Salvaging College Registrations During COVID-19 via Integer Programming

Jeffrey D. Blanchard
Grinnell College
Grinnell, Iowa, USA
jeff@math.grinnell.edu
Marc Chamberland
Grinnell College
Grinnell, Iowa, USA
chamberl@math.grinnell.edu

Summary The onset of COVID-19 in the spring of 2020 disrupted academic schedules for many colleges, including course registration for the fall of 2020. We solve the problem of salvaging the majority of an existing semester course registration when separating the semester into two terms. The problem was solved using integer programming.

Introduction

In this article we discuss an optimization problem that resulted from altering an academic calendar in response to the COVID-19 pandemic. Specifically, we address the problem of partitioning a fully complete semester-length registration into two half-length terms while maximizing the number of individual student-course registrations and enforcing faculty teaching constraints. While timetabling and scheduling problems have been extensively applied to higher education, those techniques are typically part of a planning process and either ignore student preferences or focus on students taking prescribed sets of courses. In this problem, students have already signed up for a wide range of course combinations.

In March 2020, Grinnell College was among the first schools to decide to transition to remote teaching and learning for the remainder of the year [**1, 2**]. Among the first activities scheduled for April was the annual registration for the fall semester. With the future uncertain, the College, like many around the world, moved forward with its traditional registration, albeit remotely and online. Unlike a priority registration common to most colleges, Grinnell College utilizes an open registration system where all students have equal priority to all courses and course enrollments caps are initially ignored. The open registration is followed by a cut and balance period after which some students must replace any lost courses. Combined with active, individual advising for all students, the total communal work for students, faculty, and administration is substantial. In the spring of 2020, this communal work was intensified by the need to complete the work remotely.

The ongoing difficulties presented by the pandemic forced most colleges to at least consider alternative versions for their fall 2020 semester [**3, 4**]. Grinnell College, seeing a continuing trend of increased COVID-19 spread in the surrounding region, developed multiple strategies to have students back on campus. Ultimately, prioritizing the public health benefits and flexibility associated with bringing subgroups of the student population to campus, the College made a choice to split the usual 15 week semester into two 7.5 week terms. Many colleges made similar decisions regarding their semester schedule [**5, 6, 7**], but nearly all simply canceled their registration and started anew. In order to preserve much of the total communal work from the spring, Grinnell College decided to create their split semester schedule from the completed

registration. We set upon the task of assigning the courses to the terms in a way that would retain the greatest portion of the completed registration.

In an idealized setting, the full problem of assigning the courses to time slots over two terms could be formulated as a single constrained optimization problem known as timetabling [**8**]. In this real world setting, human factors suggested that the problem be solved in phases. For example, the administration wanted to allow faculty to see which courses would be assigned to each term and allow them to make changes. There were many additional factors, such as creating a new time schedule for courses, bringing students to campus in cohorts, students working remotely around the world, changing guidance from public health organizations, and the evolving status of the virus in the community. With all of these factors at play, we accomplished the task of splitting the semester into two half-length terms by modeling the problem as a two-phase optimization problem consisting of a penalized set partitioning [**9**] followed by a pair of timetabling problems [**8**].

Phase I: Semester Partitioning

The Problem The three main objects relevant to the problem are the sets of courses, $\boldsymbol{\mathcal{C}}$, returning students registered for the courses, $\boldsymbol{\mathcal{S}}$, and the faculty teaching the courses, $\boldsymbol{\mathcal{F}}$. We represent the size of these sets as $c = |\boldsymbol{\mathcal{C}}|$, $s = |\boldsymbol{\mathcal{S}}|$, and $f = |\boldsymbol{\mathcal{F}}|$, to indicate the number of courses, students, and teaching faculty, respectively. Throughout this discussion, we use the term *courses* to indicate every distinct section offered in the semester. Furthermore, a *registered seat* is an individual student registered for a specific course. In order to salvage the total communal work involved in the registration process and maintain continuity for the student body, the problem at hand is to assign each course to one of two terms and to do so in a way that preserves the greatest number of registered seats while satisfying constraints imposed by both students and faculty.

The assignment of each course into the two terms is a set partitioning problem [**9, 10, 11**] asking us to find two disjoint subsets of $\boldsymbol{\mathcal{C}}$ whose union is all of $\boldsymbol{\mathcal{C}}$. In other words, the courses needed to be assigned to two terms we dub $\boldsymbol{\mathcal{T}_1} = \{\text{courses assigned to term 1}\}$ and $\boldsymbol{\mathcal{T}_2} = \{\text{courses assigned to term 2}\}$ where

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{T}_1} \cup \boldsymbol{\mathcal{T}_2} \qquad \text{and} \qquad \boldsymbol{\mathcal{T}_1} \cap \boldsymbol{\mathcal{T}_2} = \emptyset. \tag{1}$$

The registration can be succinctly described in the language of matrices and vectors. For this assignment problem, we utilize two assignment vectors x, y which indicate the term to which each course, C_i for $i = 1, 2, \ldots, c$, is assigned:

$$x(i) = \begin{cases} 1 & C_i \in \boldsymbol{\mathcal{T}_1} \\ 0 & C_i \notin \boldsymbol{\mathcal{T}_1} \end{cases}, \qquad y(i) = \begin{cases} 1 & C_i \in \boldsymbol{\mathcal{T}_2} \\ 0 & C_i \notin \boldsymbol{\mathcal{T}_2} \end{cases}.$$

With these assignment vectors, and the binary nature of the assignment, we can restate the disjoint union (1) as the desire to find x and y in $\{0, 1\}^c$ such that

$$x + y = \mathbb{1}_c, \tag{2}$$

where $\mathbb{1}_k$ is the vector of length k containing all ones.

