8-2007

Planning, Scheduling, and Timetabling in a University Setting

Christine Kraft
Clemson University, cstemm@clemson.edu

PLANNING, SCHEDULING, AND TIMETABLING IN A UNIVERSITY SETTING

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematical Sciences

---

by
Christine R. Kraft
August 2007

Accepted by:
Dr. James P. Jarvis, Committee Chair
Dr. Warren P. Adams
Dr. Mark E. Cawood
Dr. Herman F. Senter

ABSTRACT

Methods and procedures for modeling university student populations, predicting course enrollment, allocating course seats, and timetabling final examinations are studied and proposed. The university enrollment model presented uses a multi-dimensional state space based on student demographics and the Markov property, rather than longitudinal data to model student movement. The procedure for creating adaptive course prediction models uses student characteristics to identify groups of undergraduates whose specific course enrollment rates are significantly different than the rest of the university population. Historical enrollment rates and current semester information complete the model for predicting enrollment for the coming semester. The course prediction model aids in the system for reserving course seats for new students during summer registration sessions. The seat release model addresses how to estimate seat need each session, how to release seats among multiple course sections, and how to predict seat shortages and surpluses. Finally, procedures for creating reusable university final examination timetables are developed and compared. Course times, rather than individual courses, are used as the assignment elements because the demand for course times remains relatively constant despite changes in course schedules. Our heuristic procedures split the problem into two phases: a clustering phase—to minimize conflicts—and a sequencing phase—to distribute exams throughout finals week while minimizing the occurrence of consecutive exams. Results for all methods are compared using enrollment data from Clemson University.

ACKNOWLEDGMENTS

I give so many thanks to my adviser, Dr. James P. Jarvis, for the motivation of this study and all the help along the way, both academic and personal. My gratitude goes to my committee members: to Dr. Mark E. Cawood, for all his encouragements to me during the process; to Dr. Warren P. Adams, for turning me on to math programming with his excellent courses; and to Dr. Herman F. Senter, for bringing me back to math after a sojourn in economics. I owe much to all my professors at Clemson for teaching me more than I ever thought I would know, and for helping me begin to grasp how much more there is to learn.

I acknowledge and thank Dr. Robert Walsh for allowing me and encouraging me to advance in mathematics at a high school without an advanced program. He has been a wonderful mentor and example. I gratefully acknowledge my undergraduate professors Dr. Crump Baker, Dr. Chris Lang, and Dr. James Woeppel for allowing me to take more mathematics than their program offered and for all of their help along the way.

Thanks are not enough to give to my parents, Steve and Dorothy, for instilling in me the value of an education, for loving me through all my choices, and for giving me more than I deserve. Mom has been my rock and Dad has been my hero. I give warm acknowledgments to my mutti, Jane, who listens to me and loves me when I need her, and who did a fabulous job raising the wonderful person I call my husband. I owe many thanks to my sweet Ringo for all the joy he has brought to our lives.

To my husband, Nicholas, for believing in me, supporting my decisions, and loving me always, I will be indebted forever. I love you.

TABLE OF CONTENTS

Table of Contents (Continued)

Chapter Page

LIST OF TABLES

List of Tables (Continued)

Table Page

LIST OF FIGURES

Figure Page

List of Figures (Continued)

Figure Page

LIST OF ALGORITHMS

Algorithm Page

Chapter 1

Introduction

Planning, scheduling, and timetabling are tasks faced by all colleges and universities. Modeling university populations is necessary to allocate university resources, to examine the effects of policy changes, and to plan for future enrollments. Predicting course enrollment is imperative for accommodating student demand for courses and for planning instructor assignments to classes prior to registration. An objective system to allocate course seats to new and continuing students is important in order to create equitable enrollment choices for each student. Students and faculty desire final examination timetables that have few conflicts and few consecutive exams. Universities benefit from comprehensive planning models to address these issues.

The literature on these topics in planning, scheduling, and timetabling for universities is diverse, delving deeply into some areas and wholly ignoring others. Hopkins and Massy [1981] present a survey that addresses modeling university populations, mentions predicting course enrollment, and focuses the remainder of the book on other topics. Balachandran and Gerwin [1973] study the problem of predicting course enrollment, but give little evidence of the applicability of their models. We have found no literature on the topic of course seat allocation systems. Several authors, including Burke, Carter, Petrovic, and Schaerf, address the problem of timetabling final exams [Burke and Petrovic 2002; Carter 1986; Schaerf 1999].

In this study, we develop and present procedures for modeling university populations, predicting course enrollments, allocating course seats, and timetabling final examinations. We dedicate a chapter to each problem where we review the related work, develop the model building procedures, and present results. In the next section, we briefly describe each problem.

1.1 Research Problems

Problem 1: Modeling University Student Populations

Modeling the student population of a university involves recording the number and characteristics of students in order to observe and predict changes in the population. We present our procedure for building such models and apply these to a specific scenario. The goals of this part of our study are to design a procedure for building university population models in general, and specifically to build a model that can be used for both long- and short-term projection, while simultaneously being used to model intra-university movement. We describe our procedure in detail, specifying when alternate decisions can be made to form different university population models.

Problem 2: Predicting Course Enrollments

Predicting course enrollment is an important part of building course schedules. We describe our procedure for creating adaptive course prediction models in this study. Our goal is to create models that make accurate predictions early in the course scheduling process, and yet can be updated as information becomes available. We use student characteristics to identify groups of undergraduates whose course enrollment rates are significantly different than the rest of the university population. The model we develop utilizes historical enrollment rates and current semester information to predict course enrollment for the coming semester.

Problem 3: Allocating and Releasing Course Seats to Student Groups

Allocating course seats to specific student groups requires predicting the size of student groups, estimating the number of course seats needed by each group, and holding and releasing seats such that an equitable number of seats are available to the specific groups. We develop a system that addresses these problems. The course prediction model we develop aids in the system for allocating course seats to new students during summer registration sessions. Our goals are to estimate seat demand accurately, and to develop a system that hedges our estimates without undermining the objective of giving students similar enroll-

ment choices regardless of when they register—during early registration, any summer orientation, or make-up registration. We describe how to estimate seat demand each session, how to release seats among multiple course sections, and how to predict seat shortages and surpluses.

Problem 4: Timetabling Final Examinations

We develop a two-phase heuristic procedure to build reusable final examination timetables. We map course times, rather than individual courses, to exam periods because the demand for course times remains relatively constant despite changes in course schedules from year to year. Our approach splits the problem into two phases: a clustering phase—to minimize conflicts—and a sequencing phase—to distribute exams throughout the exam period while minimizing the occurrences of consecutive exams. Results for all methods are compared using enrollment data from Clemson University (CU).

1.2 Dissertation Outline

Because the topics covered in this study are diverse, we present the related work as a subsection of each chapter. In Chapter 2 we develop a procedure for building university population models using Markov chains. We follow the procedural description with a specific model for projecting populations at CU. In Chapter 3 we present a procedure for building course enrollment models and specific models for predicting course enrollment in four mathematics courses at CU. In Chapter 4 we describe a system to allocate course seats to specific groups of students. We give results using this system for allocating course seats for four courses at CU over multiple summer registration sessions for new students. In Chapter 5 we develop heuristics for creating reusable examination timetables. We give examples of proposed timetables for CU. In Chapter 6 we summarize our results.

Chapter 2

University Population Modeling

Predicting student populations is an important part of planning and forecasting for universities. Accurate, detailed student population models aid in revenue management, resource allocation, and course enrollment prediction. These models also allow investigation of hypothetical changes under alternate assumptions of admissions, graduation, and retention rates. This affords the opportunity to consider the consequences of proposed policy changes prior to implementation. The ability to identify groups of students, examine changes in their sizes, and predict how they will change in the future is central to university enrollment planning. Detailed student population models also aid in examining changes that have already occurred in student populations.

Creating university student population models is challenging due to the variability inherent in student groups. For example, the number of students for whom it is their first-time in college—we will refer to these students as *first-time* students—changes from year to year, as do the characteristics of the students within this group. The trade-off between accuracy and detail becomes evident as we identify the level of detail needed to model student populations. That is, we may be able to model an aggregate population, such as first-time students, very well, but we may have to sacrifice the ability to forecast a subpopulation, such as first-time students who are freshman biology majors, because more specificity in the model typically results in more variability in the prediction. We discuss this element of modeling university student populations in Section 2.3.

The goal of building university population models is to accurately predict student populations. Our criteria for accurate models is to have error rates less than those of the model's exogenous variables. The exogenous variables include the number of first-time and transfer students enrolling in the university. These variables are governed by university policy decisions and can change from year to year. Table 2.1 shows the first-time and transfer populations targeted by the administration at Clemson University (CU) along with actual

	First-time			Transfer		
	Target	Actual	Percent Difference	Target	Actual	Percent Difference
2001	2,800	2,555	8.8%	500	561	10.9%
2002	2,500	2,480	0.8%	600	694	13.5%
2003	2,800	2,758	1.5%	800	787	1.7%
2004	2,800	3,020	7.9%	800	685	16.8%
2005	2,800	2,904	3.7%	800	688	16.3%
2006	2,800	2,813	0.5%	800	782	2.3%

Table 2.1: Fall target and actual new student populations at Clemson University.

populations. The error rates for the first-time population are between .5% and 8.8% with an average of 3.9%; the error rates for the transfer population are between 1.7% and 16.8% with an average of 10.3%. Our goal is to forecast populations associated with first-time students such that our error rates are less than 3.9%. Similarly, our criteria for forecasting populations associated with transfer students is that our estimates should have error rates less than 10.3%. In general our goal is to forecast aggregate populations such that error rates are less than 5.3%, which is the weighted sum of the average first-time and average transfer error rates.

In this chapter we describe a procedure for creating a university population model, and use that procedure to build a specific model. This model forecasts student populations from the fall of one year to the fall of the next. This timing is chosen because large inputs to the university's populations primarily occur in the fall when new students and transfer students enroll. However, our model still incorporates students who graduate or enroll in the spring and summer. Furthermore, our model can be adapted easily to forecast specifically for spring enrollments.

The remainder of this chapter is structured as follows. We review the literature on university population modeling in Section 2.1. We discuss the data used in building university population models at CU in Section 2.2. In Section 2.3 we describe the procedure used to create and use our population model and we present results for predicting university populations using our model at CU in Section 2.4.

2.1 Related Work

University enrollment modeling has become increasingly important as a means to facilitate budget projections and allocate resources. The goal of enrollment modeling may affect the type of model used [Hopkins and Massy 1981]. For example, if the goal is to predict revenues from fees and tuition, the model will use aggregations with categories of students based on their pay rate. If the goal is to help plan courses and curriculum, the model must use more detail to separate students into groups based on the courses in which they are likely to enroll. The literature on enrollment modeling is segmented by the goals for which the model is built.

Some of the earliest literature in university enrollment modeling specifically targets forecasting the entire university population. Techniques for achieving this goal utilize measured populations exogenous to the university as input data for the model [Schmid and Shanley 1952]. The *ratio method* uses the state population as an approximation for the base from which its students are derived. The method requires estimates for the ratio of students enrolled in the university to the state population. These estimates are derived from trend analysis on historical ratios [Schmid and Shanley 1952]. The *cohort-survival method* also uses data from outside the university as input. It applies survival analysis on the number of students in various grade levels in schools in the state to estimate the number of students who will eventually enroll at the university [Schmid and Shanley 1952]. Because of the outside influences on state population and the prevalence of university students from other states and countries, both of these models have been overhauled to reflect new ideas in enrollment modeling, as seen by the similarly-named techniques presented by [Hopkins and Massy 1981].

Hopkins and Massy [1981] focus on modeling for revenue prediction and thus make predictions on an aggregate level. They categorize the flow models used in university enrollment planning as (1) *grade progression ratio (GPR) methods*; (2) *Markov chain models*; and (3) *cohort flow models* [Hopkins and Massy 1981].

The *GPR method* uses ratios of the number of students in one class to the number in the next lower class to predict future enrollments. Figure 2.1 shows four classes in a

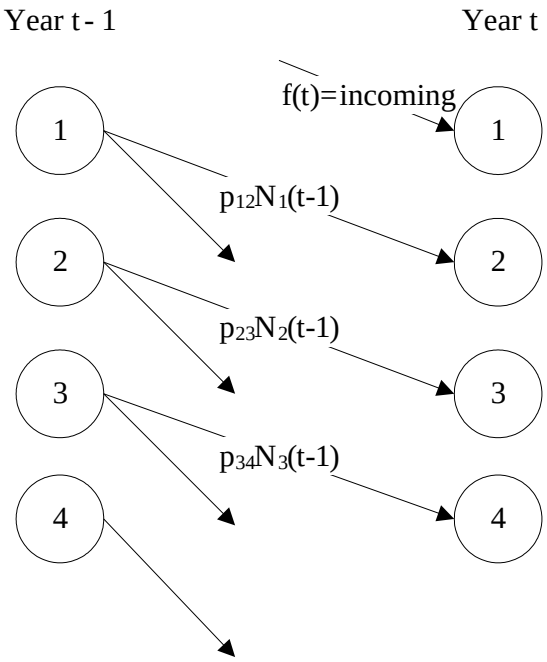


Figure 2.1: Grade progression ratio model. *This diagram shows the progression of students from one class to the next class or out of the university.*

university: (1) freshman, (2) sophomore, (3) junior, and (4) senior. Following the notation of Hopkins and Massy [1981], Figure 2.1 shows a schema of the GPR method, where $N_j(t)$ is the number of students in class j at time t and $p_{j-1,j} = N_j(t)/N_{j-1}(t-1)$. The ratios are estimated using historical data. The GPR method is straightforward and requires little data—only total counts of students in each class during each year. However, it does not account for students who stay in the same class for more than one year or for students who leave and then return. This lack of detail can cause large errors in prediction.

Before describing the Markov chain models presented by Hopkins and Massy [1981], we briefly review the definition and properties of Markov chains. First, we note that a *stochastic process* is defined as a model of a system that evolves randomly according to a probability distribution [Kulkarni 1995]. We denote a stochastic process by $\{X_n,\ n \geq 0\}$, where $X_n \in S$ is the state of the system at time n, and S, called the *state space*, is the set of possible values taken by X_n. The *Markov property*, simply stated, says that the future of a stochastic process depends only on the present. More formally, if the state space S is

countable, the Markov property is given by

$$P(X_{n+1} = x|X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = P(X_{n+1} = x|X_n = x_n), \tag{2.1}$$

where x_i is the realization of the random variable X_i.

A *Markov chain* is a stochastic process that possesses the Markov property. A *discrete time Markov chain* (DTMC) is a Markov chain with discrete, but not necessarily equally spaced, time intervals [Kulkarni 1995]. Using the notation of [Kulkarni 1995], we denote the conditional probability of moving to state j at time $n + 1$, given that $X_n = i$, for each $i, j \in S$, as follows:

$$p_{ij}(n) = P(X_{n+1} = j|X_n = i). \tag{2.2}$$

If $p_{ij}(n) = p_{ij}(k)$ for all $k \geq 0$ and for all $i, j \in S$, the DTMC is *time-homogeneous* and we denote $p_{ij}(n) = p_{ij}$. The Markov chain models we will be examining will be time-homogeneous DTMCs with finite state spaces. We define the one-step transition probability matrix, P, for a DTMC with state space $\{1, 2, \ldots, m\}$ as

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m-1} & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m-1} & p_{2m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm-1} & p_{mm} \end{bmatrix}. \tag{2.3}$$

The value $p_{ij}^{(n)}$ is the probability of moving from state i to state j in n steps and is called the n-step transition probability. The Chapman-Kolmogorov equations state that the n-step transition probabilities satisfy

$$p_{ij}^{(n)} = \sum_{r \in S} p_{ir}^{(k)} p_{ir}^{(k)} \tag{2.4}$$

where $i, j \in S$ and k is a fixed integer such that $0 \leq k \leq n$ [Kulkarni 1995]. The Chapman-Kolmogorov equations can be used to prove the following theorem, which gives a property of $P^{(n)}$, the n-step transition probability matrix. Theorem 2.1.1 will be used in Section 2.3.4.

Theorem 2.1.1. $P^{(n)} = P^n$.

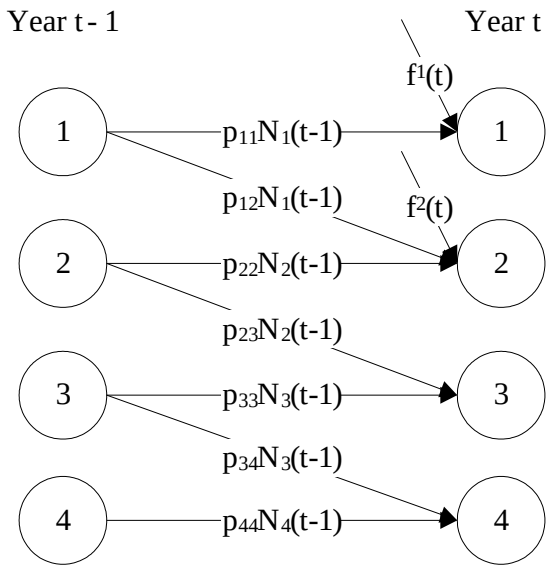


Figure 2.2: Markov chain model. *This diagrams shows a Markov chain model with four states and nine transitions.*

A matrix is called *stochastic* if each element is nonnegative and if the elements of each row sum to one [Kulkarni 1995]. A *substochastic* matrix has nonnegative elements and each row sums to less than or equal to one with at least one strict inequality. A DTMC evolves according to its transition matrix, which is stochastic.

Markov chain models can be more detailed than GPR models because they allow for more state changes than from one class to the next higher class. They can account for students who remain in a class for more than one year, for students who enter at a class level higher than one, and for students who leave the university and then return. Figure 2.2 shows a schema for a Markov chain model with four states—(1) freshman, (2) sophomore, (3) junior, and (4) senior—with new students entering the system at class (1) freshman or (2) sophomore. Note that the Markov chain models are not Markov chains due to the exogenous input (new students) to the system every year and the students who leave the system when graduating or stopping for other reasons. However, the models are similar to Markov chains and we can use some of the properties of Markov chains to aid in their analysis. Thus, we abuse the notation slightly and continue to refer to the models representing our system as DTMCs. See Section 2.3.4.

Note that we can increase the detail of the model by increasing the state space or by increasing the number of transitions. For example, instead of modeling the class level of the students in the university, we may model their semester class. This would increase the number of states to eight. Also, since some students are able to complete enough course-work in a year to move from the first class to the third class, we could add a transition to represent this. The added detail of the Markov chain models may help them perform better than the GPR models [Hopkins and Massy 1981]. A drawback to Markov chain models is the amount of data needed to calculate the proportions. Compared to the GPR method example above, which requires three parameters to be estimated, the Markov chain example given in Figure 2.2 requires nine parameters to be estimated. Actual models would most likely include more transitions than those depicted, which would result in more parameters. Determining these parameter values may be difficult depending on the data collected by a university. We begin to see the trade-off between accuracy and ease of use.

The last and most comprehensive model on enrollment that Hopkins and Massy [1981] discuss is the *cohort flow model*. A *cohort* in this context is defined as a group of students entering the university at the same time. Cohorts can be more narrowly defined by their date of enrollment and other attributes, such as their major or residency status. The cohort flow model accounts for the past states of a student [Hopkins and Massy 1981]. While the first two models use a cross-sectional approach to define states, the cohort flow model uses a longitudinal approach. *Cross-sectional* data is gathered as a snapshot of a specific point in time; *longitudinal* data is gathered over time such that records include all historical data about an individual. The Markov chain and GPR models take into account only the state of the student from the previous year. The Markov chain model allows for students who stay in a class for more than one year, but does not distinguish among students in that group—whether they have been there for one, one and a half, or two years. The Markov chain model also does not distinguish among students in the second year who are transfer students versus those who have been in the university for a year. The cohort flow model's longitudinal approach accounts for these differences [Hopkins and Massy 1981]. It categorizes students into cohorts based on their entry state into the university. These states could be new freshman, new sophomore, transfer freshman, transfer sophomore, etc. as

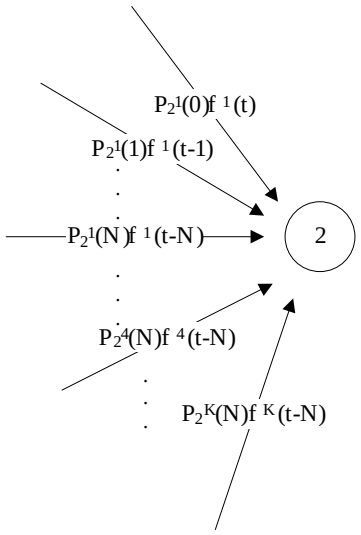


Figure 2.3: Cohort flow model: Example of constituents of one state. *This diagram shows the sources of students for one state in a cohort flow model.*

well as any other attributes that are relevant to the model. The model tracks the cohorts by keeping a cumulative total of their stay in the university. The cohorts split as their stay continues—some new freshmen become sophomores in the next year, some may still be freshmen, and some may leave the university. The cohort flow model uses *survivor fractions*, defined as $P_{jk}(s)$ = the fraction of students in cohorts of type k who are enrolled in class j, s periods after entry, to represent how students move through the university [Hopkins and Massy 1981]. Figure 2.3 shows a schema of the number of students in one class based on the cohort flow model. Following the notation of Hopkins and Massy [1981], N is the maximum number of periods after entry that a student could be enrolled, K is the number of cohorts, and $f_i(t-s)$ is the number of students in cohort i who entered the university in time period $t-s$. Note that the cohort flow model could be represented as a Markov chain model with states to represent each cohort and class level in each time period.

