August 2022

University Course Scheduling During a Pandemic and University Course Planning: Math Models and Heuristic Algorithms

Mohammad Khamechian
University of Wisconsin-Milwaukee

UNIVERSITY COURSE SCHEDULING DURING A PANDEMIC AND UNIVERSITY COURSE PLANNING: MATH MODELS AND HEURISTIC ALGORITHMS

by

Mohammad Khamechian

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Engineering

at

The University of Wisconsin-Milwaukee

August 2022

ABSTRACT

UNIVERSITY COURSE SCHEDULING DURING A PANDEMIC AND UNIVERSITY COURSE PLANNING: MATH MODELS AND HEURISTIC ALGORITHMS

by

Mohammad Khamechian

The University of Wisconsin-Milwaukee, 2022
Under the Supervision of Professor Matthew Petering

This dissertation has two chapters. In Chapter 1, we introduce two optimization problems related to university course planning. In the student course planning problem (SCPP), a student needs to design a course plan that allows him/her to graduate in a timely manner. In the department course planning problem (DCPP), an academic department needs to decide which courses to offer during which semesters to facilitate students' timely graduation. Mathematical models of these problems are developed, coded in C++, and solved with IBM ILOG CPLEX. Experiments on small, medium-sized, and large real-world and fictional problem instances show the utility of the math model.

Chapter 2 is about university course scheduling during a pandemic. Most universities have responded to the COVID-19 pandemic by offering courses in three formats: (1) online, (2) hybrid (with online and in-person components), or (3) in-person. Option 1 discourages student interaction; option 2 has low classroom utilization; and option 3 poses health risks or is limited to small courses meeting in large rooms. We propose a new approach to course scheduling which allows more than one classroom to be assigned to the same course. Our method allows all courses—even the

largest—to have a limited number of socially distanced, in-person meetings each semester in which all students in the course meet in multiple classrooms simultaneously. A math model and heuristic method are developed for implementation. Analyses of life-sized problem instances are promising.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my profound appreciation to my supervisor Professor Matthew Petering who generously offered his precious time and expertise in guiding and mentoring me step by step through the entire research process. Doctor Petering, your professional supervision, insightful comments and continuous encouragement at every stage of this study made this thesis possible for which I will always be grateful. I would also like to extend my special acknowledgement to the esteemed committee members/readers of my thesis for their time and constructive comments during the defense session.

Words fail me to express my heartfelt appreciation to my parents, whose unconditional love and continuous support have always paved the way toward my achievements. It is their constant encouragement and support from the very beginning of my life that made it possible for me to reach this stage. I will be eternally grateful to you for always believing in me and for your endless love, blessings, and support. I love you both from the bottom of my heart. I also wish to thank my brother, Mohsen, who has always been my best friend and whose everlasting love and support brighten up my life. I love you dearly and I thank you for being my loving and caring brother.

Last but not least I would like to express my heartfelt thanks to the love of my life, Sepideh, for her unwavering love and constant encouragement. Thank you for accompanying me throughout the whole process. Your love and consideration mean the world to me and give me courage to face challenges and never give up.

Chapter 1: A mathematical modeling approach to university course planning

1.1. Introduction

Education is fundamental to human civilization. The relationship between student and teacher is older than history itself, and the importance of formal education within a modern society is well recognized (Smith et al., 2017). The world's institutions of higher learning—universities and colleges—now number more than 26,000, and this number continues to rise (Sowter, 2017). The work performed at these institutions—teaching, research, and service—improves the human condition both globally and locally (Winters, 2011).

As universities become more numerous, many educational in situations are searching for better ways to use their limited resources—faculty members, staff members, physical infrastructure, time, and money—to serve students. In the United States, the desire to improve efficiency is partly motivated by recent cuts in higher education spending (Mitchell, Leachman, & Masterson, 2016). Today, many universities are realizing that they need to deliver educational programs in more efficient ways and if they wish to prosper in the future.

One aspect that affects the efficiency of a university is the way in which its courses are planned and scheduled. The structure of university degree programs, the courses offered each semester, and the assignment of courses to classrooms each semester all impact the student experience which in turn affects a university's reputation, ability to attract funding, and bottom line.

In this chapter, we introduce two optimization problems that relate to the productivity of a university and its students. In the student course planning problem (SCPP), a student needs to design one or more course plans that allow him/her to graduate in a timely manner. In the

department course planning problem (DCPP), an academic department needs to decide which courses to offer during which semesters to facilitate students' timely graduation. Mathematical models of these problems are developed, coded in C++, and solved with IBM ILOG CPLEX. Experiments on small, medium-sized, and large real-world and fictional problem instances demonstrate the math model's utility.

1.2. Literature review

The literature relevant to this study includes all papers which introduce methods for automatically doing university course scheduling, university course planning, and resource-constrained project scheduling (Fig. 1).

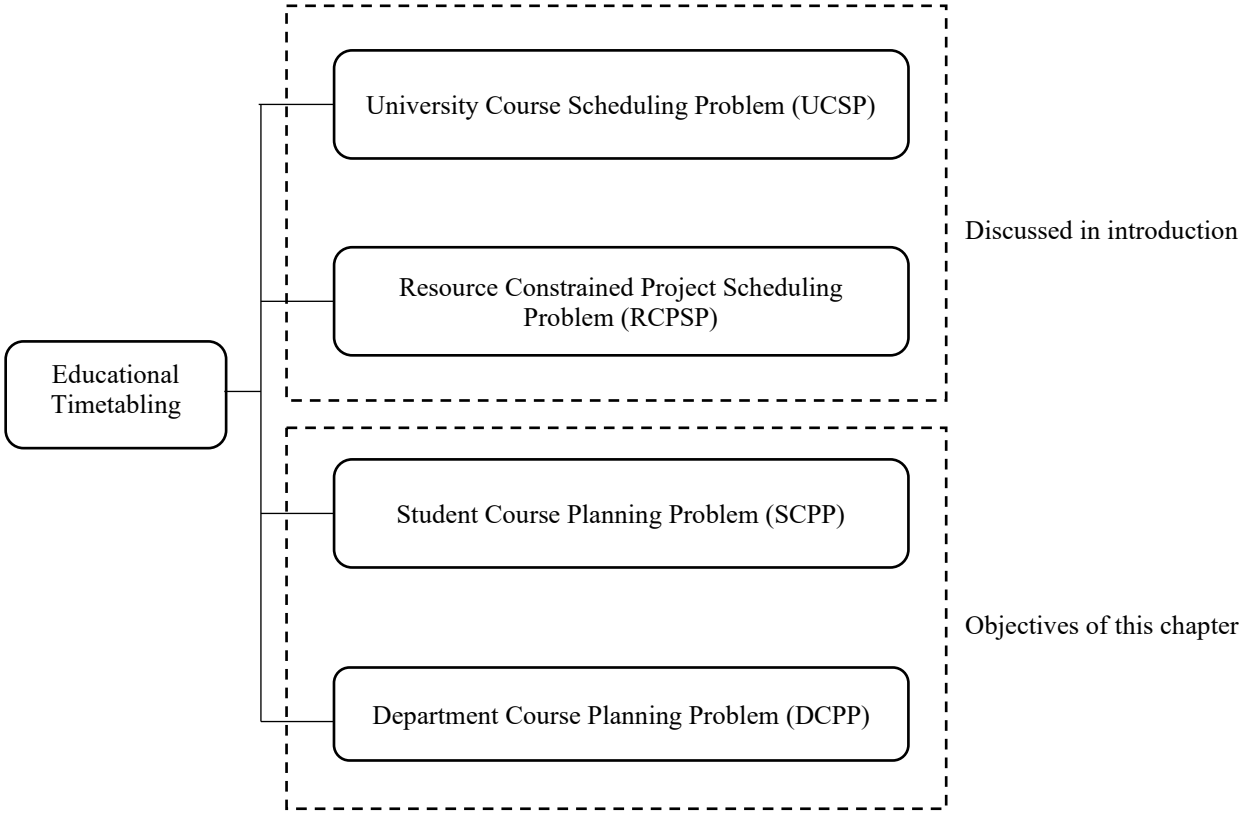

Figure 1. Branches of educational timetabling

The goal in university course scheduling is to assign university courses and laboratory sessions to classrooms and timeslots considering each room's maximum capacity, the expected number of students in a course, and other related facility issues. This topic is only tangentially related to this chapter and will be discussed in Chapter 2.

A handful of published papers introduce math models and/or algorithms for university course planning. Esteban, Zafra, and Romero (2020) develop a genetic algorithm to recommend elective courses to computer science students at the University of Cordoba. Mohamed (2015) introduces an integer program for long-term course planning (LTCP) that decides which courses to take during which semester so that a student's satisfaction and grade point average is maximized upon graduation. Morrow, Hurson, and Sarvestani (2017) propose a hierarchical, multi-stage algorithm for designing a personalized, multi-semester course plan for a student considering the degree requirements, the student's personal interests, the time required to earn the degree, and expected course grades. Shakhsi-Niaei and Abuei-Mehrizi (2020) present an integer program for deciding which courses a student takes during which semester with the goal of finishing a degree while taking his/her preferred elective courses, balancing total course complexity across the semesters, and balancing the number of credits taken across the semesters. The work described in this chapter considers university course planning from both student and department perspectives, and it has recently been published in an academic journal (Khamechian & Petering, 2022).

Our investigation of the student course planning problem (SCPP) in this paper has some overlap with the preceding three papers, but our work differs from these papers in several aspects. First, we provide a clear, thorough description of the problem. Second, we consider a different objective: minimizing the time to graduation. Third, we consider more than ten times as many problem instances as any other paper. To our knowledge, no existing work besides Khamechian

& Petering (2022) has investigated a problem similar to the department course planning problem (DCPP) that is introduced in this chapter. In other words, prior to the publication of the paper by Khamechian & Petering (2022), no paper had presented a method for helping university departments decide which course to offer in which semesters to facilitate students' timely graduation.

If we consider a university degree as a project, courses as activities (i.e., jobs, tasks), course offerings as resources, and course prerequisites as activity precedence relationships, then university course planning resembles a resource-constrained project scheduling problem (RCPSP). The RCPSP is an NP-hard combinatorial optimization problem. Due to its extensive engineering applications, the RCPSP has become an important research area in recent decades.

Araujo et al. (2020) describe three variants of the RCPSP: the single-mode resource-constrained project scheduling problem (SMRCPSP), multi-mode resource-constrained project scheduling problem (MMRCPSP), and multi-mode resource-constrained multi-project scheduling problem (MMRCMPSP). In the SMRCPSP, an activity-on-node network indicates activity precedence relationships; activities may not be preempted; activities are performed over time with limited resources; resources are renewable at each time period; and there is only one way to process each activity. The goal in a SMRCPSP is to find an activity schedule with minimum makespan that satisfies the activity precedence constraints and resource limitations (Chakrabortty, Sarker, & Essam, 2018). A branch and bound algorithm is often proposed for this problem (Brucker, Knust, Schoo, & Thiele, 1998). Researchers have also applied several types of metaheuristics to the SMRCPSP, such as priority rules (Myszkowski, Skowronski, & Podlodowski, 2013), greedy algorithms (Myszkowski & Siemienski, 2016), tabu search (Skowronski, Myszkowski, Adamski, & Kwiatek, 2013), and simulated annealing (Yannibelli & Amandi, 2013).

An extension of the conventional single-mode RCPSP is the MMRCPSP, which considers the duration of each task as a function of the level and type of resources committed to it. In the MMRCPSP, each task can be accomplished in one of several processing modes, and each processing mode has a different task duration, rate of resource usage, and/or type of resources associated with it. Two types of resources are considered: renewable resources which are available in each time period, and non-renewable resources which can lead to infeasible solutions. In the MMRCPSP, project interactions that result from the utilization of shared resources must be taken into consideration (Zapata, Hodge, & Reklaitis, 2008). A significant number of exact, heuristic, and metaheuristic approaches have been proposed for solving different MMRCPSPs (Almeida, Correia, & Saldanha-da-Gama, 2019). The MMRCPSP is distinguishably more complex than the SMRCPSP, which is itself NP-hard (Elloumi & Fortemps, 2010).

The MMRCMPSP is a generalization of the previous problem variants. The MMRCMPSP considers a multi-project environment in which there are multiple projects with assigned due dates; activities that have alternative resource usage modes; a resource dedication policy that does not allow sharing of resources among projects throughout the planning horizon; and a total budget (Besikci, Bilge, & Ulusoy, 2015).

This chapter considers two optimization problems that have at least five features which are outside the scope of a typical RCPSP: elective activities (elective courses), activity release times (course constraints relating to student seniority), cyclic resource availability (seasonal course offerings), non-strict activity precedence relationships (course corequisites), and operational interruptions (leaves of absence).

Overall, despite the existence of scores of outstanding articles on course scheduling, course planning, and the RCPSP in the academic literature, to the authors' knowledge there is no article besides Khamechian & Petering (2022) which has substantial overlap with the content of this chapter. This chapter introduces two university course planning problems: the student course planning problem (SCPP) and the department course planning problem (DCPP). Our work on the SCPP and DCPP is presented in Sections 1.3 and 1.4 respectively, and our concluding remarks are found in Section 1.5.

1.3. The student course planning problem (SCPP)

We now formally introduce the student course planning problem.

1.3.1. Problem description

Consider a student who begins a university degree. Because the student pays tuition, he/she wants to graduate as soon as possible without harming his/her learning quality. Thus, when the student plans out his/her degree by deciding which courses to take in each future semester, the main goal is to minimize the time required to complete the degree; issues relating to specific instructors and classrooms are typically not considered.

The student creates his/her plan for completing the degree based on the following information. The university course schedule is assumed to repeat annually, and courses are offered during *N sessions* each year. For example, a university that operates on a semester calendar has two sessions per year ($N = 2$), and a university that works quarterly has four sessions per year ($N = 4$). The most basic unit of academic work in the degree is a course. In other words, progress towards completion of the degree is measured in courses, not credit hours. According to university policy, the student may take a maximum of *Max* courses each session. The student begins his/her

degree during session *Start* ($1 \leq Start \leq N$); remains a student for multiple consecutive sessions after that; and is required to graduate within a maximum of S semesters (i.e., sessions) after starting his/her degree. In this chapter, the term "semester" refers to how long a student has been pursuing his/her degree, and the term "session" refers to a time during the year when courses are offered.

There are total of C available courses which are divided to two categories: required and elective. Parameter R_c equals 1 (0) if course c is (is not) a required component of the degree. Parameter E_c equals 1 (0) if course c is (is not) an elective course ($R_c + E_c = 1$ for all c). Elective courses are divided into two groups. Parameter EM_c equals 1 (0) if course c is (is not) an elective course in the major ($EM_c \leq E_c$ for all c). A total of E elective courses must be taken to graduate of which EM must be elective courses in the major. The university allows students to count courses taken at a previous institution towards completion of their degree. Parameter A_c equals 1 (0) if the student has (has not) already taken course c. Each course may refer to a specific university course or a category of courses. For example, course #5 could be "Computer Science 100," "any 100-level computer science course," or "any humanities course."

The course offerings repeat annually, and not every course is offered during every session. For example, some courses may only be offered in the fall while others are only offered in the spring. Binary parameter O_{cn} indicates if course c is offered during session n. We assume there are no time conflicts between the courses offered in the same session. In other words, the student can feasibly attend the lectures for any combination of courses that are offered in the same session.

A set of *prerequisite* and *corequisite* requirements ensure that the student takes courses in the proper sequence. Binary parameter P_{cd} indicates if course c is a prerequisite for course d, i.e., if course c must be taken before course d. Without loss of generality (and to avoid circular logic),

we assume that courses are numbered in agreement with the prerequisite requirements. That is, P_{cd} may only equal 1 if $c < d$. Binary parameter C_{cd} indicates if course c is a corequisite for course d. This parameter equals 1 (0) if course c must be taken before, or during the same semester as, course d.

Another set of restrictions relate to the seniority of the student, i.e., how many courses the student has completed. Binary parameter J_c indicates if the student must be a junior—i.e., if he/she must have completed *Junior* (e.g., 20) courses—before he/she takes course c. Binary parameter S_c indicates if the student must be a senior—i.e., if he/she must have completed *Senior* (e.g., 30) courses—before he/she takes course c ($Junior \leq Senior$).

Finally, the student has the option to take a leave of absence during his/her degree. A leave of absence is a semester when the student does not take courses. This option allows the student to take a break from his/her studies to seek temporary employment; spend additional time with family; or travel around the world. If the student takes a leave of absence, parameter LA ($2 \leq LA \leq S\text{-}1$) indicates the semester when the leave is taken; otherwise, parameter LA equals 0.

1.3.2. Mathematical model

Table 1 lists the indices, parameters, and decision variables in our integer programming (IP) formulation of the SCPP. The model has three types of indices. Index n refers to the sessions. Indices c and d represent courses. Indices s and t refer to semesters; their value indicates how long the student has been pursuing his/her degree.

Model SCPP has three sets of decision variables. Binary variable X_{cs} equals 1 (0) if the student takes (does not take) course c during his/her s^{th} semester. Binary variable Y_s equals 1 (0) if the student has not (has) completed his/her degree by the start of his/her s^{th} semester. Integer

variable Z_s equals the number of courses the student has completed by the beginning of his/her s^{th} semester. This variable allows the model to check the junior or senior standing of the student.