A student enrollment matrix, $S \in \{0, 1\}^{s \times c}$, indicates that student i is enrolled in course j when $S_{ij} = 1$; otherwise $S_{ij} = 0$. Students typically take up to four regular*

*The typical course at Grinnell College is four or five credits. While the College offers various one and two credit courses, we only consider four and five credit courses in the partitioning problem.

courses in a single semester. Thus, in the split, half-length terms, where weekly content and workload per course are doubled, a full academic load would be up to two regular courses while taking three courses in a single term is not allowed. The registration of student i, represented by the row S_i, and the partitioning defined by x and y should combine to assign at most two courses of student i to each term. In other words, for each $i = 1, 2, \ldots, s$, we want

$$\sum_{j:C_j \in \boldsymbol{\mathcal{T}_1}} S_{ij} = \sum_{j=1}^{c} S_{ij} \cdot x(j) = \langle S_i, x \rangle \leq 2 \quad \text{and}$$

$$\sum_{j:C_j \in \boldsymbol{\mathcal{T}_2}} S_{ij} = \sum_{j=1}^{c} S_{ij} \cdot y(j) = \langle S_i, y \rangle \leq 2.$$

Interpreting inequalities between vectors as component-wise inequalities, the full set of student constraints is

$$\begin{aligned} Sx &\leq 2 \cdot \mathbb{1}_s, \\ Sy &\leq 2 \cdot \mathbb{1}_s. \end{aligned} \tag{3}$$

Similarly, a faculty teaching matrix, $F \in \{0,1\}^{f \times c}$ indicates faculty member i is assigned to teach course j when $F_{ij} = 1$ with $F_{ij} = 0$ otherwise. Faculty teach no more than three courses in a normal semester, and, thus, we may divide the teaching faculty into three sets

$$\boldsymbol{\mathcal{F}}_k = \{\text{faculty teaching } k \text{ courses}\}, \text{ for } k = 1, 2, 3$$

with sizes $f_i = |\boldsymbol{\mathcal{F}}_i|$. Then we have $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}_1 \cup \boldsymbol{\mathcal{F}}_2 \cup \boldsymbol{\mathcal{F}}_3$ and $f = f_1 + f_2 + f_3$. Since we have partitioned the faculty set, we introduce two sub-matrices from the matrix F, namely

$$F_2 \in \{0,1\}^{f_2 \times c} \qquad \text{and} \qquad F_3 \in \{0,1\}^{f_3 \times c}$$

which consist of the rows of F corresponding to each subset of faculty. Faculty and administration agreed that individual teaching assignments should be spread evenly across the terms. Therefore, the faculty in $\boldsymbol{\mathcal{F}}_2$ should teach their courses one per term while the faculty in $\boldsymbol{\mathcal{F}}_3$ should teach two courses in a single term and one in the other. The group of faculty in $\boldsymbol{\mathcal{F}}_1$ do not impose a constraint, so all faculty constraints can be written as

$$\begin{aligned} F_2 x &= \mathbb{1}_{f_2}, \\ F_2 y &= \mathbb{1}_{f_2}, \\ \mathbb{1}_{f_3} \leq F_3 x &\leq 2 \cdot \mathbb{1}_{f_3}, \\ \mathbb{1}_{f_3} \leq F_3 y &\leq 2 \cdot \mathbb{1}_{f_3}. \end{aligned} \tag{4}$$

A final set of constraints for this problem is a selection of roughly 29% of courses that, for a variety of reasons imposed by the College, faculty, or administration, must appear in a specific term. For example, all incoming, first year students have the College's First Year Seminar as one of their courses in $\boldsymbol{\mathcal{T}_1}$. Since first year students are not included in $\boldsymbol{\mathcal{S}}$, and therefore these Seminars will not create student conflicts, the courses must be included in our problem to properly account for the faculty constraints (4). In terms of our assignment vectors, if C_i is a course that must be offered in $\boldsymbol{\mathcal{T}_1}$, we simply require $x(i) = 1$. These preassigned variables could, in fact, be removed

from the problem. For simplicity in identifying these constraints, and because well developed software implementations automatically perform the problem reduction, we simply list the preassigned courses as the equality constraints

$$\begin{aligned} P_1 x &= \mathbb{1}_{p_1}, \\ P_2 y &= \mathbb{1}_{p_2}, \end{aligned} \tag{5}$$

where P_1 and P_2 are the appropriate submatrices of the identity matrix to preassign the courses in the designated terms, and p_1, p_2 denote the number of courses preassigned to each term.

The simplest statement of our ideal problem to separate the registration into two terms is given by

$$\text{Find } x, y \in \{0,1\}^c \text{ subject to}$$

$$\begin{aligned} x + y &= \mathbb{1}_c, \\ P_1 x &= \mathbb{1}_{p_1}, \\ P_2 y &= \mathbb{1}_{p_2}, \end{aligned} \qquad \begin{aligned} Sx &\le 2 \cdot \mathbb{1}_s, \\ Sy &\le 2 \cdot \mathbb{1}_s, \end{aligned} \qquad \begin{aligned} F_2 x &= \mathbb{1}_{f_2}, \\ F_2 y &= \mathbb{1}_{f_2}, \\ \mathbb{1}_{f_3} \le F_3 x &\le 2 \cdot \mathbb{1}_{f_3}, \\ \mathbb{1}_{f_3} \le F_3 y &\le 2 \cdot \mathbb{1}_{f_3}. \end{aligned} \tag{6}$$

With the large number of constraints on this problem and the wide-ranging course combinations taken by students, we anticipated it would be unlikely to find a partition of the courses so that no constraint was violated. Indeed, (6) has no solution. Thus, we must relax the constraints by introducing penalties for violating the constraints. We can use these penalties to formulate a cost function that will be used to measure how close a partitioning comes to approximating this idealized setting.

