Set Union Based Formulation for Class Scheduling and Timetabling

Jesse Bukenberger, Matthew Davis, Tali Freed, and Tyler Nuss
California Polytechnic State University: San Luis Obispo
Department of Industrial and Manufacturing Engineering
San Luis Obispo, California, 93410, USA

Abstract

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 with variables for each student or group of students with identical schedules. In this paper we explore an alternative formulation that aggregates student preference data with set union principles. 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 are included.

Keywords
Operations Research, Integer linear programming, Scheduling, Course timetabling, Blocking

1. 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, but it is typically concerned with maximizing the number of students enrolled in the courses they have requested while meeting as many system constraints as possible.

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 generally not result in an inferior schedule.

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 can be fully constrained to find exact solutions, but it is typically 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 2 we will briefly review the existing literature on this topic and outline the contribution this work will make to the field, in section 3 we will explain the Course Timetabling
Problem in detail and discuss our proposed formulation, in section 4 we will fully explain our formulation in mathematical notation, in section 5 we will compare our new formulation to the pure binary programming formulation with two case studies at local schools, and in section 6 we will make our conclusions and make suggestions for future research.

2. Literature Review

2.1 Overview
There are many approaches to solving the Course Timetabling Problem, and there are several problems that are similar enough to warrant mentioning here; however, with limited space, we will recommend several surveys of the literature that we have found helpful, and comment on only a few of the myriad publications that are applicable to this problem. MirHassani and Habibi provide a good survey of the recent work on Course and Examination Timetabling [1], Qu et al. provide an extensive survey of recent work in the related problem of Examination Timetabling [2]. 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.

2.2 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. 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 [3, 4].

2.3 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.

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 [5-8]. 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.

2.4 Heuristic Solutions
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. Tabu search methods are possibly the most popular; 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 [9].

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 [10].

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 [11].

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 [12]. 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. [13]

2.5 Our Contribution
In recent years, the focus of the Course Timetabling Problem has shifted extensively to the study of various 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.

3. Problem Definition and Solution Approaches

3.1 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.

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. However, in the scheduling research community, blocks refer to partitions of the available time into discrete elements that are repeated throughout the time being
scheduled. We will use the term block in 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.

3.2 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 problems 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 ideally surveyed 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 course.

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 enroll as many students as possible to 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 be 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. 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 desirable timetables.

### 3.3 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 the CBPM has so many variables, it takes a relatively long time to run, but the solutions are 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.

### 3.4 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 10 students want to take Dance and 10 students want to take Ceramics, but 3 of these students want to take both, then only 17 students would be able to enroll if both were offered at the same time.

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 hypergeometric 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 solve time.

### 3.5 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. Therefore, we did not develop additional algorithms or formulations to disaggregate the solutions, but there likely exist superior methods of accomplishing this task.

4. Formulation

4.1 Variable Definition

This section details the specifics of the formulation in mathematical notation. We will first 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.

\[ s_{ncb} = \begin{cases} 1 & \text{if student } n \text{ takes course } c \text{ at time } b \\ 0 & \text{otherwise} \end{cases} \]

\[ y_{cb} = \begin{cases} 1 & \text{if course } c \text{ is offered at time } b \\ 0 & \text{otherwise} \end{cases} \]

\[ x_{cb} = \text{number of students taking course } c \text{ at time } b \]

<table>
<thead>
<tr>
<th>Table 1: Table of Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
</tr>
<tr>
<td>( n \in N = {1,\ldots,N} )</td>
</tr>
<tr>
<td>( c \in C = {1,\ldots,C} )</td>
</tr>
<tr>
<td>( b \in B = {1,\ldots,B} )</td>
</tr>
<tr>
<td>( t \in T = {1,\ldots,T} )</td>
</tr>
<tr>
<td>( (c, t) \in \tau )</td>
</tr>
<tr>
<td>( c \in C )</td>
</tr>
<tr>
<td>( D \subseteq N )</td>
</tr>
<tr>
<td>( P \in \mathbb{P}(C) )</td>
</tr>
<tr>
<td>( s_{ncb} )</td>
</tr>
<tr>
<td>( y_{cb} )</td>
</tr>
<tr>
<td>( x_{cb} )</td>
</tr>
<tr>
<td>( \lambda_c )</td>
</tr>
<tr>
<td>( \gamma_c )</td>
</tr>
<tr>
<td>( \alpha_c )</td>
</tr>
</tbody>
</table>

4.2 Objective Function

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

\[ \text{Maximize } Z = \sum_{b \in B} \sum_{c \in C} \sum_{n \in N} s_{ncb} \]