Table 1. Indices, parameters, and decision variables in mathematical model SCPP

Indices	
n	Session (n = 1, 2, …, N)
c, d	Course (c, d = 1, 2, …, C)
s, t	Semester: a measure of how long the student has been pursuing his/her degree (s, t = 1, 2, …, S)
Parameters	
N	Number of sessions per year (e.g., 2)
C	Number of available courses (e.g., 40)
S	Number of semesters available for completing a degree (e.g., 10)
Max	Maximum number of courses the student can take per session (e.g., 6)
$Start$	Session when the student starts his/her degree (= 1, 2, …, N)
A_c	1, if course c has already been taken (e.g., by a transfer student) 0, otherwise
R_c	1, if course c is required for graduation 0, otherwise (binary)
E_c	1, if course c is an elective course 0, otherwise (binary)
EM_c	1, if course c is an elective course in the major 0, otherwise (binary)
E	Number of elective courses needed for graduation (e.g., 4)
EM	Number of elective courses in the major needed for graduation (e.g., 2)
O_{cn}	1, if course c is offered during session n 0, otherwise (binary)
P_{cd}	1, if course c is a prerequisite for course d ($c < d$) 0, otherwise (binary)
C_{cd}	1, if course c is a corequisite for course d 0, otherwise (binary)
$Junior$	Number courses a student must have completed to be considered a junior
$Senior$	Number courses a student must have completed to be considered a senior
J_c	1, if junior standing is required for course c 0, otherwise (binary)
S_c	1, if senior standing is required for course c 0, otherwise (binary)
LA	Semester during which student takes a leave of absence (e.g., to work at a company full time) (= 2, 3, …S-1) (= 0 if no leave of absence is taken)
Decision variables	
X_{cs}	1, if the student takes course c during his/her s^{th} semester 0, otherwise (binary)
Y_s	1, if the student has not completed his/her degree by the start of his/her s^{th} semester 0, otherwise (binary)
Z_s	Number of courses student has completed by the beginning of his/her s^{th} semester

Mathematical model SCPP is shown below:

$$\text{Minimize } \sum_{s=1}^{S} Y_s \quad (1)$$

Constraints

$$Y_{s+1} \le Y_s \quad \text{for all } s \le S-1 \quad (2)$$

$$X_{cs} \le Y_s \quad \text{for all } c \text{ and } s \quad (3)$$

$$A_c + \sum_{s=1}^{S} X_{cs} \ge R_c \quad \text{for all } c \quad (4)$$

$$\sum_{c=1}^{C} [(A_c + \sum_{s=1}^{S} X_{cs}) * (E_c)] \ge E \quad (5)$$

$$\sum_{c=1}^{C} [(A_c + \sum_{s=1}^{S} X_{cs}) * (EM_c)] \ge EM \quad (6)$$

$$\sum_{c=1}^{C} X_{cs} \le Max \quad \text{for all } s \quad (7)$$

$$X_{cs} \le O_{c,\,([(s-1)+(Start-1)] \bmod N)+1} \quad \text{for all } c \text{ and } s \quad (8)$$

$$A_c + \sum_{s=1}^{S} X_{cs} \le 1 \quad \text{for all } c \quad (9)$$

$$\sum_{s=1}^{S} X_{ds} \le \sum_{s=1}^{S} X_{cs} + A_c \quad \text{for all } (c, d) \text{ such that } P_{cd} = 1 \quad (10a)$$

$$(\sum_{s=1}^{S} s * X_{cs}) + 1 \le (\sum_{s=1}^{S} s * X_{ds}) + (S+1)(1 - \sum_{s=1}^{S} X_{ds}) \quad \text{for all } (c, d) \text{ such that } P_{cd} = 1 \quad (10b)$$

$$\sum_{s=1}^{S} X_{ds} \le \sum_{s=1}^{S} X_{cs} + A_c \quad \text{for all } (c, d) \text{ such that } C_{cd} = 1 \quad (11a)$$

$$(\sum_{s=1}^{S} s * X_{cs}) \le (\sum_{s=1}^{S} s * X_{ds}) + (S)(1 - \sum_{s=1}^{S} X_{ds}) \quad \text{for all } (c, d) \text{ such that } C_{cd} = 1 \quad (11b)$$

$$Z_s = \sum_{c=1}^{C} \sum_{t=1}^{s-1} X_{ct} + \sum_{c=1}^{C} A_c \quad \text{for all } s \quad (12)$$

$$Z_s \ge Junior * X_{cs} \quad \text{for all } c \text{ and } s \text{ such that } J_c = 1 \quad (13)$$

$$Z_s \ge Senior * X_{cs} \quad \text{for all } c \text{ and } s \text{ such that } S_c = 1 \quad (14)$$

$$\sum_{c=1}^{C} X_{c,LA} = 0 \quad \text{If } LA \ne 0 \quad (15)$$

$$Y_{LA} = 1 \quad \text{If } LA \ne 0 \quad (16)$$

In model SCPP, the objective (1) is to minimize the number of semesters needed to complete the degree. Constraint (2) states that if a student has not completed his/her degree by the

beginning of his/her s+1st semester, then he/she also has not completed his/her degree by the start of his/her s^{th} semester. This constraint ensures the continuity of the student's degree. Constraint (3) ensures that if the student takes course c during his/her semester s, the student must not be finished with his/her degree by the start of semester s. Constraint (4) ensures that each required course is either taken during the program or has already been taken before the student joins the program. Constraints (5) and (6) ensure that the student takes the necessary number of elective courses and elective courses in the major. Constraint (7) ensures that the student does not take more than *Max* courses per semester.

Constraint (8) is the course availability constraint. It ensures that the student does not take course c during his/her s^{th} semester if the course is not offered during that semester. In this constraint, the expression $([(s\text{-}1) + (Start\text{-}1)] \bmod N) + 1$ converts the semester s into the appropriate session for given values of *Start* and N. For example, if $Start = 1$ and $N = 2$ (if the student starts in the fall at a university that has two sessions (fall and spring) each year) then the expression becomes $([s\text{-}1] \bmod 2) + 1$, and the expression equals (1, 2, 1, 2, …) when s equals (1, 2, 3, 4, …) respectively. If $Start = 2$ and $N = 2$, then the expression becomes $(s \bmod 2) + 1$, and the expression equals (2, 1, 2, 1, …) when s equals (1, 2, 3, 4, …) respectively.

Constraint (9) ensures that each course is taken at most once. Constraints (10a) and (10b) are the prerequisite constraints; they ensure that if course d is taken and course c is its prerequisite, then course c must be taken before course d. Constraints (11a) and (11b) are the corequisite constraints which ensure that if course d is taken and course c is its corequisite, then course c must be taken either before or during the same semester as course d. Constraint (12) ensures that Z_s equals the number of courses the student has taken by the start of his/her s^{th} semester. Constraints (13) and (14) ensure that the student already has junior or senior standing by the beginning of

semester s if he/she takes a course c that requires junior or senior standing during semester s respectively. Constraints (15) and (16) ensure that the student does not take any courses during a leave of absence but that the student is still enrolled in the degree program during a leave of absence.

1.3.3. Experimental setup

All math models in this chapter were coded into MS Visual C++2017, and IBM ILOG Concert Technology was used to call IBM ILOG CPLEX 12.9 to solve problem instances contained in text files. All experiments were run on a desktop PC with a core i7 3.4 GHz processor and 8 GB of RAM. The CPLEX computation time limit was 600 seconds for all problem instances in this chapter.

1.3.4. Case study: Industrial Engineering BSE program at UW-Milwaukee

Our first experiment applies model SCPP to the industrial engineering bachelor's degree at UW-Milwaukee. The list of requirements for this degree is available online (UW-Milwaukee Industrial Engineering curriculum, 2021).

Table 2 shows the parameter values for this degree program. Most inputs are matrices of zeros and ones with different dimensions. The university operates on a semester calendar ($N = 2$), and students are normally expected to finish their degree in ten semesters or less ($S = 10$). A total of 49 courses are available ($C = 49$) among which 37 are required and 12 are elective. To complete the degree, a student must take all required courses and four elective courses—41 courses total. Eight of the twelve elective courses are elective courses in the major, and a student must complete at least two elective courses in the major to graduate. Five of the 37 required courses are actually course categories "Art," "Humanities," "Social Sci 1," "Social Sci 2," and "Free Elective" for

which numerous options are offered each semester, most of which have no prerequisites, no corequisites, and no requirements for junior or senior standing. Without loss of generality, we aggregate all options for each category into a single course in the model.

The maximum number of courses that can be taken per semester is 6 (*Max* = 6), but many students impose their own limit of five courses per semester (*Max* = 5). Thus, we consider two values of *Max* in our experiments. Thirty-five of the 49 courses are offered in both fall and spring sessions, and 14 courses—including ten required courses—are offered in one session only. Hence, 84 (14) of the O_{cn} parameters equal 1 (0). The number of individual course-to-course prerequisite and corequisite requirements is 44 and 13, respectively. Students who complete 20 (30) courses are considered juniors (seniors), and a total of 12 (5) courses require junior (senior) standing. The following subsections discuss the results of three experiments concerning this degree.

Table 2. Experimental setup and assumptions for model SCPP case study (** indicates how many binary terms equal 1)

Parameter	Value(s) used in experiment
N	2
C	49
S	10
Max	5 or 6
Start	1 (fall) or 2 (spring)
A_c	**[8 elements = 1 (all others = 0)] **(Experiment 2 only)**
R_c	**[37 elements = 1 (all others = 0)]
E_c	**[12 elements = 1 (all others = 0)]
EM_c	**[8 elements = 1 (all others = 0)]
E	4
EM	2
O_{cn}	**[84 elements = 1 (all others = 0)] (98 elements total)
P_{cd}	**[44 elements = 1 (all others = 0)]
C_{cd}	**[13 elements = 1 (all others = 0)]
Junior	20
Senior	30
J_c	**[12 elements = 1 (all others = 0)]
S_c	**[5 elements = 1 (all others = 0)]
LA	5 **(Experiment 2 only)**

1.3.5. Experiment 1: Student with no leave of absence and no transfer courses

Our base scenario is that of a student who enters the program directly from high school (with no transfer courses); does not plan to take a leave of absence; and takes up to 6 courses per

semester (*Max* = 6). Table 3 shows our experimental results for this scenario. The left half of the table shows five optimal course schedules for a student who starts in fall (*Start* = 1), and the right half shows five optimal course schedules for a student who starts in spring (*Start* = 2). Each of these optimal solutions is obtained within one second of computation time.

The diverse course schedules shown in Table 3 were obtained by repeatedly solving math model SCPP, each time with a different, random term $[(.0001)\sum_{c=1}^{C}\sum_{s=1}^{S} W_{cs}X_{cs}]$ added to the objective function where each value W_{cs} is a random integer from 0 to 9. The W_{cs} values create arbitrary preferences for taking particular courses in specific semesters, and they are randomly generated each time the code is run. For example, if W_{35} is high, there is a preference to not take course 3 during semester 5. This may result in a different course plan than if W_{35} is low. In this paper a judgment is not made as to which course schedule is preferred; it is assumed that the diverse course schedules shown in Table 3 are equally desirable. Individual students can look at a variety of course schedules and make their own final decision regarding which course schedule they prefer.

The results in the left half of Table 3 show that a student who starts in the fall can graduate in seven semesters. However, the results in the right half of the table show that a student who starts in the spring needs a minimum of eight semesters to graduate. This asymmetry agrees with the experience of university students, faculty, and staff members who are familiar with the degree. Model SCPP was then used to identify the root causes of this asymmetry, and it was found that there were two causes: the semesters when courses are offered (O_{cn}) and the prerequisite requirements (P_{cd}). If either of these parameters were modified or loosened, both cohorts of students—those who start in fall and those who start in spring—would be able to graduate in seven semesters. Modifying P_{cd} at UW-Milwaukee is a time-consuming process because it needs

approval at multiple levels within the university government, so the easier solution is to modify the course offerings (O_{cn}). We then considered each of the ten required courses that are offered in one session only. We swapped the offering session of each of these ten courses one by one to see if it was possible for all students to graduate in seven semesters. The results indicated that all students can graduate in seven semesters if the offering session of either of the following courses is changed: IE 370 or IE 583. Currently both courses are being offered in the fall semester. If the department offers either course in the spring, all students will be able to finish their degrees in seven semesters.

Table 3 allows us to make another interesting observation, namely that the courses IE 111 and IE 112—which are intended for first-year students—could be taken during the last two or three semesters. This is because these courses are not prerequisites or corequisites for other courses. Based on this observation, we recommend that each of these courses be made a prerequisite for at least one other required course.

Table 3. Selected optimal solutions for the SCPP case study (Experiment 1)

***Start* = 1 (fall), *Max* = 6, no leave of absence and no transfer courses**				
Semester 1:	**Semester 1:**	**Semester 1:**	**Semester 1:**	**Semester 1:**
EAS 200	EAS 100	EAS 100	EAS 100	EAS 200
IE 111	IE 111	IE 111	EAS 200	IE 111
Math 116	Math 116	Math 116	Math 116	Math 116
Chem 102	Chem 102	Social Science 1	Chem 102	Chem 102
Social Science 1	Art	Social Science 2	Social Science 1	Social Science 2
Social Science 2	Social Science 1	Free Elective	English 310	Free Elective
Semester 2:	**Semester 2:**	**Semester 2:**	**Semester 2:**	**Semester 2:**
EAS 100	EAS 200	CompSci 240	CompSci 240	EAS 100
IE 112	IE 112	Math 231	MatlEng 201	IE 112
Math 231	MatlEng 201	Chem 102	Math 231	MatlEng 201
Chem 104	Math 231	Art	Chem 104	Math 231
Humanities	Social Science 2	Humanities	Humanities	Chem 104
English 310	Free Elective	English 310	Social Science 2	Art
Semester 3:	**Semester 3:**	**Semester 3:**	**Semester 3:**	**Semester 3:**
CompSci 240	CompSci 240	IE 112	IE 111	CompSci 240
IE 367	IE 350	MatlEng 201	IE 350	IE 350
Math 232	IE 367	IE 367	IE 367	IE 367
Physics 209	Math 232	Math 232	Math 232	Math 232
Free Elective	Chem 104	Chem 104	Physics 209	Physics 209
EAS 001	Physics 209	Physics 209	Art	Humanities
Semester 4:	**Semester 4:**	**Semester 4:**	**Semester 4:**	**Semester 4:**
CivEng 201	CivEng 201	CivEng 201	CivEng 201	CivEng 201
EE 301	EE 301	EE 301	EE 301	EE 301
IE 475	IE 475	IE 475	IE 475	IE 475
Math 233	IE 575	IE 575	IE 575	Math 233
EE 234	Math 233	Math 233	Math 233	EE 234
Physics 210	Physics 210	Physics 210	Physics 210	Physics 210
Semester 5:	**Semester 5:**	**Semester 5:**	**Semester 5:**	**Semester 5:**
CivEng 202	CivEng 202	IE 360	CivEng 202	CivEng 202
IE 360	IE 360	IE 350	IE 370	IE 360
IE 370	IE 370	IE 370	IE 455	IE 370
IE 455	IE 455	IE 455	IE 470	IE 455
IE 470	EE 234	IE 580	IE 580	IE 470
IE 580	IE 699	EE 234	EE 234	IE 580
Semester 6:	**Semester 6:**	**Semester 6:**	**Semester 6:**	**Semester 6:**
MatlEng 201	IE 465	CivEng 202	IE 360	IE 465
IE 465	IE 571	IE 465	IE 465	IE 571
IE 571	English 310	IE 571	IE 571	IE 575
IE 575	IE 590	IE 550	Free Elective	Social Science 1
Art	Bus Adm 330	IE 584	IE 572	IE 550
IE 572	EAS 001	Bus Adm 330	IE 587	IE 584
Semester 7:	**Semester 7:**	**Semester 7:**	**Semester 7:**	**Semester 7:**
IE 350	IE 470	EAS 200	IE 112	IE 485
IE 485	IE 485	IE 470	IE 485	IE 583
IE 583	IE 580	IE 485	IE 583	English 310
IE 590	IE 583	IE 583	EAS 001	IE 590
MechEng 301	Humanities	MechEng 474	MechEng 301	MechEng 474

***Start* = 2 (spring), *Max* = 6, no leave of absence and no transfer courses**				
Semester 1:	**Semester 1:**	**Semester 1:**	**Semester 1:**	**Semester 1:**
EAS 100	EAS 200	EAS 200	IE 111	IE 111
EAS 200	Math 116	Math 116	Math 116	Math 116
Math 116	Chem 102	Chem 102	Chem 102	Chem 102
Chem 102	Art	Humanities	Humanities	Art
Art	Humanities	Social Science 2	English 310	Social Science 1
Free Elective	Social Science 1	Free Elective	Free Elective	Social Science 2
Semester 2:	**Semester 2:**	**Semester 2:**	**Semester 2:**	**Semester 1:**
IE 111	IE 112	EAS 100	EAS 100	IE 112
CompSci 240	CompSci 240	IE 112	EAS 200	CompSci 240
Math 231	MatlEng 201	CompSci 240	CompSci 240	MatlEng 201
Chem 104	Math 231	MatlEng 201	MatlEng 201	Math 231
Social Science 1	Social Science 2	Math 231	Math 231	Chem 104
EAS 001	Free Elective	English 310	Social Science 2	Free Elective
Semester 3:	**Semester 3:**	**Semester 3:**	**Semester 3:**	**Semester 3:**
IE 112	EAS 100	IE 111	IE 112	EAS 100
MatlEng 201	IE 111	IE 367	IE 367	EAS 200
IE 367	IE 367	Math 232	Math 232	IE 367
Math 232	Math 232	Physics 209	Chem 104	Math 232
Physics 209	Physics 209	Art	Physics 209	Physics 209
English 310	EAS 001	EAS 001	EAS 001	Humanities
Semester 4:	**Semester 4:**	**Semester 4:**	**Semester 4:**	**Semester 4:**
CivEng 201	CivEng 201	CivEng 201	CivEng 201	CivEng 201
EE 301	EE 301	EE 301	EE 301	EE 301
Math 233	IE 350	IE 350	IE 350	IE 350
EE 234	Math 233	Math 233	Math 233	Math 233
Physics 210	EE 234	EE 234	EE 234	EE 234
Humanities	Physics 210	Physics 210	Physics 210	Physics 210
Semester 5:	**Semester 5:**	**Semester 5:**	**Semester 5:**	**Semester 5:**
CivEng 202	CivEng 202	CivEng 202	CivEng 202	IE 360
IE 465	IE 465	IE 360	IE 360	IE 465
IE 475	IE 475	IE 465	IE 465	IE 475
IE 571	IE 571	IE 475	IE 475	IE 571
IE 575	IE 575	IE 571	IE 571	IE 575
IE 584	IE 572	IE 575	IE 575	IE 572
Semester 6:	**Semester 6:**	**Semester 6:**	**Semester 6:**	**Semester 6:**
IE 360	IE 370	IE 370	IE 370	CivEng 202
IE 370	IE 455	IE 455	IE 455	IE 370
IE 455	IE 470	IE 470	IE 470	IE 455
IE 470	IE 580	IE 580	IE 580	IE 470
IE 580	English 310	Chem 104	IE 584	IE 580
IE 590	MechEng 474	MechEng 474	IE 699	IE 584
Semester 7:	**Semester 7:**	**Semester 7:**	**Semester 7:**	**Semester 7:**
Social Science 2	IE 360	Social Science 1	Art	English 310
MechEng 301	Chem 104	IE 584	Social Science 1	Bus Adm 330
	IE 699		IE 572	MechEng 474
Semester 8:	**Semester 8:**	**Semester 8:**	**Semester 8:**	**Semester 8:**
IE 350	IE 485	IE 485	IE 485	IE 485
IE 485	IE 583	IE 583	IE 583	IE 583
IE 583		IE 582		

1.3.6. Experiment 2: Student with a leave of absence in semester five and eight transfer courses

In this experiment we assume that *Max* = 6 and we consider a transfer student who has already taken the following courses at a previous academic institution: Math 116, Math 231, Math 232, Math 233, IE 367, CivEng 201, Chem 102, and Physics 209. Table 4 shows the results for this student if he/she takes a leave of absence in semester five and starts his/her program in the spring session (*Start* = 2). Each column of Table 4 shows a different optimal course schedule that is obtained by solving math model SCPP with a different, random term added to the objective function as described in Section 1.3.5. Less than one second of computation time is used to obtain each of these optimal solutions. Note that this student needs seven semesters—including the leave of absence—to complete his/her degree.

