required elective courses, it is not guaranteed that students will get every course they requested.

Qu et al. (2009) provide an extensive survey of recent work in the related problem of Examination Timetabling. Approaches range from the most theoretical graph coloring problem applications, to mixed integer programming (MIP) mathematical modeling. Recently, work has focused on a variety of heuristic approaches that are used to search the solution space in MIP models or even hyper heuristics that search for heuristics that may be effective. While Examination Timetabling is not identical to Course Timetabling, it is very similar and many breakthroughs for one carry over into the other. As such, they list a number of hard and soft constraints that are very similar to the others mentioned earlier.

Schaerf (1999) provides another survey that explores both course and exam timetabling while breaking course timetabling into two subsections: School Timetabling and Course Timetabling, where course timetabling is specifically for university applications. Schaerf focuses more on the automation and intractability of timetabling systems because it is uncommon for a scheduler and all the stakeholders to accept a schedule after the first run; thus, the formulation must be changed and run through several iterations. With this in mind, it is even more important for a formulation to run quickly, because it will probably need to be run several times before a schedule is accepted. Being able to easily make changes in the formulation is also an interesting new focus that has arisen, most commercial scale optimization platforms are not very easy to adjust, but that is beyond the scope of this work (Rudová et al., 2011). Schaerf (1999) also points to several areas that need additional research in the field. The estimation of how optimal different techniques are is notably uncommon, so we will make an effort to quantify our methods here. Additionally, the combination of different methods was suggested as an area for more work to be done, although this survey is slightly dated, there still seems to be plenty of opportunities in combining several of the methods we will discuss below.

Graph Theoretical Applications

The simplest scheduling problems can be thought of as graph coloring problems where each node is an event and each edge between nodes indicates some conflict. The nodes are then assigned colors in such a way that no nodes of the same color are connected by an edge (Lewis, 2008; Qu et al., 2009). This approach is very abstracted and it is difficult to apply a complete problem to a graph coloring model. It is; however, possible to use this approach to find the chromatic number of a graph which is equivalent to finding the number of time blocks needed to assign every student to their desired courses without conflicts (McDiarmid, 1979).

There is a significant body of work dealing with graph theory and how it applies to scheduling; however, many of the researchers are focused on determining the qualities that different solution spaces possess. The work is often strictly theoretical and does not apply any work to actual data (Burke et al. 2010). Meanwhile, researchers working with both MIP models and heuristics question the credibility of sources that do not demonstrate their findings either with actual institutions or with standardized data sets (Qu et al., 2009).

Mixed Integer Programming Models

When the problem is approached more directly, a Mixed Integer Program is usually the result. These models consist of numerous variables that are constrained by the requirements at a specific institution and utilize various optimization techniques to maximize or minimize some objective function. The objective functions usually seek to enroll as many students as possible, while violating as few constraints as possible. There are often soft and hard constraints, with the difference being hard constraints cannot be violated or the solution is considered infeasible, while soft constraints can be broken with some penalty applied to the objective function (MirHassani & Habibi, 2013).

This combination of soft and hard constraints is often called a Lagrangean relaxation. This method gives more power to the scheduler to weight the factors they find important more heavily. The hard constraints will not be violated by the system but the soft constraints can be violated at an arbitrary cost that the scheduler decides. These models are attractive because they are often much simpler than methods that seek to optimize enrollment and they allow the user to decide what is important and focus on many performance metrics at once. The problem with this method is that it will require significant validation to demonstrate that an optimal result in the computer corresponds to a similar result in practice. These models also require significant balancing, and they will generally not optimize anything; rather, they provide feasible solutions with an emphasis on performance goals (Tripathy, 1980).

MIP aggregated models and Lagrangean relaxations are popular because, as computers get more advanced, more realistic models can be solved in real time. Competitions are held where identical data are given to the competitors and the different formulations are compared on the basis of solve speed and number of soft constraints violated (Qu et al., 2009; Van den Broek et al., 2012).

These models have been applied over the decades in many different ways. The problems are often considered too large to be solved completely, so they are frequently simplified in a variety of ways. At many institutions, there is a large amount of symmetry in student course requests; this arises from students in the same year or program of study requiring the same courses. Many MIP models take advantage of this symmetry and aggregate student data into groups of students taking similar programs; this simplifies the problem considerably at institutions with large amounts of symmetry, but is not likely to be applicable to institutions where this is not the case. We did not find many MIP aggregations that did not require a significant amount of symmetry, which is one of the points addressed in this paper (Boland et al., 2006; Tripathy, 1984).

Additionally, the most straight-forward MIP model, which we will discuss below, is often alluded to but seldom constructed (Burke et al., 2012). It is referred to in nearly every MIP paper with varying degrees of depth and then the researchers proceed to make their changes to it without ever running it as a baseline (Rudová et al., 2011). Schaerf (1999) pointed out that very few researchers make an optimal baseline model to compare to. If researchers do compare methods, it is typically with standardized data sets, often the University of Toronto benchmark data or the Udine Course Timetabling Problem, and researchers only run their model and compare the speed and accuracy to others who have used that dataset (Burke et al., 2012; Qu et al., 2009). This practice is especially questionable because the differences in the computers' memory and processor are likely to contribute significantly to the variability in run time.

Heuristic Search Strategies

Most recent work on the Course Timetabling Problem has been focused on various heuristic search strategies, and many of these strategies have been very successful. Heuristics in this area are typically methods of exploring the vast solution space quickly and intelligently according to a few coded rules. Heuristics are typically much faster at finding good solutions than MIP models but the disadvantage is that the error cannot always be accurately estimated. Heuristics often have several parameters that need to be tuned to the specific model, so they require a greater investment at the beginning stages of the model; eventually, if the model stays similar from term to term a well-developed heuristic can be very effective.

Tabu search methods are possibly the most popular heuristic used today; Tabu searches are a type of local search that avoids getting stuck in local optima by remembering recently visited solutions and marking these locations as forbidden. The algorithm will not consider forbidden solutions and will move on to a new part of the solution space (Aladag et al., 2005).