Initial Heuristic Partitioning To quickly provide the administration with information for the decision process, preliminary work on the problem was intended to simply identify how much of the registration could possibly be salvaged when switching from a semester to two terms. A preliminary cost function was introduced to count the seats lost by a particular partitioning (i.e. violations of (3)). We developed a heuristic algorithm to find a feasible solution to the problem to obtain a lower bound on the percentage of registered seats saved. This algorithm was based on the intuitive methods that an individual would likely employ to solve the same problem with a smaller number of variables. The heuristic algorithm, similar to an implicit enumeration [**9**], completes three distinct stages. First, we proceed through the list of courses in random order, placing them into a term if no conflict arises for any constraint, or setting them aside if the course violates a constraint when assigned to either term. After passing through the list in this way, the courses set aside are forced into a term based on which one creates fewer conflicts. This process often created violations of the faculty teaching constraints which were then corrected even though it increased the total number of registered seats lost.

The full heuristic algorithm runs a few thousand times in a matter of seconds and selects the partitioning with the fewest registered seats lost (thereby also providing an initial feasible solution). This heuristic typically returns solutions that save approximately 87% of registered seats. The outcome from even the heuristic model exceeded expectations of the administration and academic planning committee, particularly when informed that further modeling and optimization would improve the results. The recognition that more than 87% of registered seats would be saved while gaining the proposed public health advantages of switching to the 7.5 week terms was

a significant factor in finalizing the decision to alter the calendar for the 2020-2021 academic year.

The Integer Program As mentioned above and as feared by most constituencies in the planning process, the partitioning of the courses into two sets with no student enrolled in more than two classes per term is already infeasible. The total communal work required to reschedule courses and start the registration process from scratch motivated our efforts to transform the problem to a minimization problem. The equality constraints assigning each course to one term (2) and pre-designating term assignment for a small set of courses (5) are non-negotiable and must remain intact. Fortunately, the remaining constraints can be relaxed via the introduction of a cost function on associated violations, and the problem formulated as an integer program (IP) [**10**].

First, recall that a *registered seat* is an instance of a single student registered for a particular course; each registered seat lost will require the communal work of finding and enrolling that student in a new course. We reframe the problem as a minimization of a cost function and replace the strict requirements from the previous sections with penalties. The cost function should increase whenever a partition causes a student to have more than two registered seats in a single term. Therefore, we introduce auxiliary cost variables associated to each student; let λ_x and λ_y be vectors representing the number of seats lost for each student in $\boldsymbol{\mathcal{T}_1}$ and $\boldsymbol{\mathcal{T}_2}$, respectively. Thus we replace the student constraints (3) with the following auxiliary constraints:

$$\begin{aligned} Sx - \lambda_x &\leq 2 \cdot \mathbb{1}_s, \\ Sy - \lambda_y &\leq 2 \cdot \mathbb{1}_s. \end{aligned} \tag{7}$$

The sum of the terms in λ_x and λ_y will represent the total number of registered seats lost. Since no student is registered for more than four courses, the penalties associated with student i naturally satisfy the bounds $\lambda_x(i) + \lambda_y(i) \leq 2$. For the sake of equity, we could have introduced an additional fairness constraint, $\lambda_x(i) + \lambda_y(i) \leq 1$, to ensure that no particular students bore the brunt of the cost. Ultimately, we decided that a manual course audit for each student forced to drop a course, combined with the relatively small number of students with $\lambda_x(i) + \lambda_y(i) = 2$, sufficiently mitigated against the need for this additional constraint.

Similarly, we replace the faculty constraints with penalized auxiliary constraints, letting $\mu_x, \mu_y \in \mathbb{R}^{f_2}$ indicate individual teaching assignment violations for faculty in $\boldsymbol{\mathcal{F}}_2$ and $\omega_x, \omega_y \in \mathbb{R}^{f_3}$ indicate such violations over $\boldsymbol{\mathcal{F}}_3$. Since $x(i) + y(i) = 1$ for each course C_i, violations of an upper bound for one constraint involving faculty teaching 3 courses, the last two lines of (4), force a violation of the lower bound for the other term; this combination renders the lower bounds unnecessary. Thus the faculty constraints (4) can be replaced with the auxiliary constraints

$$\begin{aligned} F_2x - \mu_x &\leq \mathbb{1}_{f_2}, \\ F_2y - \mu_y &\leq \mathbb{1}_{f_2}, \\ F_3x - \omega_x &\leq 2 \cdot \mathbb{1}_{f_3}, \\ F_3y - \omega_y &\leq 2 \cdot \mathbb{1}_{f_3}. \end{aligned} \tag{8}$$

We must determine how important the faculty constraints are when compared to the lost seats of the student constraints. Based on average class sizes of approximately 13 students we chose penalties of 10 registered seats for violations when teaching two courses and 20 registered seats when teaching three courses. Since faculty from $\boldsymbol{\mathcal{F}}_3$ teach two courses in a single term, it is clearly less of a problem to also permit this for

faculty in $\boldsymbol{\mathcal{F}}_2$ when it will save the registrations for many students. On the other hand, a violation of teaching assignments for faculty in $\boldsymbol{\mathcal{F}}_3$ will likely result in swapping faculty teaching duties, a potentially disruptive and costly undertaking. For simplicity, the model makes no effort to link multiple sections of the same course. Instead, any student violation which could be corrected by switching sections was accomplished in a manual post processing phase.