Table 4. Selected optimal solutions for the SCPP case study (Experiment 2)

Start = 2(spring), *LA* = 5, *Max* = 6, transferred courses: Math 116, 231, 232, 233, IE 367, CivEng 201, Chem 102, Physics 209				
Semester 1:	**Semester 1:**	**Semester 1:**	**Semester 1:**	**Semester 1:**
EAS 100	EAS 100	EAS 100	EAS 200	EAS 100
EAS 200	IE 111	IE 111	IE 111	EAS 200
IE 111	Art	Humanities	Art	Art
Art	Humanities	Social Science 1	Humanities	Social Science 1
English 310	Social Science 2	Social Science 2	Social Science 2	Social Science 2
Free Elective	English 310	EAS 001	English 310	English 310
Semester 2:	**Semester 2:**	**Semester 2:**	**Semester 2:**	**Semester 2:**
IE 112	IE 112	CivEng 202	EAS 100	IE 112
CompSci 240	CivEng 202	CompSci 240	CompSci 240	CompSci 240
EE 301	CompSci 240	IE 370	IE 370	EE 301
IE 370	EE 301	EE 234	EE 234	MatlEng 201
Chem 104	IE 370	Chem 104	Physics 210	IE 370
Physics 210	Physics 210	Physics 210	MechEng 301	Physics 210
Semester 3:	**Semester 3:**	**Semester 3:**	**Semester 3:**	**Semester 3:**
MatlEng 201	MatlEng 201	IE 112	EE 301	IE 111
IE 465	IE 465	EE 301	IE 465	IE 360
IE 475	IE 475	MatlEng 201	IE 475	IE 465
IE 571	IE 571	IE 465	IE 571	IE 475
EE 234	Social Science 1	IE 475	IE 575	IE 571
Social Science 1	EAS 001	IE 571	IE 699	IE 575
Semester 4:	**Semester 4:**	**Semester 4:**	**Semester 4:**	**Semester 4:**
IE 360	IE 360	IE 350	CivEng 202	CivEng 202
IE 350	IE 350	IE 455	MatlEng 201	IE 350
IE 455	IE 455	IE 470	IE 455	IE 455
IE 470	IE 470	IE 580	IE 470	IE 470
IE 580	IE 580	Art	IE 580	IE 580
Humanities	Chem 104	English 310	IE 584	Bus Adm 330
Semester 5:	**Semester 5:**	**Semester 5:**	**Semester 5:**	**Semester 5:**
Semester 6:	**Semester 6:**	**Semester 6:**	**Semester 6:**	**Semester 6:**
CivEng 202	EAS 200	IE 485	IE 112	IE 485
IE 485	IE 485	IE 583	IE 350	IE 583
IE 583	IE 583	Free Elective	IE 485	EE 234
IE 405	Free Elective	IE 582	IE 583	Chem 104
IE 584	IE 405	IE 699	Free Elective	Humanities
IE 590	MechEng 301	Bus Adm 330	MechEng 474	IE 590
Semester 7:	**Semester 7:**	**Semester 7:**	**Semester 7:**	**Semester 7:**
IE 575	IE 575	EAS 200	IE 360	Free Elective
Social Science 2	EE 234	IE 360	Chem 104	IE 405
IE 587	IE 590	IE 575	Social Science 1	IE 584

1.3.7. Experiment 3: Leave of absence timing

In this experiment, we explore how the timing of a leave of absence—i.e., the value of parameter *LA*—affects a student's time to graduate. We assume that the student begins from scratch with no transfer courses, and he/she starts the degree in either fall or spring. Table 5 shows how *LA* affects the optimal graduation time when *Max* = 5, and Table 6 shows how *LA* affects the optimal graduation time when *Max* = 6.

Table 5 shows that, when *Max* = 5, a leave of absence always delays a student's graduation by one semester regardless of the student's starting session or when the leave of absence is taken. Indeed, a student needs nine semesters to graduate if he/she does not take a leave of absence regardless of his/her starting session. Also, a student needs ten semesters to graduate for all values of *LA* from 2 to 9 and all values of *Start* from 1 to 2. These results are hardly surprising.

Table 6 shows a more complex situation when *Max* = 6. The rows labeled "(none)" show that, if no leave of absence is taken, a student who starts in fall (spring) can graduate in 7 (8) semesters (see Table 3). If the student starts in fall and takes a leave of absence in semester six, his/her graduation will be delayed by one semester, but if the student starts in fall and takes a leave of absence in any other semester, his/her graduation will be delayed by two semesters. On the other hand, if the student starts in spring and takes a leave of absence in semester 2, 3, 4, 5, or 7, his/her graduation will not be delayed. However, if the student starts in spring and takes a leave of absence in semester 6 or 8, his/her graduation will be delayed by two semesters.

These results show that leaves of absence need to be carefully planned. A poorly planned leave of absence can add 1–2 semesters to a student's graduation time compared to a well-planned leave. Based on these results, we advise that the course offerings (O_{cn}) and/or degree requirements

(P_{cd}, C_{cd}) be modified so that students' graduation time is less sensitive to the semester when they take a leave of absence. In the meantime, before these modifications are implemented, we recommend that new students either (1) begin the program in the fall and do not take a leave of absence; (2) begin the program in the fall and take a leave of absence during semester 6; or (3) begin the program in the spring and take a leave of absence during semester 2, 3, 4, 5, or 7.

Table 5. Effect of *LA* on the number of semesters needed to graduate for the SCPP case study when *Max* = 5 (Experiment 3)

Start =1 (fall)		*Start* =2 (spring)	
LA	#Semesters	*LA*	#Semesters
(none)	9	(none)	9
2	10	2	10
3	10	3	10
4	10	4	10
5	10	5	10
6	10	6	10
7	10	7	10
8	10	8	10
9	10	9	10

Table 6. Effect of *LA* on the number of semesters needed to graduate for the SCPP case study when *Max* = 6 (Experiment 3)

Start =1 (fall)		*Start* =2 (spring)	
LA	#Semesters	*LA*	#Semesters
(none)	7	(none)	8
2	9	2	8
3	9	3	8
4	9	4	8
5	9	5	8
6	8	6	10
7	9	7	8
		8	10

1.3.8. Experiments on fictional problem instances

We now test model SCPP on 12 fictional problem instances. These instances are categorized by their size—small, medium, large—with four instances considered for each size. We gradually tighten the four instances within each size by increasing the number of prerequisites. In other words, each problem instance within each instance size has more ones in the P_{cd} matrix compared to the preceding problem instance. For example, in instance 1 of each problem size, 2% of the elements in the upper triangle of the P_{cd} matrix equal one while this number is 5% in instance 4.

Table 7 lists the main inputs and assumptions for the fictional problem instances. For the sake of simplicity, in all fictional instances we assume that there are no elective courses in the major ($EM = 0$) and that $C_{cd} = 0$ for all c and d. We set a computation time limit of 600 seconds.

Table 7. Experimental setup and assumptions for model SCPP fictional instances

Small instances	Medium-sized instances	Large instances
$N = 3$	$N = 4$	$N = 6$
$C = 20$	$C = 50$	$C = 80$
$S = 8$	$S = 10$	$S = 12$
$Max = 6$	$Max = 6$	$Max = 6$
A_c: All elements = 0	A_c: All elements = 0	A_c: All elements = 0
R_c: 12 elements = 1	R_c: 35 elements = 1	R_c: 45 elements = 1
$E_{c:}$ 8 elements = 1	$E_{c:}$ 15 elements = 1	$E_{c:}$ 35 elements = 1
$E = 3$	$E = 5$	$E = 15$
$EM = 0$	$EM = 0$	$EM = 0$
15 courses needed for graduation	40 courses needed for graduation	60 courses needed for graduation
O_{cn}: 47 elements = 1 and all others = 0	O_{cn}: 174 elements = 1 and all others = 0	O_{cn}: 365 elements = 1 and all others = 0
P_{cd}: 2%, 3%, 4%, or 5% of elements = 1	P_{cd}: 2%, 3%, 4%, or 5% of elements = 1	P_{cd}: 2%, 3%, 4%, or 5% of elements = 1
C_{cd}: All elements = 0	C_{cd}: All elements = 0	C_{cd}: All elements = 0
$Junior = 5$	$Junior = 20$	$Junior = 30$
$Senior = 10$	$Senior = 30$	$Senior = 40$
J_c: 6 elements = 1	J_c: 14 elements = 1	J_c: 17 elements = 1
$S_{c:}$ 5 elements = 1	$S_{c:}$ 8 elements = 1	$S_{c:}$ 12 elements = 1
No leaves of absence	No leaves of absence	No leaves of absence

Table 8 shows the results of our experiments. Each instance is solved for each possible value of *Start*. As can be seen in Table 8, model SCPP produces optimal solutions for all instances in less than ten seconds. For the small instances, the optimal value is either 3 or 4. For the medium-sized instances, the optimal value generally increases from 7 to 9 from instance 1 to instance 4. For the large instances, the optimal value is either 10 or 11 depending on the session when the student starts his/her degree. Overall, it appears that a direct mathematical programming approach with default CPLEX settings is effective in solving a variety of small and large instances of the SCPP.

Table 8. Experimental results for model SCPP fictional instances.

Small instances

	Instance 1 (P_{cd}: 2% of elements = 1)			Instance 2 (P_{cd}: 3% of elements = 1)			Instance 3 (P_{cd}: 4% of elements = 1)			Instance 4 (P_{cd}: 5% of elements = 1)		
Start	1	2	3	1	2	3	1	2	3	1	2	3
Optimal value	4	3	4	4	3	4	4	3	4	4	3	4
Time elapsed (sec)	1	1	1	1	1	1	1	1	1	1	1	1

Medium-sized instances

	Instance 1 (P_{cd}: 2% of elements = 1)				Instance 2 (P_{cd}: 3% of elements = 1)				Instance 3 (P_{cd}: 4% of elements = 1)				Instance 4 (P_{cd}: 5% of elements = 1)			
Start	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Optimal value	7	7	7	7	9	8	9	8	9	8	9	8	9	8	9	8
Time elapsed (sec)	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6

Large instances

	Instance 1 (P_{cd}: 2% of elements = 1)						Instance 2 (P_{cd}: 3% of elements = 1)						Instance 3 (P_{cd}: 4% of elements = 1)						Instance 4 (P_{cd}: 5% of elements = 1)					
Start	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
Optimal value	10	10	10	10	10	10	11	10	11	10	11	10	10	10	11	10	10	10	10	11	11	11	11	11
Time elapsed (sec)	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7

1.4. The department course planning problem (DCPP)

We now introduce a second optimization problem called the department course planning problem (DCPP).

1.4.1. Problem description

Consider an academic department at a university that needs to decide when during the year it will teach courses that support an academic degree that it offers. The curriculum of this degree is defined by parameters *N, C, S, Max,* R_c, E_c, EM_c, *E, EM,* P_{cd}, C_{cd}, *Junior*, *Senior*, J_c, and S_c as described in Section 1.3.2. (The discussion of *Start*, A_c, O_{cn}, and *LA* in Section 1.3.2 is also relevant here, but these are not parameters in the DCPP.)

The department wants to facilitate the timely graduation of regular students who begin the program from scratch (without any transfer courses) and do not take any leaves of absence. In other words, the department plans its course offerings assuming that $A_c = 0$ for all c and $LA = 0$ for each student. Students are allowed to begin the degree in any session n ($1 \le n \le N$). From historical data, the department knows that the proportion of students who begin the degree in session n is W_n. For example, if 70% of students begin their studies in the fall and the university has two sessions per year—fall and spring—then $W_1 = 0.7$ and $W_2 = 0.3$. Students who begin the degree during the same session are said to be in the same *cohort*.

Courses within the degree are taught either by the department or by a unit outside of the department. Binary parameter D_c equals 1 (0) if course c is taught by (outside of) the department. Courses taught outside of the department are in good supply, and it can be assumed that every such course is offered every session. The department, on the other hand, has limited (staff, budgetary) resources to offer the courses it teaches. In particular, the maximum number of *course sections* it

can offer in a year is *CourseLimit*. Each course section is equivalent to one course offered during one session. If the department wishes to offer the same course in two different sessions, two course sections are needed.

Given the above information, the department wishes to decide the value of binary decision variable O_{cn}—which equals 1 (0) if course c is offered during session n—so as to facilitate the timely graduation of all student cohorts. We assume that O_{cn} always equals 1 if course c is taught outside of the department.

If the department is only interested in facilitating the graduation of one student cohort, this can be done by setting $W_n = 1$ for that cohort and $W_n = 0$ for all other cohorts. However, if course offerings are tailored to only one cohort, students in other cohorts may experience unnecessarily long graduation times. For this reason, our model of the DCPP is designed to consider the graduation time for all student cohorts simultaneously.

1.4.2. Mathematical models

We develop two closely linked integer programming models of the DCPP. Model DCPP I minimizes the graduation time for the average student who enters the program, and model DCPP II minimizes the maximum number of semesters needed for any student cohort to graduate. Table 9 lists the indices, parameters, and decision variables in these models. The elements in these two models are identical except that model DCPP I has one more parameter (W_n), and one less decision variable (K), than model DCPP II. We now discuss each model in detail.

Table 9. Indices, parameters, and decision variables in mathematical models DCPP I and DCPP II

Indices	
n	Session (n = 1, 2……. N)
c, d	Course (c, d = 1, 2, …., C)
s, t	Semester; a measure of how long a student has been pursuing his/her degree (s, t = 1, 2, …… S)
Parameters	
N	Number of sessions per year (e.g., 2)
C	Number of available courses (e.g., 40)
S	Number of semesters available for completing a degree (e.g., 10)
Max	Maximum number of courses student can take per semester (e.g., 6)
R_c	1, if course c is required for graduation 0, otherwise (binary)
E_c	1, if course c is an elective course 0, otherwise (binary)
EM_c	1, if course c is an elective course in the major 0, otherwise (binary)
E	Number of elective courses needed for graduation (e.g., 4)
EM	Number of elective courses in the major needed for graduation (e.g., 2)
P_{cd}	1, if course c is a prerequisite for course d ($c < d$) 0, otherwise (binary)
C_{cd}	1, if course c is a corequisite for course d 0, otherwise (binary)
$Junior$	Number courses a student needs to pass to be considered a junior
$Senior$	Number courses a student needs to pass to be considered a senior
J_c	1, if junior standing is required for course c 0, otherwise (binary)
S_c	1, if senior standing is required for course c 0, otherwise (binary)
D_c	1, if course c is taught by the department 0, otherwise (binary)
$CourseLimit$	Maximum number of course sections the department can offer in a year
W_n	Weight for the graduation time of a student who starts in session n ($\sum_{n=1}^{N} W_n = 1$) **(Model DCPP I only)**
Decision variables	
X_{ncs}	1, if a student who starts his/her degree in session n takes course c during his/her s^{th} semester 0, otherwise (binary)
Y_{ns}	1, if a student who starts his/her degree in session n has not completed his/her degree by the start of his/her s^{th} semester 0, otherwise (binary)
Z_{ns}	Number of courses a student who started his/her degree in session n has completed by the beginning of his/her s^{th} semester
O_{cn}	1, if course c is offered during session n 0, otherwise (binary)
K	The greatest number of semesters any student needs to graduate **(model DCPP II only)**

1.4.3. Model DCPP I

All parameters in model DCPP I except D_c, $CourseLimit$, and W_n are identical to those in model SCPP. Model DCPP I has four sets of decision variables. The first three are identical to the decision variables in model SCPP except that a new dimension corresponding to the session n has

been added. By adding this new dimension, the model can generate results for all student cohorts at the same time. For example, binary variable X_{ncs} equals 1 (0) if students who begin their degree in session n take course c during their s^{th} semester. Binary variables Y_{ns} and Z_{ns} have a similar relationship to variables Y_s and Z_s in model SCPP. The fourth decision variable, O_{cn}, is the focus of the DCPP. It equals 1 (0) if course c is offered during session n. This was a parameter in the SCPP. Model DCPP I is shown below:

Minimize $\sum_{n=1}^{N}\sum_{s=1}^{S} W_n * Y_{ns}$ (17)

Constraints

$Y_{n,s+1} \leq Y_{ns}$ for all n and $s \leq S-1$ (18)

$X_{ncs} \leq Y_{ns}$ for all n, c, and s (19)

$\sum_{s=1}^{S} X_{ncs} \geq R_c$ for all c and n (20)

$\sum_{c=1}^{C}\sum_{s=1}^{S} X_{ncs} * E_c \geq E$ for all n (21)

$\sum_{c=1}^{C}\sum_{s=1}^{S} X_{ncs} * EM_c \geq EM$ for all n (22)

$\sum_{c=1}^{C} X_{ncs} \leq Max$ for all s and n (23)

$X_{ncs} \leq O_{c,\,([(s-1)+(n-1)] \bmod N)+1}$ for all n, c, and s (24)

$\sum_{s=1}^{S} X_{ncs} \leq 1$ for all c and n (25)

$\sum_{s=1}^{S} X_{nds} \leq \sum_{s=1}^{S} X_{ncs}$ for all (n, c, d) such that $P_{cd} = 1$ (26a)

$(\sum_{s=1}^{S} s * X_{ncs}) + 1 \leq (\sum_{s=1}^{S} s * X_{nds}) + (S+1)(1 - \sum_{s=1}^{S} X_{nds})$ for all (n, c, d) such that $P_{cd} = 1$ (26b)

$\sum_{s=1}^{S} X_{nds} \leq \sum_{s=1}^{S} X_{ncs}$ for all (n, c, d) such that $C_{cd} = 1$ (27b)

$(\sum_{s=1}^{S} s * X_{ncs}) \leq (\sum_{s=1}^{S} s * X_{nds}) + (S)(1 - \sum_{s=1}^{S} X_{nds})$ for all (n, c, d) such that $P_{cd} = 1$ (27b)

$Z_{ns} = \sum_{t=1}^{s-1}\sum_{c=1}^{C} X_{nct}$ for all n and s (28)

$Z_{ns} \geq Junior * X_{ncs}$ for all n, c, and s such that $J_c = 1$ (29)

$Z_{ns} \geq Senior * X_{ncs}$ for all n, c, and s such that $S_c = 1$ (30)

$\sum_{n=1}^{N}\sum_{c=1}^{C} D_c * O_{cn} \leq CourseLimit$ (31)

In model DCPP I, the objective (17) is to minimize the weighted total number of semesters needed by all cohorts to graduate. There are 14 constraints. Constraints (18)-(30) are identical to constraints (2)-(14) in model SCPP except that the new dimension *n* has been added to the decision variables. Importantly, O_{cn} is a decision variable in constraint (24) but was a parameter in constraint (8) in model SCPP. Constraint (31) is a new constraint which ensures that the department offers no more than *CourseLimit* course sections each year. Note that this constraint only restricts the value of O_{cn} if $D_c = 1$, i.e., if the department teaches course c. If $D_c = 0$—if the department does not teach course *c*—then the value of O_{cn} is assumed to be 1 for all *n*.