Simulated Annealing is a method that similarly explores the solution space with a gradually decreasing temperature parameter. At high temperatures, the search is more likely to accept a move to an inferior solution, but as the temperature decreases, the algorithm becomes more selective and will prefer moving to more optimal solutions until the algorithm is essentially a local search (Abramson et al., 1999).

Genetic Algorithms attempt to mimic the natural phenomenon of evolution by starting with a number of solutions, termed individuals, which are deemed ‘fit’ if their objective function is higher than most of their competitors. The population of individuals is iterated through a number of generations where the fit individuals are mated and produce offspring with a similar solution makeup to their parents. There are different variations of this algorithm that include a probability of mutations occurring in offspring and eventually solutions that are close to optimal should arise (Beligiannis et al., 2009; Rudová et al., 2011).

Some studies have found that combinations of different search strategies perform better than either strategy individually; such combinations are sometimes known as hybrid search heuristics (Jat & Yang, 2011). Because of the success different heuristics have had, many researchers have turned to this area of the field; however, different methods perform better with different problems, and all methods have parameters that require a good deal of tuning before they become efficient. The problem of choosing and tuning an appropriate algorithm for a specific problem has led to the development of Hyper-Heuristics, which are designed to analyze the solution space and recommend search heuristics and parameters that are well suited to solving the problem effectively (Burke et al., 2003).

Our Contribution

In recent years, the focus of the Course Timetabling Problem has shifted extensively to the study of various heuristic search algorithms. Studies developing complete or slightly simplified models had largely concluded that the solution space was too large to solve efficiently at the student level with the available computer power; however, advances in computer hardware continually make larger problems feasible to solve with mixed binary-integer programs. With this paper, we intend to provide a formulation that can be solved in a reasonable time limit, but still provides accurate data at the student level; additionally, the formulation discussed below is believed to be compatible with other MIP aggregations and search heuristics, so there is room for further improvements to be made. Specifically, the work of Burke et al. (2006) is very similar to our model;

they use a similar baseline formulation and aggregate students who are taking the same courses into integer variables, while we aggregate students who are in a single section, following some other rules, into integer variables. While the two models are very similar, they are completely distinct and likely compatible at some level. We believe that our models can be combined in future work.

III. Problem Definition and Solution Approaches

Terminology

There is some variation in the terminology used in academic timetabling, and some terms have different meanings when used in an academic context than they do when used within educational institutions. We will begin our problem definition by defining the terms that are frequently used in this paper. Because we are working within the context of secondary schools, we will use language appropriate to that setting, but the methods discussed here can be applied to other settings such as universities or even institutions not related to education where timetabling is practiced. For example, conferences often have several speakers at one time and attendees must choose which of the conflicting tracks they will go to, and music festivals with several stages will have several bands performing at any time; these events would be more valuable if the events were scheduled to optimize the desirability of the schedule.

In the scheduling research community, *blocks* refer to partitions of the available time into discrete elements that are repeated throughout the time being scheduled. In a school setting, the term for period is what a scheduler would call a block. Block scheduling, at educational institutions, refers to a type of schedule where students take fewer classes each day for a longer period of time, and the classes taught each day rotate on some sort of cycle. One of the schools we worked with had this type of schedule, where each student was enrolled in 8 classes, and these classes were scheduled on a two-day cycle, so students took 4 classes each day. We will use the term blocking the way it is used in scheduling research, and we will attempt to avoid referring to block scheduling as it is known at educational institutions from this point on.

Courses will be defined here as a distinct subject at a specific level; for example, Beginning Choir, Advanced Choir and Beginning Dance would all be different courses. *Sections* will be defined here as an offering of a course. Some courses, due to limited demand or resources, will only have one section offered and others with more demand and resources will have several sections.

A *timetable* will be defined as a feasible assignment of sections to the blocks that constitute one cycle of the school's schedule. By populating all the blocks that make up one cycle, the term's entire schedule can be generated by repeating the same section assignments in each ensuing cycle.

Problem Definition

There are many factors that must be considered when creating a timetable, and this is part of what makes the problem so difficult to solve. Here we will explain some of the constraints that are enforced at the schools we worked with; the restrictions outlined here are among the most common in most timetabling works but this is by no means an exhaustive list of all impositions that are made on timetabling problems in general.

First, the courses that will be offered, and the number of sections that will be offered for each of these courses, must be declared. This is generally a function of teacher availability and student demand. Teacher availability is generally known and students are interviewed individually so their demand is known as well. At each of the schools we worked with, this was decided by the administration based on a complete tally of student requests, so we will not make recommendations in this area, and we will assume that student demand is known exactly.

Next, the sections that are being offered must be assigned several resources including: a room, a teacher, a time, and students who will enroll in the class. This assignment of resources is the main focus of most timetabling research because these resources are usually of limited availability so their assignment is somewhat competitive. The teachers at the schools we worked with had their own permanent rooms, so incorporating the room assignment into our model was unnecessary. However, we still needed to incorporate constraints for the teachers, times, and students that are assigned to each section.

The resource assignments are limited in many ways; the constraints we encountered are among the most commonly faced in the Course Timetabling Problem. A desirable timetable will allow as many students as possible to enroll in courses that they requested and will not violate any of the following restrictions:

- A course can only appear in a timetable once for each section of that course

- A course must have a section offered in a block for students to enroll in that course at that time
- Each section has a capacity and the student enrollment for that section cannot exceed this capacity
- Students can only take each course a specific number of times in a term, this number is usually one, but we did encounter some special cases where students could enroll in two sections of a course
- Students can only enroll in one class at a time
- Teachers can only teach one class at a time

After a feasible timetable is generated, it is presented to the stakeholders who will decide if the schedule is acceptable and what needs to be changed if it is undesirable in some way. The stakeholder suggestions are incorporated into the model and a new schedule is generated. For example, one institution we worked with wanted to experiment with scheduling all math teachers with a common break period. This required the formulation to be modified and run again to compare the resulting difference in enrollment. The process is repeated until everyone is sufficiently satisfied, and the final schedule is then used for the next term. This process usually takes some time and many schedules are often generated before everyone agrees that a schedule is acceptable, so the time required for a model to run is of considerable importance. In the following subsections we will explain the two formulations that we used to generate timetables.