We define the objective function as a weighted sum represented by the inner product of all auxiliary penalty variables

$$\Lambda = \begin{bmatrix} \lambda_x \\ \lambda_y \\ \mu_x \\ \mu_y \\ \omega_x \\ \omega_y \end{bmatrix} \in \mathbb{R}^{2(s+f_2+f_3)} \text{ and the weights } W = \begin{bmatrix} \mathbb{1}_{2s} \\ 10 \cdot \mathbb{1}_{2f_2} \\ 20 \cdot \mathbb{1}_{2f_3} \end{bmatrix} \in \mathbb{R}^{2(s+f_2+f_3)}.$$

Then, the objective to be minimized is the weighted penalty

$$\langle \Lambda, W \rangle = \sum_{i=1}^{s} (\lambda_x(i) + \lambda_y(i)) + 10 \sum_{i=1}^{f_2} (\mu_x(i) + \mu_y(i)) + 20 \sum_{i=1}^{f_3} (\omega_x(i) + \omega_y(i)).$$

Equipped with these reformulations, we represent our partitioning as the following integer program:

$$\begin{aligned} \underset{\Lambda \in \mathbb{R}^{2(s+f_2+f_3)}}{\text{Minimize}} \ \langle \Lambda, W \rangle \ \text{ subject to} & \\ x \in \{0,1\}^c, & \qquad Sx - \lambda_x \leq 2 \cdot \mathbb{1}_s, \\ x + y = \mathbb{1}_c, & \qquad Sy - \lambda_y \leq 2 \cdot \mathbb{1}_s, \\ P_1 x = \mathbb{1}_{p_1}, & \qquad F_2 x - \mu_x \leq \mathbb{1}_{f_2}, \\ P_2 y = \mathbb{1}_{p_2}, & \qquad F_2 y - \mu_y \leq \mathbb{1}_{f_2}, \\ y, \mu_x, \mu_y, \omega_x, \omega_y \leq 1, & \qquad F_3 x - \omega_x \leq 2 \cdot \mathbb{1}_{f_3}, \\ \lambda_x, \lambda_y \leq 2, & \qquad F_3 y - \omega_y \leq 2 \cdot \mathbb{1}_{f_3}. \\ y, \lambda_x, \lambda_y, \mu_x, \mu_y, \omega_x, \omega_y \geq 0, & \end{aligned} \tag{9}$$

Solving an Integer Program In general, integer programming is NP-complete [**12**]. For some insight into how the minimizer* is found, consider the space in which the solution exists. A linear constraint defines a hyperplane dividing the ambient space into two half-spaces with all points satisfying an inequality constraint lying in the same half-space. Thus, the intersection of all such half-spaces defines the *feasible region* containing points which satisfy all the linear constraints. Now, the integer constraints further restricts the feasible points to those sitting on the integer lattice within this feasible region. The integer lattice points within the feasible region, the *integer feasible points*, contain all optimal solutions to an integer program.

Linear Program Relaxation Without the integer constraints, we have an associated, standard linear programming problem called the *LP relaxation*. The LP relaxation can be solved efficiently, for example by the simplex method. Essentially, the simplex

*We focus here on minimization to match our problem; the intuition applies analogously to maximization.

method uses the knowledge that a minimizer of the linear objective function must lie on the boundary of the feasible region. Considering this fact iteratively, we recognize that an optimal solution will exist on one of the vertices of the polytope defined by the constraints' hyperplanes. These vertices of the feasible region are known as *Corner Point Feasible* (CPF) solutions. The simplex method moves from one CPF solution to the next by traveling along the edge which most advantageously impacts the objective function. When it arrives at a CPF solution whose neighbors all have inferior objective function values, the algorithm selects that CPF solution as a local minimum. The linear nature of the problem assures us this local optimum is the global optimal solution to the linear program.

Bounding the Integer Program The fundamental idea for solving an Integer Program is to ignore the integer constraints and solve the LP relaxation. While rounding the continuous solution can lead to disastrous outcomes*, the solution to this relaxation provides valuable information. First of all, if we are lucky enough to have a solution exclusively composed of integer values we have solved the IP. Barring this unlikely event, we do know that the minimizer of this associated relaxation must have an objective value less than any solution that also satisfies the integer constraints. Thus, we have a lower bound on the optimal IP objective value. Likewise, by evaluating the objective function at any integer feasible point, we obtain an upper bound on the optimal IP objective value.

Cutting Planes When the LP relaxation yields a non-integer solution, a new linear constraint can be added to the problem. This new constraint, called a *cutting plane*, updates the LP relaxation to cut out the non-integer solution from the feasible region without removing any integer feasible solutions. Gomory [**13**] showed that adding a finite number of cutting planes produces a linear program with an integer solution, and therefore solves the IP. This method, unfortunately, is typically inefficient as the number of cutting planes may be excessively large. In modern algorithms, cutting planes are generally used in initial phases of the problem solving process.

Branch-and-Bound The most common approach for solving an integer program is called *branch-and-bound*. When we solve the LP relaxation, we establish upper and lower bounds on the minimum objective value for an integer solution. When the solution to that relaxation has a non-integer value, e.g. $x_i \in (n, n+1)$, we formulate two new problems, or *branches*, which separate the feasible region into two parts. The first branch includes the new constraint $x_i \leq n$ and the second branch adds the constraint $x_i \geq n+1$. These branches remove a portion of the feasible region but preserve all integer feasible solutions.

At this point, LP relaxations associated to each branch provide updated information on the bounds of the optimal objective value for the IP. If the new LP relaxation objective value is larger than the upper bound on the IP objective value, no point in that branch can improve upon on the current IP objective value, and the entire branch is abandoned. Otherwise, with its restricted feasible region, the relaxation for a branch updates the lower bound on the IP objective value. If the solution to the relaxation also has non-integer values, the branching process is applied again to create two new branches.

Branch-and-bound organizes a search of the feasible region and the associated bounds provide an interval in which the optimal objective value exists. If the fea-

*It is possible to formulate problems where rounding the relaxation's solution is as far from optimal as desired both in terms of the solution and objective value, not to mention the common case where the rounded solution is not even a feasible point.