1.4.4. Model DCPP II

Model DCPP II is identical to model DCPP I except that it focuses on a different objective: minimizing the maximum time needed for any student to graduate, regardless of when he/she begins the degree. This model has one less parameter (W_n), and one more decision variable (K), than model DCPP I. Decision variable *K* represents the greatest number of semesters any student cohort takes to graduate. Overall, DCPP I minimizes the time for an average student to graduate while DCPP II minimizes the maximum time taken by any student to graduate. Math model DCPP II is shown below:

Minimize K (32)

Constraints

Subject to (18) – (31)

$K \geq \sum_{s=1}^{S} Y_{ns}$ for all n (33)

In model DCPP II, the objective (32) is to minimize decision variable *K*. There are 15 constraints, including all constraints from model DCPP I and a new constraint (33) which ensures

that K is an upper bound for the graduation time of a student who begins the degree in session n for all n.

1.4.5. Case study revisited: Industrial Engineering BSE program at UW-Milwaukee

Models DCPP I and DCPP II were deployed to gain additional insight into the case study from Sections 1.3.4-1.3.7 which considered the Industrial Engineering BSE program at UW-Milwaukee. This degree is managed by the UW-Milwaukee Department of Industrial & Manufacturing Engineering which teaches 23 courses to support it. In other words, D_c is an array with 49 binary elements, 23 (26) of which equal 1 (0).

In our first experiment, we use models DCPP I and DCPP II to see how the department should plan its course offerings assuming *CourseLimit* = 27, which represents the current (staff, budgetary) resources available to the department. We consider two values of *Max*—5 and 6—and three values of W_n—[0.9, 0.1], [0.7, 0.3], and [0.5, 0.5]—for model DCPP I. The experimental results indicate that, when *Max* = 5, courses can be offered so that both student cohorts are able to graduate in 9 semesters (regardless of W_n). Also, when *Max* = 6, courses can be offered so that both student cohorts are able to graduate in 7 semesters (regardless of W_n). These results are hardly surprising given that a total of 41 courses must be taken to complete the degree which means that 9 (7) semesters is a lower bound on the time needed to complete the degree when *Max* = 5 (6). In this case study, models DCPP I and DCPP II obtained the same results but in other cases they might be different. The decision of which model to choose depends on the department and its policy.

In our second experiment, we gradually reduce *CourseLimit* to identify the minimum value of *CourseLimit* for which students' graduation time is the same as when *CourseLimit* = 27. Table

10 shows the results of this experiment which considers the same values of *Max* and W_n as the previous experiment. The results show that, in all cases, *CourseLimit* can be reduced to 17 without lengthening students' time in the program. This is a very small number considering that the department teaches 14 required courses, each of which must be offered at least once a year. These results show that some required courses such as IE 111 and IE 112 need not be offered in both fall and spring semesters, and that the faculty teaching load—which is currently four courses per year—could likely be reduced to three courses per year without lengthening students' time in the program.

Table 10. Effect of *Max* and W_n on results for model DCPP I and DCPP II case study

Model	*Max* = 5		*Max* = 6	
	Optimal value	Minimum *CourseLimit*	Optimal value	Minimum *CourseLimit*
DCPP I (W_n = [0.9, 0.1])	9	17	7	17
DCPP I (W_n = [0.7, 0.3])	9	17	7	17
DCPP I (W_n = [0.5, 0.5])	9	17	7	17
DCPP II	9	17	7	17

1.4.6. Experiments on fictional problem instances

Models DCPP I and DCPP II were also tested on 12 fictional instances that are based on the instances from Section 1.3.8. Table 11 summarizes the parameter values in these instances. In all instances, the values of all parameters besides D_c, *CourseLimit*, and W_n equal those in the SCPP stances from Section 1.3.8. Note that the value of *CourseLimit*—which is (20, 40, 60) for the (small, medium-sized, large) instances respectively—is noticeably less than the number of unique course sections that theoretically exist each year which is (3*10, 4*25, 6*40) respectively. Thus, the value of *CourseLimit* places meaningful restrictions on the department in all instances.

Table 11. Experimental setup and assumptions for model DCPP I and DCPP II fictional instances

Small instances	Medium-sized instances	Large instances
All parameters except those below are identical to the SCPP instances.	All parameters except those below are identical to the SCPP instances.	All parameters except those below are identical to the SCPP instances.
D_c has 20 elements, 10 of which equal one.	D_c has 50 elements, 25 of which equal one.	D_c has 80 elements, 40 of which equal one.
CourseLimit = 20	*CourseLimit* = 40	*CourseLimit* = 60
W_n = [0.7, 0.2, 0.1] **(Model DCPP I only)**	W_n = [0.5, 0.3, 0.15, 0.05] **(Model DCPP I only)**	W_n = [0.35, 0.25, 0.2, 0.1, 0.07, 0.03] **(Model DCPP I only)**

The experimental results for model DCPP I (DCPP II) are displayed in Tables 12 and 13. Note that optimal values are obtained within 20 s for all small and medium-sized instances. For model DCPP I, we see that the optimal value is not an integer for medium-sized instances 2, 3, and 4. This is because of the weights allocated to each session and the number of semesters taken to graduate for each session. In these instances, students who start the degree in session 1—half of all students—need eight semesters to graduate and students who start the degree in other sessions need only seven semesters to graduate.

For large problem sizes, CPLEX identifies optimal solutions for the first three instances within 600 s. Note that the runtime gradually increases from instance 1 to instance 3. For instance 4, a feasible solution with an optimality gap of 8.33% was identified within 600 s when model DCPP II is used, but no feasible solution was found within 600 s when model DCPP I is used. Overall, a direct mathematical programming approach with default CPLEX settings is effective in solving all but the largest and most highly constrained DCPP instance that we consider.

Table 12. Experimental results for model DCPP I fictional instances

Small instances				
	Instance 1 (P_{cd}: 2% of elements = 1)	Instance 2 (P_{cd}: 3% of elements = 1)	Instance 3 (P_{cd}: 4% of elements = 1)	Instance 4 (P_{cd}: 5% of elements = 1)
Final objective value	3	3	3	3
Semesters of enrollment	(3, 3, 3)	(3, 3, 3)	(3, 3, 3)	(3, 3, 3)
Time elapsed (sec)	4	4	4	4

Medium-sized instances				
	Instance 1 (P_{cd}: 2% of elements = 1)	Instance 2 (P_{cd}: 3% of elements = 1)	Instance 3 (P_{cd}: 4% of elements = 1)	Instance 4 (P_{cd}: 5% of elements = 1)
Final objective value	7	7.5	7.5	7.5
Semesters of enrollment	(7, 7, 7, 7)	(8, 7, 7, 7)	(8, 7, 7, 7)	(8, 7, 7, 7)
Time elapsed (sec)	9	9	9	10

Large instances				
	Instance 1 (P_{cd}: 2% of elements = 1)	Instance 2 (P_{cd}: 3% of elements = 1)	Instance 3 (P_{cd}: 4% of elements = 1)	Instance 4 (P_{cd}: 5% of elements = 1)
Final objective value	10	10	10	unknown
Semesters of enrollment	(10, 10, 10, 10, 10, 10)	(10, 10, 10, 10, 10, 10)	(10, 10, 10, 10, 10, 10)	unknown
Time elapsed (sec)	76	113	297	600

Table 13. Experimental results for model DCPP II fictional instances.

Small instances

	Instance 1 (P_{cd}: 2% of elements = 1)	Instance 2 (P_{cd}: 3% of elements = 1)	Instance 3 (P_{cd}: 4% of elements = 1)	Instance 4 (P_{cd}: 5% of elements = 1)
Final objective value	3	3	3	3
Time elapsed (sec)	4	4	4	4

Medium-sized instances

	Instance 1 (P_{cd}: 2% of elements = 1)	Instance 2 (P_{cd}: 3% of elements = 1)	Instance 3 (P_{cd}: 4% of elements = 1)	Instance 4 (P_{cd}: 5% of elements = 1)
Final objective value	7	8	8	8
Time elapsed (sec)	10	10	9	11

Large instances

	Instance 1 (P_{cd}: 2% of elements = 1)	Instance 2 (P_{cd}: 3% of elements = 1)	Instance 3 (P_{cd}: 4% of elements = 1)	Instance 4 (P_{cd}: 5% of elements = 1)
Final objective value	10	10	10	12 (feasible; gap = 8.33%)
Time elapsed (sec)	233	248	258	600

1.5. Conclusion

This research introduced two university course planning problems. In the student course planning problem (SCPP), a student needs to design a course plan that allows him/her to graduate in a timely manner. In the department course planning problem (DCPP), an academic department needs to decide which courses to offer in which semester to facilitate students' timely graduation. Three closely linked integer programming models of these problems were developed, coded in C++, and solved with IBM ILOG CPLEX. Experiments on small, medium, and large real-world and fictional instances showed that these models provide swift insight into a university degree program and help identify ways to modify a program to better meet the needs of students and faculty. Future work might consider other objectives such as minimizing the number of courses taken per semester while achieving a specified a graduation date (e.g., allowing students to work full- or part-time while taking courses). More experiments that consider additional scenarios could also be conducted. In addition, it might be possible to develop math models and/or heuristic methods for planning course offerings within a college that houses several departments. Finally, the analysis might be taken one step further to develop models and methods for course planning from the perspective of an entire academic institution with multiple colleges.

Chapter 1 References

Almeida, B. F., Correia, I., & Saldanha-da-Gama, F. (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical comparison. *International Transactions in Operational Research*, 26(3), 946-967.

Araujo, J. A., Santos, H. G., Gendron, B., Jena, S. D., Brito, S. S., & Souza, D. S. (2020). Strong bounds for resource constrained project scheduling: Preprocessing and cutting planes. *Computers & Operations Research*, 113, 104782.

Besikci, U., Bilge, Ü., & Ulusoy, G. (2015). Multi-mode resource constrained multi-project scheduling and resource portfolio problem. *European Journal of Operation Research*, 240, 22-31.

Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 107(2), 272-288.

Chakrabortty, R. K., Sarker, R. A., & Essam, D. L. (2018). Single mode resource constrained project scheduling with unreliable resources. *Operational Research*, 1-35.

Elloumi, S., & Fortemps, P. (2010). A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 205(1), 31-41.

Esteban, A., Zafra, A., & Romero, C. (2020). Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization. *Knowledge-Based Systems*, 194, 105385.

Khamechian, M., & Petering, M. E. H. (2022). A mathematical modeling approach to university course planning. *Computers & Industrial Engineering*, 168, 107855.

Mitchell, M., Leachman, M., & Masterson, K. (2016). Funding down, tuition up: State cuts to higher education threaten quality and affordability at public colleges. *Washington, DC. Report by the Center of Budget and Public Priorities.*

Mohamed, A. (2015). A decision support model for long-term course planning. *Decision Support Systems*, 74, 33-45.

Morrow, T., Hurson, A. R., & Sarvestani, S. S. (2017). A multi-stage approach to personalized course selection and scheduling. *IEEE International Conference on Information Reuse and Integration* (pp. 253-262).

Myszkowski, P. B., & Siemieński, J. J. (2016). GRASP applied to multi–skill resource–constrained project scheduling problem. *International Conference on Computational Collective Intelligence.*

Myszkowski, P. B., Skowroński, M. E., & Podlodowski, Ł. (2013). Novel heuristic solutions for multi-skill resource-constrained project scheduling problem. *2013 Federated Conference on Computer Science and Information Systems.*

Shakhsi-Niaei, M., & Abuei-Mehrizi, H. (2020). An optimization-based decision support system for students' personalized long-term course planning. *Computer Applications in Engineering Education*, 28(5), 1247-1264.

Skowroński, M. E., Myszkowski, P. B., Adamski, M., & Kwiatek, P. (2013). Tabu search approach for multi-skill resource-constrained project scheduling problem. *2013 Federated Conference on Computer Science and Information Systems.*

Smith, W. C., Fraser, P., Chykina, V., Ikoma, S., Levitan, J., Liu, J., & Mahfouz, J. (2017). Global citizenship and the importance of education in a globally integrated world. *Globalisation, Societies and Education*, 15(5), 648-665.

Sowter, B. (2017). How to claim a place amongst the top 1% of world universities? https://www.qs.com/claim-place-amongst-top-1-world-universities/, website accessed September 22, 2021.

UW-Milwaukee Industrial Engineering Curriculum. (2021). https://catalog.uwm.edu/engineering-applied-science/industrial-manufacturing-engineering/industrial-engineering-bse/#requirementstext, website accessed September 22, 2021.

Winters, J. V. (2011). Human capital, higher education institutions, and quality of life. *Regional Science and Urban Economics*, 41(5), 446-454.

Yannibelli, V., & Amandi, A. (2013). Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem. *Expert Systems with Applicatio*ns, 40(7), 2421-2434.

Zapata, J. C., Hodge, B. M., & Reklaitis, G. V. (2008). The multimode resource constrained multiproject scheduling problem: Alternative formulations. *AIChE Journal*, 54(8), 2101-2119.

Chapter 2: University course scheduling during a pandemic

2.1. Introduction

Education is one of the most important aspects of human life in a modern society. Universities and educational institutes play a vital role in this regard. In late 2019 a new coronavirus named SARS-CoV-2 (i.e., COVID-19) was identified in China which quickly became a pandemic and affected the whole world and has killed more than 900,000 people in the United States and 6.2 million people worldwide (Dong et al., 2020). Before facing the pandemic, the latest research on education shows the importance of in-person classroom environments that facilitate discussions to enhance critical thinking and communication skills (Freeman et al., 2014). However, having students physically in classrooms to engage with their instructor and peers is in direct conflict with the research on the COVID-19 pandemic, which has shown that transmission of the coronavirus is highest when people are sitting indoors for a long period and talking (de Oliveira et al., 2021).

Around the world, governments have taken drastic steps to slow the spread of the virus by closing most of the organizations which require face-to-face interaction. Although universities and academic institutions require face-to-face interaction, closing universities altogether was not an option, so universities had to re-think how they offered their courses. The main problem was that classroom capacities suddenly decreased by about six fold because students had to socially distance to meet the U.S. Centers for Disease Control and Prevention (CDC) guidelines. Most universities responded to this situation by offering courses in three formats (with 0 or 1 classrooms assigned to a course): (a) online, (b) hybrid, and (c) in-person. Each format has its own disadvantages. With an online format, there is little student interaction. A hybrid format has multiple cons such as low classroom utilization, health risks, and/or overworked teaching staff. An

in-person format, on the other hand, poses health risks if social distancing is not enforced or is limited to low-enrollment courses being scheduled in large classrooms if social distancing is enforced.

This research proposes an alternate framework for offering university courses during a pandemic in which multiple classrooms may be assigned to the same course. In this approach, students in the same course gather for a limited number of socially distanced, in-person meetings called *face-to-face meetings* (i.e., f2f meetings) each semester. During each f2f meeting, all students in the course simultaneously spread out across multiple classrooms in a socially distanced manner. The instructor teaches in one classroom, and a video of the instructor is displayed in all rooms simultaneously. Alternatively, an exam could be scheduled during a f2f meeting, in which case one or more proctors are present to monitor the exam in each classroom.

In this chapter we introduce this alternative framework for university course scheduling during a pandemic. We develop a mathematical model of the problem and compare two methods for solving it, exact and heuristic. In the exact method, we use the IBM ILOG CPLEX solver to solve problem instances. We also develop a heuristic approach which uses simulated annealing principles to get a high quality solution using a reasonable amount of computation time. Both methods are coded in C++. Experiments on small, medium-sized, and large fictional instances show promising results.

2.2. Literature review

This research relates to the general area of educational timetabling. In the previous chapter we discussed four problems that relate to educational timetabling (Figure 2). Whereas Chapter 1 focused on the resource-constrained project scheduling problem (RCPSP), student course planning

problem (SCPP), and department course planning problem (DCPP), this chapter deals with the university course scheduling problem (UCSP).

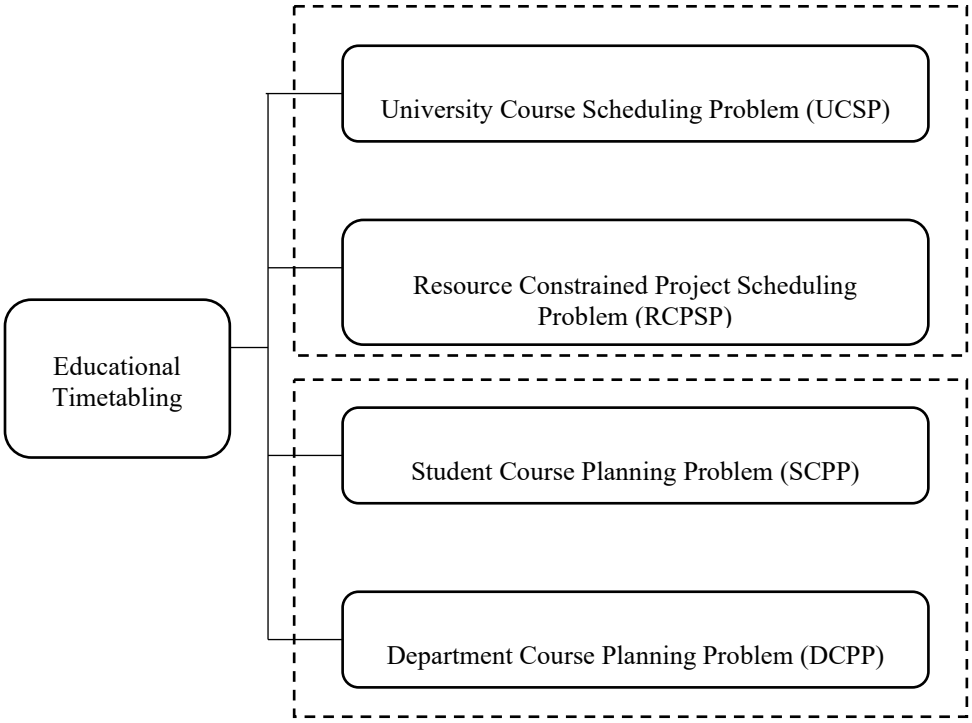

Figure 2. Research related to course planning and scheduling.

The university course scheduling problem (UCSP) is a well-known and highly constrained real-world problem. It is a timetabling problem that deals with scheduling a predetermined number of courses to time slots and resources (i.e., resources) considering several constraints (Chiarandini et al., 2006; Imran Hossain et al., 2019; Mencía et al., 2016; Tang et al., 2018). The goal of the UCSP is to assign all university lectures and laboratory sessions to rooms and timeslots (and possibly instructors) considering each room's maximum capacity, the expected number of students enrolled in each course, and other related facility related issues (Feizi-Derakhshi et al., 2012; Naji Azimi, 2005). The UCSP is an NP-complete problem, meaning that approaches that are guaranteed to provide an optimal solution are often too time-consuming, so heuristic and metaheuristic approaches are often utilized (Goh et al., 2017). Hard and soft constraints may vary from institution

to institution based on their resources and facilities. Typical hard constraints in the UCSP require that at most one lecture or laboratory session be assigned to each classroom during each timeslot and that students attend at most one course at any time. As an example of a soft constraint, Shiau (2011) solved a UCSP in which instructors and students indicate their course preferences, along with their preferred days and times for attending courses.