The obvious disadvantage of the cohort flow model is the amount of data needed to estimate the survivor fractions. However, according to Hopkins and Massy [1981] cohort flow models make the most reliable predictions of the three models presented and are robust in the presence of changing student bodies.

The models that Hopkins and Massy [1981] present can be used for both long- and short-term projections whereas other models are constructed specifically for one or the other. Models used for long-term projections often depend on variables exogenous to the university including the business cycle, national policies for financial aid, and projected state budgets. Time series analysis, yields from populations components, and curve fitting are methods used in long-term modeling [Salley 1979; Armstrong and Nunley 1981]. Models used for short-term projections, for example those presented in [Weiler 1980; Salley 1979], utilize information unavailable in the long term such as new student applications, admissions, and realized enrollment. The model presented by Salley [1979] utilizes a time series analysis to identify the trend and the variation in university enrollment. By identifying the trend, short-term variations are not extrapolated into the future, which could cause significant prediction errors.

Marshall and Oliver [1970] present a Markov chain model called a *constant-work model* in which students either complete a unit of work and move to the next class level, or they do not complete the unit of work and remain in the same class. This model has less detail than the Markov chain models already discussed. Marshall [1973] compares a Markov chain model and a cohort flow model used for predictive purposes. He finds that the two models have similar results when the cohort sizes remain constant, but that the cohort model yields better predictions when no assumptions about the cohort sizes are made. Marshall [1973] finds that there are considerable problems in building the cohort model due to data requirements, and uses averages for some of the parameters that are not available. The forecasts made by both models have large errors—up to more than ten percent.

Campbell [1975] presents a programming model for enrollment planning that tracks students based on their program of study similar to the cohort flow model previously discussed. The objective of the model is to find an optimal make-up of the student body categorized into cohorts. Thus, the model does not make forecasts, but instead produces a plan for student admissions which will, theoretically, reach a steady state. Heiberger [1993] presents a Markov chain model predicting university enrollment that groups students similar to a cohort flow model. The model then uses survival analysis to determine how long a student remains in a state [Heiberger 1993].

Many university enrollment models focus on student retention, for example [Brazziel 1987; Murtaugh et al. 1999; Titus 2004]. Murtaugh et al. [1999] present a method for predicting the retention of university students that could be used as part of an enrollment prediction model. They use survival analysis and historical data to derive retention rates of groups of students.

Lacking in the literature is a robust model for university enrollment that can be used for both long- and short-term projections while simultaneously being used to model intra-university movement. Furthermore, a general procedure to create such a model is needed. This provides the background and motivation for our study.

2.2 Data

The data needed to model university populations depends on the purpose and detail of the model. The data available to us for this study is that recorded by Clemson University. We made use of the records on all undergraduates in each semester and summer session from fall 1998 to spring 2007. A complete listing of fields within each record is presented in Table 2.2. Note that the variables are split into common information and term information. The common information is that which typically does not change from term to term such as race, gender, and SAT score. The term information is expected to change each semester and includes credit hours attempted in the current semester and credit load. Possible values of selected fields are presented in Tables 2.3, 2.4, 2.5.

The fields we predominantly use include semester class, college, enrollment status, residency status, and credit load. *Enrollment status* is an attribute assigned to each student by the university identifying him or her as a first-time, transfer, returning, continuing or transient student. *Residency status* identifies each student as in-state or out-of-state, and *credit load* identifies each student as full-time or part-time. The average number of students in a fall semester within our study data is 13,861.

COMMON INFORMATION	TERM INFORMATION
ID number (CUID)	Term
State	Enrollment On Campus
Race	Enrollment Off Campus
Gender	Credit Load
Birth date	College
Original Entry Status	Major
Honors Program	Minor
Home County	Termination
Marital Status	Academic Status
Religious Preference	Residency Status
Original Entry Major	Semester Class
First Year GPR	Enrollment Status
Degree Sought	Semester Hours Attempted
SAT Verbal	Semester Hours Earned
SAT Math	Semester Grade Points Earned
ACT Composite	Semester GPR
Athlete	Credit Hours Attempted to Date
Academic Suspensions	Credit Hours Earned to Date
Satisfactory Progress	Grade points Earned to Date
Curriculum Year	GPR to Date
Curriculum Version	Special Program
Academic Renewal Term	Student Fee
Veteran	Housing Status
Date Enrolled	

Table 2.2: Fields included in records stored for each student at Clemson University each semester.

Academic Status (Code)	Housing Status (Code)
Good Standing (0)	Not in University Housing ( )
Academic Renewal (1)	Reservation Canceled (R)
Admitted on Probation (3)	Residence Hall (1)
Appealed Suspension (6)	Married Student Housing (2)
Suspension (7)	Broken Contract (3)
Appealed Dismissal (8)	Offer Housing (4)
Dismissal (9)	Other (6)
	Co-op (7)
	Withdrawn for Practice Teaching (8)
	Withdrawn from School (9)

Table 2.3: Possible values of Academic Status and Housing Status fields.

Semester Class (Code) (Hours)	Enrollment Status (Code)
Unclassified (0)	Transient (0)
1^{st} Sem. Freshman (1) (0-15)	First-time (1)
2^{nd} Sem. Freshman (2) (16-29)	Transfer (2)
1^{st} Sem. Sophomore (3) (30-45)	Returning (3)
2^{nd} Sem. Sophomore (4) (46-59)	Continuing (4)
1^{st} Sem. Junior (5) (60-77)	
2^{nd} Sem. Junior (6) (78-94)	
1^{st} Sem. Senior (7) (95-111)	
2^{nd} Sem. Senior (8) (112-999)	
Postgraduate (12)	
Post baccalaureate (15)	

Table 2.4: Possible values of the Semester Class and Enrollment Status fields.

College (Code)	Termination (Code)
Agriculture, Forestry and Life Sciences (10)	Candidate (C)
Architecture, Arts, and Humanities (30)	Discharged (D)
Business and Behavioral Sciences (40)	Deficient (F)
Engineering and Science (50)	Graduated (G)
Health, Education and Human Development (70)	Grad Posthumously (P)
Undergraduate School (80)	Deceased (X)

Table 2.5: Possible values of College and Termination fields.

2.3 University Population Modeling Procedure

A main goal of our study is to build a model for university enrollment that can be used for both long- and short-term projection. Potentially, this model could take on many different forms. One model may simply predict total population and movement between class levels without differentiating between college, course load, and residency status. Another more detailed model may track the movement of students by college, major and class. Our design is based on the Markov chain models discussed in Section 2.1 because of the flexibility they afford—from very simple GPR models to very detailed cohort flow models.

The other main goal of our study is to design a procedure for creating university population models. As noted by Hopkins and Massy, the goal of enrollment modeling may affect the type of model used [1981]. Furthermore, the goal of the model may affect the amount of data gathered and the student characteristics studied. Because each university population model may serve a different purpose, very little has been published on procedures to create university population models in general. In this section we present a straightforward procedure for creating university population models. Utilizing this procedure, we give examples from a particular population model built for Clemson University.

The purpose of the model we build is threefold: (1) to predict the total university population and several subgroups of the population with accuracy in the short term; (2) to project total university population and several sets of subgroups of the population in the long term given assumptions about incoming students and graduation and drop out rates; and (3) to predict population changes resulting from changes in transition rates including freshman retention and senior graduation rates. A university population model that meets these criteria can be used to project scenarios under alternate assumptions, which aids in developing policy decisions and other university planning.

The procedure we have developed for creating university enrollment models has the four major steps outlined in Algorithm 2.1. Each step is presented individually in the next four subsections.

Algorithm 2.1: Procedure for modeling university populations.

1. State identification.
2. Transition matrix estimation and verification.
3. Historical verification.
4. Model usage.

2.3.1 State Identification

The first step in modeling university population using a Markov chain model is to identify the state space. We consider two sets of student characteristics in this step: those that must be distinguished in the model results and those that are needed to model the population with accuracy. The second set of characteristics may not be needed for reporting purposes, but aids in creating states that have significantly different transition rates than the rest of the population. Deciding which characteristics to consider is guided by experience and testing. Experience may indicate that part-time students move through the university at a different rate than full-time students. Testing might identify other characteristics that lead to a more accurate or robust model.

The procedure to identify the state space begins with choosing sets of characteristics to consider. The next step is to test for correlation between the characteristics we initially select and the events we want to model. This step may have no effect on attributes that must be distinguished in the model results, but plays a role in the testing of the other attributes. A variable that approximates the events to be modeled may be used for this step. Because we are modeling population movement and total population in the university, we choose to measure correlation of potential state variables (characteristics) to the event of moving to a higher semester class. For example, if we find that students with different residency statuses move to the next higher class with significantly different rates, we may include residency status in our model. We use the χ^2 test for independence for testing. The two-way χ^2 statistic tests the null hypothesis of independence between the row variable and the column variable of a contingency table. When the sample size is large—typically greater than 30 with at least five elements in each cell—and the null hypothesis is true, the test statistic's distribution is approximately χ^2 with $(c-1)(r-1)$ degrees of freedom, where

	Remain in current class	Move to higher class	Total
In-state	1,885 21%	7,089 79%	8,974
Out-of-state	1,188 31%	2,627 68%	3,815
All	3,073 24%	9,716 75%	12,789

Table 2.6: Contingency table for residency status versus the event moving to a higher semester class.

c is the number of columns and r is the number of rows [Ross 2002]. Table 2.6 shows the contingency table from a χ^2 test on residency status versus our event variable class movement. The test shows that the row and column variables are not independent, and thus residency status is a significant variable.

The product of the number of possible values for each characteristic gives an upper bound on the number of states. It is an upper bound and not the actual number because some combinations could be infeasible or could be excluded by choice. An example of the former situation is that it is not possible for a first-time student to be a senior at CU because he or she has not had the opportunity to accumulate enough credits, even with AP credits. Note that there is a drawback associated with specifying many states. The counts of students in each state become low and the variability in predicting the associated transitions becomes high, which reduces the accuracy of the model. Thus, practicing parsimony is important when choosing characteristics and identifying states. For example, we choose not to split part-time students by class and college because there are few part-time students.

At this stage of model building, there is often much trial and error to decide on the final set of states. For example, because there are too many majors to use individually as states in our model, we attempt to identify and cluster together majors with significantly different rates of movement. This could capture the populations of students with different movement due to major, but would not require as many states. To accomplish this, we cluster majors based on the courses in which they enroll and then use this grouping as states. We first order all possible courses—suppose there are n—and assign each a unique number. We represent major m by an n-vector where the jth component represents the number

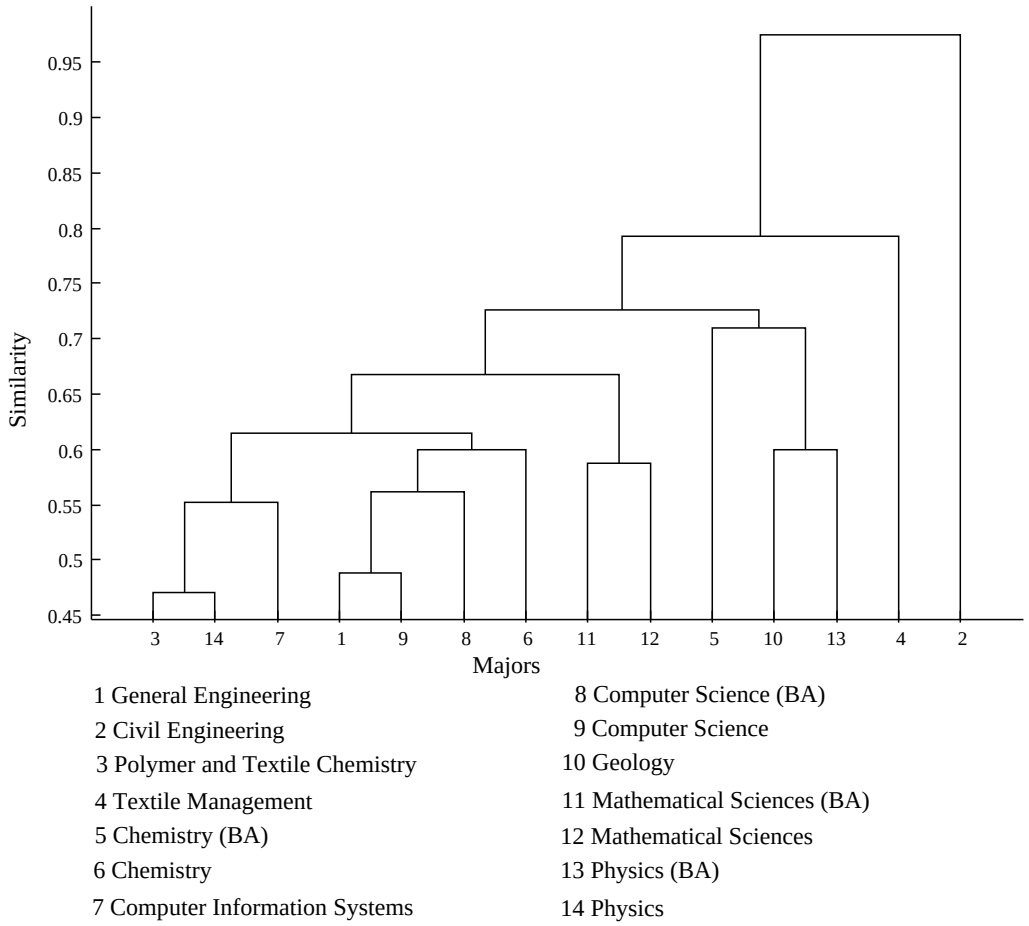


Figure 2.4: Dendrogram of clustering on majors of first semester students within the college of Engineering and Science in fall 2005.

of m majors taking course j. We measure the distance between two majors as the angle between their representative vectors. To group the majors, we use a hierarchical agglomerative clustering technique, which joins the closest pairs iteratively until all elements are grouped into one cluster. The graphic representation of this clustering technique is called a *dendrogram*. Figure 2.4 shows a dendrogram of a clustering where no clear groupings are evident. Two problems arise from attempting to group on major. The first is that few of the clusterings result in obvious groupings. The second problem is that because semester class and college are two characteristics we want to keep in the model, we first group majors by semester class and college. This reduces the individual cell counts and ultimately leads us to abandon having major as part of the state space.

Semester	College	Enrollment Status	Residency Status	Number of states
1 2 3 4	AFLS AAH BBS ENSC HEHD	First-time Transfer Continuing	In-state Out-of-state	120
5 6	AFLS AAH BBS ENSC HEHD	Transfer Continuing	In-state Out-of-state	40
7 8	AFLS AAH BBS ENSC HEHD	Continuing	In-state Out-of-state	20
			Total number of states	180

Table 2.7: Description and number of states for university population model.

In our model the state identification step resulted in 204 states: ten states for incoming students, six states for leaving students, and 188 states for students within the university. The ten states for incoming students include designating each student as transient, first-time, transfer, returning, or 'continuing from a semester not in the model' and in-state or out-of-state. The six leaving states include graduating, leaving from freshman, sophomore, junior, or senior class, and leaving from any other state. Four of the 188 states for students within the university are part-time students who are returning students or not, who are in-state or out-of-state. Four of the remaining 184 states for students within the university are full-time students who are returning or transient, who are in-state or out-of-state. The other 180 states are detailed in Table 2.7.

2.3.2 Transition Matrix Estimation and Verification

The second step in modeling university population using a Markov chain model is to estimate the transition matrix. We do this by first counting the number of students transitioning between each pair of states for each pair of years that we have data available. Note that if the model is to be used by semester, we would count for each semester. Table 2.8 shows

$$\begin{bmatrix} 0 & 3,060 & 360 & 180 & 0 & 0 \\ 0 & 330 & 2,310 & 330 & 0 & 330 \\ 0 & 0 & 435 & 1,885 & 290 & 290 \\ 0 & 0 & 0 & 660 & 1,980 & 660 \\ 0 & 0 & 0 & 0 & 1,080 & 2,520 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0.85 & 0.1 & 0.05 & 0 & 0 \\ 0 & 0.1 & 0.7 & 0.1 & 0 & 0.1 \\ 0 & 0 & 0.15 & 0.65 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0.2 & 0.6 & 0.2 \\ 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 2.8: Example of transition counts and transition matrix.

a simple example of transition counts for a model with six states, where state 0 represents incoming students, states 1-4 represent the freshman through senior classes, and state 5 is the exiting state. We convert the transition counts into a transition matrix by dividing each element by its row sum.

The procedure just described finds transition matrices based on one year of data. This may not be the best transition matrix to use for predictive purposes because of the variability that could occur from one year to the next. To find an average transition matrix for two or more years, we find the weighted average of the respective transition matrices by adding the transition count matrices from these years and calculating the transition matrix in the same manner as before. Note that we cannot simply add two transition matrices because the resulting matrix would not necessarily be stochastic or substochastic. We must decide how many years of data to include in the transition matrix for the model. To guide in making this decision, we test and compare four methods with available data for one to five year periods. The four methods include using the transition matrix from (1) the previous year, (2) a two-year moving average, (3) a three-year moving average, and (4) the average of all data available. Figures 2.5, 2.6, 2.7, and 2.8 show the results of these comparisons.

Figure 2.5 shows the sum of absolute error rates for predicting the total population and several subgroups of the population one year in advance using the four methods over five years. The error rates for the freshman are high because we use the number of incoming students estimated by the admissions office for future predictions. These estimates sometimes have high error rates themselves as we showed at the beginning of this chapter. Figures 2.6 and 2.7 show the error rates for predicting the total population one and two years in advance, respectively, using the four methods. Figure 2.8 shows the average square root of the sum of squared error, which is the Euclidean distance between the actual values and the

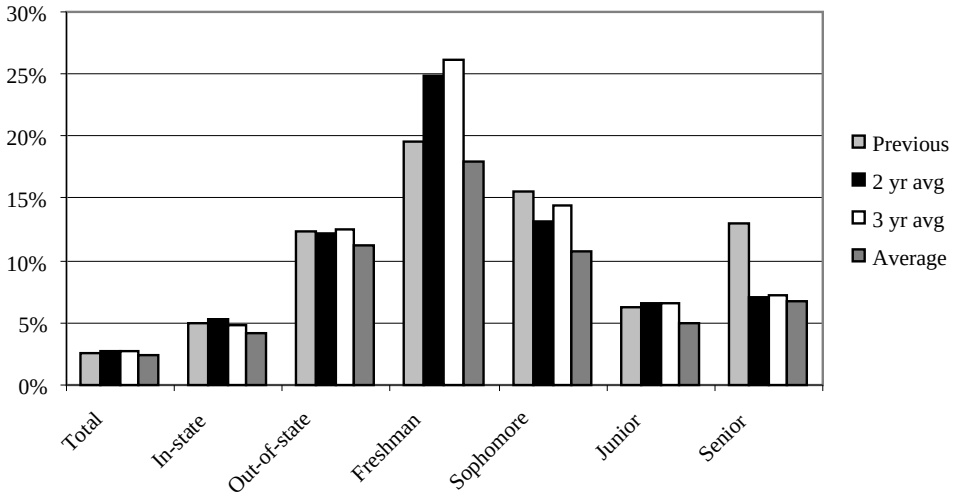


Figure 2.5: Sums of one year in advance error rates over years predicted for several groups using four methods.

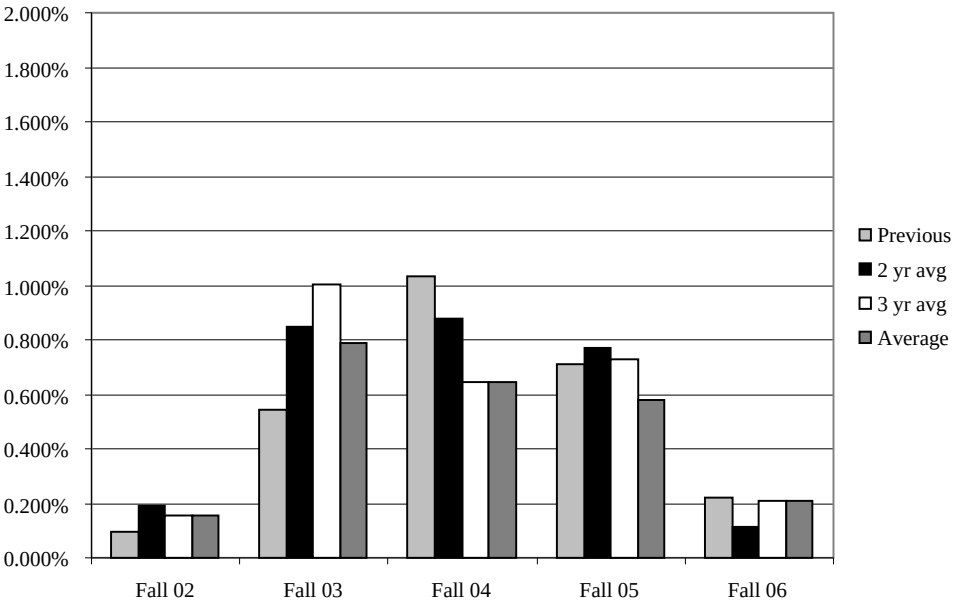


Figure 2.6: Error rates for total population when predicting one year forward using four methods.