	Random Perturb.	Cutting Planes	Branches Created	Relaxations Solved	Simplex Iterations	Optimality Gap (%)
A	No	2	1,348,205	1,266,138	409,986,094	14.62
B	No	4	1,417,327	1,325,648	409,435,633	14.41
C	Yes	5	2,380,516	2,242,821	769,773,065	11.97
D	Yes	8	2,532,475	2,376,589	778,706,493	12.02

TABLE 1: Computational Data reported by Mosek after a timed termination of 8.5 hours working on (9). Instances A and B did not perturb the objective function.

sible region is bounded, this process will eventually terminate, but the exponential growth on the number of branches makes it computationally impractical to complete the search for the optimal solution. Fortunately, the length of interval defined by the bounds, known as the *optimality gap*, can be used to decide when to stop. Once our gap is within some chosen tolerance, we can take our best known objective value and declare the outcome a near optimal solution.

Random Perturbation An integer program with equal weights for the cost variables is prone to having a high degree of symmetry [**14**]. This symmetry often means many solutions exist where trading some small set of components from one solution to another will not impact the objective function. When a CPF solution has many neighbors with the same objective function value, the search can get bogged down selecting which direction to travel in search of improving the objective value. While more sophisticated approaches exist for dealing with symmetry [**14**], adding positive, random values to all weights can perturb the objective function [**15**] and often breaks the symmetry.

Implementation and Results The final data set consisted of $f = 197$ faculty teaching $c = 358$ courses comprising 4542 registered seats for $s = 1297$ distinct students. Of these faculty, there were $f_2 = 91$ faculty teaching two courses and $f_3 = 37$ teaching three courses. The College designated $p_1 = 71$ courses* to be fixed in $\boldsymbol{\mathcal{T}_1}$ with $p_2 = 33$ courses fixed in $\boldsymbol{\mathcal{T}_2}$. The optimization problems were implemented in Matlab [**19**] while employing the Mosek Optimization Toolbox for Matlab [**16**].

We began by running two instances of the algorithm, one with and one without random perturbation†. The randomization inherent in the heuristic algorithm and the randomly weighted objective function ensured that we had two distinct integer feasible solutions when stopping the method after 8 hours. Each of the solutions was then presented as an initial feasible solution in two more instances of the algorithm, one with and one without random perturbation. Mosek then worked on each the problem for an additional 8.5 hours. In Table 1 we see that every instance performed hundreds of millions of simplex iterations to solve millions of LP relaxations from millions of branches. In all instances, the chosen integer feasible solution was found in the first four hours, with the additional work shrinking the optimality gap by solving LP relaxations in the branch-and-bound process.

Near Optimal Partitions At the culmination of this process, we had four feasible solutions outlined in Table 2. As with most large integer programs, we did not expect to certify a particular feasible solution as the optimal solution. With the naive heuristic algorithm providing initial feasible solutions preserving 87% of registered seats,

*The apparent lopsided nature of the number of fixed courses is related to the 31 First Year Seminars preassigned to $\boldsymbol{\mathcal{T}_1}$.

†In our implementation, we perturbed the objective with absolute values of samples from Matlab's normal pseudorandom number generator, `randn`.

	# Sections $\mathcal{T}_1$	# Sections $\mathcal{T}_2$	Seats Saved (%)	Students Unchanged (%)	Faculty Violations
A	195	163	90.66	69.93	5
B	192	166	90.82	70.62	4
C	192	166	91.00	71.24	6
D	196	162	90.62	70.39	3

TABLE 2: Four Near Optimal Solutions: Number of sections assigned to each term, percentage of seats saved, percentage of students who do not need to make any changes to their registration, and number of two course faculty constraints violated. (No solutions had a three course faculty violation.)

we ran Mosek [**16**] on the relaxed problem which ignored faculty constraints (8) to obtain a clear upper bound of salvaging at most 93% of registered seats. With time constraints and anticipated alterations to the partition from the faculty approval process, we decided to accept any solution salvaging 90% of registered seats as a near optimal solution [**17, 18**].

We selected Option D despite it having the lowest percentage of seats saved. The reasoning was twofold. First, the characteristics listed in Table 2 are roughly equivalent for each of the options, except for the faculty constraint violations. In particular, Option D had the fewest such violations which will either require the faculty to be reassigned or a course to be moved. Second, each of the faculty violations associated with Option D would have moved a class from $\mathcal{T}_1$ to $\mathcal{T}_2$ and therefore would further balance the offerings. The apparent lopsided nature of the splitting was related to the College's First Year Seminar all being pre-assigned to $\mathcal{T}_1$ and having no student conflicts with any other courses in the registration. Option D provided the best balance between registered seats saved, percentage of students with intact registrations, balancing of sections across terms, and minimal faculty constraint violations. Option D was designated as our preferred solution for the semester partitioning problem (9) saving more than 90% of registered seats and preserving the entire registration of over 70% of the student population even after rectifying the three faculty constraint violations.

Manual Post Processing The near optimal solution was translated into what was deemed the *seeded schedule* and presented to the faculty. The College, seeking to maintain the ethos of a faculty controlled curriculum and course schedule, presented the seeded schedule to departments for their review, revision, and potential faculty reassignments. While the overwhelming majority of course sections remained in their assigned term, various factors compelled some courses to be swapped between terms, canceled, or moved to the spring. Unfortunately, the relatively small number of manual changes had a significant impact decreasing the percentage of seats saved to 84.39%. The required faculty approval of the schedule prevented reformulating the optimization problem with this new information to improve the final solution. After departmental changes were finalized, the registrar staff conducted an academic audit to find the most academically advantageous schedule alterations for the 30% of students for whom the partition caused a conflict.

The Final Partition The preferred solution, Option D, saved 4116 of the 4542 registered seats. In addition to the 426 seats lost in finding this near optimal solution, 283 seats were lost in the post processing. The majority of these secondary lost seats stemmed from moving or canceling courses. At the end of all post-processing, the changes to the seeded schedule (Option D) combined with balancing, cutting, and leaves of absence, the conflict-free, split-semester registration retained 3833 of the

original 4542 registered seats (84.39%). Moreover, less than one third of students had one of their courses dropped and only 3% of students lost two courses.