Articles on the UCSP consider either a prior-enrollment or post-enrollment perspective. The prior-enrollment perspective groups students according to their study curriculum and student grade; hence it is also named curriculum-based scheduling (Jamal, 2020). This approach consists of the weekly scheduling of the lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curriculum published by the university and not based on enrollment data. In post-enrollment course scheduling, course times and locations are decided after students have enrolled in courses. This type of scheduling considers individual students and faculty members, and the main goal is to schedule courses so that all students can feasibly attend the courses in which they have enrolled. (Bettinelli et al., 2015).

Many methods for addressing the prior-enrollment UCSP have been proposed including integer programing formulations (da Fonseca et al., 2017) and heuristics such as simulated annealing (Abramson, 1991), local search (da Fonseca et al., 2016), harmony search algorithms (Al-Betar & Khader, 2012), genetic algorithms (Lewis & Paechter, 2005), and adaptive tabu search algorithms (Lü & Hao, 2010). Other researchers have used swarm intelligence optimization methods such as ant colony optimization (Ayob & Jaradat, 2009), honey-bee mating optimization (Sabar et al., 2012), and particle swarm algorithms (Imran Hossain et al., 2019). Chiarandini et al.

(2006) compares the performance of various metaheuristic algorithms for the prior-enrollment UCSP including simulated annealing, variable neighborhood descent, and tabu search.

A different approach is post-enrollment scheduling, in which individual students' enrollment is explicitly accounted for. In post-enrollment course scheduling, course times and locations are decided after students have enrolled in courses. An important constraint is that no student may attend more than one event at the same time. The objective is typically to minimize the penalty for undesirable situations in which a student has a single class on a day, more than two classes in a row, or a class in the last time slot of a day (Jat & Yang, 2011). Méndez-Díaz et al. (2016) propose an integer programming formulation that is heuristically solved to produce high-quality solutions. Goh et al. (2017) solve a post-enrollment UCSP by combining two different search algorithms into an iterative two-stage procedure. In stage 1, tabu search with sampling and perturbation generates feasible solutions. In stage 2, simulated annealing with reheating is used to improve the quality of feasible solutions. Gonzalez et al. (2018) explain that advances in integer programming solvers such as CPLEX have made exact approaches possible.

Recently, due to room capacity constraints, the question of assigning students to courses has sparked interesting work in mechanism design (Budish et al., 2017). For instance, Atef Yekta & Day (2020) introduced five new algorithms for course allocation problem using various combinations of existing methods such as matching algorithms.

As we mentioned before, the recent pandemic has opened a new window of research opportunities to optimally utilize university classroom seats. During the pandemic, classroom capacities have decreased by a factor of about six to comply with CDC guidelines, and the effective utilization of scarce classroom space has become more important than ever.

To our knowledge, there are only three published works that consider university course scheduling during a pandemic. Barnhart et al. (2021) propose a unified model for university course scheduling under a two-stage framework. They use integer optimization combined with enrollment data from thousands of past students in the MIT Sloan School of Business. Their model's objective is to maximize the number of courses that students can take, with a preference for an on-campus experience. Notable assumptions in their study are as follows:

- A four-fold reduction in classroom capacity
- A maximum of two classrooms can be used for the same course
- Some courses' preferred format is fully online

Johnson & Wilson (2022) propose a multi-objective assignment model for scheduling classrooms during COVID-19 at the Spears School of Business at Oklahoma State University. The authors surveyed students, faculty, and staff to learn about their preferences and concerns. The results indicated that a majority of students wished to return to campus. In their model, instead of allowing simultaneous teachings, they used rotations. For example, in a Tuesday-Thursday class, half of the students might attend physically on Tuesday, whereas the other half might attend physically on Thursday. In this example, there are two rotations. Students might attend remotely or learn other online modules during their off-rotation days. Their model has three objectives: (1) maximize the number of courses that meet f2f with (two or more) rotations; (2) minimize the number of f2f courses that have three or more rotations; and (3) maximize the number of courses that stay in the same classroom.

Navabi-Shirazi et al. (2022) propose an integer programming model for simultaneously assigning course modes and classrooms to class sections when classroom capacities are reduced by 75-80%. They define four possible teaching modes:

- Residential: The section is held in-person with all students attending every class.
- Hybrid Split: The section is simultaneously taught online and in-person with students attending in-person on a rotating basis.
- Hybrid Touch Point: Most class delivery takes place online, but a few f2f meetings are scheduled each semester, so students can touch base with the instructor
- Remote: The section is fully online

Their work assumes that at most one classroom is assigned to a course section and all course sections are delivered in their previously assigned time slots due to existing registrations and ease of administrative implementation. They use hierarchical optimization to handle multiple optimization criteria according to priorities.

This chapter presents an alternate approach to university course scheduling during a pandemic in which multiple classrooms may be assigned to a course and all courses—even the largest—have one or more f2f meetings each semester. Our approach has the following desirable features.

- Practical: To the authors' best knowledge, no one has solved this problem in a way that guarantees that all courses—even the largest—have an in-person component. The aforementioned methods from the literature generally assign the largest courses to a fully online teaching mode so their classrooms can be used for smaller courses.

- Flexible: Our approach is flexible and always provides a high-quality feasible solution. The preferred timing of a course's f2f meetings for each course can be customized. If the preferred timing of f2f meetings cannot be perfectly satisfied, our approach recommends other nearby timeslots for the course's f2f meetings. Additionally, the model can be used to schedule a minimum number of f2f meetings for each course.
- Multi-criteria: Our approach considers seven different objectives that are weighted to reflect changing priorities identified by faculty members and university administrators.
- Scalable: Our approach can handle small and large problem instances. Our smallest problem instance has 40 courses, 10 classrooms, 4 weeks, and 15 timeslots per week, and our largest problem instance has 600 courses, 60 classrooms, 16 weeks, and 15 timeslots per week.

2.3. Problem description

We call our problem the university course scheduling problem during a pandemic (UCSPDP). Consider a university that wants schedule face-to-face (f2f) lectures (i.e., meetings) for a set of courses that it offers during a pandemic. All classrooms have a lower capacity than usual to be able to practice social distancing as recommended by the CDC or other public health organization. Although classroom capacity is limited, the university still wants students in all courses—even the largest—to have a limited number of socially distanced f2f meetings during the semester when *all* students in the course meet in person (in one or more rooms) *simultaneously*. We assume the university is not interested in splitting a course into separate groups that meet at separate times.

The university wishes to decide when and where all of the coming semester's f2f meetings will take place. It makes these decisions based on the following information. There are total of R

rooms available to be assigned. The parameter C_r indicates the (reduced) capacity of room r with social distancing guidelines in place. There are total of C courses (i.e., course sections) that need to be scheduled. The parameter S_c indicates the number of students enrolled in (or expected to be enrolled in) course c. Parameters W and T refer to the total number of weeks and timeslots that are being used in the schedule, respectively. Each time slot refers to a different continuous time period during the week when a f2f meeting can take place. For example, if the university wants to assign classrooms to courses for a 16-week semester and there are five business days each week and five 2-hour timeslots beginning at 08:00, 10:00, 12:00, 14:00, and 16:00 each day, then $W = 16$ and $T = 25$.

Instructors must inform the university about how many f2f meetings (including in-person exam sessions) they want during each week. Parameter DN_{cw} indicates the number of f2f meetings for course c that are desired to take place during week w. Parameter $CuDN_{cw}$ indicates the cumulative number of f2f meetings for course c that are desired to take place during weeks 1–w combined. For example, if the number of desired f2f meetings for course 3 during an 8-week semester is DN_{3w} = [0, 1, 0, 2, 0, 3, 0, 1], then $CuDN_{3w}$ = [0, 1, 1, 3, 3, 6, 6, 7]. Parameters DN_{cw} and $CuDN_{cw}$ reflect what the instructor of course c desires which may deviate from what is possible. If all course instructors desire many f2f meetings, it will not be possible to schedule all such meetings. However, if instructors desire roughly one f2f meeting every six class sessions then it will likely be possible to schedule most such meetings (assuming that classroom capacities are reduced by a factor of no more than six).

Parameter $Dist_{rs}$ is the distance between classrooms r and s, and parameter $DistOff_{cr}$ is the distance between the office of the professor who teaches course c and room r. These parameters are implemented to reflect the difficulty of traveling from one room to another which may differ

greatly from the actual distance (in meters) between the centroids of the rooms. Binary parameter J_{cr} equals 1 if course c is eligible to be scheduled in room r and equals 0 otherwise. Binary parameter I_{ct} equals 1 if course c is eligible to have a f2f meeting during time slot t and equals 0 otherwise. Parameters J_{cr} and I_{ct} are resource availability parameters. Parameter $MaxTS_c$ is the maximum number of weekly time slots that can be associated with course c in the timetable published by the university.

The goal of the UCSPDP is to schedule f2f meetings for each course (a) in a set of rooms that do not violate J_{cr} that have enough combined seats to host the selected course; (b) during weeks when the meetings are desired as specified by DN_{cw}; (c) during time slots that agree with I_{ct}; and (d) so that two courses do not use the same classroom during the same time slot in the same week. Additional goals of the UCSPDP are to (e) minimize the distance between rooms assigned to a course ($Dist_{rs}$); (f) minimize the distance from the office of the professor who teaches a course and the classrooms used for the course ($DistOff_{cr}$); (g) minimize the number of rooms assigned to a course; and (h) schedule as many f2f meetings as possible for each course regardless of when they take place. The UCSPDP is a highly constrained, nontrivial optimization problem with multiple objectives. Advanced methods are therefore needed to address this challenging problem.

2.4. Exact solution approach using a mathematical model

We propose two methods to solve this problem. The first method is to develop a mathematical model—in particular an integer programming model—and then call the IBM ILOG CPLEX solver to solve it for various problem instances. The second method, described in Section 2.5, is a heuristic algorithm aimed at obtaining good solutions quickly.

Table 14 lists the indices, parameters, and decision variables in our integer programming (IP) formulation of the problem. The integer program, which we call model UCSPDP, has four categories of indices. Indices *r* and *s* represent classrooms. Two indices are needed to be able to represent the distance between rooms. Index *c* refers to the courses. Indices *w* and *k* refer to weeks. Index *t* refers to the time slots within each week.

Table 14. Indices, parameters, and decision variables in mathematical model UCSPDP

Indices	
r, s	Rooms
c	Courses (i.e., course sections)
w, k	Weeks
t	Time slots
Parameters	
R	Total number of available rooms
C_r	Capacity of room r (during COVID-19)
C	Number of courses for which at least one f2f meeting is desired
W	Total number of weeks in the semester (e.g., 16)
T	Number of unique time slots when a f2f meeting can occur within a week (e.g., 15)
S_c	Number of students enrolled in (or expected to be enrolled in) course c
DN_{cw}	Number of f2f meetings for course c that are desired to take place during week w
$CuDN_{cw}$	Cumulative number of f2f meetings for course c that are desired to take place during weeks 1-w combined
$Dist_{rs}$	Distance between rooms r and s
$DistOff_{cr}$	Distance between office of professor who teaches course c and room r
J_{cr}	= 1 if course c is eligible to be scheduled in room r = 0 otherwise (binary)
I_{ct}	= 1 if course c is eligible to be scheduled during time slot t = 0 otherwise (binary)
$MaxTS_c$	Maximum number of weekly time slots that can be associated with course c in the timetable published by the university
$\alpha1, \alpha2 \ldots \alpha7$	Weights for objective function components (real, > 0)
Decision variables	
X_{ctw}	= 1 if course c has a f2f meeting during time slot t in week w = 0 otherwise (binary)
Y_{cr}	= 1 if room r is used for course c = 0 otherwise (binary)
Z_{crtw}	= 1 if course c has a f2f meeting in room r during time slot t in week w = 0 otherwise
U_c	= 1 if at least one f2f meeting is scheduled for course c during the semester = 0 otherwise (binary)
V_{ct}	= 1 if the university's published timetable states that course c has f2f meetings during time slot t = 0 otherwise (binary)
AN_{cw}	Actual number of f2f meetings for course c that are scheduled to take place during week w (integer, ≥ 0)
$CuAN_{cw}$	Cumulative number of f2f meetings for course c that are scheduled to take place during weeks 1-w combined (integer, ≥ 0)
$Diff_{cw}$	Deviation between the actual and desired number of f2f meetings for course c that are scheduled to take place during weeks 1-w combined (integer, ≥ 0)
$NumRm_c$	Number of rooms assigned to course c (integer, ≥ 1)
$WastedSeats_c$	Number of empty seats in course c's room assignment (integer, ≥ 0)
M_c	Maximum distance between rooms assigned to course c (real, ≥ 0)
N_c	Maximum distance between office of professor who teaches course c and any room assigned to course c (real, ≥ 0)

The model has twelve sets of decision variables. Binary variable X_{ctw} equals 1 (0) if course c has (does not have) a f2f meeting during time slot t in week w. Binary variable Y_{cr} equals 1 (0) if room r is used (is not used) for course c's f2f meetings. Binary variable Z_{crtw} equals 1 (0) if course c has (does not have) a f2f meeting in room r during time slot t in week w. Binary variable U_c equals 1 if at least one f2f meeting is scheduled for course c during the semester. Binary variable V_{ct} equals 1 (0) if the university's published timetable states that course c has f2f meetings during time slot t. Integer variable AN_{cw} equals the actual number of f2f meetings for course c that are scheduled to take place during week w. Integer variable $CuAN_{cw}$ equals the cumulative number of f2f meetings for course c that are scheduled to take place during weeks 1–w combined. Integer variable $Diff_{cw}$ equals the deviation between the actual and desired number of f2f meetings for course c that are scheduled to take place during weeks 1-w combined. Integer variables $NumRm_c$ and $WastedSeats_c$ equal the number of rooms assigned to course c and the number of empty seats that course c's room assignment, respectively. Real variables M_c and N_c equal the maximum distance between rooms assigned to course c and the maximum distance between the office of the professor who teaches course c and a room assigned to course c, respectively. Our integer programming model is shown below.

Minimize :

$$
\begin{aligned}
&(\alpha_1) * \left(\sum_{c=1}^{C} S_c * NumRm_c\right) + \\
&(\alpha_2) * \left(\sum_{c=1}^{C} S_c * M_c\right) + \\
&(\alpha_3) * \left(\sum_{c=1}^{C} S_c * N_c\right) + \\
&(\alpha_4) * \left(\sum_{c=1}^{C} S_c * WastedSeats_c\right) + \\
&(\alpha_5) * \left(\sum_{c=1}^{C} S_c * (CuDN_{cW} - CuAN_{cW})\right) + \\
&(\alpha_6) * \left(\sum_{c=1}^{C} \sum_{w=1}^{W} S_c * Diff_{cw}\right) + \\
&(\alpha_7) * \left(\sum_{c=1}^{C} S_c * (1 - U_c)\right)
\end{aligned}
\tag{34}
$$

Constraints:

$\sum_{c=1}^{C} Z_{crtw} \leq 1$	for all *r*, *t*, and *w*	(35)
$\sum_{r=1}^{R} C_r * Y_{cr} \geq S_c$	for all *c*	(36)
$V_{ct} \leq I_{ct}$	for all *c* and *t*	(37)
$X_{ctw} \leq V_{ct}$	for all *c*, *t,* and *w*	(38)
$Y_{cr} \leq J_{cr}$	for all *c* and *r*	(39)
$Z_{crtw} \leq Y_{cr}$	for all *c*, *r*, *t*, and *w*	(40)
$Z_{crtw} \leq X_{ctw}$	for all *c, r, t*, and *w*	(41)
$Z_{crtw} \geq Y_{cr} + X_{ctw} - 1$	for all *c, r, t*, and *w*	(42)
$\sum_{w=1}^{W} AN_{cw} \leq \sum_{w=1}^{W} DN_{cw}$	for all *c*	(43)
$Dist_{rs} \leq M_c + (\text{BigM})*(2 - Y_{cr} - Y_{cs})$	for all *c*, *r*, and *s*	(44)
$Diff_{cw} \geq CuDN_{cw} - CuAN_{cw}$	for all *c* and *w*	(45)
$Diff_{cw} \geq CuAN_{cw} - CuDN_{cw}$	for all *c* and *w*	(46)
$\sum_{t=1}^{T} X_{ctw} = AN_{cw}$	for all *c* and *w*	(47)
$CuAN_{cw} = \sum_{k=1}^{w} AN_{ck}$	for all *c* and *w*	(48)
$\sum_{t=1}^{T} V_{ct} \leq MaxTS_c$	for all *c*	(49)
$Z_{crtw} \leq V_{ct}$	for all *c, r, t*, and *w*	(50)
$DistOff_{cr} \leq N_c + (\text{BigM})*(1 - Y_{cr})$	for all *c* and *r*	(51)
$U_c \leq \sum_{t=1}^{T} \sum_{w=1}^{W} X_{ctw}$	for all *c*	(52)
$NumRm_c = \sum_{r=1}^{R} Y_{cr}$	for all *c*	(53)
$WastedSeats_c = (\sum_{r=1}^{R} C_r * Y_{cr}) - S_c$	for all *c*	(54)

The objective function (34) has seven parts that are weighted by positive values α1 to $\alpha 7$. Weights can be modified based on the importance of the objectives in different applications. Each part of the objective function is a sum of C individual penalty values, one for each course. The penalty value for course c is weighted by the number of students in the course, S_c. The seven main parts of the objective function emphasize the following goals respectively.

1. Minimize the number of rooms used for a course.
2. Minimize the maximum distance between rooms assigned to a course.
3. Minimize the maximum distance between the office of the instructor who teaches a course and any room assigned to the course.
4. Minimize the number of wasted seats in a course's room assignment.
5. Minimize the difference between the total number of f2f meetings scheduled for a course and total the number of f2f meetings that were desired.
6. Optimize the timing when f2f meetings are held. In other words, minimize the sum of $Diff_{cw}$ over all w for each course c (more details are provided in the paragraph below).
7. Minimize the number of courses with no f2f meetings scheduled

We now provide a few examples to illustrate these goals. Regarding goals 2 and 3, consider a set of three rooms assigned to a course with the pairwise distances, and distances from the instructor's office to each room, shown below on the left and right, respectively.

$$Dist_{rs} = \begin{bmatrix} 0 & 536 & 445 \\ 536 & 0 & 653 \\ 445 & 653 & 0 \end{bmatrix} \qquad DistOff_{cr} = [456 \quad 214 \quad 785]$$

According to the above values, the maximum distance between rooms assigned to the course is 653 and the maximum distance between the office of the professor who teaches the course and a

room assigned to it is 785. Regarding goal 4, consider a course with 45 students (S_c = 45) that is assigned to three rooms with capacities of 23, 18, and 10. In this case the number of wasted seats for this course is 6 (= 23 + 18 + 10 – 45). Goal 4 is to assign each course to a set of rooms in a way that minimizes the number of wasted seats. To explain goals 5 and 6, consider an 8-week scenario with the following values for parameters DN_{cw} and $CuDN_{cw}$ and decision variables AN_{cw} and $CuAN_{cw}$ for a particular course c:

DN_{cw}: 0 1 0 2 1 0 3 1 $\quad$ $CuDN_{cw}$: 0 1 1 3 4 4 7 8

AN_{cw}: 0 1 0 1 1 0 2 0 $\quad$ $CuAN_{cw}$: 0 1 1 2 3 3 5 5

$Diff_{cw}$: 0 0 0 1 1 1 2 3

Goal 5 is to minimize the difference between the total number of f2f meetings scheduled and the number of f2f meetings that were desired. For the above data this difference is 3 (= 8 – 5). Goal 6 is to optimize the timing of a course's f2f meetings. The timing of the meetings is evaluated by computing $Diff_{cw}$ (shown above) which is the absolute value of the difference between $CuDN_{cw}$ and $CuAN_{cw}$ for each week w. The $Diff_{cw}$ values for all w are then summed. In the above case, the sum of the $Diff_{cw}$ values is 8 (= 0 + 0 + 0 + 1 + 1 +1 + 2 + 3) which is the total amount of deviation in the timing of courses c's actual, scheduled f2f meetings compared to the timing that was desired.