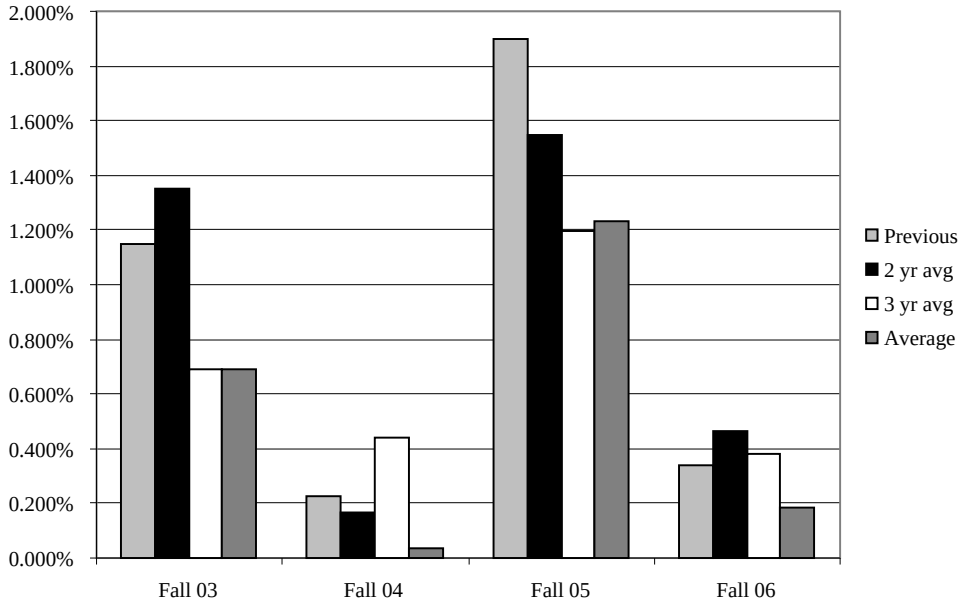


Figure 2.7: Error rates for total population when predicting two-years forward using four methods.

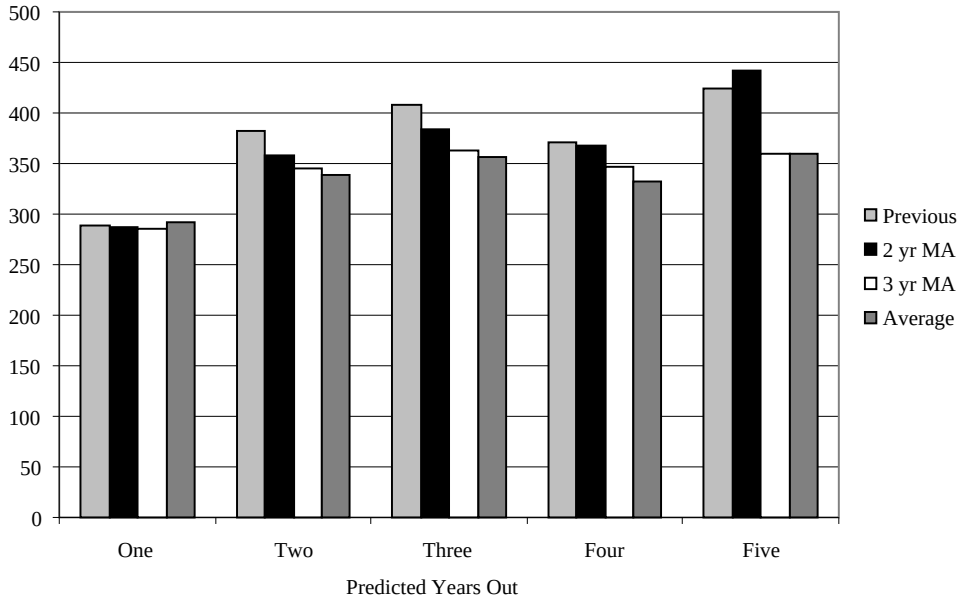


Figure 2.8: Average sum of Euclidean distances from actual population using four methods.

predicted values, for each of the four methods when used to predict future populations one to five years in the future. Because we have limited available data, the number of elements in the sum is fewer when predicting out farther. Thus, we must take the average of the sums of error in order to have commensurate data. Note that the error tends to increase with more extrapolation, but the three-year moving average and the total average tend to have the lowest error within each year.

Because the projections based on the previous year's data tend to be volatile in terms of error rates, we eliminate it from consideration. The transition matrices that depend on all available data require too much data to be used consistently in the future when many more years of data are available. Furthermore, as university policies change earlier data will become irrelevant to future transition probabilities. Thus, we eliminate that method from consideration and are left with the two and three-year moving averages. Because the three-year moving average has consistently lower sum of squared error for future predictions than the two-year moving average, we choose the three-year moving average to estimate our transition probability matrices.

2.3.3 Historical Verification

The third step in modeling university population using a Markov chain model is to verify the accuracy of the model on historical data. If the accuracy is not within the desired range, steps one and two may need to be repeated. We measure the accuracy of our model on predicting the total university population, several subgroups, and the graduating population. We present the results for total population predictions in Figure 2.9 and Table 2.9. The lines in Figure 2.9 represent the projections made from a given year. We assume that predictions for the following year are made after the start of the semester of the current year. Thus, we know the counts for the current year when predicting forward. For example, in fall 2002 we know the total university population for fall 2002 and project out one to four years to predict the population for fall 2003 to fall 2006, respectively. The series marked by triangles represent these projections. Table 2.9 shows that the error rates are within 1% when projecting out one year and are within 1.5% when projecting out up to five years. Because the populations of current and previous semesters are known with certainty, the

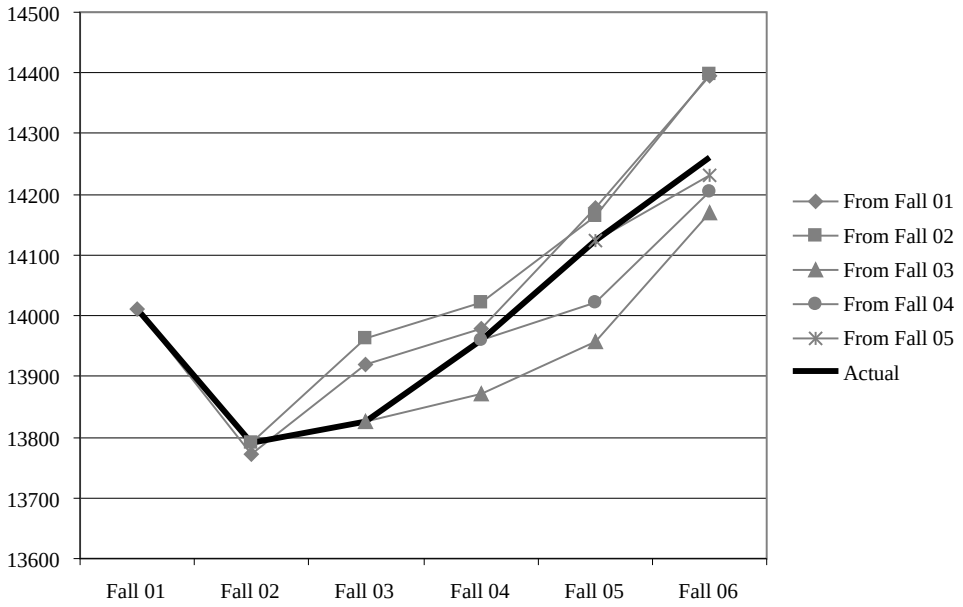


Figure 2.9: Actual and predicted total university population.

From To \	Fall 01	Fall 02	Fall 03	Fall 04	Fall 05
Fall 02	-0.2%				
Fall 03	0.7%	1.0%			
Fall 04	0.1%	0.4%	-0.6%		
Fall 05	0.4%	0.3%	-1.2%	-0.7%	
Fall 06	0.9%	1.0%	-0.6%	-0.4%	-0.2%

Table 2.9: Error rates for total university population by year.

table is lower triangular reflecting that we are forcasting forward. The error rates are well within the 5% error rate criteria that we set for forecasting aggregate populations.

Figure 2.10 shows the actual and predicted full-time students in the university by class. On average, the absolute error rates for predicting one year forward are less than 4% for freshmen, 2% for sophomores, 1% for juniors, and 1% for seniors. These error rates are within or near the bounds of 3.9% for populations associated with first-time students, 10.3% for populations associated with transfer students, and 5.3% for aggregate populations. Note that the freshman population is made up of continuing, first-time, and transfer students.

Figure 2.11 shows the actual and predicted graduating students in the university. The error rates are within 5.3% for all projections. Note that from these and other forecasts we can estimate the retention and graduation rates of specific populations, which we do in the

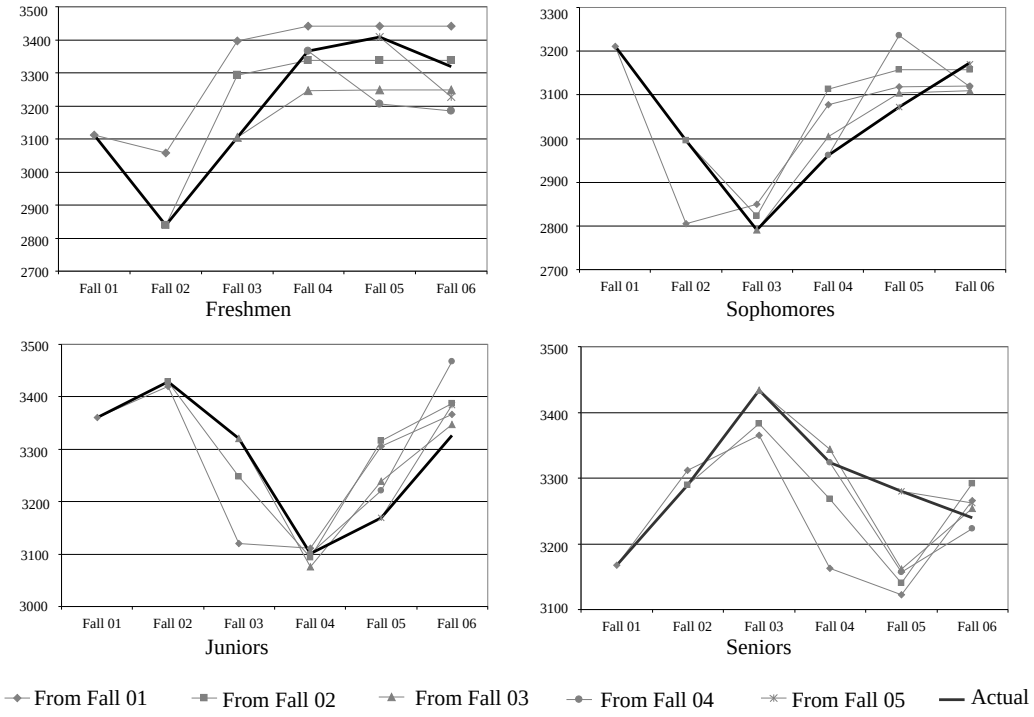


Figure 2.10: Actual and predicted full-time students by class.

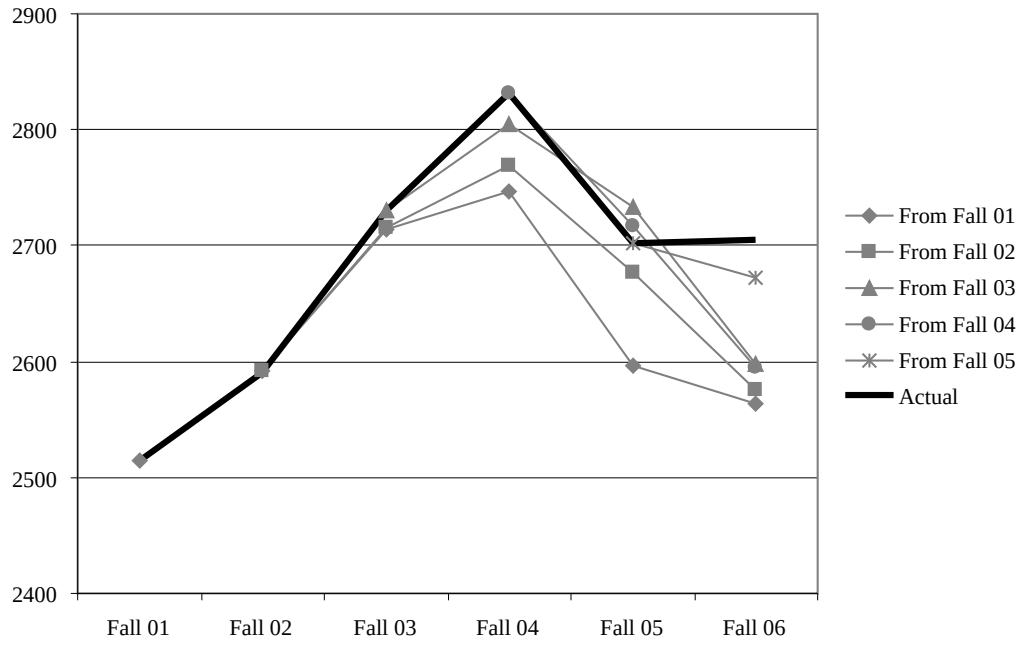


Figure 2.11: Actual and predicted graduating students.

next section. Because the errors for predicting the number of graduates seem to increase for the later years we predict, we suspect that qualitative changes, such as a decrease in graduation requirements, may have occured that are not yet being recorded by the transition matrices. In the next section we discuss investigating scenarios under alternate assumptions, which can aid in making accurate projections when policy changes occur.

2.3.4 Model Usage

The fourth step in modeling university population using a Markov chain model is to use the model for forecasting student populations. In Section 2.3.3 we tested the model by projecting forward within the data we have available and testing for accuracy. The final step is to project beyond the available data. Because we are using a Markov chain model, we can take advantage of certain properties to aid in this projection.

We now introduce more properties and notation about DTMCs using the notation and definitions of [Kulkarni 1995]. Let S be the state space of a DTMC, P be the transition matrix, and $i, j \in S$. If, for some $n \geq 0$, $p_{ij}^{(n)} > 0$, then state j is *accessible* from state i; we denote this as $i \rightarrow j$. If i is accessible from state j as well, then i is said to *communicate* with j; we denote this as $i \leftrightarrow j$. A *communicating class* $C \subseteq S$ has the properties (1) for all $i, j \in C$, $i \leftrightarrow j$ and (2) every element that communicates with an element in C is also in C. If for all $i \in C$ and for all $j \notin C$, j is not accessible from i, then C is called a *closed* communicating class. If all the states of a DTMC are in one closed communicating class, the DTMC is called *irreducible*. Otherwise, it is called *reducible*. A state is called *transient* if there is a positive probability that once the system has left the state, it will never return. A state is called *recurrent* if this probability is zero. Note that a finite Markov chain cannot have all transient states.

We revisit the simple modeling example from Table 2.8 by showing the transition diagram in Figure 2.12. This is a simplified version of our university population model. Recall that state 0 represents incoming students, states 1-4 represent the freshman through senior classes, and state 5 is the exiting state. States 0-4 are clearly transient because after following an arrow from one of these states, there is no directed path to return to it. The transition matrix governing the four internal states is as follows:

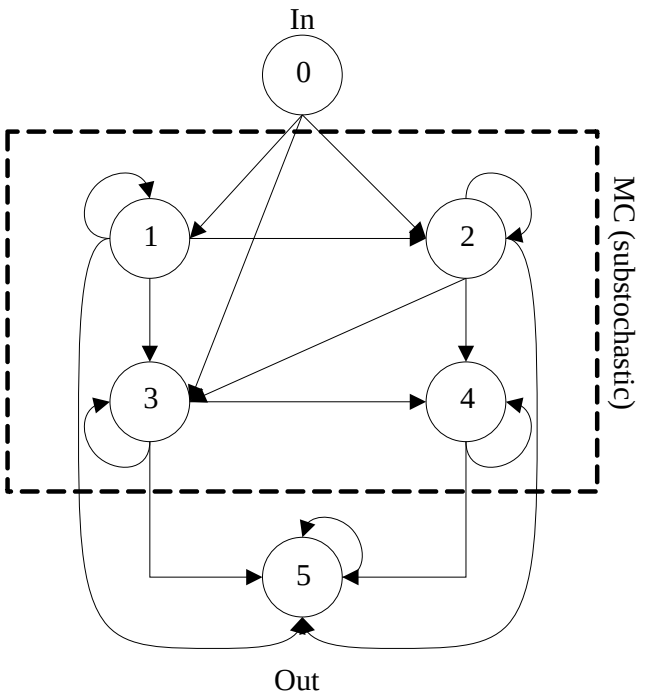


Figure 2.12: Transition diagram for modeling example given in Table 2.8.

$$\mathrm{P} = \begin{bmatrix} 0.1 & 0.7 & 0.1 & 0 \\ 0 & 0.15 & 0.65 & 0.1 \\ 0 & 0 & 0.2 & 0.6 \\ 0 & 0 & 0 & 0.3 \end{bmatrix}.$$

Note that P is substochastic because at least one row sum is less than one. In Figure 2.12 we represent state 5 as an absorbing state, which means that once a student has left the system, he or she remains gone. This assumption is relaxed in our actual model where we allow returning students.

Let us focus on the internal transient states of our model. Let $\Pi(n)$ be the vector of university populations at some time n. In our simple example, $\Pi(n)$ is the vector of students in states 1-4. Let P be the transition matrix of the internal states. Note that P is substochastic because students leave the system and, therefore, P has at least one row that sums to less than one. Furthermore, because all students eventually leave the system, all states are transient meaning for some $n > 0$ the n-step transition probability $p_{ij}^n = 0$ for all internal states i, j. Thus, we have that $\lim_{n\to\infty} P^n = 0$.

Continue to define P and $\Pi(n)$ as above. Let Δ be the vector of incoming students each year and $\Pi(0) = C$ be the vector of students at time zero, both of which are represented in the same manner as $\Pi(n)$ above. We solve the following system of equations to find the population at time $n \geq 0$.

$$\begin{aligned}
\Pi(1) &= \Pi(0)P + \Delta \\
&= CP + \Delta \\
\Pi(2) &= \Pi(1)P + \Delta \\
&= (CP + \Delta)P + \Delta \\
&= CP^2 + \Delta P + \Delta \\
&\vdots \\
\Pi(n) &= \Pi(n-1)P + \Delta \\
&= CP^n + \Delta P^{n-1} + \Delta P^{n-2} + \ldots + \Delta P + \Delta \\
&= CP^n + \Delta \textstyle\sum_{i=0}^{n-1} P^i.
\end{aligned}$$

To find the equilibrium population, we take the limit of $\Pi(n)$ as n increases without bound as follows:

$$\begin{aligned}
\lim_{n\to\infty} \Pi(n) &= \lim_{n\to\infty} \left[CP^n + \Delta \textstyle\sum_{i=0}^{n-1} P^i\right] \\
&= C \lim_{n\to\infty} P^n + \Delta \lim_{n\to\infty} \textstyle\sum_{i=0}^{n-1} P^i \\
&= \Delta \textstyle\sum_{i=0}^{\infty} P^i \\
&= \Delta \left[I - P\right]^{-1}.
\end{aligned} \tag{2.5}$$

The final equality of 2.5 is a consequence of two theorems on matrix sequences and series whose proofs can be found in [Cullen 1990]. We take the relevant parts of those to form the following theorem, which shows that $(I - P)$ is invertible.

Theorem 2.3.1. *Let $\{A_m\}$ be a sequence of matrices from $C_{n\times n}$. If the matrix sequence $\{A_m\} \to 0$, then $(I - A)$ is invertible and $(I - A)^{-1} = \sum_{m=0}^{\infty} A^m$.*

Because $\{P_n\} \to 0$, it follows from Theorem 2.3.1 that $(I-P)$ is invertible and $(I-P)^{-1} = \sum_{n=0}^{\infty} P^n$. We forgo the proof because it is beyond the scope of this study.

Using (2.5) we can find the equilibrium population assuming that transition rates and the make-up of the incoming class remain constant. Although we expect changes in the

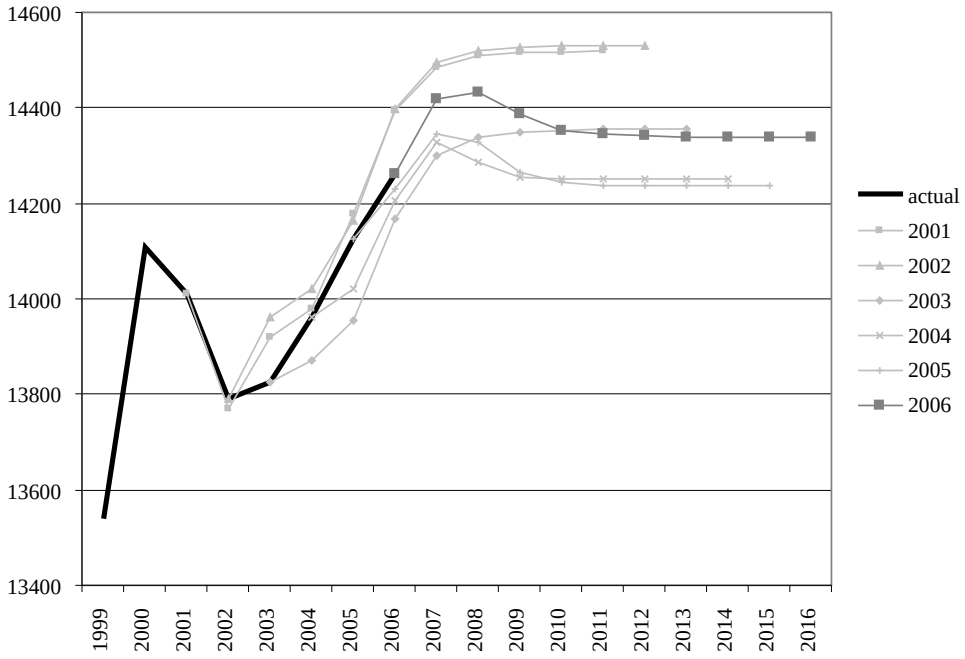


Figure 2.13: Actual and predicted total university population.