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; this objective function is shown below in equation (5). It is worth noting that this function does not necessarily provide exact numbers on enrollment; 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 \( Z = \sum_{b \in B} \sum_{c \in C} \sum_{n \in N} S_{n \in b} \sum_{b \in B} \sum_{c \in C} x_{c \in b} \) 

(5)

4.3 Constraints

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

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

\[
\sum_{b \in B} y_{c \in b} \leq \gamma_c \quad \forall c \in C
\]

(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 \( \lambda_c \). Equation (8) represents the same capacity constraints for courses with only one section in the ASM.

\[
\sum_{n \in N} S_{n \in c \in b} - \lambda_c \cdot y_{c \in b} \leq 0 \quad \forall c \in C \quad \forall b \in B
\]

(7)

\[
x_{c \in b} - \lambda_c \cdot y_{c \in b} \leq 0 \quad \forall c \in C \quad \forall b \in B
\]

(8)

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

\[
\sum_{b \in B} S_{n \in c \in b} \leq \alpha_c \quad \forall n \in N \quad \forall c \in C
\]

(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.

\[
\sum_{c \in C} S_{n \in c \in b} \leq 1 \quad \forall n \in N \quad \forall b \in B
\]

(10)

\[
\sum_{c \in C} y_{c \in b} \leq 1 \quad \forall b \in B \quad \forall t \in T
\]

(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 classes when there are several being offered. The exact number of students that can enroll in at least one class is not fully constrained by this equation until the union extends to a tier past the number of classes that are offered in a block, which is not known before the problem is solved. However the equation does provide a good estimate for how many students can enroll in each block. For those unfamiliar with this notation, the equation roughly states that the sum of all students taking a subset of courses must be less than the number of students who want to take at least one of the courses for all times and all subsets of courses.

\[
\sum_{c \in P \in N \in W} S_{n \in c \in b} + \sum_{c \in P} x_{c \in b} \leq \bigcup_{c \in P} D_c \quad \forall b \in B \quad \forall P \in \mathbb{P}(C)
\]

(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.

5. Application to Local Schools

5.1 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 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. 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 [14], and the results, summarized in Table 2 below, show the ASM solving in less than half the time with a nearly identical solution.

<table>
<thead>
<tr>
<th></th>
<th>Complete Binary Programming Model</th>
<th>Aggregated Student Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Variables</td>
<td>2,760</td>
<td>1,500</td>
</tr>
<tr>
<td>Number of Constraints</td>
<td>1,193</td>
<td>2,652</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>12.98</td>
<td>7.37</td>
</tr>
<tr>
<td>% Solve Time of CBPM</td>
<td>100%</td>
<td>56.78%</td>
</tr>
<tr>
<td>Number of Students Enrolled</td>
<td>624</td>
<td>621</td>
</tr>
<tr>
<td>% Student Enrollment of CBPM</td>
<td>100%</td>
<td>99.52%</td>
</tr>
</tbody>
</table>

5.2 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 [14], and the results, summarized in Table 3 below, show a larger disparity in the quality of the solutions but a dramatically improved run time from the CBPM to the ASM.

<table>
<thead>
<tr>
<th></th>
<th>Complete Binary Programming Model</th>
<th>Aggregated Student Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Variables</td>
<td>55,456</td>
<td>35,184</td>
</tr>
<tr>
<td>Number of Constraints</td>
<td>14,201</td>
<td>15,283</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>10,231</td>
<td>366</td>
</tr>
<tr>
<td>% Solve Time of CBPM</td>
<td>100%</td>
<td>3.58%</td>
</tr>
<tr>
<td>Number of Students Enrolled</td>
<td>5,294</td>
<td>5,137</td>
</tr>
<tr>
<td>% Student Enrollment of CBPM</td>
<td>100%</td>
<td>97.0%</td>
</tr>
</tbody>
</table>
6. Conclusions

6.1 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 balance the accuracy and run time of the ASM and compare the results to the Complete Binary Programming Model.

We found that the ASM is successful when employed at institutions with a diverse range of student demand; this is expected because the CBPM runs faster when there is a large amount of symmetry in the student demand, while the ASM does not benefit from this symmetry in the same way. The ASM also excels where a high proportion of courses only have one section; this is because we only aggregated student data for courses with one section, so it is easy to see why the benefits of the ASM are enhanced when most courses only have one section.

We conclude that the ASM, like most solution strategies to the Course Timetabling Problem, is best 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.

6.2 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 both 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.
References