We now discuss the constraints of the math model in detail. Constraint 35 ensures that no more than one course meets in the same room at the same time. Constraint 36 is a capacity constraint that ensures that the combined capacity of all rooms assigned to a course is greater than the number of students in that course. In other words, it ensures that each course is assigned to rooms with enough combined capacity to host it. Constraint 37 ensures that the university's published timetable may state that course c has f2f meetigns during time slot t only if I_{ct} = 1.

Constraint 38 ensures that all f2f meetings for course c are scheduled during timeslots that agree with the university's published timetable. Constraint 39 ensures that course c can only be scheduled in room r if $J_{cr} = 1$. Constraints 40, 41, and 42 ensure that the values of the Z variables agree with those of the Y and X variables. If either the Y or X variable is zero, the Z variable must be zero (constraints 40 – 41). If the Y and X variables are both 1, the Z variable must be 1 (constraint 42). Constraint 43 ensures that the total number of actual, scheduled f2f meetings for course c does not exceed the total number that is desired. Constraint 44 ensures that variable M_c is properly computed. In particular, if course c is scheduled in both rooms r and s, this constraint ensures that M_c is greater than or equal to the distance between rooms r and s. If course c is not scheduled in both rooms r and s, this constraint does not restrict any decision variables because the "BigM" is multiplied by a nonzero value on the right side. Constraints 45 and 46 make sure that $Diff_{cw}$ is properly computed. In particular, they ensure that $Diff_{cw}$ is greater than or equal to the deviation between the cumulative actual and cumulative desired number of f2f meetings for coure c during weeks 1-w. For all c and w, constraint 47 ensures that AN_{cw} is properly computed based on X_{ctw}. Constraint 48 ensures that $CuAN_{cw}$ is properly computed based on AN_{cw}. Constraint 49 ensures that $MaxTS_c$ is the maximum number of weekly time slots that can be allocated to course c in the timetable published by the university. Constraint 50 ensures that the values of the Z variables agree with the values of the V variables. Constraint 51 ensures that N_c is properly computed. In particular, if course c is assigned to room r, then the distance between the office of the instructor who teaches course c and room r is less than or equal to N_c. Constraint 52 ensures that U_c is correctly computed based upon the X_{ctw} variables. Constraint 53 ensures that $NumRm_c$ is correctly computed. Constraint 54 ensures that $WastedSeats_c$ is properly computed.

2.5. Heuristic method

The experimental results in Section 2.6.2 show that directly applying the mathematical model to instances of the UCSPDP is not effective. In particular, the IBM ILOG CPLEX solver fails to obtain satisfactory solutions to large instances of the UCSPDP within an hour. Thus, a heuristic method was developed for addressing the UCSPDP.

2.5.1. Overall structure of the heuristic method

Figure 3 shows the overall procedure of our heuristic. The sections that follow describe various steps in detail.

Overall heuristic method for the UCSPDP:

1 For each course c, use an exact or heuristic method to generate a set of *potential room assignments* (*PRAs*) (i.e., sets of rooms where the course could feasibly be held) that have good values for $NumRm_c$, $WastedSeats_c$, M_c, and N_c.

2 Create an initial schedule for all courses in which as many f2f meetings as possible are scheduled, assuming that each course's first PRA is used. This is the *current solution.*

3 Compute the objective value of the current solution

4 If the time limit has been reached, STOP and display the best solution that was found. If not, go to step 5.

5 Create a *neighboring solution* by (a) removing all f2f meetings for a subset of courses from the current solution, (b) selecting a new, random PRA for each such course, (c) forming a waitlist of one or more unscheduled f2f meetings, and (d) placing as many of the waitlisted f2f meetings as possible back into the schedule.

6 Compute the objective value difference between the neighboring and current solutions. Use simulated annealing principles to decide if the neighboring solution replaces the current solution. Go to step 3.

Figure 3. Heuristic pseudocode

2.5.2. *Step 1: Generate potential room assignments (exact method)*

The first step in the heuristic method is to generate different room combinations for each course that are able to host all students in the course. We develop two methods to automatically generate these potential room assignments (PRAs). In first method, we use a mathematical model to generate different room combinations ranked from best to worst. The model runs several times, and the result of each run is the best room assignment that has not yet been generated. We set a limit on how many different room combinations are computed. Table 15 lists the indices, parameters, and decision variables in the mini math model that generates the PRAs.

Table 15. Indices, parameters, and decision variables in the mini math model that generates room possibilities for each course

Indices	
r, s	Rooms
Parameters	
R	Total number of available rooms
S	Number of students enrolled in (or expected to be enrolled in) the course at hand
C_r	Capacity of room r (during COVID-19)
$MaxM$	Maximum allowed value of the decision variable M
$MaxN$	Maximum allowed value of the decision variable N
$Dist_{rs}$	Distance between rooms r and s
$DistOff_r$	Distance between room r and office of instructor who teaches the course at hand
J_r	= 1 if the course on hand is eligible to be scheduled in room r = 0 otherwise (binary)
$\alpha 1, \alpha 2, \alpha 3, \alpha 4$	Weights for parts of the objective function (real, ≥ 0)
Decision variables	
Y_r	= 1 if room r is used for the course at hand = 0 otherwise (binary)
M	Maximum distance between rooms assigned to the course at hand
N	Maximum distance between the office of the professor who teaches the course and a room assigned to it
$WastedSeats$	Number of empty seats in course c's room assignment (integer, ≥ 0)

Most of the indices, parameters, and decision variables are similar to elements in Table 14. The model is applied to each course individually. Hence, the course index c is removed. Our integer programming model is shown below.

Minimize: $(\alpha_1) * (\sum_{r=1}^{R} Y_r) + (\alpha_2) * M + (\alpha_3) * N + (\alpha_4) * WastedSeats$ (55)

Constraints**:**

$$\sum_{r=1}^{R} C_r * Y_r \geq S \quad (56)$$

$DistOff_r \leq N + (\text{BigM}) * (1 - Y_r)$ for all r (57)

$Y_r \leq J_r$ for all r (58)

$Dist_{rs} \leq M + (\text{BigM}) * (2 - Y_r - Y_s)$ for all r and s (59)

$M \leq MaxM$ (60)

$N \leq MaxN$ (61)

$WastedSeats = (\sum_{r=1}^{R} C_r * Y_r) - S$ (62)

This model has four objectives that are weighted by nonnegative values $\alpha 1$, $\alpha 2$, $\alpha 3$, and $\alpha 4$ respectively:

1. Minimize the number of rooms used for the course
2. Minimize the distance between rooms assigned to the course
3. Minimize the distance between the office of the professor who teaches the course and any room used for the course
4. Minimize the total number of wasted seats in the course's room assignment

Constraint 56 is a capacity constraint that ensures that the combined capacity of all rooms assigned to the course is greater than the number of students enrolled in the course. Constraint 57 ensures that *N* is an upper bound on the distance between the office of the instructor who teaches

the course and any room assigned to it. Constraint 58 ensures that room *r* may only be used for the course if room *r* is compatible with the course. Constraint 59 ensures that *M* is an upper bound on the distance between any two rooms assigned to the course. Constraints 60 and 61 make sure that *M* and *N* are less than *MaxM* and *MaxN*, respectively. Constraint 62 ensures that *WastedSeats* is properly computed.

The model is applied to each course separately. For each individual course, the model is solved many times. Each time an optimal solution is identified, one additional constraint is added to the model to forbid that solution from appearing when the model is solved again. Let the parameter *NumRooms* equal the number of rooms used (e.g., 4) in the optimal solution that was most recently generated when the model was solved, and let R_i be the i^{th} room used in the most recently generated solution. Then the new constraint that is added to the model each time is:

$$\sum_{i=1}^{NumRooms} Y_{R_i} \leq NumRooms - 1. \qquad (63)$$

This constraint states that no more than *NumRooms* – 1 of the *NumRooms* that were used in the most recently generated optimal solution may be used in the next optimal solution. In other words, constraint 63 ensures that the most recently generated optimal solution is no longer feasible the next time the model is solved. A constraint of this type is added to the model each time we run the model. For example, if 20 PRAs are to be generated, a constraint of this type is added a total of 19 times to the model to make sure that each previously generated solution is never generated again.

2.5.3. Step 1: Generate potential room assignments (heuristic method)

Using an exact method for generating PRAs has its own advantages and disadvantages. The main advantage is the quality of the generated PRAs. In other words, the exact method generates high quality PRAs, but the disadvantage of this method is the run time. When developing our heuristic approach to solve the UCSPDP (Figure 3), we aimed to allocate at most 1/3 of the total computation time for generating PRAs (i.e., for performing step 1 in Figure 3). With this restriction, only a limited number of PRAs could be generated for each course if they were generated using the mini math model. Therefore, we also developed a heuristic approach to generate PRAs. Figure 4 shows the logic of this heuristic. The heuristic is guided by the values of five input parameters: *PRA_limit*, *MaxRoom*, *MaxM*, *MaxN*, and *MaxWS*.

Heuristically generating PRAs

For each course *c*

Create a list of rooms available for course *c* based on the room availability matrix (J_{cr})
Let *#AvailRooms* equal the number of rooms in the list
Let *#PRAsMade* = 0, *#RoomsUsed* = 0, and *AllRecentlyMadePRAsRedundant* = false
While (*#PRAsMade* < *PRA_limit* & *#RoomsUsed* < *MaxRoom* & *AllRecentlyMadePRAsRedundant* = false)
    #RoomsUsed ++
    Let *#PossibitiesWithThisNumRooms* = choose(*#AvailRooms*, *#RoomsUsed*)
    Let *iterator* = 0
    Let *AllRecentlyMadePRAsRedundant* = true
    While (*iterator* < *#PossibitiesWithThisNumRooms* & *#PRAsMade* < *PRA_limit*)
        ** Generate the next PRA in the sequence **
        Iterator ++
        If (total seats in PRA < S_c)
            AllRecentlyMadePRAsAreRedundant = false
        Else
            If the PRA still has enough seats to accommodate course *c* even if one of its rooms is removed from it
                PRAisRedundant = true
            Else
                PRAisRedundant = false
            Compute the values of *M, N*, and *WastedSeats* for this PRA as defined in Table 15
            If (*PRAisRedundant* = false & $M \leq MaxM$ & $N \leq MaxN$ & $WastedSeats \leq MaxWS$)
                The current PRA is approved and joins the list of the PRAs generated for course *c*
                #PRAsMade ++
                AllRecentlyMadePRAsRedundant = false

Figure 4. Pseudocode for heuristically generating PRAs

The method generates PRAs exhaustively beginning with those with the fewest rooms. Only PRAs that have enough seats to accommodate the course are considered. Among these PRAs, only non-redundant PRAs that do not have enough seats to accommodate a course if any room is removed from the PRA are considered. Among these PRAs, only those with values of *M*, *N*, and *WastedSeats* (as defined in Table 15) that are below *MaxM*, *MaxN*, and *MaxWS*, respectively, are approved for use in the heuristic shown in Figure 3. The method terminates when then number of

approved PRAs reaches *PRA_limit;* all PRAs have been exhaustively considered; or all PRAs in the most recent PRA cohort (i.e., all PRAs with a given number of rooms) are determined to be redundant.

For example, consider a case in which only 3 rooms may be used for a course (according to J_{cr}). First, the method starts with room combinations with just 1 room. Then it computes total number of possible combinations consisting of just one room as (3 choose 1) which is 3. It starts with the first combination to be just {room1}. Then it increases the room number to reach the total number of possible combinations of just one room which is 3. Now we have three room combinations of {room1}, {room2}, and {room3}. Then so far possible room combinations are {1},{2}, and {3}. Then while number of PRAs and the number of rooms considered in each combination are within their threshold and the number of rooms considered in each combination is less than available rooms, it increases the number of rooms considered in each combination by 1. Then by incrementing each value in each combination, other combinations are being created. For each room combination M_1, N_1, and *WastedSeats* are computed to check it's within the limit. If not, this combination will not be considered. Also, it checks for redundancy in each combination as well by comparing each room capacity in each combination with number of students enrolled in the course. If there is a room with enough capacity for all student in the course in room combinations with two or more rooms, then this combination is redundant.

It should be mentioned that we considered two main approaches for heuristically generating the PRAs. In the first approach, we divide campus into quadrants, each with roughly the same supply of classroom seats and the same total demand for seating during the semester. We then require that a demand in one quadrant must be satisfied by the available classrooms in the same quadrant. The second method is to not divide the campus to quadrants. Depending on which

method is used, the room availability matrix J_{cr} (see line 1 in Figure 4) is modified to check the availabilities in each quadrant separately or for the entire campus. Preliminary experiments indicated that the quadrant approach was inferior to the non-quadrant approach, so the quadrant approach was abandoned.

2.5.4. Step 2: Create an initial schedule

Once the PRAs are created, the next step is to create an initial feasible solution. The initial feasible solution is constructed one f2f meeting at a time, not one course at a time. We first build a giant list of all desired f2f meetings (for all courses), each specified by a course number and week number. The week number for each desired f2f meeting comes from parameter DN_{cw}. The list is then randomly scrambled. We then proceed sequentially through the list and attempt to schedule each f2f meeting one at a time, during the week when it is desired, using the first PRA for the course that is associated with the f2f meeting. All eligible timeslots (indicated by I_{ct}) within the week at hand and for the course at hand are considered. If any timeslot works, the f2f meeting is placed in the schedule. If no timeslots work, the f2f meeting is not placed in the schedule. After considering all f2f meetings in such manner, we then revisit those which were not placed in the schedule, and we try to place each of them, one at a time, in the schedule one week before or after the f2f meeting's desired week. After considering all f2f meetings in this manner, we then revisit those not in the schedule, and we try to place each of them, one at a time, in the schedule two weeks before or two weeks after the f2f meeting's desired week. This process continues until all of the f2f meetings are scheduled or no more f2f meetings can be feasibly scheduled in any week.

2.5.5. Step 5: Create neighboring solution

Steps 3 and 4 of the heuristic (see Figure 3) are self explanatory. The next step (step 5) is to create a neighboring solution by (a) removing all f2f meetings for a subset of courses from the current solution, (b) selecting a new, random PRA for each such course, (c) forming a waitlist of one or more unscheduled f2f meetings, and (d) placing as many of the waitlisted f2f meetings as possible back into the schedule.

In (a), we define three options for deciding *how many* courses' f2f meetings should be removed from the current schedule:

1. A certain percentage of courses
2. A certain number of courses
3. All courses

In (a) we also need to decide *which* courses' f2f meetings to remove from schedule. We define six options for doing this.

1. The *N* courses with the greatest number of unscheduled f2f meetings (ties broken in favor of removing larger courses' f2f meetings)
2. The *N* courses with the lowest fraction of f2f meetings scheduled (ties broken in favor of removing larger courses' f2f meetings)
3. The *N* largest courses that have at least one unscheduled f2f meeting
4. The *N* courses whose scheduled f2f meeting timing has the highest total deviation from desired

5. The *N* courses whose scheduled f2f meeting timing has the highest average deviation from desired
6. *N* random courses

In step (c), we define three options for deciding which unscheduled f2f meetings are added to the waitlist.

1. All unscheduled f2f meetings
2. Only the unscheduled f2f meetings related to the course(s) removed from the schedule in step (a)
3. Only the unscheduled f2f meetings related to the course(s) with no f2f meetings scheduled right now

Step (d) proceeds very much like the creation of the initial feasible solution in step 2 of Figure 3. First, we randomly scramble the (unscheduled) f2f meetings in the waitlist. We then proceed sequentially through the waitlist and attempt to schedule each f2f meeting, one at a time, during the week when it is desired, using the PRA that was most recently selected for the course that is associated with the f2f meeting. As before, all eligible timeslots within the week at hand are considered. After considering all f2f meetings in such a manner, we revisit those that were not placed in the schedule, and we try to place each of them, one at a time, in the schedule one week before or after the f2f meeting's desired week. If any f2f meetings in the waitlist are still not scheduled, we revisit those f2f meetings, and we try to place each of them, one at a time, in the schedule two weeks before or two weeks after the f2f meeting's desired week. The process continues until all waitlisted f2f meetings are scheduled or no more waitlisted f2f meetings can be scheduled in any week.

2.5.6. Step 6: Decide if neighboring solution replaces current solution

In step 6, we compare the neighboring solution's objective value to the current solution's objective value. If the neighboring solution has a lower objective value or its objective value is within the simulated annealing (SA) acceptance range, the neighboring solution is accepted, and the neighboring solution replaces the current solution. The procedure then returns to step 3. This process continues until the time limit is reached. When the time limit is reached, the best solution that was found in displayed. Figure 5 shows the pseudocode for the heuristic with the simulated annealing steps shown. Two input parameters—*StartTemp* and *TempFactor*—are used in the procedure.

```
Perform steps 1 and 2 in Figure 3
Let S_Current = initial feasible soloution and Obj_Current = initial objective value
Let S_Best = S_Current and Let Obj_Best = Obj_Current
Let T = StartTemp
Let Iter = 0
While (time has not yet expired)
    Perform step 5: Generate neighboring solution (S_Next, Obj_Next)
    If (Obj_Next < Obj_Current) then
        S_Current = S_Next, Obj_Current = Obj_Next
    Else
        Let Rand be a random real number between 0 and 1
        Let Δ = Obj_Next − Obj_Current
        If (Rand < e^(−Δ/T))
            S_Current = S_Next, Obj_Current = Obj_Next
    If (Obj_Current < Obj_Best) then
        S_Best = S_Current, Obj_Best = Obj_Current
    Iter = Iter + 1
    T = T * TempFactor
Return S_Best and Obj_Best
```

Figure 5. Heuristic pseudocode with simulated annealing steps shown

2.6. Experimental setup, results, and discussion

The math model and heuristic method were tested on a variety of problem instances. In this section we describe the experimental setup, present the results, and discuss their significance.