Phase II: Assigning Time Slots

As mentioned in the Introduction, an idealized timetabling problem would simply assign all courses to both the term and the time-slot from the outset. In our situation, the governance process imposed that we solve the problem in two phases as the term assignments needed approval, and, at the start of the problem, no time slots had been approved for the accelerated terms. A typical semester course lasts 15 weeks. For the split semester schedule, the terms are 7.5 weeks, necessitating twice as much meeting time per week. Thus, the typical College time slots could not be used. Ultimately, seven time slots listed in Table 3 were chosen to cover three general time blocks of morning (t_1, t_2, t_3), afternoon (t_4, t_5, t_6), and evening (t_7). Both the morning and afternoon blocks were divided into two 110-minute time slots; a third 230-minute time slot designated for laboratories spans the full block. The evening block could not reasonably support two 110-minute time slots, so a single evening time slot was designated with laboratory courses permitted.

Morning		Afternoon		Evening	
(110 min.)	(230 min.)	(110 min.)	(230 min.)	(110 min.)	(230 min.)
t_1 t_2	t_3	t_4 t_5	t_6	t_7	

TABLE 3: Time slots: Within each general time slot of morning, afternoon, or evening were full time period laboratory time slots conflicting with shorter time slots; the morning and afternoon included two non-conflicting 110 minute time slots.

The Time Slot Problem Again the primary objective is the preservation of registered seats. To preserve seats, courses must be assigned to time slots with no student assigned to two courses in the same time slot. To ensure faculty avoid time conflicts, the problem is also constrained by the faculty. However, the faculty constraints satisfied by the near optimal partitioning from Phase I ensured that very few faculty were teaching more than one course in either $\boldsymbol{\mathcal{T}_1}$ or $\boldsymbol{\mathcal{T}_2}$; thus, the faculty constraint is not substantial. Finally, since first year students had not yet registered for courses, all sections of the First Year Seminar were removed* from this time slot assignment phase.

This problem must be solved for each term $\boldsymbol{\mathcal{T}_1}$ and $\boldsymbol{\mathcal{T}_2}$, but the problem formulation is identical. We let $\tilde{S}$ denote the student enrollment matrix with columns representing only those courses assigned to the relevant term. Similarly, $\tilde{F}_2$ is the matrix of faculty assignments only for those faculty teaching two courses in the current term. Likewise, we let $\tilde{c}, \tilde{s}, \tilde{f}_2$ represent the number of courses, students, and faculty teaching two courses in the appropriate term. To assign each course to a single time slot, we start with the problem

*In a standard academic year, First Year Seminars are all assigned the same protected time slot. In this splitting, faculty teaching Seminar were allowed to select their time slot after all other courses were given their designated time slot. This provided a range of time slots for first year students learning online and spread around the world.

$$
\begin{aligned}
&\text{Find } t_1, t_2, t_3, t_4, t_5, t_6, t_7 \in \{0,1\}^{\tilde{c}} \text{ subject to} \\
&\quad t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 = \mathbb{1}_{\tilde{c}}, \\
&\quad \tilde{S}(t_1 + t_3) \leq \mathbb{1}_{\tilde{s}}, \qquad \tilde{F}_2(t_1 + t_3) \leq \mathbb{1}_{\tilde{f}_2}, \\
&\quad \tilde{S}(t_2 + t_3) \leq \mathbb{1}_{\tilde{s}}, \qquad \tilde{F}_2(t_2 + t_3) \leq \mathbb{1}_{\tilde{f}_2}, \\
&\quad \tilde{S}(t_4 + t_6) \leq \mathbb{1}_{\tilde{s}}, \qquad \tilde{F}_2(t_4 + t_6) \leq \mathbb{1}_{\tilde{f}_2}, \\
&\quad \tilde{S}(t_5 + t_6) \leq \mathbb{1}_{\tilde{s}}, \qquad \tilde{F}_2(t_5 + t_6) \leq \mathbb{1}_{\tilde{f}_2}.
\end{aligned} \tag{10}
$$

Faculty Preference and Laboratory Constraints Faculty were asked to indicate their preference of morning, afternoon, or evening; alternatively, faculty could indicate a willingness to teach in any time slot or only during the daytime (only morning or afternoon). No other combinations of preference were submitted. Only courses with a designated laboratory or studio component were designated for the lab time slots. Similar to the definition of the pre-assignment matrices in (5), we define five matrices which preassign a course to a specific category of time slot: mornings M, afternoons A, evenings E, daytime D, and laboratory L. The number of courses for each designation, i.e. the number of rows of these matrices, is denoted by the corresponding lower case letter: m, a, e, d, l.

For example, M is the sub-matrix of the $\tilde{c} \times \tilde{c}$ identity matrix obtained by extracting those rows corresponding to the courses with the morning designation. The other matrices are defined similarly. Clearly, a course assigned to the morning must have a 1 in exactly one of the first three vectors, t_1, t_2, t_3. Thus, this morning designation constraint is simply $M(t_1 + t_2 + t_3) = \mathbb{1}_m$. The following is the full set of pre-assignment equality constraints for a given term:

$$
\begin{aligned}
M(t_1 + t_2 + t_3) &= \mathbb{1}_m, \\
A(t_4 + t_5 + t_6) &= \mathbb{1}_a, \\
E(t_7) &= \mathbb{1}_e, \\
D(t_1 + t_2 + t_3 + t_4 + t_5 + t_6) &= \mathbb{1}_d, \\
L(t_3 + t_6 + t_7) &= \mathbb{1}_l.
\end{aligned} \tag{11}
$$