Complete Binary Programming Model

The Complete Binary Programming Model (CBPM) uses binary variables to keep track of both course and student assignments. There is a binary variable for each course-block combination and for each student-desired course-block combination; these variables will be explained in more detail later in this paper.

The CBPM is a straightforward approach to solving the problem; it is clear from the variables when each course is being offered and which students are taking what course at what time. This transparency makes enforcing the constraints above fairly simple, as will be seen in the constraints section of this paper.

Because building a timetable is an NP-Complete problem (Qu et al., 2009), and the CBPM has so many variables, it takes a relatively long time to run; with the computing power available today, it could take several years to find the optimum solution at larger institutions. But when a solution is found, it is exact and known to be optimal. Most models studied today, including ours, are decompositions of the CBPM or heuristic search strategies that are tuned to quickly find solutions on the CBPM. As such, the CBPM will provide a good benchmark to compare the Aggregated Student Model against.

Aggregation of Student Variables into Sets

The Aggregated Student Model (ASM) is similar to the CBPM in that there is a binary variable for each course-block combination, but the two differ slightly in the creation of student variables. The ASM attempts to aggregate students into integer variables that do not track exactly which student can enroll in a course at a given time, but only how many students can enroll in a course at that time. This replaces many binary student variables, with a single integer aggregated student variable. This simplifies the solution of the problem computationally so an optimal timetable can be generated quickly. Once an optimal timetable is found, the problem of disaggregating student data and providing individual enrollment assignments can be easily resolved.

The ASM approaches most constraints in the same way the CBPM does, but the constraints concerning individual students are now not possible because of the removal of individual student variables. This poses a problem for courses with multiple sections because it becomes difficult to track which students have already been enrolled in a different section of the same course. To avoid this problem, we chose to only aggregate student variables for courses with a single section, and model courses with multiple sections in the same way the CBPM does.

Aggregated variables also confound section capacity with the constraint that students cannot take multiple courses at the same time. For example, if 5 students want to take Dance and 5 students want to take Ceramics, but 3 of these students want to take both, then only 7 students would be able to enroll if both were offered at the same time; however, the ASM will attempt to enroll ten students in this situation unless a new constraint is introduced. A diagram illustrating this is shown in Figure 1 below.

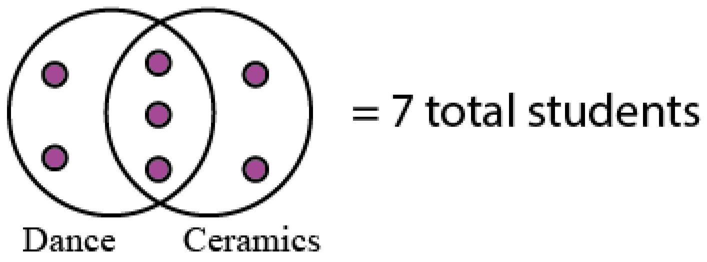

Figure 1: Enrollment when conflicting courses are given in the same block

To properly enforce the two confounded constraints, we must incorporate a new constraint that limits the possible enrollment in concurrent courses to the total number of students who want at least one of the courses. To ensure exact solutions, there must be such a constraint for each combination of multiple courses. Conceptually, it helps to refer to these constraints as belonging to disjoint tiers, where the first tier is the course capacity, the second tier covers all the combinations of two courses, the third tier covers all combinations of three courses, and so on. For large institutions, there will be many tiers each with a number of constraints that grow in a hypergeometic fashion. This will eventually become an unreasonable requirement; however, with each tier of constraints that are included, the solution will be bounded closer to the true optimal solution. We therefore recommend relaxing the upper tiers of this constraint, which will reduce the ability of the model to find optimal solutions while decreasing the time required to build and read the model.

Disaggregation into a Complete Solution

Once an optimal timetable is found in the ASM, the data needs to be disaggregated to determine exactly how many students can be enrolled in their desired courses and which students are to be enrolled in each section. To accomplish this, we exported the timetable from the ASM to the CBPM, and found that this bounded the solution space so sufficiently that an optimal solution was found in a time that was negligible when compared to the full computing time of either model. Therefore, we did not develop additional algorithms or formulations to disaggregate the solutions, but there likely exist superior methods of accomplishing this task.

IV. Formulation

Variable Definition

This section details the specifics of the formulation in mathematical notation. We will fist describe the variables that are used, then we will give equations for the two objective functions for the CBPM and the ASM, and finally we will detail equations for all the constraints used in the formulations. The exact details for the variables are seen in equations (1), (2) and (3) below and; after that, Table 1 contains a summary of the sets, variables, subscripts, and parameters that are used in the formulation. Please note that the variables indicated in equation (3) only appear in the ASM.

$$s_{n\,c\,b} = \begin{cases} 1 \text{ if student } n \text{ takes course } c \text{ at time } b \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

$$y_{c\,b} = \begin{cases} 1 \text{ if course } c \text{ is offered at time } b \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

$$x_{c\,b} = \text{number of students taking course } c \text{ at time } b \tag{3}$$

Table 1: Table of Symbols

Symbol	Description
$n \in N = \{1,\ldots,N\}$	Set of all students, *n* is an individual student
$c \in C = \{1,\ldots,C\}$	Set of all courses, *c* is a specific course
$b \in B = \{1,\ldots,B\}$	Set of all time blocks, *b* is a specific time block
$t \in T = \{1,\ldots,T\}$	Set of all teachers, *t* is an individual teacher
$(c, t) \in \tau$	Set of course-teacher assignments, (*c*, *t*) exists if course *c* is taught by teacher *t*
$c \in C_t = \{c \in C: (c, t) \in \tau\}$	Set of courses taught by teacher *t*, *c* is a specific course
$c \in C^M = \{c \in C: \gamma_c > 1\}$	Set of courses with more than one section available
$D_c \subseteq N$	Set of students who wish to enroll in course *c*
$P \in \mathbb{P}(C)$	Power set of C, $\mathbb{P}(C)$ consists of every subset of C
$s_{n\,c\,b}$	Binary variable indicating that student *n* is enrolled in course *c* at time *b*
$y_{c\,b}$	Number of sections of course *c* offered at time *b*
$x_{c\,b}$	Number of students enrolled in course *c* at time *b*
λ_c	Capacity (number of students) of course *c*
γ_c	Number of sections of course *c* that can be offered
α_c	Number of times course *c* can be taken by a single student