2.6.1. General experimental setup

The math model and heuristic method were coded into MS Visual C++ 2015, and IBM ILOG Concert Technology was used to call IBM ILOG CPLEX 12.10 to solve instances of the math model contained in text files. All experiments were run on a desktop PC with an 11th generation intel core i7 (3.00 GHz) processor and 32 GB of RAM. Random cartesian (X, Y) coordinates ranging from 0 to 800 were generated to define the location of each classroom and the office of the professor who teaches each course, and the Euclidean distance formula is used to calculate the distance between each pair of rooms ($Dist_{rs}$) and the distance between each room and each professor's office ($DistOff_{cr}$).

For testing our approaches, we created 135 instances. These instances are categorized by their *size*—small, medium, large—based on the total available supply of seat-timeslots during the semester as computed in equation 64. All small (medium; large) instances have a seat-timeslot supply ranging from 6000-9000 (55,000-70,000; 370,000-420,000) for the entire semester. Each problem size is divided into three categories based on the room and time slot *availability* – 100%, 75%, 50% – which indicates the percentage of J_{cr} and I_{ct} values that are 1. Each category is further divided into to three subcategories, each with a different *demand level*: low demand (LD), medium demand (MD), and high demand (HD). For the purposes of computing the demand level, we define the *Supply* and *Demand* in a problem instance as follows.

$$Supply = (\sum_{r=1}^{R} C_r) * W * T \tag{64}$$

$$Demand = (\sum_{c=1}^{C} S_c * CuDN_{cW}) \tag{65}$$

The value of *Demand*/*Supply* in all LD (MD, HD) problem instances falls within the range 0.45-0.6 (0.65-0.8, 0.85-1). The three problem sizes, three availability levels, and three demand levels create 27 subcategories of problem instances. Five instances are created for each subcategory, making 135 instances altogether. It should be noted that set of I_{ct} and J_{cr} values that equal 1 in each instance with 50% availability is a strict subset of the set of I_{ct} and J_{cr} values that equal 1 in one of the instances with 75% availability.

Table 16 shows the values of the main input parameters—including R, C_r, C, W, T, and S_c—that define the problem instances. In this table, "DU" refers to the discrete uniform distribution. The expression "X, Y = DU(0,800)" means that each classroom and professor's office was given an X, Y coordinate ranging from 0 to 800, and parameters $Dist_{rs}$ and $DistOff_{cr}$ were computed as the straight line distance between two points. Also, the expression "P(DN_{cw}) = [.7, .15, .1, .05]" means that, for each course in all problem instances, the number of desired f2f meetings in any given week has a 70% chance being of 0, 15% chance of being 1, 10% chance of being 2, and 5% chance of being 3. Thus, the average number of f2f meetings desired per week is 0.5 (=0.7*0 + 0.15*1 + 0.1*2 + 0.05*3) for each course in all problem instances.

For example, each of the five "Large–100%–HD" instances (see the bottom-left portion of Table 16) has 60 rooms (R), classroom capacities generated from the DU(5, 50) distribution, 600 courses (C), 16 weeks (W), 15 time slots per week (T), and DU(10, 140) students in each course. In experiments testing the math model and heuristic, we set a computation time limit of 300, 1200, and 3600 seconds for small, medium, and large instances, respectively.

In all experiments, the values of α_1, α_2, α_3, α_4, α_5, α_6, and α_7, were set to 10, 0.01, 0.01, 10, 100, 10, and 100000, respectively (see equation 34).

Table 16. Experimental setup and assumptions for the problem instances

Small (supply = 6000 - 9000 seat timeslots)

J_{cr} & I_{ct}: 100%			J_{cr} & I_{ct}: 75%			J_{cr} & I_{ct}: 50%		
LD: 45-60%	**MD: 65-80%**	**HD: 85-100%**	**LD: 45-60%**	**MD: 65-80%**	**HD: 85-100%**	**LD: 45-60%**	**MD: 65-80%**	**HD: 85-100%**
R = 10	R = 10	R = 10	R = 10	R = 10	R = 10	R = 10	R = 10	R = 10
C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)	C_r = DU (5,20)
C = 40	C = 55	C = 70	C = 40	C = 55	C = 70	C = 40	C = 55	C = 70
W = 4	W = 4	W = 4	W = 4	W = 4	W = 4	W = 4	W = 4	W = 4
T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15
S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)	S_c = DU (10,80)
X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)
P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]

Medium (supply = 55,000 – 70,000 seat timeslots)

J_{cr} & I_{ct}: 100%			J_{cr} & I_{ct}: 75%			J_{cr} & I_{ct}: 50%		
LD: 45-60%	**MD: 65-80%**	**HD: 85-100%**	**LD: 45-60%**	**MD: 65-80%**	**HD: 85-100%**	**LD: 45-60%**	**MD: 65-80%**	**HD: 85-100%**
R = 30	R = 30	R = 30	R = 30	R = 30	R = 30	R = 30	R = 30	R = 30
C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)	C_r = DU (5,30)
C = 140	C = 190	C = 240	C = 140	C = 190	C = 240	C = 140	C = 190	C = 240
W = 8	W = 8	W = 8	W = 8	W = 8	W = 8	W = 8	W = 8	W = 8
T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15
S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)	S_c = DU (10,110)
X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)
P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]

Large (supply = 370,000 – 420,000 seat timeslots)

J_{cr} & I_{ct}: 100%			J_{cr} & I_{ct}: 75%			J_{cr} & I_{ct}: 50%		
LD: 45-60%	**MD: 65-80%**	**HD: 85-100%**	**LD: 45-60%**	**MD: 65-80%**	**HD: 85-100%**	**LD: 45-60%**	**MD: 65-80%**	**HD: 85-100%**
R = 60	R = 60	R = 60	R = 60	R = 60	R = 60	R = 60	R = 60	R = 60
C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)	C_r = DU (5,50)
C = 340	C = 470	C = 600	C = 340	C = 470	C = 600	C = 340	C = 470	C = 600
W = 16	W = 16	W = 16	W = 16	W = 16	W = 16	W = 16	W = 16	W = 16
T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15	T = 15
S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)	S_c = DU (10,140)
X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)	X,Y= DU (0, 800)
P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]	P(DN_{cw}) = [.7, .15, .1, .05]

2.6.2. Math model experimental setup, results, and discussion

The math model was tested on all problem instances in which the size is small (S) or large (L), the availability is 100% or 50%, and the demand level is low (LD) or high (HD). Thus, the math model was tested on all instances in eight problem subcategories (40 instances total). The CPLEX computation time limit for small and large problem sizes was set to 300 and 3600 seconds, respectively, and the node parameter was set to 3 to minimize the likelihood of CPLEX reaching an out-of-memory status.

Before discussing the results, it is important to mention that we divide courses into four groups based on how well their f2f meetings are scheduled in the best solution identified by CPLEX.

1. *Unscheduled courses* (UC): Courses with no scheduled f2f meetings.
2. *Partially scheduled courses* (PSC)*:* Courses in which the total number of scheduled f2f meetings is less than the desired number of f2f meetings.
3. *Fully scheduled courses* (FSC): Courses in which all desired f2f meetings are scheduled but not all in the desired weeks.
4. *Perfectly scheduled courses* (PeSC): Courses in which all desired f2f meetings are scheduled and they are scheduled in the desired weeks.

Table 17 shows the results of the experiments that tested the math model. Each row in the table refers to a different problem subcategory, and the results in each row are average results for five problem instances. Table 17 has 21 columns. The first column refers to the demand density of the problem instance which equals *Demand*/*Supply* as computed in equations 64-65. The next

two columns show the number of binary variables and constraints in the IP formulation that remains after CPLEX finishes preprocessing the initial IP formulation shown in equations 34-54. The next four columns show the time needed to solve the IP's LP relaxation, CPLEX gap for the best solution identified by CPLEX, number of wasted seats for all courses combined in the best solution identified by CPLEX, and density of the best solution found by CPLEX, respectively. The density equals the total number of seat time slots used divided by the total number of seat time slots available. The next column, "Density Gap," shows the difference between the instance density and the best solution density. The next four columns show the average number of unscheduled, partially scheduled, fully scheduled, and perfectly scheduled courses, respectively, in the best solution identified by CPLEX. The next seven columns show the values of the seven parts of the objective function (described in Section 2.4) in the best solution identified by CPLEX. The second to the last column shows the overall objective value which is the summation of the values in the previous seven columns. Finally, last column shows the objective value without objective 4 which is related to the wasted seats. The reason we added this column is that minimizing the number of wasted seats is an intermediate priority that is generally less important than other objectives such as the number of meetings lacking.

Table 17. Math model results

	Instance density (%)	#Binary vars	# Const	LP relax Soln Time (sec)	Cplex Gap (%)	# Wasted seats	Best Soln Density (%)	Density Gap (%)	#UC	#PSC	#FSC	#PeSC	Obj1	Obj2	Obj3	Obj4	Obj5	Obj6	Obj7	Best OV	Best OV w/o Obj 4
S-100%-LD	54.14	26816	75165	0.4	14.12	19	54.14	0	0	0	2.2	37.8	63776	7897	9659	7730	0	1490	0	90552	82822
S-100%-HD	94.00	47087	131215	0.9	93.26	42	61.92	32.08	2.2	25.6	2	40.2	109656	16489	16672	18728	283640	70422	13080000	13595606	13576878
S-50%-LD	54.14	4973	12085	0.17	0.03	86	54.14	0	0	0	3	37	66370	9380	10690	39168	0	1412	0	127020	87852
S-50%-HD	94.00	11070	27769	0.95	54.43	193	76.91	17.09	0.2	16.6	12.4	40.8	115712	18282	18047	94896	151340	46366	1540000	1984763	1889747
L-100%-LD	52.77	Failed to generate feasible solution because of out-of-memory status in 3 out of 5 instances																			
L-100%-HD	92.44	8820710	27057756	2770	100	4670	0	92.44	600	0	0	0	170666	306005	286482	2748198	37093800	31609188	4570260000	4644010336	4641262138
L-50%-LD	52.77	1266371	3834490	38	99.76	1202	20.67	32.10	53.4	205.8	12	68.8	665132	88596	104580	942372	12683960	10965900	519160000	544610541	543668169
L-50%-HD	92.44	2257226	6819602	75	99.96	3496	11.35	81.09	325.8	216	17.2	41	121383	187636	214015	2716758	32554960	27841382	2916980000	2981708585	2978991827

As Table 17 shows, the math model performs well on small instances with low demand (with 100% or 50% availability). However, performance drastically worsens when the problem size or demand level increases. Within the same instance size and demand level, the number of binary variables and constraints in the math model decreases when availability decreases from 100% to 50%. This happens due to having fewer room and timeslot options to check. The average CPLEX gap and density gap (gap between the best solution density and instance density) for all small cases is 40.46% and 12.29% respectively. The average time to solve the LP relaxation of the integer program is less than one second for all small instances. Even though we expect the math model to solve all small instances to optimality, the math model generates non-optimal feasible solutions with 0.6 unscheduled courses on average per small instance.

In large instances with 100% availability and low demand (L-100%-LD), CPLEX reached an out-of-memory status in 3 out of 5 instances. In the other three subcategories of large instances, the math model did not generate acceptable results. Indeed, the average CPLEX gap and density gap is 100% and 68.4% for those instances, respectively. Interestingly, for the HD instances, the time needed to solve the LP relaxation dramatically increases (from 75 seconds to 2770 seconds) when the room and timeslot availability increases from 50% to 100%. Overall, CPLEX does a terrible job solving the large problem instances, leaving more than half of the courses unscheduled on average.

The results in Table 17 verify the absolute need for a heuristic approach than can generate high quality solutions within a reasonable amount of time. This is especially the case for large instances with a large number of binary variables and constraints which are similar in size to real-life problems.

2.6.3. Heuristic method experimental setup, results, and discussion

A significant number of preliminary experiments were performed to determine the best way to generate PRAs in step 1 of the heuristic (Figure 3). Results overwhelmingly showed that heuristically generating PRAs (Section 2.5.3) was better than using the mini math model (Section 2.5.2) because less computation time was used. Regarding step 5 of the heuristic (Section 2.5.5), preliminary experiments revealed the following.

- Options 1 and 3 for deciding how many courses' f2f meetings to remove from the current schedule do not help to create better solutions, and they slow down the algorithm. We decided to ignore these options and just use option 2 which is to remove all f2f meetings for a certain number of courses. The number of courses is random integer from 1 to 7.
- Options 1-6 for deciding which courses' f2f meetings to remove from the schedule were all beneficial. Based on the results of preliminary runs, the likelihood of selecting option (1, 2, 3, 4, 5, 6) when generating a neighboring solution was (15%, 25%, 3%, 7%, 15%, 35%).
- Options 1-3 for deciding which unscheduled f2f meetings are added to the waitlist were all beneficial. Based on preliminary experiments, the likelihood of selecting option (1, 2, 3) was selected to be (40%, 30%, 30%).

A start temperature (*StartTemp*) of 30,000 and temperature factor (*TempFactor*) of 0.99999 were used in all experiments. In all experiments the parameter *PRA_limit* (see Figure 4) was set to infinity; there was no predefined limit on the number of PRAs generated for each course.

Tables 18-24 show the results for the heuristic algorithm on all 135 problem instances. Table 18 shows the settings and quality of the initial feasible solutions that were generated in the experiments on the small (S) problem instances. Table 19 shows the quality of the best feasible solutions that were found in the experiments on the small instances. Tables 20-21 are analogous to Tables 18-19 and show the results of the experiments on the medium-sized (M) instances. Tables 22-23 are analogous to Tables 18-19 and show the results of the experiments on the large (L) instances. Table 24 shows a detailed breakdown of the objective value on a "per course" basis for all instances. Each row in each table is the average result for five instances within the same subcategory. For each instance subcategory, the values of the parameters *MaxM*, *MaxN*, *MaxWS*, and *MaxRoom* (Figure 4) are modified to be able to generate at least one PRA for each course in all 5 instances within each subcategory.

Tables 18, 20, and 22 have nine columns with the same headers. The first two columns show the problem instance demand density and computation time limit. The next column refers to the settings used to generate the PRAs (Figure 4). The setting components are *MaxM*, *MaxN*, *MaxWS*, and *MaxRoom*. The next column shows the time elapsed to generate the initial feasible solution. The next three columns show the number of unscheduled courses, density, and number of wasted seats in the initial feasible solution, respectively. The density equals the total number of seat time slots used divided by the total number of seat time slots available. The second to the last column shows the objective value of the initial feasible solution. Finally, the last column shows the total number of PRAs generated for all courses combined (in the procedure shown in Figure 4).

Tables 19, 21, and 23 have 20 columns with the same headers. The first three columns show the total number of iterations, iterations in which the neighboring solution is accepted, and

iterations in which a neighboring solution with a better objective value is accepted, respectively. The next three columns show the number of wasted seats, best solution density, and density gap (i.e., difference between the demand density of the problem instance and the best solution density), respectively. The next four columns show the average number of unscheduled, partially scheduled, fully scheduled, and perfectly scheduled courses, respectively. The next seven columns show the values of the seven parts of the objective function in the best solution identified by the heuristic. The next two columns show the overall objective value (which is the summation of the values in the previous seven columns) and objective value improvement compared to the initial feasible solution. Finally, the last column shows the objective value without objective 4 which is related to the wasted seats. Minimizing the number of wasted seats is an intermediate priority that is generally less important than other objectives. The last column shows the overall objective value without considering the number of wasted seats in each course's room assignment.

Table 24 shows how the objective value breaks down on a "per course" basis for all problem instances. Each column in the table refers to a different objective function component (objectives 1-7). The values in the table are computed by dividing the objective values in columns Obj1, Obj2, Obj3, Obj4, Obj5, Obj6, and Obj7 in Tables 19, 21, and 23 by their weights ($\alpha 1 - \alpha 7$), then dividing the result by the number of courses, and lastly dividing the result by the average number of students in a course. For example, the values in row "M-100%-LD" are computed by first dividing the corresponding values in Table 21 by the appropriate weight ($\alpha 1 - \alpha 7$), then dividing the result by 140 (the number of courses), and finally dividing the result by 60 (the average number of students per course which is the expectation of the DU(10,110) distribution). The values in this row of the table are interpreted as follows.

- Objective 1 is 3.56 which means each course occupies about 3.56 rooms on average.
- Objective 2 is 303.63 which means that the maximum distance between rooms assigned to each course is about 303.63 meters on average (in a campus measuring 800m x 800m.)
- Objective 3 is 474.21 which means that the maximum distance between the office of the professor who teaches a course and any room assigned to it is about 474.21 meters on average.
- Objective 4 is 0.24 which means that about 0.24 seats are wasted in each course's assignment on average.
- Objective 5 is 0 which means that all desired f2f meetings have been scheduled for all courses in all instances in this problem subcategory.
- Objective 6 is 0.07 which means that the deviation between when f2f sessions are desired and when they are scheduled is about 0.07 on average. In other words, the average course has one f2f meeting that is scheduled 0.07 weeks away from when it is desired, and all other f2f meetings are scheduled in the exact weeks when desired.
- Objective 7 is 0 which means that there are no unscheduled courses in any instance in this problem subcategory.

Tables 18-24 show that the heuristic performs well across all subcategories of problem instances. The values in column "UC" in Tables 19, 21, and 23 shows that, in all instances, the heuristic is able to schedule one or more socially distanced, in-person meetings each semester (when all students in a course gather in multiple rooms simultaneously) for each course. As expected, within the same instance size and availability, when the demand level increases, the gap between the best solution density and the instance density increases. For the small, medium, and

large instances–across all availability levels and demand levels–the average gap between the best solution density and the instance density is 4.04%, 6.99%, and 5.98%, respectively. The 5.98% gap for the large instances indicates that the heuristic approach works very well for large instances (which are closest to real-life problems) if seat-timeslot utilization is the main goal. The average seat-timeslot utilization across all HD instances (shown in column "Best Soln Density" in Tables 19, 21, and 23) is 78.1%. Across all instance sizes, on average the objective value of the best solution is 99.31% better than that of the initial feasible solution.

Table 24 shows that, in all problem subcategories, courses are scheduled in two to four rooms on average (Obj1). The values of M_c and N_c (Obj2 and Obj3) vary due to different settings used in each scenario but are generally low considering that the university campus is assumed to be an 800x800 square (with diagonal length $1131 = 800\sqrt{2}$) and classrooms are randomly scattered within the campus. As expected, the number of wasted seats (Obj4), number of meetings lacking (Obj5), and timing deviation (Obj6) generally increase as the demand increases (from LD to MD to HD). The most important information in this table which verifies the power of the heuristic is shown in the third-last and last columns. The last column shows that, in all instances, there are no unscheduled courses. In other words, there are no courses without any f2f meetings scheduled. The third-last column shows that, in 24 of the 27 problem subcategories, the average number of f2f meetings lacking per course is less than 1. A comparison of columns "Best OV" and "Best OV w/o Obj4" in Tables 17, 19, and 23 shows that the heuristic clearly outperforms CPLEX across all problem subcategories.