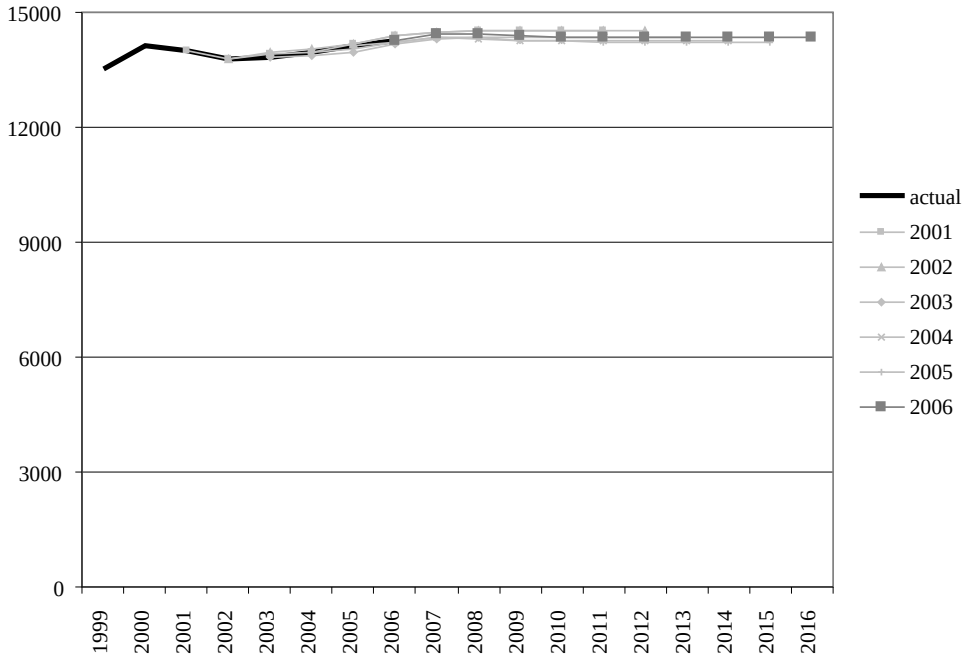


Figure 2.14: Actual and predicted total university population scaled.

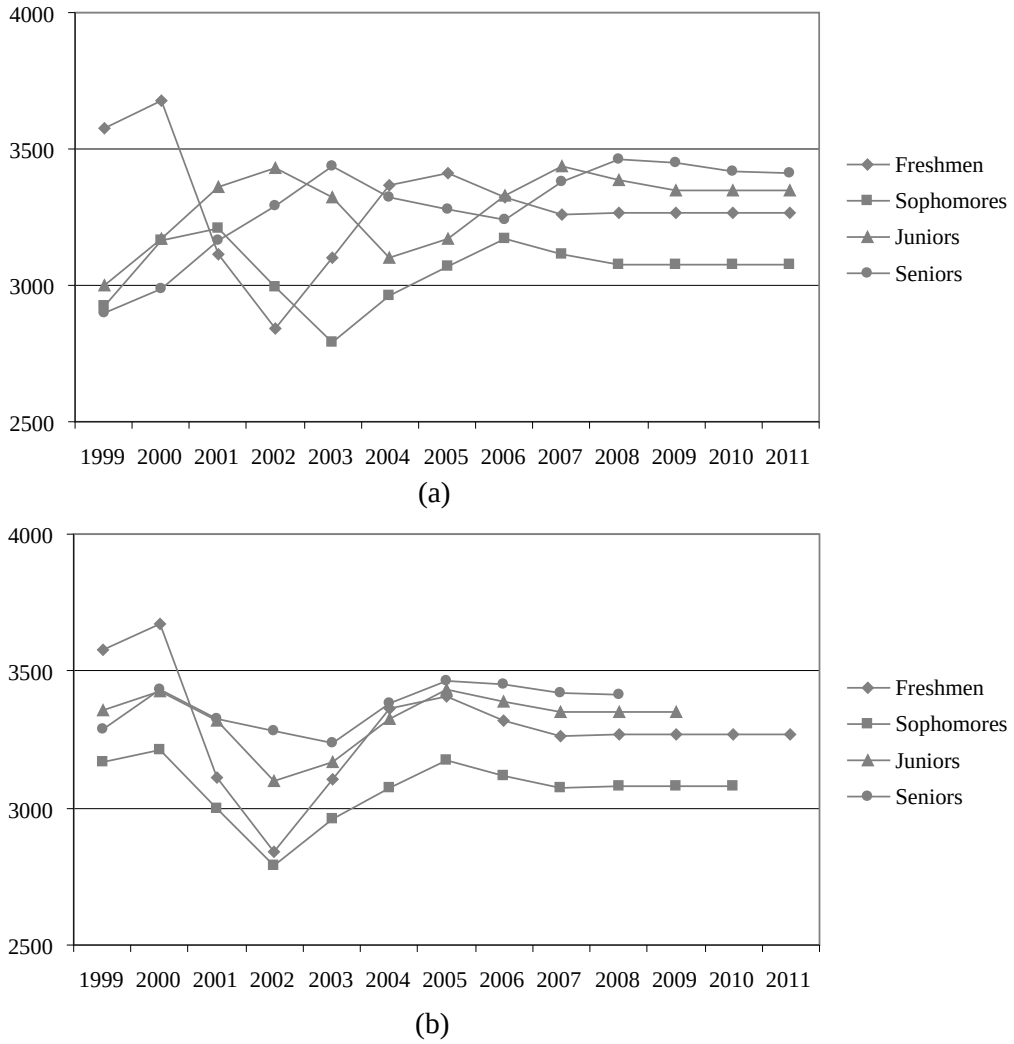


Figure 2.15: Actual and predicted full-time university population by class.

system in the form of differing incoming classes, changing graduation rates, etc., finding the equilibrium populations aids in developing policy decisions by setting target populations. Figures 2.13 and 2.14 show the actual and predicted total university population for Clemson University. The series in the figures predict out ten years starting at fall 2001 to fall 2006, each of which has different transition matrices and input. The constant predictions near the end of the series indicate that the equilibrium population has been approximately reached. Figure 2.14 shows Figure 2.13 to scale and gives us an appreciation of the accuracy which our projections possess.

Figure 2.15 shows the actual and predicted values of the full-time university population by class. The values are actual from 1999 to 2006 and are predicted from 2007 and beyond. Figure 2.15(a) shows each class prediction by year. We can see that a large freshman class results in a large sophomore class the following year, a large junior class in two years, and a large senior class in three years. To visualize these ripple effects better, Figure 2.15(b) has the freshman series on the actual year and each higher class shifted back one year. We can see that after the high or low inputs have rippled through and a constant incoming class is predicted, an equilibrium population becomes apparent. However, our model can also account for incoming classes of varying sizes, which may be the norm.

In the following section we give more results based on alternate assumptions for the input, output, and transition probabilities of the model.

2.4 Hypothetical Analyses

Section 2.3.3 showed test results for our model on how accurately it predicts total university populations, full-time populations by class, and graduating populations within the data available. Section 2.3.4 gave model results on predicting forward to examine how input changes filter through the model and how the system settles on an equilibrium population. In this section, we analyze equilibrium populations formed from alternate assumptions, and also discuss how to derive other relevant information, such as probability of graduating and average time to graduate, using the properties of our model.

Example 1: Varying admission

Since 2003, Clemson University has had a target new student enrollment of 2,800 first-time students and 800 transfer students each fall. However, actual enrollments were different than the targets in several years by up to two hundred first-time students and three hundred transfer students. We have seen the effects of these perturbations in the previous section as ripples through class levels. Although we do not expect actual enrollments to match target enrollments every year, examining the equilibrium populations resulting from differing incoming populations aids in making admissions policy decisions. Table 2.10

	First-time	2,800	2,600	(-7%)	3,000	(7%)	2,800	(%)
	Transfer	800	800	(0%)	800	(0%)	700	(-13%)
Class	Freshmen	3,268	3,055	(-7%)	3,481	(7%)	3,237	(-1%)
	Sophomores	3,079	2,912	(-5%)	3,246	(5%)	3,008	(-2%)
	Juniors	3,349	3,181	(-5%)	3,518	(5%)	3,269	(-2%)
	Seniors	3,411	3,246	(-5%)	3,575	(5%)	3,331	(-2%)
College	AFLS	2,305	2,177	(-6%)	2,433	(6%)	2,259	(-2%)
	AAH	2,055	1,944	(-5%)	2,166	(5%)	2,011	(-2%)
	BBS	3,673	3,481	(-5%)	3,865	(5%)	3,591	(-2%)
	ENSC	3,040	2,865	(-6%)	3,214	(6%)	2,995	(-1%)
	HEHD	2,035	1,928	(-5%)	2,142	(5%)	1,990	(-2%)
Load	Full-time	13,363	12,646	(-5%)	14,081	(5%)	13,100	(-2%)
	Part-time	975	941	(-3%)	1,009	(3%)	951	(-3%)
	Total	14,339	13,587	(-5%)	15,090	(5%)	14,050	(-2%)

Table 2.10: Equilibrium population counts and percentage difference by varying first-time and transfer assumptions.

shows the equilibrium population targeted by four sets of proposed incoming populations assuming that transition probabilities remain constant. That is, the outcomes in Table 2.10 would result from the actual first-time and transfer populations equaling the number of first-time and transfer students given, with no other changes in the system. We use 2,800 first-time students and 800 transfer students as the base enrollment and show the percentage difference when using alternate enrollment assumptions. A seven percent increase or decrease in the number of first-time students results in a five percent increase or decrease, respectively, in the equilibrium total population. In recent years the number of new transfer students at CU has been nearer to 700 than the target of 800. The last column of Table 2.10 shows the result of transfer admissions remaining at 700, which includes a drop in total population of about 300 students.

Example 2: Changing retention rates

CU has adopted several new teaching methods in largely freshman courses that have lowered the failing rates in those courses significantly. If lower failing rates translate into lower freshman drop-out or stop rates (higher freshman retention), this will have an effect on the total university population. For example, between fall 2005 and fall 2006 the university drop-out rate for full-time, in-state, first-time freshmen in the college of engineering and science was 14.5%. If that rate decreases by 10% of the current rate to 13.05%, more

	Change in Freshman Drop-out Rates	0%	-10%		10%	
	First-time	2800	2800		2800	
	Transfer	800	800		800	
Class	Freshmen	3,268	3,272	(.1%)	3,265	(-.1%)
	Sophomores	3,079	3,117	(1.2%)	3,041	(-1.2%)
	Juniors	3,349	3,390	(1.2%)	3,309	(-1.2%)
	Seniors	3,411	3,451	(1.2%)	3,371	(-1.2%)
College	AFLS	2,305	2,326	(.9%)	2,284	(-.9%)
	AAH	2,055	2,072	(.9%)	2,037	(-.9%)
	BBS	3,673	3,709	(1.%)	3,636	(-1.%)
	ENSC	3,040	3,070	(1.%)	3,009	(-1.%)
	HEHD	2,035	2,052	(.8%)	2,018	(-.8%)
Load	Full-time	13,363	13,487	(.9%)	13,240	(-.9%)
	Part-time	975	984	(.9%)	966	(-.9%)
	Total	14,339	14,472	(.9%)	14,206	(-.9%)

Table 2.11: Equilibrium population counts and percentage difference by varying freshman drop-out rate assumptions.

students would remain in the university, assuming all other transition rates from this state are adjusted and all other rates are held constant. Table 2.11 shows the equilibrium population for no change, a ten percent decrease, and a ten percent increase in freshman drop-out rates relative to the current rates and assuming that other transition probabilities remain constant.

Example 3: Changing graduation rates

CU has adopted several curriculum changes in the past two years including changing general education requirements and lowering the minimum number of credit hours required to graduate. These changes imply that students' academic careers at CU could be shortened and thus graduation rates may increase. Recall that in Figure 2.11 the number of students graduating in 2006 is higher than the predictions. Table 2.12 shows the equilibrium population for no change, a ten percent increase, and a ten percent decrease in graduation rates relative to the current rates and assuming that other transition probabilities remain constant. For example, between fall 2005 and fall 2006 the graduation rate for full-time, in-state, continuing first semester seniors in the college of engineering and science was 37.41%. If that rate increases by 10% of the current rate to 41.15%, fewer students would remain in the

	Change in Graduation Rates	0%	10%		-10%	
	First-time	2800	2800		2800	
	Transfer	800	800		800	
Class	Freshmen	3,268	3,268	(-.01%)	3,269	(.01%)
	Sophomores	3,079	3,076	(-.1%)	3,082	(.1%)
	Juniors	3,349	3,336	(-.4%)	3,364	(.4%)
	Seniors	3,411	3,230	(-5.3%)	3,612	(5.9%)
College	AFLS	2,305	2,280	(-1.1%)	2,333	(1.2%)
	AAH	2,055	2,022	(-1.6%)	2,092	(1.8%)
	BBS	3,673	3,618	(-1.5%)	3,733	(1.6%)
	ENSC	3,040	2,990	(-1.6%)	3,096	(1.9%)
	HEHD	2,035	2,000	(-1.7%)	2,074	(1.9%)
Load	Full-time	13,363	13,157	(-1.5%)	13,594	(1.7%)
	Part-time	975	926	(-5.1%)	1,031	(5.7%)
	Total	14,339	14,083	(-1.8%)	14,625	(2.%)

Table 2.12: Equilibrium population counts and percentage difference by varying graduation rate assumptions.

university assuming that all other transition rates from this state are adjusted and all other rates are held constant.

Example 4: Simultaneous rate changes

Our model is also able to examine the effects of multiple changes at once, which is a more realistic assumption than only one change occurring at a time. Table 2.13 shows the equilibrium population for two combined changes in graduation rates and freshman drop-out rates assuming that other transition probabilities remain constant. Figure 2.16 shows the total population for several changes to freshman drop-out rates and graduation rates.

Example 5: Making use of the Markov property

Because of the properties of DTMCs, we can use our model not only to predict the outcome of policy decisions but also to back into policy decisions given the desired results or predicted changes. For example, suppose graduation rates are expected to increase by five percent, which would eventually lower the total university populations. If administrators want to keep the university population heading toward the estimated equilibrium population, we can use our model to find the number of new students needed each year to keep the

	Change in Freshman Drop-out Rates	0%	-5%		-10%	
	Change in Graduation Rates	0%	5%		10%	
	First-time	2800	2800		2800	
	Transfer	800	800		800	
Class	Freshmen	3,268	3,270	(.1%)	3,271	(.1%)
	Sophomores	3,079	3,096	(.6%)	3,114	(1.1%)
	Juniors	3,349	3,363	(.4%)	3,377	(.8%)
	Seniors	3,411	3,338	(-2.1%)	3,268	(-4.2%)
College	AFLS	2,305	2,303	(-.1%)	2,301	(-.2%)
	AAH	2,055	2,047	(-.4%)	2,039	(-.8%)
	BBS	3,673	3,663	(-.3%)	3,654	(-.5%)
	ENSC	3,040	3,029	(-.4%)	3,020	(-.7%)
	HEHD	2,035	2,025	(-.5%)	2,016	(-.9%)
Load	Full-time	13,363	13,319	(-.3%)	13,278	(-.6%)
	Part-time	975	954	(-2.2%)	934	(-4.2%)
	Total	14,339	14,273	(-.5%)	14,213	(-.9%)

Table 2.13: Equilibrium population counts and percentage difference by varying freshman drop-out and graduation rate assumptions.

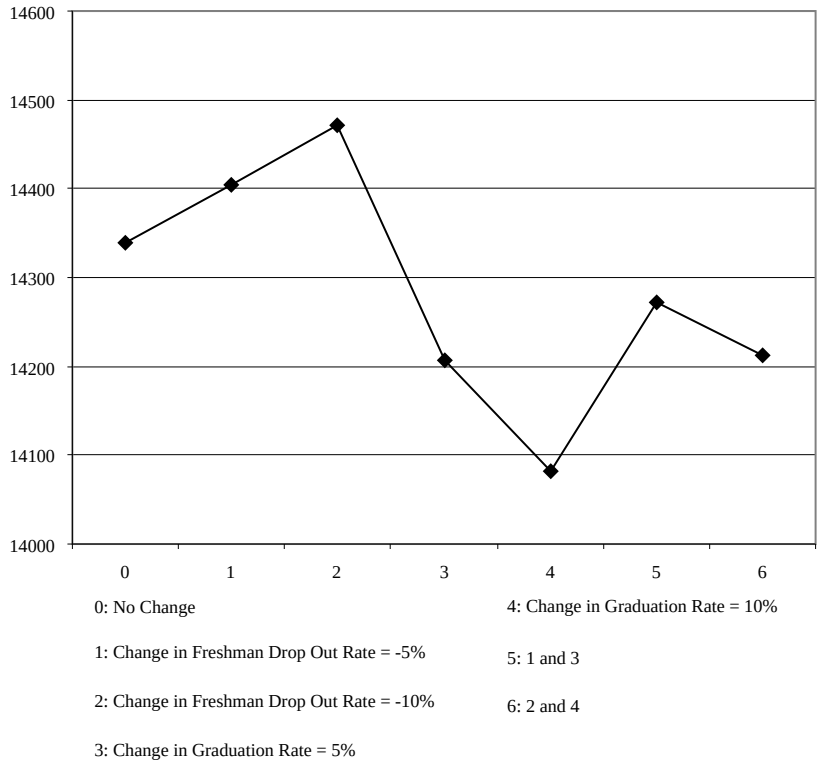


Figure 2.16: Equilibrium total population by varying freshman drop-out and graduation rate assumptions.

population on the desired path. We return to our simple example from Table 2.8 to illustrate the computation necessary to derive this input. First we find the equilibrium population toward which the current system is moving. Note that the row sum of row zero of the transition counts matrix is 3,600, which indicates that 3,600 new students enroll each year. Furthermore, the zero row gives us Δ =[3,060, 360, 180, 0], which is the distribution of the new students. Using (2.5), the given Δ, and P from Section 2.3.4, we find that the equilibrium population is $\Pi \approx$[3,400, 3,224, 3,269, 3263]. Next we adjust P to reflect the five percent increase in graduation rates using the formula $\hat{P}_{ij}=\left(P_{ij}/\sum_{i=1}^{4} P_{ij}\right)(1 - 1.05P_{i5})$. The new transition matrix is given as

$$\hat{P} = \begin{bmatrix} 0.0995 & 0.696 & 0.0995 & 0 \\ 0 & 0.149 & 0.6465 & 0.0995 \\ 0 & 0 & 0.1975 & 0.5925 \\ 0 & 0 & 0 & 0.265 \end{bmatrix}.$$

To find the number of new students needed each year for the population to continue toward the given equilibrium population, we solve $\Pi = \hat{\Delta}(I - \hat{P})^{-1}$ for $\hat{\Delta}$, which yields $\hat{\Delta}$ = [3,061, 377, 201, 141]. Thus, the number of new students needed each year is approximately 3,780. Notice that $\hat{\Delta}$ has a different distribution of incoming students than Δ, including a number of new students entering as seniors. If the administration thinks that the distribution of incoming students will not change, but they still want to adjust admissions such that the total university population moves toward the previous equilibrium population, we can still use our model to derive the number of new students needed each year. We solve the equation $\Delta(I - P)^{-1}ONE = \alpha\Delta(I - \hat{P})^{-1}ONE$, where ONE is a column vector of ones, for α. We find that $\alpha \approx 1.023$ and, thus, $\hat{\Delta} = \alpha\Delta \approx$ [3,130, 368, 184, 0] indicating that the number of new students needed each year is approximately 3,682. The difference between the two proposed inputs is that the first will produce an equilibrium population with the same distribution across class levels as the initial target while the second input will produce an equilibrium population with the same total population, but a different distribution across class levels.

In practice, we use the latter approach because few, if any, new students enter as seniors and the distribution of incoming students is difficult to control. From Table 2.13 we see that

the equilibrium population toward which Clemson is currently moving is 14,339. If freshman drop-out rates decrease by ten percent and graduation rates increase by ten percent, the total equilibrium population decreases to 14,212. Using the second approach to find the input necessary for specific populations, we find that to balance these changes the number of first-time and transfer students must increase from 2,800 and 800 to approximately 2,828 and 808, respectively. For this case $\alpha \approx 1.0099$.

We can also use our model to find the average time students are in the system and the probability that they reach a certain state. Specifically, we can find the probability and average time for a student to graduate by representing the graduating state as an absorbing state. Again we use our example to illustrate this procedure assuming that the leaving state represents graduating. To find the probability, u, of graduating beginning at any of the internal states, we must solve $(I - P)u = b$ for u, where b is the transition submatrix representing transitions from the internal states to the graduating state. In our case, $b' = [.1, .1, .2, .7]$. Because $(I - P)$ is invertible, it follows that $u = (I - P)^{-1}b$. For our example, $u' = [1, 1, 1, 1]$, as expected. Using the three-year average transition matrix from the 2004-2006 CU data, we find that the probability of graduation for a first-time student who enters the university as a first-semester freshman is .66.