Objective Function

As mentioned above, we define the desirability of the timetable as the number of students that can enroll in a desired course summed across all blocks. For the CBPM this simply requires us to sum each student variable as seen in equation (4) below.

$$Maximize\ Z = \sum_{b \in B} \sum_{c \in C} \sum_{n \in N} S_{n\,c\,b} \tag{4}$$

For the ASM; however, there are not binary student variables for each of the classes that are offered. Instead, there are variables that represent the number of students taking a class at a given time. For each class with more than one section, there are binary student variables, and for classes with only one section, there are aggregated student variables; this objective function is shown below in equation (5). It is worth noting that this function does not necessarily provide exact numbers on enrollment unless the problem is fully constrained as shown in the Appendix; what we gain from the new formulation is the timetable that can be used to constrain the CBPM which will provide an exact number on enrollment.

$$Maximize\ Z = \sum_{b \in B} \sum_{c \in C^M} \sum_{n \in N} s_{n\,c\,b} + \sum_{b \in B} \sum_{c \in C} x_{c\,b} \tag{5}$$

Constraints

In this section we express the constraints on the model mathematically. Each of these constraints appears in both the CBPM and the ASM except for the equations containing any $x_{c\,b}$ variables; these constraints are only in the ASM.

A course can only be offered a number of times given by γ_c. This is represented in equation (6) below.

$$\sum_{b \in B} y_{c\,b} \leq \gamma_c \qquad \forall c \in C \tag{6}$$

Equation (7) below captures two separate impositions on the model: students cannot enroll in a course at a given time unless it is being offered in that same time block, and the course's enrollment is capped at a specific number given by λ_c. Equation (8) represents the same capacity constraints for courses with only one section in the ASM.

$$\sum_{n \in N} s_{n\,c\,b} - \lambda_c * y_{c\,b} \leq 0 \qquad \forall c \in C \quad \forall b \in B \tag{7}$$

$$x_{c\,b} - \lambda_c * y_{c\,b} \leq 0 \qquad \forall c \in C \quad \forall b \in B \tag{8}$$

Students cannot enroll in the same course multiple times. Normally students cannot enroll more than once in any course, but we did encounter special cases where students could enroll in the same course over several blocks; thus, a student may not enroll in the same course more than a number of times given by α_c which is shown in equation (9) below.

$$\sum_{b \in B} s_{n\,c\,b} \leq \alpha_c \qquad \forall n \in N \quad \forall c \in C \tag{9}$$

Students may not enroll in more than one class at a time and teachers may not teach more than one class at a time. These constraints are represented for students and teachers in equation (10) and equation (11) respectively. Note that to avoid the inclusion of teacher variables, we constrain classes that are taught by the same teacher; this will not be possible at every institution and teacher variables will be necessary.

$$\sum_{c \in C} s_{n\,c\,b} \leq 1 \qquad \forall n \in N \qquad \forall b \in B \tag{10}$$

$$\sum_{c \in C_t} y_{c\,b} \leq 1 \qquad \forall b \in B \qquad \forall t \in T \tag{11}$$

Equation (12) represents the major distinction between the CBPM and the ASM; this constraint determines how many students can enroll in one of their preferred courses when there are several courses being offered at the same time. The exact number of students that can enroll in at least one course is not fully constrained by this equation until the union extends to a tier past the maximum number of courses that are offered in a block, which is not known before the problem is solved. However, even if the model is not fully constrained, the constraint provides a very close estimate for how many students can enroll in one of their desired courses each block. For those unfamiliar with this notation, the equation roughly states that the sum of all students who can enroll in a collection of courses with only one section must be less than or equal to the number of students who want to take at least one of the courses for all time blocks and all subsets of courses with one section.

$$\sum_{c \in C^M} \sum_{n \in N} s_{n\,c\,b} + \sum_{c \in P} x_{c\,b} \leq \left| \bigcup_{c \in P} D_c \right| \qquad \forall b \in B \qquad \forall P \in \mathbb{P}(C) \qquad (12)$$

Additionally, all variables are constrained to be non-negative and are restricted to either binary or integer values; all student variables are binary, the aggregated student variables all integers, and the course variables are binary if there is only one section of the course and integer if there are multiple sections.

V. Example

Overview

In this section, we will create an example to better illustrate how the two models function. Imagine a small school with only eight students; this school has four classes: Art, Band, Ceramics, and Dance. Art, Band, and Ceramics are offered once, while Dance is offered twice. The school only has two time blocks for these classes to be offered. The details of the school are shown in Table 2 below.

Table 2: Example Course Details

ID	Course	Sections
1	Art	1
2	Band	1
3	Ceramics	1
4	Dance	2

Each of the eight students has submitted requests to be enrolled in their two most preferred classes as shown below in Table 3 below.

Table 3: Example Student Preferences

ID	Student	Requested Classes	
1	Aly	Art	Band
2	Ben	Band	Ceramics
3	Cole	Art	Dance
4	Dan	Art	Dance
5	Emma	Art	Dance
6	Fay	Band	Dance
7	Gail	Ceramics	Dance
8	Hal	Ceramics	Dance

Complete Binary Programming Model

The CBPM for this school results in 32 student enrollment variables and 8 course variables for a total of 40 variables. Table 4 below shows these variables in more detail. The Objective function for the CBPM is then to maximize the sum of the student variables.