Table 18. Heuristic method results for small instances: settings and initial feasible solution

	Instance density (%)	Allowed time (sec)	Heuristic settings [*MaxM, MaxN, MaxWS, MaxRoom*]	IFS time elapsed (sec)	IFS # Unscheduled courses	IFS density (%)	IFS # Wasted seats	IFS OV	Total # PRA Generated
S-100%-LD	54.14	300	[700, 9999, 10, 6]	0	5.8	34.20	171	27,227,560	2779
S-100%-MD	71.08	300	[700, 9999, 10, 6]	0	14.6	35.87	215	62,468,020	2779
S-100%-HD	94.00	300	[400, 9999, 10, 5]	0	24.4	34.66	156	112,312,140	4112
S-75%-LD	54.14	300	[900, 9999, 15, 6]	0	3.2	41.93	244	17,615,560	1333.4
S-75%-MD	71.08	300	[900, 9999, 15, 6]	0	6.4	44.70	364	34,352,100	1627
S-75%-HD	94.00	300	[900, 9999, 20, 6]	0	20	45.23	261	92,605,939	2102
S-50%-LD	54.14	300	[1000, 9999, 15, 6]	0	1	46.92	308	8,029,595	294
S-50%-MD	71.08	300	[900, 9999, 20, 6]	0	5	50.24	533	26,872,443	363
S-50%-HD	54.14	300	[1000, 9999, 20, 6]	0	11	53.57	453	64,269,011	724

Table 19. Heuristic method results for small instances: best solution obtained

	# Iter	# Acp	# Better	# Wasted seats	Best soln density (%)	Density Gap (%)	#UC	#PSC	#FSC	#PeSC	Obj1	Obj2	Obj3	Obj4	Obj5	Obj6	Obj7	Best OV	OV improve (%)	Best OV w/o Obj4
S-100%-LD	801555	161965	78268	23	54.14	0	0	0	0.8	39.2	63192	7609	9665	1866	0	510	0	82842	99.62	80976
S-100%-MD	521276	65709	31918	135	70.28	0.80	0	1.2	8.6	38.6	73108	11297	13087	18192	35040	5670	0	168394	99.73	138202
S-100%-HD	612313	32374	15169	252	84.72	9.28	0	10.4	24.4	35.2	114990	16178	17486	39226	83280	22360	0	293532	99.73	254294
S-75%-LD	805394	156346	73069	50	54.14	0	0	0	0.6	39.4	65776	8437	10,059	7256	0	513	0	87955	99.39	84682
S-75%-MD	556176	70198	31677	234	71.08	0	0	0	12.2	42.8	90608	13602	14298	42296	0	10274	0	171077	99.44	128781
S-75%-HD	584931	52248	20245	298	83.5	10.5	0	12	26	32	117602	18659	18242	57008	92680	21914	0	324105	99.65	269096
S-50%-LD	672253	213453	73687	218	54.14	0	0	0	2.6	37.4	66402	9430	10725	39892	0	1478	0	125598	84.86	88035
S-50%-MD	626090	188602	42926	507	68.98	2.1	0	2	15	37	92452	14607	14893	97682	17360	12180	0	249174	99.05	151492
S-50%-HD	601765	116729	28398	487	80.3	13.7	0	12	22	36	117242	18326	18300	102802	117800	16624	0	391094	99.34	288292

Table 20. Heuristic method results for medium-sized instances: settings and initial feasible solution

	Instance density (%)	Allowed time (sec)	Heuristic settings [*MaxM, MaxN, MaxWS, MaxRoom*]	IFS time elapsed (sec)	IFS #Unscheduled courses	IFS Density (%)	IFS #Wasted seats	IFS OV	Total # PRA Generated
M-100%-LD	53.58	1200	[400, 9999, 5, 5]	3.6	23.2	24.17	456	166,798,280	39336
M-100%-MD	74.97	1200	[400, 9999, 5, 5]	4	63	20.98	506	422,917,400	41753
M-100%-HD	90.42	1200	[400, 9999, 5, 5]	5.8	90.2	25.10	652	598,762,800	49634
M-75%-LD	53.58	1200	[500, 9999, 5, 5]	1	16	30.46	593	112,315,400	35061
M-75%-MD	74.97	1200	[500, 9999, 5, 5]	2	44.2	28.79	723	303,847,600	41390
M-75%-HD	90.42	1200	[500, 9999, 5, 5]	2	64	31.60	865	448,614,400	46968
M-50%-LD	53.58	1200	[800, 9999, 5, 6]	1	9	36.04	699	68,933,300	75982
M-50%-MD	74.97	1200	[800, 9999, 5, 6]	1	25	38.42	931	186,538,200	120880
M-50%-HD	90.42	1200	[800, 9999, 5, 6]	1	45	38.28	967	327,108,600	120145

Table 21. Heuristic method results for medium-sized instances: best solution obtained

	# Iter	# Acp	# Better	# Wasted seats	Best soln density (%)	Density Gap (%)	#UC	#PSC	#FSC	#PeSC	Obj1	Obj2	Obj3	Obj 4	Obj5	Obj6	Obj7	Best OV	OV improve (%)	Best OV w/o Obj4
M-100%-LD	583283	98053	47596	178	53.58	0	0	0	7.2	132.8	298670	25505	39833	19878	0	6012	0	389,898	99.73	370,021
M-100%-MD	619978	25309	12372	794	71.62	3.33	0	19.8	79	91.2	456588	34097	57447	108646	212580	205176	0	1,054,535	99.75	1,027,689
M-100%-HD	669318	15351	9238	1196	77.98	12.44	0	65	75.6	99.4	539976	42357	72348	181246	788500	259564	0	1,883,990	99.67	1,702,745
M-75%-LD	571465	85238	41133	204	53.58	0	0	0	23.6	116.4	301952	30293	42251	20280	0	21564	0	416,340	99.47	396,060
M-75%-MD	643528	25145	18000	848	69.97	4.99	0	29.4	76.4	84.2	438464	42232	59364	120190	318040	222320	0	1,200,610	99.60	1,080,421
M-75%-HD	679675	17500	7293	1131	76.37	14.04	0	71.4	71.4	97.2	541116	52815	74827	177810	892400	269142	0	2,008,108	99.54	1,830,300
M-50%-LD	448555	58813	28372	328	53.45	0.13	0	0.4	48	91.6	315368	44805	47645	39236	8140	62522	0	498,951	98.91	478,480
M-50%-MD	688482	23135	11369	996	66.85	8.12	0	41	64.8	82.2	453156	66150	54567	141182	519880	234668	0	1,481,603	99.20	1,328,421
M-50%-HD	580797	18052	8947	1280	70.58	19.84	0	79.4	74.4	86.2	557606	82341	84601	2007981261240		239238	0	2,425,818	99.24	2,225,026

Table 22. Heuristic method results for large instances: settings and initial feasible solution

	Instance density (%)	Allowed time (sec)	Heuristic settings [MaxM, MaxN, MaxWS, MaxRoom]	IFS time elapsed (sec)	IFS #Unscheduled courses	IFS density (%)	IFS #Wasted seats	IFS OV	Total # PRA Generated
L-100%-LD	52.77	3600	[200, 9999, 5, 4]	23.2	40.6	20.24	2740	373,467,800	26519
L-100%-MD	69.39	3600	[200, 9999, 5, 4]	29.6	57.4	25.74	3644	483,591,200	31967
L-100%-HD	92.44	3600	[200, 9999, 5, 4]	37.8	145.2	22.62	2897	1,338,280,200	35560
L-75%-LD	52.77	3600	[200, 9999, 15, 4]	7	40.8	21.75	6064	352,080,080	24878
L-75%-MD	69.39	3600	[200, 9999, 15, 4]	12.2	69	25.57	7332	525,608,600	29821
L-75%-HD	92.44	3600	[200, 9999, 15, 4]	14.2	136	25.54	8242	1,137,623,800	33413
L-50%-LD	52.77	3600	[500, 9999, 15, 4]	4.4	30.8	22.27	6279	286,841,400	247128
L-50%-MD	69.39	3600	[500, 9999, 15, 4]	5	55.8	24.40	6625	482,577,600	305907
L-50%-HD	92.44	3600	[500, 9999, 15, 4]	6.2	100.8	28.27	8487	880,899,000	363502

Table 23. Heuristic method results for large instances: best solution obtained

	# Iter	# Acp	# Better	# Wasted seats	Best soln density (%)	Density Gap (%)	#UC	#PSC	#FSC	#PeSC	Obj1	Obj2	Obj3	Obj4	Obj5	Obj6	Obj7	Best OV	OV improve (%)	Best OV w/o Obj4
L-100%-LD	346051	70576	33652	2944	52.77	0	0	0	79.4	260.6	624,776	37,056	112,137	222,946	0	94,206	0	1,242,602	99.60	868,175
L-100%-MD	266169	26862	13183	7041	69.39	0	0	0	333.6	136.4	1,043,274	50,958	162,091	611,366	0	1,123,814	0	2,991,504	99.33	2,380,137
L-100%-HD	361659	7551	4507	8898	80.36	12.08	0	187.6	286.4	126	1,372,894	66,307	208,137	903,750	4,777,720	2,717,692	0	10,046,492	99.10	9,142,749
L-75%-LD	316298	47758	23282	5669	52.77	0	0	0	143.2	196.8	778,684	38,146	113,422	429,210	0	205,758	0	1,565,222	99.31	1,136,010
L-75%-MD	353957	17195	8467	13911	68.89	0.50	0	15.6	344	110.4	1,049,352	50,743	161,550	1,168,472	193,180	1,553,148	0	4,176,444	99.17	3,007,973
L-75%-HD	409893	12373	5182	16132	77.30	15.15	0	215.2	282.6	102.2	1,380,914	66,555	211,486	1,586,878	6,020,060	2,777,112	0	12,043,040	98.84	10,456,126
L-50%-LD	283006	66823	13100	8158	52.73	0.04	0	0.4	277.8	61.8	816,846	93,646	133,027	660,914	13,160	987,986	0	2,705,576	99.01	2,044,665
L-50%-MD	464032	12451	6278	13608	64.09	5.30	0	99.6	274.2	96.2	1,078,406	124,002	183,581	1,248,738	2,118,900	2,036,384	0	6,790,010	98.48	5,541,273
L-50%-HD	351107	10244	5244	17105	71.66	20.79	0	268.4	272.2	59.4	1,420,294	165,715	239,038	1,697,154	8,271,600	2,941,906	0	14,735,700	98.28	13,038,553

Table 24. Detailed objective value breakdown

	Obj1: Num rooms	Obj2: M_c (meters)	Obj3: N_c (meters)	Obj4: Wasted seats	Obj5: Meetings lacking	Obj6: Timing deviation	Obj7: Unscheduled courses
S-100%-LD	3.51	422.72	536.94	0.10	0.00	0.03	0.00
S-100%-MD	2.95	456.45	528.76	0.74	0.14	0.23	0.00
S-100%-HD	3.65	513.58	555.11	1.25	0.26	0.71	0.00
S-75%-LD	3.62	463.00	548.04	0.38	0.00	0.03	0.00
S-75%-MD	3.66	549.56	577.68	1.71	0.00	0.42	0.00
S-75%-HD	3.73	592.34	579.10	1.81	0.29	0.70	0.00
S-50%-LD	3.67	527.05	589.18	2.39	0.00	0.06	0.00
S-50%-MD	3.74	590.19	601.74	3.95	0.07	0.49	0.00
S-50%-HD	3.72	581.79	580.94	3.26	0.37	0.53	0.00
M-100%-LD	3.56	303.63	474.21	0.24	0.00	0.07	0.00
M-100%-MD	4.01	299.10	503.92	0.95	0.19	1.80	0.00
M-100%-HD	3.75	294.15	502.42	1.26	0.55	1.80	0.00
M-75%-LD	3.59	360.63	502.99	0.24	0.00	0.26	0.00
M-75%-MD	3.85	370.46	520.74	1.05	0.28	1.95	0.00
M-75%-HD	3.76	366.77	519.63	1.23	0.62	1.87	0.00
M-50%-LD	3.75	533.39	567.21	0.47	0.01	0.74	0.00
M-50%-MD	3.98	580.27	478.66	1.24	0.46	2.06	0.00
M-50%-HD	3.87	571.81	587.51	1.39	0.88	1.66	0.00
L-100%-LD	2.45	145.32	439.75	0.87	0.00	0.37	0.00
L-100%-MD	2.96	144.56	459.83	1.73	0.00	3.19	0.00
L-100%-HD	3.05	147.35	462.53	2.01	1.06	6.04	0.00
L-75%-LD	3.05	149.59	444.79	1.68	0.00	0.81	0.00
L-75%-MD	2.98	143.95	458.30	3.31	0.05	4.41	0.00
L-75%-HD	3.07	147.90	469.97	3.53	1.34	6.17	0.00
L-50%-LD	3.20	367.24	521.67	2.59	0.01	3.87	0.00
L-50%-MD	3.06	351.78	520.80	3.54	0.60	5.78	0.00
L-50%-HD	3.16	368.26	531.20	3.77	1.84	6.54	0.00

2.7. Conclusion

In response to the COVID-19 pandemic, academic institutions offered courses in three formats: in-person, online, and hybrid. In each format, no more than one classroom could be assigned to a course. However, we believe that a more sophisticated course format is needed. In this chapter, we propose a method for university course scheduling during a pandemic in which multiple classrooms may be assigned to each course. The goal is for all courses to have limited number of socially distanced, face-to-face (f2f) meetings each semester when all students in the course simultaneously spread out across multiple classrooms. We develop a mathematical model that can be used to schedule such face-to-face meetings for all courses. The model considers COVID-19-reduced classroom capacities, the number of students enrolled in each course, distances between rooms, and other practical constraints. Seven optimization criteria are considered including the timing of f2f meetings, distances of classrooms from instructor offices, and the distances between classrooms assigned to the same course. A heuristic algorithm is also developed. The heuristic method significantly outperforms a direct approach in which the math model is solved using standard integer programming software. The heuristic obtains excellent results on life-sized instances (with up to 600 courses and 60 classrooms), allowing all courses—even the largest—to have one or more socially distanced, in-person meetings each semester in which all students in the course simultaneously gather in multiple rooms.

Future work might consider applying the methods proposed in this chapter to real-life settings to help real universities prepare for future pandemics. More experiments that consider additional scenarios could also be conducted. Design-of-experiment methods (such as Taguchi methods) could also be applied to find the best settings for the heuristic.

Chapter 2 References

Abramson, D. (1991). Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, *37*(1), 98–113.

Al-Betar, M. A., & Khader, A. T. (2012). A harmony search algorithm for university course timetabling. *Annals of Operations Research*, *194*(1), 3–31.

Atef Yekta, H., & Day, R. (2020). Optimization-based mechanisms for the course allocation Problem. *INFORMS Journal on Computing*, *32*(3), 641–660.

Ayob, M., & Jaradat, G. (2009). Hybrid ant colony systems for course timetabling problems. *2009 2nd Conference on Data Mining and Optimization*, 120–126.

Barnhart, C., Bertsimas, D., Delarue, A., & Yan, J. (2021). Course scheduling under sudden scarcity: Applications to pandemic planning. *Manufacturing & Service Operations Management*, *24*(2), 727–745.

Bettinelli, A., Cacchiani, V., Roberti, R., & Toth, P. (2015). An overview of curriculum-based course timetabling. *TOP*, *23*(2), 313–349.

Budish, E., Cachon, G. P., Kessler, J. B., & Othman, A. (2017). Course Match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation. *Operations Research*, *65*(2), 314–336.

Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, *9*(5), 403–432.

da Fonseca, G. H. G., Santos, H. G., Toffolo, T. Â. M., Brito, S. S., & Souza, M. J. F. (2016). GOAL solver: a hybrid local search-based solver for high school timetabling. *Annals of Operations Research*, *239*(1), 77–97.

da Fonseca, G. H. G., Santos, H. G., Carrano, E. G., & Stidsen, T. J. R. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research*, *262*(1), 28–39.

de Oliveira, P. M., Mesquita, L. C. C., Gkantonas, S., Giusti, A., & Mastorakos, E. (2021). Evolution of spray and aerosol from respiratory releases: theoretical estimates for insight on viral transmission. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *477*(2245), 20200584.

Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, *20*(5), 533–534.

Feizi-Derakhshi, M.-R., Babaei, H., & Heidarzadeh, J. (2012). A survey of approaches for university course timetabling problem. *Proceedings of 8th International Symposium on Intelligent and Manufacturing Systems, Sakarya University Department of Industrial Engineering, Adrasan, Antalya, Turkey*, 307–321.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, *111*(23), 8410–8415.

Goh, S. L., Kendall, G., & Sabar, N. R. (2017). Improved local search approaches to solve the post enrolment course timetabling problem. *European Journal of Operational Research*, *261*(1), 17–29.

Gonzalez, G., Richards, C., & Newman, A. (2018). Optimal course scheduling for United States Air Force academy cadets. *Interfaces*, *48*(3), 217–234.

Imran Hossain, Sk., Akhand, M. A. H., Shuvo, M. I. R., Siddique, N., & Adeli, H. (2019). Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert Systems with Applications*, *127*, 9–24.

Jamal, A. (2020). Global optimization using local search approach for course scheduling problem. In *Scheduling Problems - New Applications and Trends*. IntechOpen.

Jat, S. N., & Yang, S. (2011). A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. *Journal of Scheduling*, *14*(6), 617–637.

Johnson, C., & Wilson, R. L. (2022). Practice summary: A multiobjective assignment model for optimal socially distanced classrooms for the Spears School of Business at Oklahoma State University. *INFORMS Journal on Applied Analytics*, *52*(3), 295–300.

Lewis, R., & Paechter, B. (2005). Application of the grouping genetic algorithm to university course timetabling (pp. 144–153).

Lü, Z., & Hao, J.-K. (2010). Adaptive tabu search for course timetabling. *European Journal of Operational Research*, *200*(1), 235–244.

Mencía, R., Sierra, M. R., Mencía, C., & Varela, R. (2016). Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators. *Integrated Computer-Aided Engineering*, *23*(3), 269–285.

Méndez-Díaz, I., Zabala, P., & Miranda-Bront, J. J. (2016). An ILP based heuristic for a generalization of the post-enrollment course timetabling problem. *Computers & Operations Research*, *76*, 195–207.

Naji Azimi, Z. (2005). Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, *163*(2), 705–733.

Navabi-Shirazi, M., El Tonbari, M., Boland, N., Nazzal, D., & Steimle, L. N. (2022). Multicriteria course mode selection and classroom assignment under sudden space scarcity. *Manufacturing & Service Operations Management.* https://doi.org/10.1287/msom.2022.1131

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2012). A honey-bee mating optimization algorithm for educational timetabling problems. *European Journal of Operational Research*, *216*(3), 533–543.

Shiau, D.-F. (2011). A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences. *Expert Systems with Applications*, *38*(1), 235–248.

Tang, Y., Liu, R., Wang, F., Sun, Q., & Kandil, A. A. (2018). Scheduling optimization of linear schedule with constraint programming. *Computer-Aided Civil and Infrastructure Engineering*, *33*(2), 124–151.