To find the average time to graduation, m, we solve $m = ONE + Pm$, where ONE is a column vector of ones. Because $(I - P)$ is invertible, it follows that $m = (I - P)^{-1}ONE$. For our example, $m' \approx [3.8, 3.1, 2.3, 1.4]$. The first entry of m indicates that if students start as freshmen, they spend an average of 3.8 years in the system. We can use this procedure to find the average time that a student spends at Clemson. Using the same data as above, we find that on average a first-time student who enters the university as a first-semester freshman stays for 3.76 years. This includes students who drop-out as well as graduate. The average time until graduation for a first-time student who enters the university as a first-semester freshman is 4.10 years.

In summary, the error rates for the projections presented for the various subgroups of the population are representative of the results for all the categories into which we split the population. The university population model we built can be used for projecting forward

in the short term of one to five years to examine how changes to the systems will ripple through the university. It can also be used in the long term to set targets for enrollment management when considering policy changes. The results indicate that the model yields accurate results in practice. Furthermore, the inherent properties of Markov chains allow us to use the model to find conditional probabilities of graduation, average time to graduation, and inputs necessary to achieve specific populations. The procedure we outlined can be used to produce similar Markov chain models with varied purposes and state spaces.

Chapter 3

Predicting Course Enrollment

Scheduling university courses first requires predicting the number of students expected to enroll in each course. The predictions guide the number and size of sections to be offered. Since course schedules are prepared and submitted well in advance of the start of the semester, predicting course enrollment is a problem faced by every department at every university. Models to assist with enrollment prediction can be extremely simple. If schedules are rolled over from a previous semester, we can simply use the course enrollments from the previous semester as the prediction. At the other extreme, course enrollment models can predict for each individual student, resulting in great detail. In this chapter we present our procedure for creating accurate, robust models that predict course enrollment on an aggregate level. Although this work has wide applicability to other colleges and universities, the following discussion refers specifically to Clemson University (CU).

At CU, schedules for fall semester are due in February, a time when admissions estimates, course pass rates, and student retention are uncertain. The schedules are published in March, early registration for continuing students begins in April, spring grades are recorded in May, and freshmen and transfer registration begins in mid-June. Adjusting the number of instructors or adding and canceling sections after June is not desirable, and may not be possible. Making accurate predictions at the time the schedules are constructed, and adjusting these predictions as soon as changes become apparent, are extremely important. Another need for detailed course prediction models is to identify and predict the number of enrolled students in distinct groups, such as first-time and transfer students. Because many lower level courses have space reserved for first-time students, this information could be used to reserve a more accurate number of course seats for the incoming class, whose enrollment is set by the admissions office. Furthermore, if the enrollment models were more detailed, first-time seats could be released based on the expected demand of the students attending each orientation session instead of uniformly over all sessions as is the current practice.

Although complete information relative to student enrollment in the following semester is not available at the time of prediction, available information includes course rolls and student information such as major and *enrollment status*, which is a code the university uses to identify each student as first-time, transfer, continuing, or returning. The admissions office also releases the estimates of the number of new first-time and transfer students expected to enroll in the fall. This information, along with analysis of historical data, makes it possible to construct generic course prediction models that are robust and accurate.

A challenge that all *service departments*—those departments that teach courses for students majoring in other disciplines—face is the uncertainty of the admissions estimates of first-time and transfer student enrollment. At CU these numbers have differed from their targets by more than 300 students in recent years. These variations appear after the schedule is set in the spring or during freshmen orientation when students are already registering. If admissions estimates are high and therefore more sections than necessary have been scheduled, department and university resources are not used efficiently. If the estimates are low, it is difficult to accommodate the unexpected students because too few sections have been scheduled. Therefore, predicting enrollment using a range of possible university enrollment numbers is desirable to ensure adequate capacity in each section.

The future total university population is not known at the time of course enrollment estimation because schedules must be submitted early in the previous semester. Even the number of students who will be continuing from the present semester must be estimated because students graduate, transfer, or leave for other reasons, thus changing enrollment numbers. Since course enrollment estimates depend on university population estimates, our models must determine these estimates to ensure accurate course enrollment predictions.

The goal of our course prediction models is to make accurate predictions early in the scheduling process and to update predictions as new information becomes available. Similar to our university prediction model discussed in Chapter 2, our criteria for accurate predictions is based on the accuracy of the exogenous variables. In general, our goal is to predict course enrollments such that error rates are less than 5.3%, which is the weighted sum of the average first-time and average transfer error rates as given in Table 2.1.

The remainder of this chapter is structured as follows. We review the literature on predicting course enrollment in Section 3.1. In Section 3.2 we describe the data used in studying course prediction at CU. We discuss our course prediction procedure and models in Section 3.3 in the context of a lower-level math course with large enrollment at CU. We present results for course prediction models at CU in Section 3.4.

3.1 Related Work

Most of the research published on enrollment modeling focuses on the entire student body. There is little published work that specifically examines course enrollment prediction. Hopkins and Massy [1981] claim that the same techniques used for predicting on the aggregate level can be used at the department and course level, but they do not develop the necessary models. We briefly review their university population models here as a precursor to other more specific models; a full description was given in Chapter 2. Hopkins and Massy [1981] present three categories of flow models used in university enrollment planning: the grade progression ratio (GPR) method, Markov chain models, and cohort flow models. The GPR method and the Markov chain models use cross-sectional data and historical yields to predict the university population. The cohort flow models use longitudinal data to track students through the university and use historical yields for predictive purposes. They do not develop or test these models on course enrollment.

Balachandran and Gerwin [1973] present three variable-work models for predicting course enrollments: the *work model*, the *eligible-work model accounting for prerequisites*, and the *eligible work model with program requirements*. The authors claim that all three models can predict course enrollments semesters ahead using estimates for certain probabilities, forecasts of new students each semester, and the initial distribution of students [Balachandran and Gerwin 1973]. The *work model* uses the conditional probability that a student will take a course, given that he has not already, to predict how many students will enroll in that course. The *eligible-work model accounting for prerequisites* pares down the number of students eligible to take a course by accounting for prerequisites and then uses the conditional probability that they will take the course for predictive purposes. The

eligible work model with program requirements categorizes eligible and ineligible students based on how much work they have completed in progress toward a degree. Conditional probabilities are used for each group to predict the total number of students who will enroll in the course in question. In testing of the models, the parameters—the conditional probabilities—were estimated using the average rates from one year, 1971. Note that each model is more detailed that the last and thus requires more data for parameter estimation. Balachandran and Gerwin [1973] test the work and eligible-work models on graduate courses. Because the enrollments are so low, the error rates are often high and the effectiveness of the models is not clear because of the minimal amount of testing. The authors concede that the models are not a replacement for intuition, but instead a complement [Balachandran and Gerwin 1973].

The course prediction models currently used in the Mathematical Sciences Department at Clemson University are similar to the Markov Chain models described by Hopkins and Massy [1981], but focus on conditional probabilities. The models use historical proportions to estimate the number of students in one specific group who will enroll in the course; we call this the primary estimate. Then the model backs into the total course enrollment estimate using historical proportions and the primary estimate. For example, as shown in Figure 3.1, in fall 2003 there were 3,241 freshmen enrolled at Clemson, 740 freshmen were enrolled in Calculus I (MATH 106), and 842 total students were enrolled in MATH 106. That is, 22.83% of the freshmen were enrolled in MATH 106 and 87.89% of the students in MATH 106 were freshmen. An estimated 3,450 freshmen were expected to enroll in fall 2004. To predict the number of students in MATH 106, we multiply the number of expected freshmen by 22.83% to get 788. This is the number of freshmen we expect to enroll in MATH 106 in fall 2004, the primary estimate. To find the total course enrollment estimate, we multiply the primary estimate, 788, by 1/87.89% to get 896 students. Figure 3.1 shows this example in detail. The schema under the fall 2003 heading shows the information available at the time of fall 2004 enrollment estimation. The percentages shown on the edges are multiplied by the number at the tail of the edge to arrive at the number at the head of the edge. The schema under the fall 2004heading shows the flow of estimations using

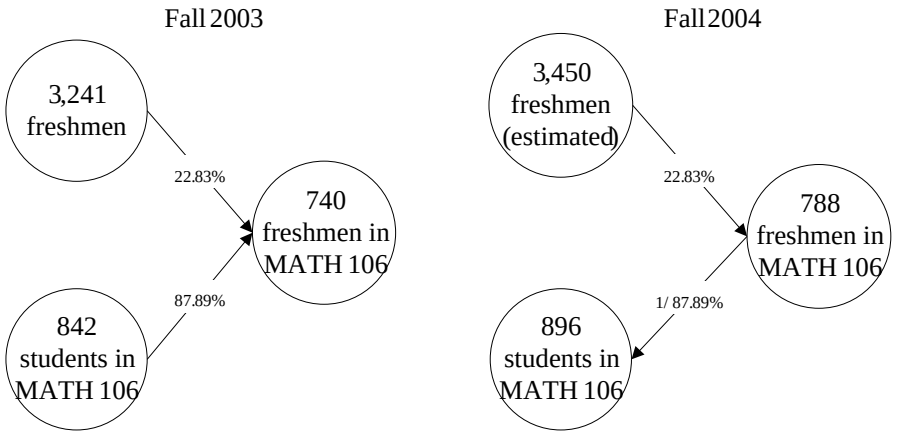


Figure 3.1: Partial diagram of Clemson University Mathematical Sciences department's current model for predicting Calculus I enrollment.

the data from fall 2003. Notice that the direction of the second edge is reversed in the fall 2004 schema to depict that the model backs into the final estimate.

Because the literature involving course enrollment models is limited, more work in this area is warranted especially in creating procedures for robust, detailed course prediction models and in describing case studies using these models.

3.2 Data

We use three sets of data to create the course enrollment models. The first is the estimated new student populations, including first-time and transfer students. This information, which is usually a targeted number set by the admissions office, is not derived from historical data. These estimates may be volatile and often change before the semester begins. For this reason, we may want to use a range of estimates centered on the targeted numbers.

The second set of data is the current course enrollments. From these we obtain the number and characteristics of the students currently enrolled in each course, its prerequisites, and the rest of the university. Some of the variables available to us for each student through the enrollment data are major, SAT score, ACT score, class, gender, race, age, course load, quantity of semester hours earned this term, and quantity of credit hours earned to date. Because we do not know the course pass and fail rates for the current semester courses at the time of prediction, we must estimate these. We update our model after the grades are

recorded at the end of the current semester to reflect the actual pass and fail rates and then adjust our predictions at that time.

The third and largest set of data needed for the model is historical enrollment data. There are 12,000 to 14,000 undergraduates enrolled at CU in any fall or spring semester and 4,000 to 6,000 students enrolled in courses taught by the Mathematical Sciences department each semester. From the current enrollment data and the historical enrollment data, we are able to determine which student characteristics and enrollments are significant predictors for modeling course enrollment. Because course descriptions, course pass rates, and program curricula change, we use just a few preceding years (typically three) to estimate parameters for the current year in light of these qualitative changes. Examples from CU of qualitative changes affecting course enrollment are the fall 2005 decrease in the required number of hours for graduation—from approximately 135 hours to just over 120 hours—and the decrease in the number of required math courses for some programs from two to one.

Thus far, we have focused on models used for predicting fall enrollment. However, the models we develop can be used for both semesters. The differences are slight and primarily include differing sizes of student populations. The fall semester has a large new student population with many first-time students, and course enrollment depends heavily on such admissions. New student populations are much smaller in the spring. At that time, course enrollment predictions rely more heavily on students in prerequisite courses and the population of continuing students. In developing generic models adaptable for both the fall and spring, we typically include the significant groupings needed for both semesters.

3.3 Course Enrollment Prediction Procedure

Most of the course enrollment models currently used in the Department of Mathematical Sciences at Clemson University predict future populations using the previous semester's enrollments and an appropriate conditional probability model. There are exceptions where regression on historical aggregate data provides the model. Regression is often not appropriate because of the changing curriculum requirements on the number of hours required to

graduate and the number of math courses required, as well as fluctuating fail and withdrawal rates. Using multivariate or logistic regression on individual students results in extremely high variability. In order to isolate these variations, we split students into groups with similar enrollment rates and identify groups whose rates have been fluctuating. In this study, we take advantage of extensive data that includes detailed information on student characteristics and course enrollments. The results of this study are applicable to other departments, universities, and institutions in need of enrollment prediction models.

We aggregate students into groups instead of predicting on an individual basis because there is higher variability in the latter approach. Furthermore, we ensure that the groups are well populated. Some attributes relevant to course enrollment, such as major, divide the population into a large number of groups with few students in each. These fine divisions again have too much variability to be useful predictors. We instead use broader categories to determine the significant student groupings. We use goodness-of-fit tests and past experience of enrollment behavior as our guides in determining which groups to use.

Figure 3.2 shows the groupings of a single variable Calculus I (MATH 106) model, where continuing students and returning students are denoted "Cont" and "Return", respectively. The figure shows a decision tree that yields final groupings of students for predicting course enrollment. General engineering majors (GE Major), other students whose majors require MATH 106 (106 Group), and all the rest of the first-time students (Other) are the final groups into which first-time students are divided for the MATH 106 model. From the final groupings we record the conditional probabilities of students in the group enrolling in the course from historical data. Then we use this information to aid in estimating the conditional probabilities for the coming year. We multiply these proportions by the estimated university enrollment of the respective groups to determine total course enrollment. The number of nodes in the diagram is course dependent, and varies depending on the behavior of students enrolling in the course to be modeled. More parsimonious models may be used in the case of limited data. For example, a simpler model may only include the nodes extending two branches from the root of the tree—using parameters for first-time, transfer, and continuing student characteristics.

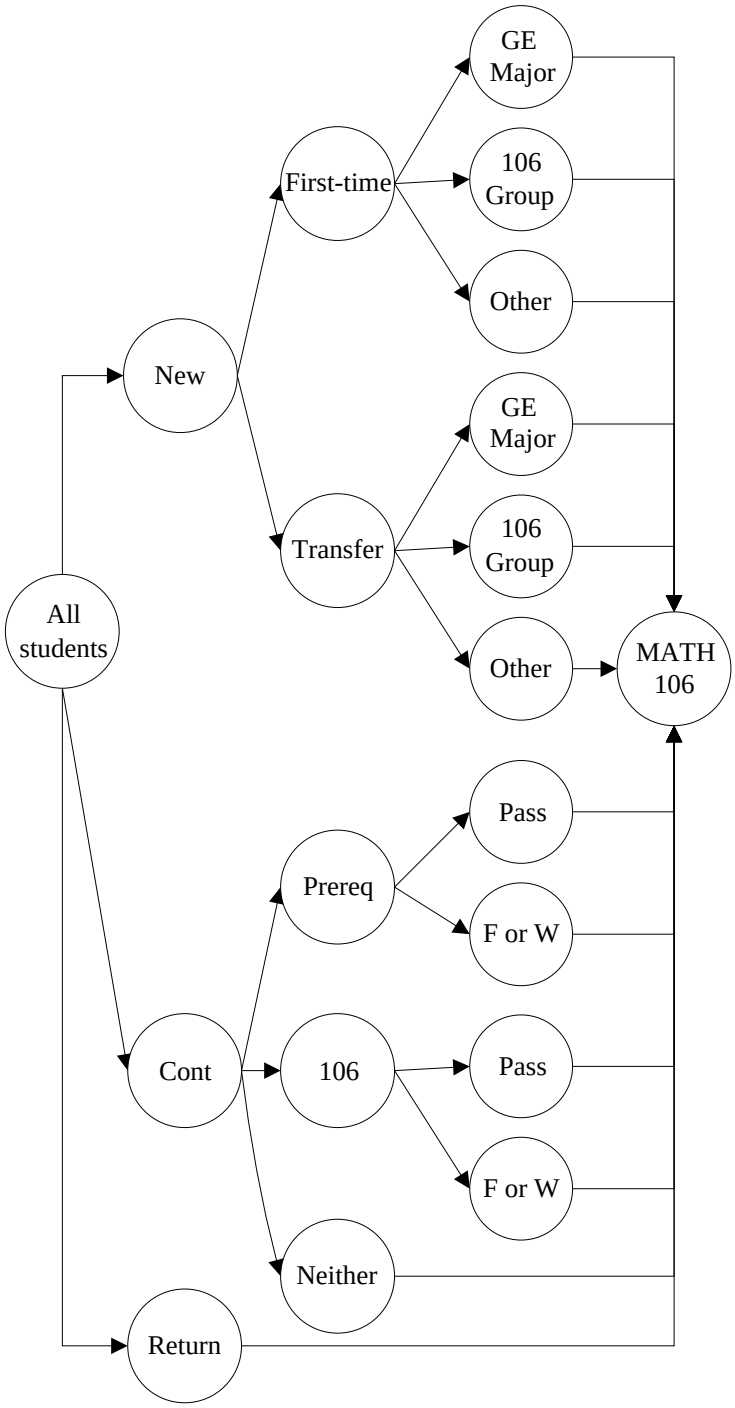


Figure 3.2: Template for course enrollment model. *This diagram shows the template for the course enrollment model using MATH 106 as the guide. Nodes could be added or removed depending on the detail of the model.*

The procedure we develop for creating course prediction models has four major steps outlined in Algorithm 3.1. Each step is presented individually in the next four subsections.

Algorithm 3.1: Procedure for predicting course enrollment.

1. Identification of significant factors.
2. Parameter estimation and verification.
3. Historical verification.
4. Model usage.

3.3.1 Identification of Significant Factors

Identifying student characteristics that are significant in predicting course enrollment is the first step in creating course enrollment prediction models. We do this by testing for correlation between the grouping variable and enrollment in the course we are modeling. Specifically, we use a χ^2 test for independence. With this test we are able to determine if the outcome—enrollment in the course we are predicting—and the variable—the student characteristic we are testing—are correlated or independent.

To keep the model simple, we only use the most significant factors in predicting course enrollment and exclude information that increases the variability of our model's predictions. For example, we have the major codes for each student and could use these individually to estimate rates of enrollment. Because there are over 100 majors, many such groups are small and prone to high variability in their enrollment behavior. Thus, instead of using major codes individually to group students, we group students with similar enrollment behavior. For the MATH 106 model, we group students as general engineering majors, majors required to take MATH 106, or all other majors. We do not include general engineering majors with all the other majors requiring MATH 106 because the admissions office provides an estimate of first-time general engineering majors that is not derived from historical data, but set by the university. We use this target for predicting enrollment in courses highly populated by first-time general engineering students. Furthermore, this MATH 106 grouping yields better estimates than grouping all the first-time students together because first-time

		Conditional Enrollment Rates	
		% in Group	% in MATH 106
New Students		Fall 2003	Fall 2003
First-time	GE Majors	25%	55%
	106 Group	29%	29%
	Everyone Else	46%	8%
Transfer	GE Majors	10%	22%
	106 Group	24%	11%
	Everyone Else	66%	2%
Continuing Students		Spring 2003	Spring 2003
In Prerequisite (Previous Term)	Pass	61%	49%
	Fail or Withdraw	39%	14%
In Course, GE Major (Previous Term)	A, B, C	44%	0%
	D, F or Withdraw	56%	16%
In Course, 106 Group (Previous Term)	Pass	58%	5%
	Fail or Withdraw	42%	20%
In Course, Other Major (Previous Term)	Pass	58%	0%
	Fail or Withdraw	42%	0%
Everyone Else (Previous Term)	Sem 1, 2, 3	12%	1%
	Sem ≥ 4	88%	0%
Other Students		Fall 2003	Fall 2003
All Students	Other Students	2%	1%

Table 3.1: University and MATH 106 enrollment rates for designated groups (fall 2003).

general engineering majors enroll in MATH 106 at a significantly higher rate than all other first-time and continuing students.

Many model changes are made while determining which grouping variables to include in the model. We want groupings that make sense and show promise empirically to be good predictors. The χ^2 tests guide us in model creation. However, there are instances where we use grouping variables despite a test result indicating independence relative to the grouping. Because we typically want the same model for both the spring and fall semesters, and because certain groupings may be significant in only the fall semesters or only the spring semesters, we also use past experience and empirical results to guide us. For example, the number of new first-time students is high in the fall and very low in the spring. Thus, first-time groupings found to be significant in the fall may not be significant in the spring because the cell counts in the spring χ^2 tests are too low to yield conclusive results. In these

cases, past experience and empirical results showing that groupings yield accurate predictions aid in our modeling decisions.

Table 3.1 shows the final groupings for the MATH 106 model. Note that the conditional enrollment rates for new students are given for fall 2003 while the rates for continuing students are given for spring 2003. This is because, for the fall prediction models, new students first enter the university in the fall and we therefore use the conditional probabilities from that time. However, in the current semester when we predict for the following semester, the number of current students who will continue from the spring to the fall is unknown. Thus, the conditional probability for current students is the probability that they will return the following semester and enroll in the specific course, given that they are currently enrolled.

3.3.2 Parameter Estimation and Verification

After determining groups whose course enrollment rates are significantly different than the rest of the population, our second step is to estimate the parameters of the model. We base our initial prototype model solely on data from one year by recording university and mathematical sciences course enrollment rates for the identified groupings. When we use the model in practice, we first estimate the number of students in each of the groups and then predict the number of students who will enroll in the course from each of the groups using conditional probabilities estimated from historical data. In our prototype model, we gather both the university and course enrollment rates from each of the final groupings for one year. Table 3.1 also gives the MATH 106 model parameters derived from 2003 data. Note that the university and course enrollment rates are conditional on being in the group on the left. For example 25% of the first-time students are general engineering (GE) majors and 55% of the first-time GE majors enroll in MATH 106.