Table 4: Complete Binary Programming Model Example Variables

Student Variables			
$s_{Aly\ Art\ 1}$	$s_{Aly\ Art\ 2}$	$s_{Aly\ Band\ 1}$	$s_{Aly\ Band\ 2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$s_{Hal\ Ceramics\ 1}$	$s_{Hal\ Ceramics\ 2}$	$s_{Hal\ Dance\ 1}$	$s_{Hal\ Dance\ 2}$
Course Variables			
$y_{Art\ 1}$	$y_{Band\ 1}$	$y_{Ceramics\ 1}$	y_{Dance1}
$y_{Art\ 2}$	$y_{Band\ 2}$	$y_{Ceramics\ 2}$	$y_{Dance\ 2}$

Equation (13) constrains the number of sections offered for each course, so all courses will have at most one section except for Dance, which can have two. There is one formula for each course offered, so in this example 4 constraints are added to the formulation.

$$\begin{gathered} y_{Art\ 1} + y_{Art\ 2} \leq 1 \\ \vdots \\ y_{Dance\ 1} + y_{Dance\ 2} \leq 2 \end{gathered} \tag{13}$$

Equation (14) enforces the capacity of each course and also prevents students from being enrolled in a course that is not being offered; for this example, each section has an enrollment capacity of four students per section offered. Because there is a constraint for each course at each time block, there are a total of 8 constraints added to manage the capacity of the sections.

$$\begin{gathered} s_{Aly\ Art\ 1} + s_{Cole\ Art\ 1} + s_{Dan\ Art\ 1} + s_{Emma\ Art\ 1} - 4 * y_{Art\ \ 1} \leq 0 \\ \vdots \\ s_{Cole\ Dance\ 2} + s_{Dan\ Dance\ 2} \dots + s_{Hal\ Dance\ 2} - 4 * y_{D2} \ \leq \ 0 \end{gathered} \tag{14}$$

Equation (15) prevents students from being enrolled in a course several times in different blocks. Because there is a constraint for each course each student is taking, there are 16 of these constraints in this model; however, the courses with only one section will never be bounded by these constraints, so we could safely remove all but the Dance class constraints here for a total of just 6 constraints. For simplicity, we will only show the necessary constraints in Equation (15) below.

$$\begin{gathered} s_{Cole\ Dance\ 1} + s_{Cole\ Dance\ 2} \leq \ 1 \\ \vdots \\ s_{Hal\ Dance\ 1} + s_{Hal\ Dance\ 2} \leq \ 1 \end{gathered} \tag{15}$$

Equation (16) prevents students from being enrolled in several courses in the same block, because there is a constraint for each block by the number of students, there are 16 of these constraints in this model.

$$
\begin{gathered}
s_{Aly\ Art\ 1} + s_{Aly\ Band\ 1} \leq 1 \\
\vdots \\
s_{Hal\ Ceramics\ 2} + s_{Hal\ Dance\ 2} \leq 1
\end{gathered}
\tag{16}
$$

The last constraint that is needed for the CBPM is shown in Equation (17) and it prevents a teacher from teaching two sections at the same time. In this example, there is only one teacher for Art and Ceramics, so these courses must not be offered at the same time. We could also include a constraint to prevent dance from being offered twice in one block, but because the dance variable is binary, this constraint is not needed. There is one constraint needed for each teacher for each block, so we only need 2 constraints in this model.

$$
\begin{gathered}
y_{Art\ 1} + y_{Ceramics\ 1} \leq 1 \\
y_{Art\ 2} + y_{Ceramics\ 2} \leq 1
\end{gathered}
\tag{17}
$$

Some inspection reveals that there are multiple solutions to this model, one of which is found by offering Art and Dance first, and then offering Band, Ceramics and Dance in the second block. Every student can be enrolled in the second block, and only Ben cannot be enrolled in the first block, giving an optimal enrollment number of 15. The specific solution chosen is shown in Table 5.

Table 5: Complete Binary Programming Model Example Solution

Block 1		**Block 2**	
Courses	**Students**	**Courses**	**Students**
Art:	Aly	**Band:**	Aly
	Cole		Ben
	Dan		Fay
	Emma		
		Ceramics:	Gail
Dance:	Fay		Hal
	Gail		
	Hal	**Dance:**	Cole
			Dan
			Emma

Aggregated Student Model

The ASM eliminates the need for many of the student variables needed in the CBPM; the ASM requires 12 student variables, 6 aggregated student variables, and 8 course variables for a total of 26 variables, shown in Table 6 below. The objective function is to maximize the sum of both the individual student variables and the aggregated student variables.

Table 6: Aggregated Student Model Example Variables

Student Variables			
$s_{Cole\ Dance\ 1}$	$s_{Dan\ Dance\ 1}$	$s_{Emma\ Dance\ 1}$	
$s_{Fey\ Dance\ 1}$	$s_{Gail\ Dance\ 1}$	$s_{Hal\ Dance\ 1}$	
$s_{Cole\ Dance\ 2}$	$s_{Dan\ Dance\ 2}$	$s_{Emma\ Dance\ 2}$	
$s_{Fey\ Dance\ 2}$	$s_{Gail\ Dance\ 2}$	$s_{Hal\ Dance\ 2}$	
Aggregated Student Variables			
$x_{Art\ 1}$	$x_{Band\ 1}$	$x_{Ceramics\ 1}$	
$x_{Art\ 2}$	$x_{Band\ 2}$	$x_{Ceramics\ 2}$	
Course Variables			
$y_{Art\ 1}$	$y_{Band\ 1}$	$y_{Ceramics\ 1}$	y_{Dance1}
$y_{Art\ 2}$	$y_{Band\ 2}$	$y_{Ceramics\ 2}$	$y_{Dance\ 2}$

The ASM has the same constraints as the CBPM for the sets shown in (13), (15) and (17) with no changes. Equation (18) shows the capacity constraints for the ASM; for courses with only one section, such as Art, the variable denoting how many students can enroll is all that is needed, but for courses with several sections, such as Dance, we must use the variables for individual students.

$$\begin{gathered} x_{Art\ 1} - 4 * y_{Art\ \ 1} \leq 0 \\ \vdots \\ s_{Cole\ Dance\ 2} + s_{Dan\ Dance\ 2} \cdots + s_{Hal\ Dance\ 2} - 4 * y_{D2} \leq 0 \end{gathered} \tag{18}$$

The constraints given by Equation (16) are still needed in the ASM but only with the individual student variables that still exist in the ASM. However, because the only student variables that exist are for enrolling in Dance, there is no way to schedule a student in two classes at once, and these constraints are not needed.