The Optimization Problem This strict time slot problem has no feasible solution since every assignment results in some student conflicts. To solve the problem and minimize the loss of registered seats, we formulate the problem as a timetabling problem [**8, 20, 21**], which considers student conflicts a soft constraint incurring a penalty. Due to challenges with teaching remotely and online, faculty preferences were deemed non-negotiable (a hard constraint) and the faculty preferences were not relaxed. Likewise, the courses with a laboratory component must be placed into a laboratory designated time slot. Additionally, the number of faculty constraints were small enough that these constraints were left intact. Therefore the time slot problem was formulated as a timetabling problem to minimize the number of seats lost with the auxiliary cost variables associated only with student conflicts. The four time slot conflicts from Table 3 were assigned a vector λ_j containing the number of registered seats lost for each student. We solve the minimization problem

$$\underset{\Lambda\in\mathbb{R}^{4\tilde{s}}}{\text{Minimize}}\,\langle\Lambda,\mathbb{1}_{4\tilde{s}}\rangle=\sum_{i=1}^{\tilde{s}}\sum_{j=1}^{4}\lambda_j(i)\quad\text{subject to}$$

$$\begin{aligned}
t_1,t_2,t_3,t_4,t_5,t_6&\in\{0,1\}^{\tilde{c}},\\
t_1+t_2+t_3+t_4+t_5+t_6+t_7&=\mathbb{1}_{\tilde{c}},\\
M(t_1+t_2+t_3)&=\mathbb{1}_m,\\
A(t_4+t_5+t_6)&=\mathbb{1}_a,\\
E(t_7)&=\mathbb{1}_e,\\
D(t_1+t_2+t_3+t_4+t_5+t_6)&=\mathbb{1}_d,\\
L(t_3+t_6+t_7)&=\mathbb{1}_l,\\
t_7,\lambda_1,\lambda_2,\lambda_3,\lambda_4&\geq 0,
\end{aligned}
\qquad
\begin{aligned}
\tilde{S}(t_1+t_3)-\lambda_1&\leq\mathbb{1}_{\tilde{s}},\\
\tilde{S}(t_2+t_3)-\lambda_2&\leq\mathbb{1}_{\tilde{s}},\\
\tilde{S}(t_4+t_6)-\lambda_3&\leq\mathbb{1}_{\tilde{s}},\\
\tilde{S}(t_5+t_6)-\lambda_4&\leq\mathbb{1}_{\tilde{s}},\\
\tilde{F}_2(t_1+t_3)&\leq\mathbb{1}_{\tilde{f}_2},\\
\tilde{F}_2(t_2+t_3)&\leq\mathbb{1}_{\tilde{f}_2},\\
\tilde{F}_2(t_4+t_6)&\leq\mathbb{1}_{\tilde{f}_2},\\
\tilde{F}_2(t_5+t_6)&\leq\mathbb{1}_{\tilde{f}_2}.
\end{aligned}
\tag{12}$$

Implementation and Results The integer program for each term's time slot assignment problem (12) was both smaller and less constrained than the set partitioning problem (9) in Phase I. Mosek [**16**] certified an optimal solution for each term in a matter of seconds; the simpler problems were more amenable to cutting plane reductions and combined for roughly half of a million simplex iterations. These solutions preserved 1758 of 1829 (96.12%) registered seats in $\boldsymbol{\mathcal{T}_1}$ and preserved 1898 of 2004 (94.71%) registered seats in $\boldsymbol{\mathcal{T}_2}$. Thus, the assignment into time slots, even when permitting faculty to select general times of day, only resulted in the loss of 177 additional seats. Combined with the semester partitioning of Phase I, the near optimal solution derived from this two-phase, integer program preserved 3656 of 4542 registered seats.

	Cutting Planes	Branches Created	Relaxations Solved	Simplex Iterations	Time (sec.)	Seats Lost
$\boldsymbol{\mathcal{T}_1}$	21	11,493	8,585	354,513	26.04	71
$\boldsymbol{\mathcal{T}_2}$	34	4,396	3,424	176,680	14.25	106

TABLE 4: Computational Data reported by Mosek on the Time Slot Problem (12). The optimal solution was found for each problem in less than 30 seconds.

Conclusion

With regional public health measures indicating the potential for significant spread of COVID-19, Grinnell College joined other colleges in altering their academic calendar from a traditional semester to two terms of half the length. This change presented many challenges including how to salvage the significant effort already given to course registration. By approaching the problem as an optimization problem and formulating the task as a series of integer programs, the College was able to salvage over 80% of registered seats, exceeding expectations.

Academic challenges and uncertainty caused by COVID-19 saw enrollments decline across much of higher education [**22, 23**]. While the number of students taking a leave of absence at Grinnell College increased from previous years, the increase was less dramatic than at many similar institutions. Anecdotal information suggests that preserving the full academic plan for over two-thirds of the student body, and the majority of courses for nearly all students, contributed substantially to their decisions to return for the split-term fall academic period. While priority registration systems at other institutions enabled them to discard completed registrations and start fresh, it is possible such a process failed to account for the effort and work of students, faculty, and staff engaging in two registrations. As the current COVID-19 pandemic is unlikely to be the last challenge faced by higher education, employing integer programming to salvage completed registration, as was done here, may improve future responses to academic disruptions.

Acknowledgements

The authors gratefully acknowledge the significant contributions of Jason Maher, Grinnell College Registrar, throughout the project. They thank Andrew Beveridge, Barry Thomas, and Coralia Cartis for conversations helping with model formulation, perturbation heuristics, and framing the problem in the literature. Lastly, the authors thank the anonymous referee for critique and recommendations that improved the article.