We test the accuracy of model predictions and the robustness of the model by using the parameter values estimated from a single year to predict course enrollment for other years whose results are known. For our example, we use the parameter estimates in Table 3.1 for MATH 106 and test the model on data from fall 2002 and fall 2004. After grouping the data for the previous semesters, we count the number of students in each group enrolled in the university. For initial testing, we use the actual numbers of students in each group to

Semester	Fall 2002	Fall 2004
Actual Enrollment	792	937
Predicted Enrollment	802	915
Error	10	-22
% Error	1%	-2%

Table 3.2: Preliminary test results for MATH 106 model with parameters derived from fall 2003 data.

test the course prediction rates. Recall that in the final model we must estimate the size of these groups since complete university enrollment information is not available at the time course enrollments are predicted. We multiply the parameter values by the counts to get the predicted number of students in the course for that semester. Table 3.2 compares the predicted and observed values. Note that the error rates are well within the 5.3% error rate criteria we set based on the error rates of exogenous variables.

We also perform goodness-of-fit tests to ensure that the groupings significant in the fall and spring of the given year are also significant for the other years. After modification of the variables, we find the final groupings that are significant relative to course enrollment.

3.3.3 Historical Verification

The third step of the creating course prediction models is verifying the model with historical data. The final model uses university and course enrollment rates from several years to determine parameter estimates. We typically use data from the previous three years. We do not include less recent data because of possible changes in courses, retention, and university population. Such qualitative changes undermine the applicability of data that is older than a few years.

The probability distribution of course enrollment rates is unknown. To understand better the amount of variation in the rates, we calculate the minimum, maximum, and average historical rates for university and course enrollment over the range of years we have chosen. Table 3.3 shows the rates of new students in MATH 106 for the fall semesters of 2002, 2003, and 2004. There are noticeable differences between the rates of enrollment of GE majors, the 106 group, and the rest of the population. Notice that the percentage of transfer students who are GE majors and enrolled in MATH 106 changed significantly each of the three

	Enrollment Rates			Variation over Fall 2002 to Fall 2004		
	Fall 2002	Fall 2003	Fall 2004	Minimum	Maximum	Average
First-time						
GE Major	56%	55%	57%	55%	57%	56%
106 Group	31%	29%	30%	29%	31%	30%
Other	7%	8%	8%	7%	8%	8%
Transfer						
GE Major	8%	22%	32%	8%	32%	21%
106 Group	9%	11%	8%	8%	11%	9%
Other	1%	2%	2%	1%	2%	2%

Table 3.3: Course enrollment rates for new students in MATH 106.

years recorded, while the percentage of first-time GE majors enrolled in MATH 106 stayed relatively constant. Grouping students and observing rates over multiple years allows us to identify these differences.

To complete the historical verification step, we test the model using both the university and course enrollment rates derived from the recent data. We test the model on past data as it would have been used for predicting future enrollments. Thus we use only the information that we would have known at the time the schedule was submitted. We estimate university enrollment for each group in our model that we can not count or obtain at the time of prediction. For the MATH 106 model, we count from the historical data the numbers of first-time, transfer, and continuing students in their respective groupings, such as those who are in MATH 106 and are GE majors. We then estimate the number of students in the final groupings by multiplying the average university enrollment rates derived from historical data by these counts. In the MATH 106 model we estimate the number of students who pass the prerequisite by multiplying the number of students in the prerequisite by the average pass rate. Finally, we predict the course enrollment from each group by multiplying the average course enrollment rates with the group estimates. Summing the number of students expected to take the course in each group gives us our total prediction.

The MATH 106 model predictions shown in Table 3.4 were within one section of the actual enrollment for each year we tested, with error between 10 and 19 students. If the error for a model is large at this point in the procedure, the model should be changed before continuing to the prediction step. High error may be a sign that the groups in the model do

Semester	Fall 2002	Fall 2003	Fall 2004
Actual Enrollment	792	860	937
Predicted Enrollment	808	870	918
Error	16	10	-19
% Error	2%	1%	-2%

Table 3.4: Final test results for MATH 106 model.

not capture the significant groups of students who enroll in the course. High error rates may also indicate that the course has changed over the years of data used and a shifting mix of students is enrolling in the course.

3.3.4 Model Usage

The final step of the procedure is using the model for course enrollment prediction. In predicting future enrollment we follow the procedure used for final testing, except that we do not automatically use the average rate from the historical data for predictive purposes. Instead, we observe the minimum, maximum, and average historical rates multiplied by the current populations as well as any previous trend that is apparent in the absolute enrollments. We make our estimate for each group based on these factors. Table 3.5 shows the university and course predictions for first-time GE students in MATH 106 for fall 2005. The numbers that appear below the term date are the past enrollments and enrollment rates of first-time GE students in the university and in MATH 106, respectively. The numbers in the fourth through sixth columns of the university enrollment row are the past minimum, maximum, and average rates of enrollment multiplied by the admissions estimate of 2,800 new first-time students. Because a target is set for the number of first-time GE students, we use that estimate of 725 rather than the average of the previous years. However, the percentage of first-time GE students who enroll in MATH 106 has been relatively constant over the past three years. Our estimate for first-time GE MATH 106 enrollment reflects that trend by choosing a rate similar to the historical rates.

We make estimates using the same method for each group in the course enrollment model. The sum of these estimates yields the total prediction for the course. We can also find the sum of the minimum, maximum, and average estimates to give perspective to the

First-time Students (Target of 2800 for Fall 2005)							
	Enrollment Rates			Variation			Fall 2005
	Fall 2002	Fall 2003	Fall 2004	Min	Max	Average	Estimate
University Enrollment	660	696	734	680	744	710	725
(GE Major)	27%	25%	24%	24%	27%	25%	26%
MATH 106 Enrollment	369	381	421	397	416	406	410
(GE Major)	56%	55%	57%	55%	57%	56%	57%

Table 3.5: Model for University and MATH 106 enrollment of first-time general engineering students.

number we ultimately decide to use. To calculate the variance of our estimate, we collapse the decision tree for our model groupings to find the probability that a student will enroll in the course, given no information about which group the student is in. We treat whether or not a student enrolls in the course as a Bernoulli random variable. As such, we can estimate the variance and construct confidence intervals for our prediction.

Let X be the probability that a student in a given group enrolls in a given course. Let Y be the number of students in the group who will enroll in the course. Y is a binomial random variable with parameters n and X, where n is the number of students in the group and X is the binomial proportion. Thus, Y is distributed as $B(n, X)$. The expectation of Y is given by

$$\begin{aligned} E[Y] &= \int_0^1 E[Y|X = x]P[X = x]dx \\ &= \int_0^1 nxP[X = x]dx \\ &= n\int_0^1 xP[X = x]dx \\ &= nE[X] \end{aligned} \tag{3.1}$$

which is equivalent to the calculations performed above to estimate the number of students expected to enroll in the course each session. From [Ross 2002] we have that

$$Var[Y] = E[Var(Y|X)] + Var[E(Y|X)] \tag{3.2}$$

Prediction	926
Observed	889
Error	37
% Error	4%
80% Confidence Interval	[889, 963]
95% Confidence Interval	[870, 982]

Table 3.6: MATH 106 prediction and result for fall 2005.

Therefore, the variance of Y is given by

$$\begin{aligned} Var[Y] &= E[Var(Y|X)] + Var[E(Y|X)] \\ &= E[nX(1-X)] + Var[nX] \\ &= E[nX] - E[nX^2] + n^2 Var[X] \\ &= n(E[X] - E[X^2]) + n^2 Var[X] \\ &= n(E[X] - E[X^2] + E[X]^2 - E[X^2]) + n^2 Var[X] \\ &= n(E[X] - Var[X] - E[X^2]) + n^2 Var[X] \\ &= n(E[X](n-1)Var[X] - E[X^2]) \end{aligned} \tag{3.3}$$

The prediction we made for fall 2005 MATH 106 enrollment is given in Table 3.6 along with the 80% and 95% confidence intervals. The actual enrollment of 889 is 37 students—about four percent or one section—less than our initial prediction. The actual enrollment is captured by both the 80% and 95% confidence intervals, and is equal to the lower bound on the 80% confidence interval. The error is within the 5.3% error rate criteria. We used the course prediction model on three other large, primarily freshman courses and the error rates of our predictions for those courses were also low. In the next section we give further results for other courses and years.

3.4 Results

Four of the largest enrollment courses in the Department of Mathematical Sciences at Clemson University are MATH 101, 102, 106, and 108, titled Essential Mathematics, Introduction to Mathematical Analysis, Calculus of a Single Variable I, and Calculus of a Single

Course	Early (March)	Mid (May)	Late (July)	Final (August)	Actual
101	584 (-11%)	684 (5%)	677 (4%)	657 (1%)	653
102	734 (-11%)	734 (-11%)	756 (-8%)	804 (-2%)	823
106	926 (4%)	951 (7%)	934 (5%)	921 (4%)	889
108	380 (3%)	380 (3%)	394 (7%)	366 (-1%)	369

Table 3.7: Fall 2005 predicted and actual enrollment and percentage error for four times during the planning period.

Variable II, respectively. Tables 3.7 and 3.8 show the predictions made for each of these courses for fall 2005 and 2006 using models built from the procedure just described. The tables show predictions made at four different times during the planning period: early in February or March, mid May or in June after early registration and spring grades have been recorded, late in July after new student registrations have finished, and finally in August just before courses begin. Note that in the majority of cases the predictions are more accurate as more information becomes available. All of the predictions have percentage errors near our 5.3% criteria except for the early prediction of MATH 101 enrollment for fall of 2006, and to a lesser degree the early prediction of MATH 101 for fall 2005. These errors are high for qualitative reasons—there was a change in the curriculum beginning in fall of 2005 that continued to show itself in the continuing student enrollment behavior in the fall of 2006. Some qualitative changes can be addressed by investigating the groups of students affected by the change and adjusting conditional enrollment rates accordingly. For example, prior to 2005 education majors were not required to earn credit in MATH 101. If this information had been available to us at the time of prediction, we could have allowed for this change in our prediction. It became apparent after the early registration period and we were able to adjust our estimates.

In summary, we have developed a procedure to build models for predicting course enrollment that uses student characteristics as predictors of enrollment behavior. We use the models on an aggregate level to predict the total number of students enrolling in a course. These models are sufficiently detailed and robust to predict courses with low enrollment

Course	Early (Feb.)	Mid (June)	Late (July)	Final (August)	Actual
101	676 (23%)	605 (10%)	552 (1%)	566 (3%)	548
102	804 (8%)	754 (1%)	751 (1%)	756 (2%)	744
106	919 (9%)	896 (6%)	883 (5%)	870 (3%)	844
108	388 (-9%)	379 (-11%)	390 (-8%)	405 (-5%)	426

Table 3.8: Fall 2006 predicted and actual enrollment and percentage error for four times during the planning period.

or to predict for specific groups of students such as those attending a single summer orientation. The next chapter will use the course prediction models as a starting point for developing seat allocation and release systems for summer registration sessions.

Chapter 4

Seat Allocation and Release Systems

Similar to course scheduling, which requires prediction of the number of students expected to enroll in each course, allocating and releasing space in a given course to student groups requires predicting the number of students in each group expected to enroll in the course. This prediction is the first step in allocating seats to student groups. Reserving space and then releasing that space to specific groups as they register are the next steps. *Seat allocation* is reserving or holding back space in courses in the form of course seats and then releasing those seats at a later time to specific student groups. Seat allocation systems are applicable to colleges and universities that face the problems of predicting, reserving, and releasing seats for new students or other student groups. Although this work has wide applicability to other colleges and universities, the following discussion refers specifically to Clemson University.

Seat allocation systems give students equal access to the most desired course sections. Reserving seats for new students during continuing student registration ensures that continuing students do not take all the seats in the limited number of desirable sections, leaving fewer time choices for new students during summer registration. Releasing seats gradually during the orientation sessions gives students at each orientation session equitable enrollment choices. Seat allocation systems also hedge fall course predictions by partially filling each section over time rather than filling each section in sequence. This is important because, if during the orientations we realize that one more course section will be needed, we may open it knowing that there is room not only in the newly opened section that occurs at one given time, but there is also capacity in the remaining sections. If there was no seat release system, the most desirable sections would fill quickly, leaving students in remaining orientation sessions with fewer choices. Opening one new section may not fulfill the demand for the course because of time conflicts with other courses.

The problem we address first considers which courses would benefit from the use of a seat allocation system. We find that reserving and then releasing seats is appropriate for courses that have many sections as well as significant expected new student enrollment. We then address the problem of how to release seats to students during summer orientation sessions, at which time registration occurs. The release of course seats during the multiple orientation sessions depends on the fall enrollment predictions. We present a new model for seat allocation that addresses how to estimate seat needs each session, how to allocate seats for multiple course sections, and how to predict seat shortages and surpluses.

The remainder of this chapter is structured as follows. We review related work on seat allocation systems in Section 4.1. In Section 4.2 we present criteria for determining whether or not a course could benefit from a seat allocation system. In Section 4.3 we describe the data used in allocating seats. We discuss our allocation procedure and resulting system in Section 4.4. Finally, we present results for this system at CU in Section 4.5.

4.1 Related Work

To our knowledge, literature on seat allocation systems is nonexistent at the time of this writing, although the problem of reserving and releasing seats is common. The American Association of Collegiate Registrars and Admissions Officers (AACRAO) and the Association for Institutional Research have many publications detailing enrollment management systems, strategic planning, and student tracking, for example see [Glover and Krotseng 1993; Ewell 1995; Sevier 2000; Black 2004; Dooris et al. 2004; Rodgers and Zimar 1993]. However, we have found nothing detailing the process of reserving and releasing seats to groups of students. To give background for our work, we have surveyed a sample of registrars at universities in the United States. We also describe in detail the seat allocation system currently in use at CU.

Table 4.1 shows the seat allocation survey responses we collected from eleven American universities. We use the shorthand N/A to refer to not applicable. All of the universities reserve course seats or course sections in at least one course for new students. Most of the respondents said that they reserve seats or sections in basic mathematics and English

	Reserve seats /sections for new students (in ≥ 1 course)	Release seats to new students	Process for releasing seats
Duke University	Yes	N/A	New student registration prior to orientation using on line registration
Indiana University	Yes	Yes	Release during orientation sessions
Louisiana State University	Yes	Unknown	Unknown
Mississippi State University	Yes	Yes	Release during orientation sessions
North Carolina State University	Yes	No	Students are pre-registered for core classes
Ohio State University	Yes	Yes	Release during orientation sessions
University of Central Florida	Yes	Yes	Release during orientation sessions
University of Connecticut	Yes	Yes	Release during orientation sessions
University of Minnesota	Yes	Unknown	Unknown
University of Virginia	Yes	Yes	Release during orientation sessions
Vanderbilt University	Yes	No	N/A

Table 4.1: Seat allocation survey answers from several American universities.

courses populated primarily by new students. Many of the universities release these seats using some system during orientation sessions. Note that North Carolina State University does not release seats directly to students for registration. Instead they pre-register students for core courses, for which seats have been reserved. Duke University also uses a novel approach for new student registration. Their new students use an on-line registration system prior to orientation that assesses their interests and uses that assessment in conjunction with their placement scores to guide them in registering for courses. All of the other universities could potentially benefit from a seat allocation system that considers the demand for courses at each orientation session.

The Office of Registration Services at CU manage the seat release program, which gradually releases space during summer orientation sessions in courses that reserve seats for new

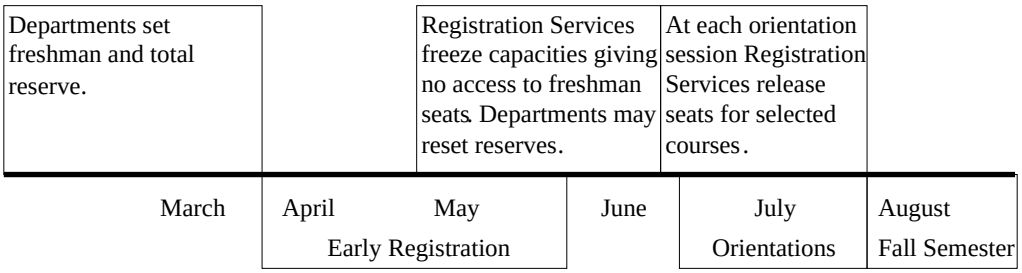


Figure 4.1: Registration time line at Clemson University. *This diagram shows a description of the activities involving registration over time at Clemson University.*

students. The system releases the reserved seats uniformly during the orientation sessions; that is, it releases $\frac{k}{n}$ of all the reserved seats during the k^{th} orientation session, where there are n sessions total. Figure 4.1 shows the time line for registration at Clemson University. The course schedule for the following fall is built prior to early registration, which begins in April. Building the schedule involves setting the number of seats to be reserved for new students. Thus, during early registration, continuing students do not have access to seats intended for freshmen. Departments have opportunities to reset these reserves until orientations start in mid-June. At this time, the Office of Registration Services uniformly releases seats to new students during orientation sessions.

4.2 Criteria

We consider several criteria in deciding whether or not to use a seat release system. These include the number of seats needed for new students, the number of course sections, and the number of seats to be released on average per section per orientation session. We assume that the seat allocation system will release seats just before each session. However, this assumption may be relaxed if there is a need for releasing at other times.

The first consideration in developing a seat allocation system is the number of reserved seats. If the number of reserved seats is low, then the seat need per section is also low. This implies that the system would reserve only a few seats per section, and then release a fraction of those each session. Thus, a minimum number of reserved seats is part of the

criteria in deciding whether or not to choose a seat allocation system. This minimum number may depend on the number of releases that are planned.

A second criterion is the number of sections of the course in question. If the number of sections is low, there are limited choices for class times, even assuming that the sections are scheduled at different times during the course week. Since one of the major reasons for using a seat allocation system is to give students at each registration session similar enrollment choices, few sections already hinder this goal. Thus, we assert that a minimum number of sections are required to motivate the use of a seat allocation system.

Another consideration for using a seat allocation system is the seat need per session per section for the course. For a seat allocation system to be worthwhile, seats must be released each session that are representative of the entire schedule for that course. Thus, on average, we would like to release at least one seat per section at every orientation session. This requirement, along with the number of sections and orientation sessions, gives a lower bound on the number of seats we must reserve in order for a release system to be applicable. Depending on the type of orientations held, this criterion may be split to accommodate the needs of the university. If there are fewer orientation sessions for transfer students than for first-time students, the minimum seat need per session per section may be less for the first-time student sessions than for the transfer sessions and yet represent a significant proportion of students enrolling in a course. For example, if the seat need per session per section for transfer sessions is one and the number of sections is 20, the total seat need is 40, assuming there are two transfer orientation sessions. Assuming a capacity of 35 per section, the total capacity is 700; thus, 40 students is fewer than six percent of the total. However, if there are nine first-time sessions and the seat need per session per section for those sessions is $\frac{3}{4}$ and the number of sections is 20, the total first-time seat need is 135, or about 19 percent of the total.

At CU, after reviewing the data on lower-level mathematics courses including the number of sections, average class size, new student enrollment in each course, and number of orientation sessions, we set the criteria levels. For background, note that the orientation schedule includes nine summer orientation sessions from mid-June to mid-July, a month-long break, and then two make-up sessions just prior to the start of the semester. The nine

		Student counts					Seat need per session per section		
Course (MATH)	sections	cont	first time	transfer	(new)	total	first time	transfer	(new)
101	22	179	339	69	408	587	1.93	1.05	1.69
102	29	179	498	96	594	773	2.15	1.10	1.86
106	27	98	829	48	877	975	3.84	0.59	2.95
108	11	152	185	20	205	357	2.10	0.61	1.69
117	6	37	82	32	114	151	1.71	1.78	1.73
118	3	76	1	5	6	82	0.04	0.56	0.18
119	3	63	7	4	11	74	0.29	0.44	0.33
203	9	206	17	20	37	243	0.24	0.74	0.37
206	13	345	89	36	125	470	0.86	0.92	0.87
207	12	243	44	35	79	322	0.46	0.97	0.60
208	6	182	0	9	9	191	0.00	0.50	0.14
301	8	234	28	16	44	278	0.44	0.67	0.50
302	4	105	0	5	5	110	0.00	0.42	0.11
309	9	296	6	10	16	312	0.08	0.37	0.16
311	4	121	2	0	2	123	0.06	0.00	0.05
TOTAL	179	2,674	2,754	451	3,205	5,879			

Table 4.2: Breakdown of mathematics course enrollment by enrollment status per section per orientation session, 2004.

summer orientation sessions include seven two-day sessions designed for first-time students and two one-day sessions for transfer students. The two make-up sessions are each one day; one is for first-time students and the other is for transfer students. In all, there are eight first-time sessions and three transfer sessions. We considered the overall seat need per session per section instead of breaking that number into transfer and first-time components because of the relatively few transfer sessions in CU's orientation schedule. We set the minimum number of sections criterion to eight because CU has fourteen regularly scheduled class times, and eight is more than half of that. Assuming that the sections are spread out in the schedule, eight sections ensure course time diversity. We use one as the minimum seat need per section per session. Thus, the minimum number of reserved seats is set to eighty-eight. These minimums are set as guides, and can be adjusted if appropriate.

Table 4.2 shows the breakdown of course enrollment for mathematical sciences courses in fall 2004 by enrollment status, which describes a student as continuing, transfer, or first-time. The last column of the table is the weighted average of the first-time and transfer seat need per respective session per section. The first four rows indicate the courses for which CU uses a seat allocation system. Note that the four courses that use the seat release

system, MATH 101, 102, 106, and 108, have more than eight sections and that the seat need per session per section is above the criteria set. Two courses that might benefit from a seat allocation system appear to be 117 and 206. Each satisfies two of the three criteria set. MATH 117 does not meet the minimum number of sections required, and MATH 206 does not meet the minimum seat need per session per section. Again, if appropriate, these criteria may need to be relaxed.