The last set of constraints needed for the ASM is the union constraints. Figure 2 shows the value of D_c for several combinations of courses, and Equation (19) shows how these are programmed into the model. Parts a and c of Figure 2 show the standard union with sections of two courses offered just once. Part b shows the union where one course is offered once and the other course is offered multiple times. We must use the individual student variables here, but we only need the student variables for students who want to take both courses.

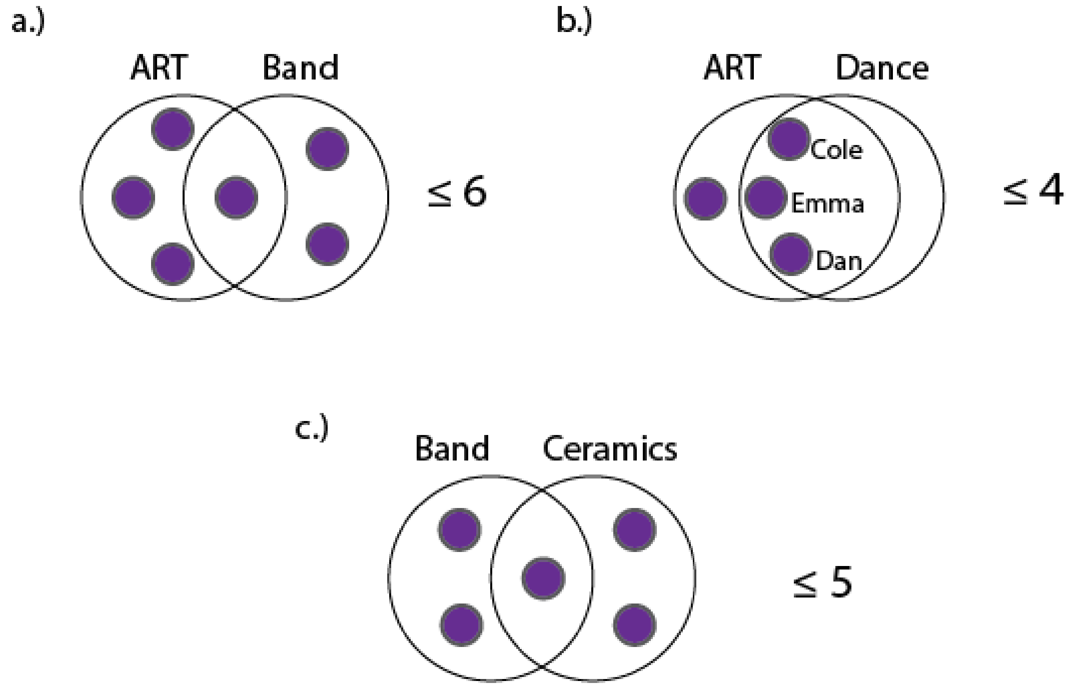

Figure 2: Aggregation Constraint Examples

Because Art and Ceramics cannot be taught at the same time, we only need 7 constraints for each block. All 7 of the constraints needed for the first block are shown in Equation (19), and another identical set is needed for the second block.

$$
\begin{gathered}
x_{Art\ 1} + x_{Band\ 1} \leq 6 \\
x_{Art\ 1} + s_{Cole\ Dance\ 1} + s_{Dan\ Dance\ 1} + s_{Emma\ Dance\ 1} \leq 4 \\
x_{Band\ 1} + x_{Ceramics\ 1} \leq 5 \\
x_{Band\ 1} + s_{Fay\ Dance\ 1} \leq 3 \\
x_{Ceramics\ 1} + s_{Gail\ Dance\ 1} + s_{Hal\ Dance\ 1} \leq 3 \\
x_{Art\ 1} + x_{Band\ 1} + s_{Cole\ Dance\ 1} + s_{Dan\ Dance\ 1} + s_{Emma\ Dance\ 1} + s_{Fay\ Dance\ 1} \leq 6 \\
x_{Band\ 1} + x_{Ceramics\ 1} + s_{Fay\ Dance\ 1} + s_{Gail\ Dance\ 1} + s_{Hal\ Dance\ 1} \leq 5
\end{gathered}
\tag{19}
$$

As mentioned earlier, there are several optimal solutions for this example. The solution given by the ASM is when Art and Dance are offered in the first block and Band, Ceramics, and Dance are offered in the second block. The aggregated solution is shown in Table 7 and the disaggregated solution, which is found by constraining the CBPM to the timetable found in the ASM, is found to be the same as the solution of the CBPM alone, but with the two blocks switched, which was shown in Table 5.

Table 7: Aggregated Student Example Solution

Student Variables			
$s_{Cole\ Dance\ 1} = 0$	$s_{Dan\ Dance\ 1} = 0$	$s_{Emma\ Dance\ 1} = 0$	
$s_{Fey\ Dance\ 1} = 1$	$s_{Gail\ Dance\ 1} = 1$	$s_{Hal\ Dance\ 1} = 1$	
$s_{Cole\ Dance\ 2} = 1$	$s_{Dan\ Dance\ 2} = 1$	$s_{Emma\ Dance\ 2} = 1$	
$s_{Fey\ Dance\ 2} = 0$	$s_{Gail\ Dance\ 2} = 0$	$s_{Hal\ Dance\ 2} = 0$	
Aggregated Student Variables			
$x_{Art\ 1} = 4$	$x_{Band\ 1} = 0$	$x_{Ceramics\ 1} = 0$	
$x_{Art\ 2} = 0$	$x_{Band\ 2} = 3$	$x_{Ceramics\ 2} = 2$	
Course Variables			
$y_{Art\ 1} = 1$	$y_{Band\ 1} = 0$	$y_{Ceramics\ 1} = 0$	$y_{Dance\ 1} = 1$
$y_{Art\ 2} = 0$	$y_{Band\ 2} = 1$	$y_{Ceramics\ 2} = 1$	$y_{Dance\ 2} = 1$

VI. Application to Local Schools

Atascadero Fine Arts Academy

The first school we worked with was a relatively small institution that schedules its elective courses separately from its core courses. The school is small enough that the core courses are not particularly difficult to schedule, but there is a significant amount of diversity in the demand for elective courses. What we mean by diversity is that very few students want exactly the same combination of courses, as opposed to symmetry, where many students want the same course combination. Demand diversity makes it far more difficult to create desirable schedules manually, so we were asked to survey student demand, and schedule only the elective courses.