REFERENCES

1. A. Cannon, A. Breaux, To stave off coronavirus' spread, Grinnell College cancels in-person classes for rest of year, instructs students to leave campus, Des Moines Register, Mar. 10, 2020. URL `https://www.desmoinesregister.com/story/news/health/2020/03/10/coronavirus-iowa-Generic-college-asks\\-students-leave-campus/5012500002/`
2. E. Redden, Colleges ask students to leave campuses, Inside Higher Ed, Mar. 11, 2020. URL `https://www.insidehighered.com/news/2020/03/11/harvard-cornell-mit-and-others-ask-students-leave-campus-due-coronavirus`
3. E. J. Maloney, J. E. Kim, 15 fall scenarios, Inside Higher Ed, Apr. 22, 2020. URL `https://www.insidehighered.com/digital-learning/blogs/learning-innovation/15-fall-scenarios`
4. B. McMurtrie, The next casualty of the coronavirus crisis may be the academic calendar, Chronicle of Higher Education, Apr. 16, 2020. URL `https://www.chronicle.com/article/the-next-casualty-of-the-coronavirus-crisis-may-be-the-academic-calendar/`
5. E. Redden, Beloit redesigns its academic calendar to give itself more flexibility if covid-19 forces closures, Inside Higher Ed, Apr. 20, 2020. URL `https://www.insidehighered.com/news/2020/04/20/beloit-redesigns-its-academic-calendar-give-itself-more-flexibility-if-covid-19`
6. Staff, Flexible approach to fall, Macalester College, Aug. 18, 2020. URL `https://www.macalester.edu/news/2020/08/flexible-approach-to-fall/`
7. C. Roepke, Administration and faculty present "module-style" learning proposal for fall 2020, Mount Holyoke News, May 6, 2020. URL `https://www.mountholyokenews.com/news/2020/5/6/administration-and-faculty-present-module-style-learning-proposal-for-fall-2020`
8. D. de Werra, An introduction to timetabling, European Journal of Operational Research 19 (1985) 151–162. doi.org/10.1016/0377-2217(85)90167-5
9. E. Balas, M. W. Padberg, Set partitioning: A survey, SIAM Rev. 18 (4) (1976) 710–760. doi.org/10.1137/1018115
10. S. Chopra, M. R. Rao, The partition problem, Mathematical Programming 59 (1) (1993) 87–115. doi.org/10.1007/BF01581239
11. F. S. Hillier, G. J. Lieberman, Introduction to Operations Research, 8th Ed., McGraw-Hill, New York, 2005.
12. R. M. Karp, Reducibility among Combinatorial Problems. In: Miller R.E., Thatcher J.W., Bohlinger J.D. (eds) Complexity of Computer Computations. Springer, Boston, MA. (1972) 85–103. doi.org/10.1007/978-1-4684-2001-2_9
13. R. E. Gomory, Outline of an algorithm for integer solutions to linear programs, Bull. Amer. Math. Soc. 64 (1958) 275–278. doi.org/10.1090/S0002-9904-1958-10224-4

14. F. Margot, Symmetry in Integer Linear Programming, In: J unger M. et al. (eds) 50 Years of Integer Programming 1958-2008. Springer, Berlin, Heidelberg, (2010) 647–686. doi.org/10.1007/978-3-540-68279-0_17
15. A. Ghoniem, H. D. Sherali, Defeating symmetry in combinatorial optimization via objective perturbations and hierarchical constraints, IIE Transactions 43 (8) (2011) 575–588. doi.org/10.1080/0740817X.2010.541899
16. Mosek ApS, The MOSEK optimization toolbox for MATLAB manual. Version 9.2.10. (2019). URL `http://docs.mosek.com/9.2/toolbox/index.html`
17. T. Serra, J.N. Hooker, Compact representation of near-optimal integer programming solutions, Mathematical Programming, 182 (1) (2020) 199–232. doi.org/10.1007/s10107-019-01390-3
18. P. Voll, M. Jennings, M. Hennen, N. Shah, A. Bardow, The optimum is not enough: A near-optimal solution paradigm for energy systems synthesis, Energy, 82 (2015) 446–456. doi.org/10.1016/j.energy.2015.01.055
19. The MathWorks Inc., Matlab (Version 8.6, 2015b).
20. V. Pereira, H. Gomes Costa, Linear integer model for the course timetabling problem of a faculty in Rio de Janeiro, Advances in Operations Research (2016) 7597062. doi.org/10.1155/2016/7597062
21. M. Oktavia, A. Aman, T. Bakhtiar, Courses timetabling problem by minimizing the number of less preferable time slots, IOP Conf. Ser.: Mater. Sci. Eng. 166 (2017) 012025. doi.org/10.1088/1757-899x/166/1/012025
22. T. Sedmak, Fall 2020 college enrollment declines 2.5%: Nearly twice the rate of decline of fall 2019, National Student Clearinghouse, Dec. 17, 2020. URL `https://www.studentclearinghouse.org/blog/fall-2020-college-enrollment-declines-2-5-nearly-twice-the-rate-of-decline-of-fall-2019/`
23. D. Berrett, Fall's enrollment decline now has a final tally. Here's what's behind it., Chronicle of Higher Education, Dec. 17, 2020. URL `https://www.chronicle.com/article/falls-enrollment-decline-now-has-a-final-tally-heres-whats-behind-it`

JEFF BLANCHARD (MR Author ID: 856216) is an associate professor of mathematics at Grinnell College. He has been an NSF International Research Fellow, a Project NExT Fellow, and a Harris Faculty Fellow. His primary research areas are compressed sensing, sparse approximation, and high performance computing with graphics processing units.

MARC CHAMBERLAND (MR Author ID: 335642) is the Myra Steele Professor of Mathematics at Grinnell College. He has published in various research areas, including differential equations, number theory, classical analysis, and experimental mathematics. He has also sought to popularize math with a book, mathematical artwork, and a YouTube channel. A current project is writing a book on the number π.