4.3 Data

The seat release system we present requires three sets of data. The first is produced by the model we created for predicting course enrollment. The model specifies the groupings that are significant and gives conditional enrollment rates for these groups. The second set of data needed is the current enrollments in the course by section. Note that when orientation begins, continuing students will have already had the opportunity to register for courses during early registration.

The third set of data we need is the number and characteristics of students attending each orientation session. All new students are required to attend an orientation session. Most do this during the summer, although there are two make-up orientation sessions held just prior to the start of the semester. Since the attendance at each orientation session is limited, students must pre-register. Although there are additions (walk-ins) and deletions (no-shows), these numbers are typically small. Hence, the number and characteristics of students attending an orientation is very accurate just prior to that orientation, but less so for future sessions because the orientation registration process may not be complete. Thus, after each orientation session we update the orientation and course enrollments to obtain current information on the change in orientation enrollments and the number of remaining course seats.

We group students attending each orientation session by their characteristics as we did in the course prediction model. The numbers of students in each group are significantly smaller than the numbers we encountered in the course prediction model since we are predicting for each of several orientation sessions.

We use data from students registered for every session, not just the current session. Although we have less confidence in the estimates for more distant sessions, the counts for the future sessions help us predict the overall number of seats we need and allow us to confirm or alter our previous predictions.

4.4 Seat Allocation Procedure

Ideally, our predictions made in the spring would be exactly realized, and we could use those predictions to reserve seats for new students. We could then set section capacities based on the seat need predictions for continuing students to allow them to register in the spring and could release seats uniformly to new students during summer orientation sessions. An example of an ideal system is given in Table 4.3. The total prediction for the course is 200 students—40 continuing and 160 new. During spring registration, we allocate seats uniformly for 40 students among the four sections-ten per section. Since our predictions are correct and the students register promptly (because they are ideal), all continuing students have taken their seats in the course by the first orientation session, and we have only to release seats for new students. In our ideal scenario, students attend orientation sessions uniformly and we therefore release seats in the same manner—eight per session per section. Table 4.3 shows the capacities of the sections and reflects the cumulative release of seats. In practice, section capacities usually are not this high and a seat release system may not be appropriate for this example because there are only four sections.

	Section Capacities			
Session	1	2	3	4
Spring	10	10	10	10
1	18	18	18	18
2	26	26	26	26
3	34	34	34	34
4	42	42	42	42
5	50	50	50	50

Table 4.3: Ideal course release, initially 10 and increasing 8 per session per section, for a course with four sections over five orientation sessions.

Obviously this ideal situation is not realistic. At least four sets of imbalances prevent the system from working so smoothly: (1) we cannot allocate the exact number of seats needed since we do not know that number; (2) the attendance at orientation sessions is uncertain and therefore must be estimated; (3) our seat release is not uniform across sections because enrollments are not uniform and room capacities differ; and (4) some sections are favored by students (e.g., not eight AM). Our seat release system shows how to overcome these imbalances to obtain accurate estimates of seat need and how to allocate seats to students. Figure 4.2 shows the schema for our seat allocation procedure and Algorithm 4.1 outlines the major steps of the procedure. Each step will be presented individually in the following subsections after the terminology used in the procedure is presented.

Algorithm 4.1: Procedure for allocating course seats.

1. Initialization.
2. Seat need estimation.
3. Capacity adjustment.
4. System usage.

4.4.1 Terminology

Course reserve is the number of seats reserved for a certain group, such as all, new, or continuing students. *Course capacity* is the number of seats currently available and is typically increased as orientations progress to make room for incoming students. The total capacity may not exceed the total reserve. The total reserve must be adjusted to allow for more capacity. *Seat release* refers to making available seats in a given course by changing section capacities.

4.4.2 Initialization

Preparation for summer orientation seat release begins in the spring prior to early registration. At this time, we use our course enrollment predictions for new and continuing students to set course reserves. We reserve a portion of seats in each section for new students and the remaining is reserved for continuing students. During early registration we set the section

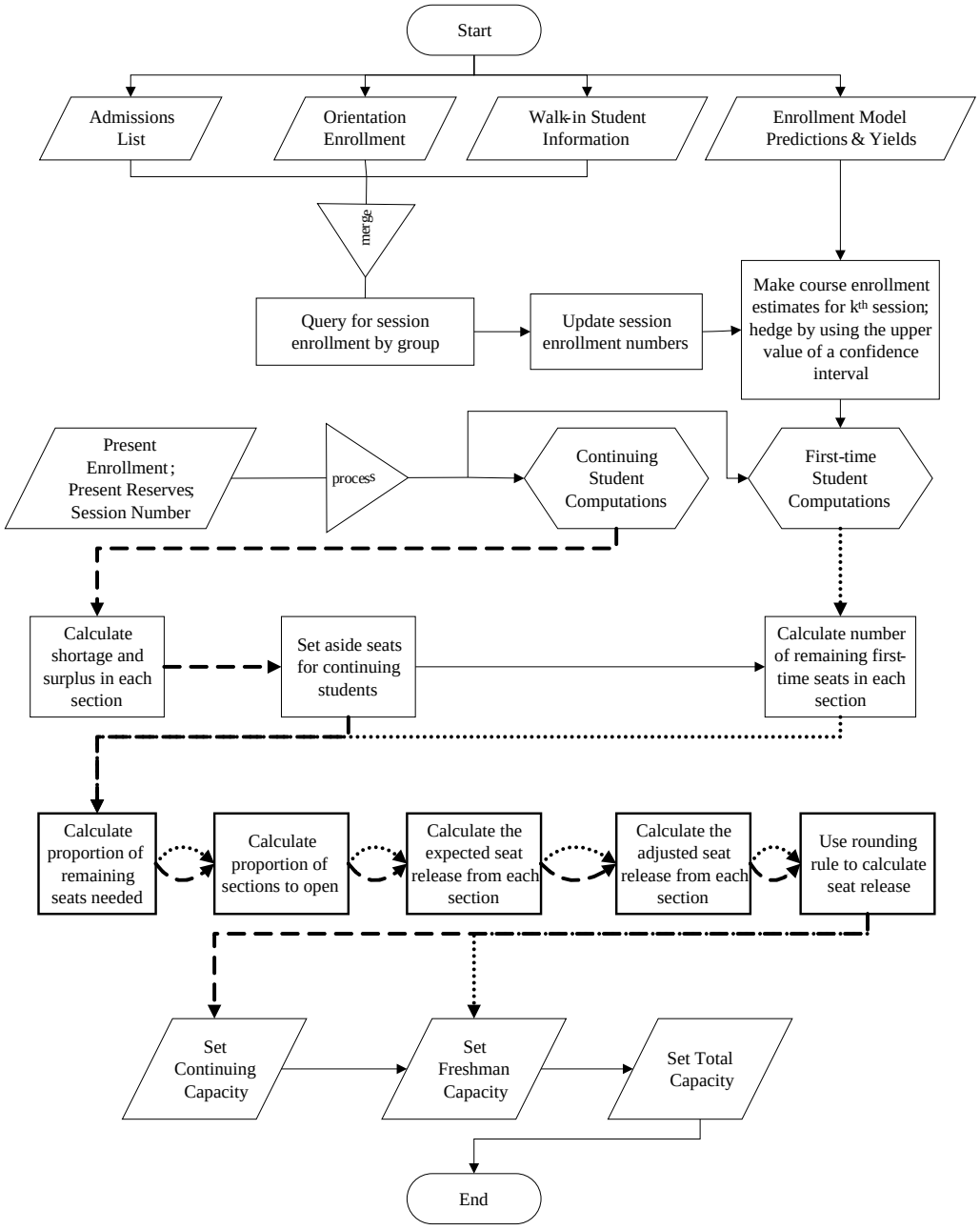


Figure 4.2: Schema of seat allocation procedure. *This diagram shows the data and procedures involved in releasing seats to students during summer orientation sessions at Clemson University.*

	Orientation Session					Total	
Student group	1	2	3	4	5	First-time	Transfer
GE Major	27	12	25	4	2	56	14
106 Group	40	16	34	18	6	92	22
Other	45	46	53	12	16	110	62
TOTAL	112	74	112	34	24	258	98

Table 4.4: Example of orientation enrollments prior to session 3 by group.

capacities to make available only enough seats to accommodate the continuing students. That is, we release all the continuing student seats at one time since they have access to registration at the same time. We only release seats for continuing students so that the residual capacity is held for new students who will register later in the summer.

Before orientation sessions begin, we update our course predictions using updated admissions information and pass rates that have now been recorded. Recall that early registration is prior to the end of the spring semester when grades are issued. We also observe and reset, if necessary, the section capacities based on continuing student registrations. If our continuing student prediction was too high, we may need to lower the continuing student reserves and the section capacities.

With the number and characteristics of students registered for each orientation session and the course enrollment model, we group students into the groupings detailed in the model. Table 4.4 shows an example of first-time and transfer student orientation enrollment using the MATH 106 model groupings in preparation for the third orientation session. Sessions two and five are transfer sessions. Note that we may have first-time and transfer students attending the same session, but for simplicity we will keep them separate in our example. Suppose there are three summer orientation sessions (shown in the first three columns) and there are two make-up sessions (shown in the fourth and fifth columns). Since in our example we are preparing for the seat release of the third orientation session, the registrations for the two make-up sessions—held much later in the summer—are low in comparison to the regular session enrollments because students have yet to register for an orientation session.

Using our predictions for total fall enrollment, we estimate the number of students, by group, who have not yet registered for an orientation session by calculating the difference

	Orientation Enrollment Predictions		Estimated Number of Students Yet to Enroll	
Student group	First-time	Transfer	First-time	Transfer
GE Major	73	23	17	9
106 Group	109	29	17	7
Other	145	94	35	32
TOTAL	327	146	69	48

Table 4.5: Example of orientation enrollment and expectations for future enrollments prior to session 3 by group.

between the group prediction and the sum of orientation registrants so far in that group. For our example, the expected number of first-time GE majors yet to register for orientation is 17, and is shown in Table 4.5. This estimate of 17 is the difference between 73—shown in Table 4.5 as the total prediction for first-time GE majors—and 56—shown in Table 4.4 as the total number of first-time GE majors already registered for orientation. As orientation sessions progress, the estimates of students who have yet to register for an orientation session will decrease. If they become negative, we have underestimated the number of students in that particular group and must make adjustments. If they remain large, we have overestimated and also need to adjust. Observing these totals will help us gauge the accuracy of our predictions and help us make adjustments as soon as the need for corrections becomes apparent.

At CU there is an extended period of time between the last summer orientation session and the make-up sessions. During that interim time, updates of the number of students expected for the make-up sessions can be made. In the interim period, some transfer students may forgo the orientation process and register for courses. We need to identify these students and exclude them from the total number that we expect to attend the late orientation sessions. As these students are identified, we subtract them from the estimates of students yet to register to keep an accurate count of the students that we still expect to register for courses.

	Orientation Session					Estimated Seat Need for Students Yet to Register for Orientation	
Student group	1	2	3	4	5	First-time	Transfer
GE Major	15	3	14	2	0	10	2
106 Group	12	2	10	5	1	5	1
Other	3	1	4	1	0	3	1
Total	30	6	28	8	1	18	3
95% CI	40	13	37	14	6	24	9

Table 4.6: Example of expected seat need by group by session.

4.4.3 Seat Need Estimation

Just before each orientation session, we update course and orientation enrollment numbers so that we have current information. Using the recorded counts of students attending each session, we estimate the number of course seats needed for each group in the current and future orientation sessions using the course enrollment model. We construct a confidence interval for our estimate using the same method used in the course enrollment model prediction. An example of these calculations is given in Table 4.6, where the last two columns give the expected seat need for students who have not yet registered for an orientation session. The last row in Table 4.6 gives the upper bound on the confidence interval. To hedge our estimate, we may want to release slightly more seats than we expect to need.

4.4.4 Capacity Adjustment

We make seats available by adjusting section capacities using our estimates of seat need and the current course and section enrollments. The course enrollments include both continuing and new students, and thus adjusting section capacities is the method for releasing seats to both groups.

Continuing students have had access to course seats since early registration, and many of the seats intended for them will be filled prior to new student orientation. Because at CU new students may take seats intended for continuing students if they are open, but continuing students only have access to seats set aside for them, we take any empty seats intended for continuing students back prior to the first orientation session and then gradually release

them over the summer. In this way, we are still giving continuing students opportunities to register and change sections over time, but we are also protecting their seats from being used by new students.

We use the estimates we have calculated to release seats for new students and use a proportion—such as session number over n, where n is the total number of sessions—to release the seats held for continuing students. Thus, at each orientation session, we reset the total capacity in each course section to allocate seats for new and continuing students.

Unlike our ideal example given in Table 4.3, almost certainly current section enrollments will not be uniform. Therefore, we increase the capacity in each section proportional to the number of remaining seats available. Since new and continuing students have differing numbers of seats intended for them based on the course model predictions, we calculate their releases separately and then adjust the section capacities using the two allocations. We use the estimated new student seat need divided by the total number of remaining seats for new students as the proportion of seats to release from the seats in each section remaining for new students. The expected release for new students, then, is simply their estimated seat need. Again, to hedge against a lack of available seats for new students, we may choose to use the upper bound on an appropriate confidence interval instead of the estimated seat need.

After each orientation, we update the course enrollment numbers. When we adjust the section capacities for the next release, we use these current enrollment numbers, not the capacities we previously set. That is, the allocation of seats is always based on current enrollment and predicted needs. Hence, over or under estimation of needs in a single session is not cumulative. All empty seats, whether there are more or fewer than we expected, are incorporated into future allocations through the enrollments at the time of release.

4.4.5 System Usage

The seat release system we have presented has three main components: estimation, allocation, and recovery. We estimate seat need for each session using the course prediction model; we allocate seats using proportions of remaining seats; and we recover from hedging or unforeseen demand by using current enrollments instead of prior predictions.

	Section	1	2	3	4	TOTAL
Initialize	New Reserve	48	32	40	40	160
	Continuing Reserve	12	8	10	10	40
	Total Reserve	60	40	50	50	200
Update each session	New Enrollment	21	11	19	14	65
	Continuing Enrollment	8	5	7	9	29
	Total Enrollment	29	16	26	23	94

Table 4.7: Example of course enrollment data needed each orientation session.

We extend the example from Tables 4.4, 4.5, and 4.6 to finish the illustration of the seat release system. Table 4.7 shows the course data needed for initializing the seat release system and the course data needed to be updated just before each session. The initialization includes recording the reserves for new and continuing students-shown as new reserve and continuing reserve, respectively in Table 4.7. The sum of these makes up the total course reserve. Each session we update the current enrollment for new and continuing students. This current information helps us in allocating seats for the next orientation session and may alert us of potential problems in our course enrollment predictions.

The calculations for new and continuing student seat release are shown for our example in Table 4.8. There, the remaining continuing seats and the remaining new seats are shown. We calculate the remaining number of available seats by subtracting the current enrollment from the appropriate reserve. (Calculating remaining seats may be more involved depending on the enrollment rules at the university using the system. Holding back seats for continuing students or other groups may complicate the calculation.) Our example is for the third of five orientation sessions, so we release three-fifths of the remaining continuing student seats. The proportion of remaining new student seats we release is 37—the upper bound on the 95% confidence interval shown in Table 4.5 for session 3—divided by 95—the total number of remaining new seats, which is shown in Table 4.8. The updated capacity is the total current enrollment plus the continuing and new student releases. Table 4.8 shows these sums for our example. Note that using the current enrollment after each orientation session as the base for calculating remaining seats, instead of using the estimates on what we expect to happen or a pre-conceived plan, allows for recovery from unforeseen demand or other abnormalities.

Section	1	2	3	4	TOTAL
Total Enrollment	33	25	30	37	125
Remaining Continuing Seats	4	3	3	1	11
Continuing Release					
(proportion = 3/5)	2	2	1	1	6
Remaining New Seats	27	21	21	26	95
New Release					
(proportion = 37/95)	11	8	8	10	37
Updated Capacity	42	26	35	34	138

Table 4.8: Example of adjusted section capacities after new and continuing student seat release.

A desirable side effect of our seat allocation and release system is that fewer seats are released from sections with more than average enrollment. This is because we release a fixed proportion of seats from the remaining seats in each section. Sections with more than average enrollment have fewer than average remaining seats, resulting in fewer seats being released. This is especially helpful in keeping uniform the distribution of students enrolled in each section because students tend toward the desirably timed sections.

4.5 Results

We used the seat allocation system presented in this chapter at CU during the summer of 2006. The system released seats for four of the largest enrollment mathematical sciences courses with success in making accurate estimates of seat need and releasing equitable distributions of course seats each orientation session. The first measure of this success is shown in Table 4.9. There, the estimated new student enrollment is recorded for each of the four courses at four times in 2006 along with the actual enrollment at the start of the fall semester. With the aid of the course prediction model, we made the first two estimates. The seat allocation system helped in adjusting these estimates during the orientation sessions, and in most cases improved the estimates. The error rates are, for the most part, near our 5.3% criteria. The error rates for MATH 108 are higher than the others in part due to relatively low enrollment compared to the other courses.

A second measure of success for our seat allocation system is the actual release in comparison to the uniform release that had been taking place. Note that the uniform release

Course	Early (Feb.)	Mid (June)	Late (July)	Final (August)	Actual Actual
101	460 -(8%)	449 -(6%)	448 -(5%)	446 -(5%)	425
102	580 -(4%)	559 (0%)	559 (0%)	561 (0%)	560
106	809 -(11%)	791 -(8%)	786 -(7%)	787 -(8%)	732
108	219 (15%)	219 (15%)	211 (18%)	239 (7%)	258

Table 4.9: Estimated new student enrollment in four mathematical sciences courses at four times in 2006.

allots approximately the same number of seats to release each session. The number may not be exactly the same because the number of orientation sessions may not divide the number of reserved seats. In practice, the effective release will vary because the number of seats that were left over (or taken beyond capacity) from the previous session will still be available (or already taken). The last column of Table 4.10 shows examples of where the effective release is greater than or less than the uniform release. Our seat allocation system does not suffer from differing effective and actual releases because it releases seats based on the number of students currently enrolled on the day of the release; it does not depend on previous estimates to make the current release. Table 4.10 shows the actual MATH 102 enrollment each session along with the number of seats released by our model. The enrollment during a session can be negative if more students drop the course than add it. In the fourth column we have included the number of seats that would be released had we used a smaller version of our model that lacks some of the detail in the course prediction. The stripped model does not include the breakdown of students into major groups. It simply uses whether a student is classified as transfer or first-time to make predictions. The fifth column of Table 4.10 shows the number of seats that would have been released if the seats had been released uniformly. The numbers given in parentheses are the differences between the release and the actual enrollment. The uniform release performs poorly because it uses the estimate for new students from June to release seats throughout the orientation sessions. There is no update during the summer as estimates change. Note that the stripped model does almost as well as the full model. One more point is that the number of seats we released is greater

session	actual enroll.	model release	stripped model release	uniform release	effective uniform release
1	34	47 (13)	46 (12)	54	54 (20)
2	61	80 (19)	81 (20)	54	74 (13)
3	63	86 (23)	84 (21)	54	67 (4)
4	58	89 (31)	91 (33)	54	58 (0)
5	69	108 (39)	106 (37)	53	53 (-16)
6	39	94 (55)	90 (51)	54	38 (-1)
7	42	110 (68)	107 (65)	54	53 (11)
8	65	107 (42)	114 (49)	54	65 (0)
9	80	95 (15)	109 (29)	54	54 (-26)
Interim	32	20 (-12)	20 (-12)	0	-26 (-58)
10	-9	49 (58)	49 (58)	53	-5 (4)
11	17	58 (41)	58 (41)	54	58 (41)

Table 4.10: Enrollment, seat release, and possible seat releases by session for MATH 102 during 2006.

than the number of students we expected. To hedge against too few seats available at a session, we released the upper bound on a 95% confidence interval of our estimate at each session. We used a 95% confidence interval in the simulation of our stripped model as well.