We surveyed 186 students' desire to enroll in 19 courses, 13 of which have only one section. There were a total of 31 sections that needed to be timetabled over 4 blocks. There were no students who wanted all the same courses so the model proposed by Boland et al. (2006) would not simplify the formulation; however the ASM managed to remove 1,260 variables. We constructed the CBPM and the ASM with the constraint represented by equation (12) enforced to the fourth tier. Both models were built in Gurobi (2014), and the results are summarized in Table 8 below.

Table 8: Atascadero Fine Arts Academy Comparison of CBPM vs. ASM

	Complete Binary Programming Model	Aggregated Student Model
Number of Variables	2,760	1,500
Number of Constraints	1,193	2,652
Solve Time (s)	12.98	7.37
% Solve Time of CBPM	100%	56.78%
Number of Students-Sections Enrolled	624	621
% Student Enrollment of CBPM	100%	99.52%

This output is encouraging for several reasons; the ASM greatly reduced the number of variables in the model, and while the number of constraints also rose substantially, the solve time for the ASM was roughly half that of the CBPM. Additionally, the quality of the final solution is nearly identical between the two models; the ASM fails to enroll three students in one block that the CBPM successfully enrolls, and this could likely be amended by enforcing the fifth tier of the constraint represented by equation (12).

Templeton High School

Next, we worked with a much larger school that required both core and elective classes to be scheduled. This school is large enough that computers are required to find good solutions, and even with the aid of computers, optimal solutions may not be found in times that are considered reasonable. Because core courses are being scheduled, there is a large amount of symmetry involved in the solution space, but there are still many classes that are only offered once so the diversity of student demand is also high.

Data on 782 students was provided for 101 courses, 72 of which have only one section. There were a total of 183 sections that needed to be timetabled over 8 blocks. We constructed the CBPM and the ASM with the constraint represented by equation (12) enforced to the first tier. The models were both built in Gurobi (2014), and the results, summarized in Table 9 below, show a larger disparity in the quality of the solutions but a dramatically improved run time from the CBPM to the ASM.

Table 9: Templeton High School Comparison of CBPM vs. ASM

	Complete Binary Programming Model	Aggregated Student Model
Number of Variables	55,456	35,184
Number of Constraints	14,201	15,283
Solve Time (s)	10,231	366
% Solve Time of CBPM	100%	3.58%
Number of Students-Sections Enrolled	5,294	5,137
% Student Enrollment of CBPM	100%	97.0%

These results demonstrate that even with the ASM running at its most inaccurate setting, the results are still reasonable. The Constraint represented by equation (12) was only enforced to the first tier, which is essentially only using course capacity as a guideline for when to schedule courses, but the quality of the solution is only 3% worse. It is likely that most of this disparity would be eliminated with the enforcement of the second or third tier of constraints. Because the complete generation of a hypergeometic constraint takes quite some time, it is expected that the ASM will take longer to generate and read than the CBPM, but this is expected to be comparatively small when the entire process is considered.

Another noteworthy aspect of these results is the run time; the ASM ran in less than 4% of the time that the CBPM. Reduction in run time was the primary motivation for creating a new formulation, and it is possible that once the model is more constrained, it will run even faster because the size of the solution space will be reduced.

VII. Conclusions

Analysis of Results

We have seen that by aggregating student data, with a method independent of symmetry, it is possible to considerably reduce the time required to find a solution to the Course Timetabling Problem. We believe that optimal solutions can still be found with the Aggregated Student Model, but the focus of this study was to reduce the run time of the ASM while maintaining comparable results to the Complete Binary Programing Model.

We found that the ASM is successful when employed at both small and mid-sized institutions; the run time advantage of the ASM appears to be better as the institution grows in size, and the error appears to increase as the model is less constrained. Because some of these variables are confounded though, it is possible that the size of the institution or the degree to which the model is constrained do not have the expected impact. Furthermore, we only have data from two institutions, so we cannot be entirely confident in any of our findings because it is possible that unexpected factors had an influence on our results.

We were unable to determine if constraining the model further reduces the run time or extends it, so a follow up study will be needed. As has been seen in other studies, the bounds of error for this problem are too large and variable to be useful. Practically, we have shown that the error at the least accurate setting is around

3% but a more exact analysis with several data sets, and theoretical proofs would be needed for any confident conclusions here.

The ASM, like most solution strategies to the Course Timetabling Problem, is better equipped to deal with some problems than others. Specifically, the ASM is ideal for institutions with many courses that have a single section and a diverse range of course demand. But it seems to be an improvement from the CBPM whenever speed is a primary factor.

Scalability

The scalability of the problem is difficult to estimate because the size of the models, and thus the solution time, is dependent on a number of factors that are different from institution to institution. Because the timetabling problem presented in this paper is NP-Complete (Qu et al., 2009), the time required to find an optimal solution at large institutions would take several years with the computing power available today. As the number of courses offered, blocks used, and students grows, the solution space grows in an exponential manner, so the solution time will also grow at a similar rate. Both the CBPM and the ASM will take an unreasonable time to solve at large institutions, but the ASM will always solve faster than the CBPM as long as there is at least one course with only one section. The ASM will take longer to load into the computer than the CBPM, but so far, this time has been insignificant compared to the actual solve time.

Potential for Future Research

Most work on the Course Timetabling Problem focuses on heuristic search strategies applied to the CBPM; however there are many possibilities for search algorithms to be applied to decomposed models. It is likely that heuristics or hyper-heuristics tuned to decomposed models will run much faster than a decomposed model or a heuristic alone, so there is potential for such work to be done on the ASM. Additionally, we would like to see our work combined with that of Boland and Hughes [5]; they have developed a model that benefits from symmetry in the solution space while the ASM is well suited to demand diversity. We believe that the ASM is compatible with their model, and together they would compose a more robust program than either individually, benefiting from both demand diversity and symmetry where present.

In this paper, we chose not to aggregate student data for all courses with multiple sections to avoid the need to track which students have already enrolled in a previous section of the course, but this simplification is not inherently necessary and it may be possible to constrain aggregated data for these courses. This would make the ASM efficient when working with large institutions that offer several sections for most courses.

Finally, the speed and accuracy of the ASM may depend largely on the tier to which the union constraints on enrolment are enforced. The model can find accurate solutions quickly if the model is constrained to the right degree, but it is difficult to know where that is. This paper has set up the basics for analytically determining the range of optimality the solution is bounded within, but additional work is needed to determine the relationship between solution time and accuracy of solution.

List of References

Abramson, D., Mohan, K., & Dang, H. (1999). Simulated Annealing cooling Schedules for the School Timetabling Problem. *Asia – Pacific Journal of Operational Research. 16*(1). 1-22.

Aladag, C., & Hocaoglu, G., (2005). A Tabu Search Algorithm to Solve a Course Timetabling Problem. *Hacettepe Journal of Mathematics and Statistics. 36*(1). 53-64.

Beligiannis, G., Moschopulos, C., & Likothanassis, S. (2009). A Genetic Algorithm Approach to School Timetabling. *The Journal of the Operational Research Society. 60*(1). 23-42.

Boland, N., Hughes, B., Merlot, L., & Stuckey, P. (2006). New integer linear programming approaches for course timetabling. *Computers & Operations Research. 35.* 2209-2233.

Burke, E., Kendall, G., & Soubeiga, E. (2003) A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics. 9*(6). 451.

Burke, E., Marecek, J., Parkes, A., & Rudova, H. (2010). A SuperNodal Formulation of Vertex Colouring with Applications in Course Timetabling. *Annals of Operations Research. 179*(1). 105-130.

Burke, E.,Marecek, J., Parkes, A., & Rudova, H. (2012). A Branch-and-Cut Procedure for the Udine Course Timetabling Problem. A*nnals of Operations Research. 194*(1). 71-87.

Gurobi Optimization, Inc. (2014) Gurobi Optimizer Reference Manual. <http://www.gurobi.com>

Jat, S., & Yang, S., (2011). A Hybrid Genetic Algorithm and Tabu Search Approach for Post Enrolment Course Timetabling. *Journal of Scheduling. 14*(6). 617-637.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum, 30*(1), 167-190. doi:http://dx.doi.org/10.1007/s00291-007-0097-0

McDiarmid, C. (1979). Determining the Chromatic Number of a Graph. *SIAM Journal on Computing. 8*(1). 1-14.

Mirhassani, S. A., & Habibi, F. (2013). Solution approaches to the course timetabling problem. *The Artificial Intelligence Review, 39*(2), 133-149. doi:http://dx.doi.org/10.1007/s10462-011-9262-6

Qu, R., Burke, E. K., Mccollum, B., Merlot, L. T., G., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling, 12*(1), 55-89. doi:http://dx.doi.org/10.1007/s10951-008-0077-5

Rudová, H., Müller, T., & Murray, K. (2011). Complex university course timetabling. *Journal of Scheduling, 14*(2), 187-207. doi:http://dx.doi.org/10.1007/s10951-010-0171-3

Schaerf, A. (1999). A survey of automated timetabling. *The Artificial Intelligence Review, 13*(2), 87-127. doi:http://dx.doi.org/10.1023/A:1006576209967

Tripathy, A. (1980). A Lagrangean Relaxation Approach to Course Timetabling. *The Journal of the Operational Research Society. 31*(7). 599-603.

Tripathy, A. (1984). School Timetabling – A Case in Large Binary Integer Linear Programming. *Management Science. 30*(12). 1473.

Van den Broek, J J , J., Hurkens, C. A., & J. (2012). An IP-based heuristic for the post enrolment course timetabling problem of the ITC2007. *Annals of Operations Research, 194*(1), 439-454. doi:http://dx.doi.org/10.1007/s10479-010-0708-z

Appendix

Theorem:

Let b ∈ B be the block with the largest number of courses offered in a schedule, let the number of courses offered during b be $|C_b| = r$. Let the sum of all x_c be constrained so that

$$\sum_{c \in C_b} x_c \leq \left| \bigcup_{c \in C_b} D_c \right|$$

Therefore,

$$MAX\left[\sum_{b \in B}\sum_{c \in C}\sum_{n \in N} S_{n\,c\,b}\right] = MAX\left[\sum_{b \in B}\sum_{c \in C} x_{c\,b}\right]$$

and the objective function of the ASM is equivalent to that of the CBPM.

Proof:

Since the definition of the Desirability set is the subset of students who want to take course c, the cardinality of the union of the desirability set across all courses offered this block C_b is the sum of all students who want to take at least one course $c \in C_b$

$$MAX\left[\sum_{c\in C_b}\sum_{n\in N} S_{n\,c}\right] = \left|\bigcup_{c\in C_b} D_c\right|$$

Because x_c is upward bounded by the union of the desirability set, the maximum of x_c is of course

$$MAX\left[\sum_{c\in C_b} x_c\right] = \left|\bigcup_{c\in C_b} D_c\right| \rightarrow\ MAX\left[\sum_{c\in C_b}\sum_{n\in N} S_{n\,c}\right] = MAX\left[\sum_{c\in C_b} x_c\right]$$

Now because b is the block with the largest number of courses offered, all other blocks have at most r courses offered; thus, we can repeat the above procedure for each block and the union of the Desirability set will sufficiently constrain the sum of all x_c. Therefore,

$$MAX\left[\sum_{b\in B}\sum_{c\in C}\sum_{n\in N} S_{n\,c\,b}\right] = MAX\left[\sum_{b\in B}\sum_{c\in C} x_{c\,b}\right]$$

and the objective function of the ASM is equivalent to that of the CBPM. ■