Figure 4.3 shows the results for the MATH 102 seat allocation for both the full and the stripped model. The lines with markers show the actual student enrollment, the actual release using the full model, and the simulated release using the stripped model respectively. The lines without markers show the expected enrollment in the remaining sessions based on future orientation enrollments at each session. Note that these increase and tend toward the actual enrollment as the sessions continue. Recall that this is because students register for sessions up until that session. Therefore, we have more confidence in our estimates

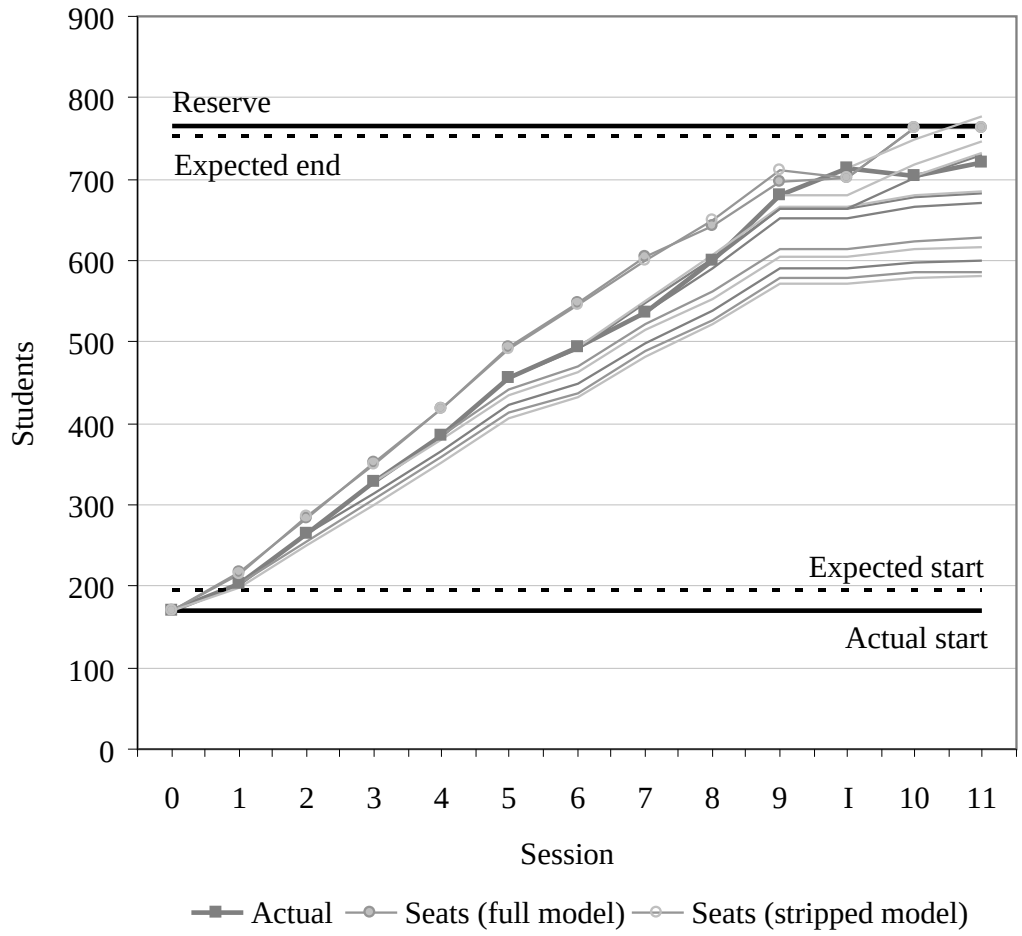


Figure 4.3: Results for MATH 102 seat allocation using 95% confidence interval. *This graph shows the increasing expectation of student enrollment in MATH 102 as students register for orientation sessions. Note that the actual number of students at the end of the orientation sessions is slightly less than the expected ending point.*

of session registration the closer we are to that session; the more confidence in session registration implies more confidence in course enrollment estimates. The line indicating the reserve in Figure 4.3 is the number of seats allotted for the course by the department. The "Expected End" is the number of seats we expect to need. The "Expected Start" is the number of continuing students we expected to have already enrolled by the time orientation sessions began. The "Actual Start" is the number of continuing students who had enrolled at the time of the first orientation. The graph indicates that our course prediction model and our seat allocation system yield accurate predictions when used together.

In summary, the results presented for MATH 102 are representative of the results for all four courses for which we used the seat allocation system. The seat allocation system we presented is a realistic approach to managing course seats during multiple registration sessions. The system of estimation, allocation, and recovery allows for imbalances in orientation attendance and course registration, which inevitably occur. The results indicate that the model yields accurate results in practice.

Chapter 5

Final Examination Timetabling

Timetabling final examinations at most universities is an arduous task if new schedules are considered and analyzed each semester. Finding a conflict-free schedule is impossible for many universities because of the limited time in which the final exam timetable must fit. Furthermore, the gap between the theory and practice of examination timetabling is large due to the practices of rolling over old exam schedules or using ad hoc techniques. In this chapter we develop reusable examination timetabling methods that are straightforward and robust.

Full timetabling problem formulations are specific to the institution for which they are used. Depending on the institution, the problem formulation may include many other constraints. The literature considers *hard* and *soft constraints*. *Hard constraints* are those which cannot be violated; *soft constraints* lead to desirable features in a timetable, but may be violated in order to produce a feasible solution [Burke and Petrovic 2002]. An example of a hard constraint is a maximum number of students sitting for an exam in any one period due to room capacity constraints. An example of a soft constraint is no student having more than one exam per day.

We develop straightforward algorithms for examination timetabling that use *course times*, rather than individual courses, as the assignment elements for building exam schedules. *Course times* are sets of course meeting times. For example, a set might be all courses that meet at 8:00 MWF or 8:00 MW. Using course times rather than the set of all courses to build the final exam schedule allows us to use the built-in structure of the *course timetable*, which is the schedule of courses for the university for a given term. That is, we can take advantage of the fact that one person cannot sit for two courses at the same time. Furthermore, most universities have many more individual courses than course times. By using course times as the unit of measurement, we reduce the exam timetabling problem size.

We demonstrate our algorithms by creating several alternative timetables for final examinations at Clemson University. Unlike some universities that recreate the exam timetable each semester based on that semester's course enrollment, CU uses an exam schedule that remains constant for several semesters—requiring update only every few years. Using course times as the unit of measurement facilitates this goal since the demand for course times remains relatively constant despite changes in course schedules that occur from semester to semester. Furthermore, using course times as the assignment elements allows us to take advantage of the inherent structure of the course timetable including the assignment of rooms and non-overlap of students in courses scheduled at the same time. Using course times instead of individual courses may also yield fewer constraints in formulations since these constraints may have already been resolved through the constraints on students schedules imposed by the course timetable. This approach is appropriate for other schools and universities with the same goal of reusing exam schedules and is especially appropriate where exam timetables are developed university wide, but course timetables are not. Also, this approach can be modified easily for updating timetables each semester if that is desired.

The criteria we use to develop the exam timetables and measure their quality include the number of conflicts generated, the frequency of consecutive exams, and the number of multiple exams in a single day. We also analyze the placement of common exams during exam week and the number of students sitting for exams in each exam period. We consider the exam timetabling problem in two distinct ways. We first formulate the exam timetabling problem using a multi-criteria approach where we measure conflicts, consecutive exams, and constraints in one formulation. In the approach we develop, we split the problem and develop algorithms to build timetables in two phases. In the first phase, we group course times into the desired number of exam periods with the objective of minimizing conflicts when course times are grouped together. We present two distinct algorithms for this purpose. In the second phase, we sequence the groups in order to minimize the number of consecutive exams. We present two distinct formulations for this phase as well.

The remainder of this chapter is structured as follows. We first introduce the mathematical background for the exam timetabling problem in the next section. We review the litera-

ture on examination timetabling in Section 5.2. In Section 5.3 we describe the data used in studying exam timetabling at CU. We discuss our timetabling algorithms and formulations in a general setting in Section 5.4. We present results for alternative exam timetables at CU in Section 5.5.

5.1 Mathematical Background

The basic *examination timetabling problem* is defined here as the mapping of courses to exam periods in order to minimize conflicts produced by courses with common students being mapped to the same exam period. A simplification of this basic problem, finding a conflict-free timetable with no restriction on the number of exam periods, can be formulated as a graph coloring problem by letting each vertex in a graph represent a course and requiring that courses with common students have an edge between their vertices. The *graph coloring problem* is finding the minimum number of colors required such that all vertices are colored, and no two vertices connected by an edge have the same color. This number is known as the *chromatic number* of the graph. The graph coloring formulation produces a conflict-free timetable with the number of exam periods equal to the chromatic number of the graph. Finding a minimum coloring of a graph is an NP-hard problem [Karp 1974], which makes even the most basic exam timetabling problems difficult to solve.

When the length of the examination timetable is fixed, we can represent the examination timetabling problem as a quadratic program (QP) with binary variables as follows:

$$\begin{aligned} \min \quad & x'Qx \\ \text{s.t.} \quad & \textstyle\sum_{j\in J} \hat{x}_{ij} = 1, i \in I \\ & \hat{x}_{ij} \text{ binary } \forall i \in I, \forall j \in J. \end{aligned} \tag{5.1}$$

In (5.1) we assume we have c courses, $I = \{1, \ldots, c\}$, and we want to schedule $e < c$ exam periods, $J = \{1, \ldots, e\}$. The vector x is a $c \cdot e$ by one vector where $x_m = \hat{x}_{ij}$ with $m = (i-1)e + j$, $m \in \{1, \ldots, c \cdot e\}$, and $\hat{x}_{ij} = \{1$ if course i is assigned to exam period j; 0 otherwise$\}$, $i \in I$, $j \in J$. The coefficient matrix Q is given by $Q_{mn} = \hat{Q}_{(ij)(kl)}$ where

$m = (i-1)e + j$, $n = (k-1)e + l$, and $m, n \in \{1, \ldots, c \cdot e\}$. We define $\hat{Q} = \left[\hat{Q}_{(ij)(kl)}\right]$ by

$$\hat{Q}_{(ij)(kl)} = \begin{cases} f(d_{ik}) & \text{if} \quad j = l \text{ and } (ij) \prec (kl) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where $i, k \in I$, $j, l \in J$, the distance matrix $D = [d_{ik}]$ where d_{ik} = the number of students enrolled in both courses i and k, f is a penalty function for conflicts, and $(ij) \prec (kl)$ is defined as $(i-1)e + j < (k-1)e + l$. Note that the number of binary variables in this formulation is equal to $c \cdot e$, and Q is upper triangular.

The objective function in (5.1) is the number of conflicts associated with a given mapping. The constraints in the formulation ensure that all courses are mapped to exactly one exam time. Because Q is not necessarily positive definite or positive semi-definite, the objective function is not necessarily convex. The indefinite nature of Q makes solving (5.1) increasingly difficult as the sets I and J get large.

The exam timetabling problem may also be represented as a multi-criteria program. We give the definition of a multi-criteria program here, and present the specific formulation for an exam timetabling problem in Section 5.4.1. In general a multi-criteria problem may be formulated as follows:

$$\begin{aligned} \min \quad & F(x) = [f_1(x), f_2(x), \ldots, f_n(x)] \\ \text{s.t.} \quad & x \in X \\ \text{where} \quad & F : \mathfrak{R}^m \to \mathfrak{R}^n \\ & X \subset \mathfrak{R}^m. \end{aligned} \tag{5.3}$$

A worsening of one criteria, i.e. f_1, may result in a better value of another criteria, i.e. f_2. A solution $\hat{x}$ is said to be a *Pareto efficient* solution of (5.3) if there does not exist another $x \in X$ such that $f_i(x) \leq f_i(\hat{x})$, $i \in \{1, \ldots, n\}$ with strict inequality for at least one i. The set of Pareto efficient solutions is called the Pareto set, and from this set the decision maker may choose a solution.

The formulations presented in this section will aid in understanding the diverse approaches to the problem, which we discuss as we review the literature in the next section.

5.2 Related Work

The literature on exam timetabling includes several comprehensive surveys of the topic. Carter et al. [1996] present a survey of final examination timetabling studies that discusses various strategies for solving graph coloring problems—the basic formulation of exam timetabling problems. They test these strategies on real examination timetabling problems from several universities accounting for conflicts generated. They refer to conflicts as costs. They do not account for side constraints that are university specific. Carter et al. [1996] also test the strategies on the related graph-coloring problems (without costs) derived from the real timetabling problems and on randomly-generated cases. Carter et al. [1996] note that one major difference in graph-coloring heuristics is the method used for deciding which vertex to color next. Table 5.1 describes five sorting criteria used in list sorting methods of graph-coloring problems that are presented and compared in [Carter et al. 1996].

Carter et al. [1996] also discuss other strategies to be used in conjunction with the list processing methods presented in Table 5.1 to solve the exam timetabling problem using the graph coloring formulation. The first strategy is to run the graph coloring algorithm with proximity costs as proposed by [Laporte and Desroches 1984]. This is no longer a graph coloring problem, but an examination timetabling problem. The second strategy is to find large *cliques* in the examination timetabling data and schedule the elements in the cliques prior to running the graph coloring algorithm. A *clique* is a set of pairwise adjacent vertices. This strategy is used to schedule exams with the most course time conflicts first [Carter et al. 1996]. The third strategy is to use *backtracking* while running the graph coloring algorithm.

Sorting criteria	Description	Properties
Largest degree (LD)	Largest number of conflicting exams	This and (SD) yields minimum conflict timetables for most cases in [Carter et al. 1996]
Saturation degree (SD)	Largest number of conflicting periods	Yields minimum number of periods for most cases in [Carter et al. 1996]
Largest weighted degree (LWD)	Largest number of common students	Non remarkable for most cases in [Carter et al. 1996]
Largest enrollment (LE)	Largest enrollment	Second to (LD) and (SD) in terms of minimum conflict timetables.
Random ordering (RO)	Randomly chosen	Primarily used for test cases. Yields worst results for all performance measures.

Table 5.1: Sorting criteria used in graph-coloring algorithms tested in [Carter et al. 1996].

Backtracking is described by Carter et al. [1996] as undoing some assignments in order to schedule an examination that is in conflict with every examination so far scheduled. Thus, Carter et al. [1996] tested forty different strategies: five sorting criteria, with or without cliques, using proximity costs or not, and using backtracking or not. They found that using cliques is beneficial on real problems but not on randomly generated ones, that using backtracking yields shorter exam schedules, and that using proximity costs yields longer exam schedules. They propose that the clique strategy is not useful on randomly generated problems because of the uniform spacing of the elements, which is not typically the case in real problems. The authors believe that backtracking yields shorter exam schedules because backtracking corrects "mistakes" previously made during the coloring algorithm [Carter et al. 1996]. Finally, they contend that using proximity costs yields longer exam schedules because the algorithms try to evenly distribute the exams instead of clustering them together [Carter et al. 1996]. This article updates a previously published examination timetabling survey [Carter 1986].

The *saturation degree* criteria shown in Table 5.1 was first proposed by Brelaz [1979], where he defines the "saturation degree of a vertex as the number of different colors to which it is adjacent (colored vertices)." A number of other researchers have presented graph-theoretic approaches as methods for solving final examination timetabling problems including [Mehta 1981; de Werra 1985; Burke et al. 1994]. In particular [Mehta 1981] uses graph-theoretic methods as a starting solution and then uses a compression algorithm to fit the timetable into a specified number of exam periods. We use a similar compression method for our hierarchical clustering algorithm.

Schaerf [1999] presents a survey of timetabling problems that includes school timetabling, course timetabling, and examination timetabling. He notes the similarity in solution techniques of the three problems. Schaerf [1999] defines *direct heuristics* as those which are meant to solve problems by simulating the methods used by humans to solve the problems. Schaerf describes successive augmentation as extending a timetable course by course until all courses are assigned an exam period and classifies this technique as a direct heuristic [Schaerf 1999]. He also classifies integer programming techniques, network flow techniques, and graph coloring techniques in this category. Schaerf classifies simulated an-

Approach	Description	Examples
Sequential methods	Order and schedule events sequentially minimizing conflicts.	Graph coloring techniques.
Cluster methods	Events are grouped to satisfy hard constraints, then sequenced to satisfy soft constraints.	None given.
Constraint-based approaches	Find a feasible solution using deductive reasoning to decrease the search space.	None given.
Meta-heuristics	Start with initial solution, search feasible region with efforts to avoid local minima.	Simulated annealing, tabu search, genetic algorithms, hybrid approaches.

Table 5.2: Overview of approaches for examination timetabling in the literature as presented in [Burke and Petrovic 2002].

nealing, tabu search, genetic algorithms, and constraint satisfaction as artificial intelligence techniques used to solve timetabling problems [Schaerf 1999]. Simulated annealing is a search technique in which the randomness of the search is reduced, or annealed, in a controlled fashion as the search continues. Tabu search techniques attempt to avoid previously searched regions of the feasible region in order to find better search paths. Genetic algorithms consider populations of solutions that evolve according to certain mutation and crossover rules. Schaerf [1999] gives examples of studies using techniques from both categories, but cannot compare solution qualities as the test cases are not commensurate.

Burke and Petrovic [2002] present the work of the Automated Scheduling, Optimisation and Planning Research Group (ASAP) at the University of Nottingham, which focuses on course and final examination timetabling. They first present a brief overview of past timetabling methods and then present their novel approaches to the problem. Table 5.2 outlines the past timetabling methods and includes sequential methods, cluster methods, constraint-based approaches and meta-heuristics. Sequential methods attempt to schedule the courses to exam periods in sequence. Cluster methods attempt to solve the problem in phases, focusing on the hard constraints first. Constraint-based approaches search for feasible solutions by satisfying constraints and thus reducing the search space. Meta-heuristics begin with an initial solution and search the feasible region using methods to avoid local minima. Table 5.3 outlines new methods presented by Burke and Petrovic [2002]. These methods include hybrid heuristic methods, which combine multiple heuristics for solving

Approach	Description
Hybrid heuristic methods	Combination of graph-theoretic approaches and selection techniques such as tournament selection and bias selection.
Genetic and memetic algorithms	Genetic algorithms are global optimization algorithms that work on a population of solutions and employ methods similar to those found in evolutionary biology to search the feasible region. Memetic algorithms work with genetic algorithms on neighborhood search.
Multi-criteria approaches	Account for hard and soft constraints at once in the evaluation of the solution. Specifically, compromise programming is used to find solutions closest to the ideal point.
Case-based reasoning	Solving new problems based on similar previously solved problems.

Table 5.3: Heuristic and meta-heuristic approaches for examination timetabling as presented in [Burke and Petrovic 2002].

the timetabling problem. Genetic algorithms attempt to find global optimal solutions by considering populations of solutions; memetic algorithms assist genetic algorithms in local neighborhood searches. Multi-criteria approaches to the exam timetabling problem consider multiple objectives at once—for example, minimizing conflicts, consecutive exams, and multiple exams per day. Case-based reasoning approaches attempt to use previously solved exam or graph-coloring problems that are similar to the present problem to build a new timetable. Burke and Petrovic [2002] do not compare the quality of the solutions found by the methods presented.

Other studies on multi-criteria approaches to timetabling problems include [Silva et al. 2004; T'kindt and Billaut 2002]. Silva et al. [2004] present meta-heuristic multi-criteria methods for machine, personnel, and educational scheduling problems and includes an overview of multi-criteria modeling and decision making. Finding the Pareto set is the goal of many multi-criteria approaches, although some approaches rank the importance of criteria prior to the search and look for one solution that takes this ranking into account [Silva et al. 2004]. Silva et al. [2004] characterize multi-phase approaches, compromise programming, and multi-criteria evolutionary algorithms as multi-criteria approaches. Multi-phase approaches attempt to optimize one or more criteria in each phase, reducing the feasible region at each phase. The authors define compromise programming as a method to "find compromise solutions that are close to the ideal point," which is the vector with the best value for each criteria [Silva et al. 2004]. Multi-criteria evolutionary algorithms measure

Approach	Description	Properties
Mathematical programming	Usually binary linear programs.	High computational demands.
Graph coloring	Vertices are events, edges are conflicts. Many methods for list sorting.	Heuristics are easy to implement.
Cluster methods	Two phase approach: first satisfy hard constraints, then soft.	May yield poor solutions.
Constraint-based	Assign values until an infeasible solution is reached; backtrack and reassign; continue search.	None given.
Meta-heuristics	Starting with initial solution(s), search for optimal solution while avoiding local optima.	Significant parameter tuning, high computational cost.
Multi-criteria	Criteria measure the violation of constraints. Can solve problem in stages or simultaneously.	Allows for explicit ranking of criteria.
Case-based reasoning	Use previous solutions to similar problems and backtracking to make up for new constraints.	Measuring similarity of problems is difficult.

Table 5.4: Approaches used in examination timetabling as presented in [Petrovic and Burke 2004].

multiple criteria in the evaluation of the population. The authors give examples and discussions of each approach. The studies they cite are not commensurate and therefore no comparison between the three methods are given. We present a multi-criteria formulation for the examination timetabling problem in Section 5.4.1, and multi-phase models in the rest of Section 5.4.

The review by Petrovic and Burke [2004] updates the surveys by Carter et al. [1996] and Burke and Petrovic [2002] and is the most recent survey to date. Again, Petrovic and Burke [2004] focus on both course and examination timetabling in their study. Table 5.4 presents the timetabling approaches discussed in [Petrovic and Burke 2004], which include mathematical programming, graph coloring, cluster methods, constraint-based programming, meta heuristics, multi-criteria algorithms, and case-based reasoning. The authors specifically focus on and write in detail about multi-criteria approaches and case-based reasoning approaches to timetabling, which they claim are the state-of-the-art timetabling algorithms [Petrovic and Burke 2004]. Petrovic and Burke [2004] do not compare the quality of solutions from any of the methods presented.

Our study focuses on clustering and sequencing methods applied to final examination timetabling. White and Chan [1979] use a two-phase approach to timetabling similar to the one we propose. Their first phase clusters courses and the second phase sequences the

clusters to obtain an exam timetable. Our approaches differ in that we use course times instead of courses in our timetabling algorithm and their clustering algorithm uses both hard and soft constraints to develop their clustering [White and Chan 1979]. Lofti and Cerveny [1991] also use a multi-phase approach to solving exam timetabling problems. Their first phase clusters courses using a quadratic assignment problem (QAP) formulation without column constraints [Lofti and Cerveny 1991]. The next three phases seek to minimize various criteria that we minimize in our sequencing phase. The clustering references we predominantly use are [Kaufman and Rousseeuw 1990] and [Hartigan 1975]. The algorithms we use or extend from these references are explained in Subsection 5.4.2.

Several researchers [Mehta 1981; Fisher and Shier 1983; Carter 1986] have suggested using a traveling salesman problem (TSP) formulation to minimize consecutive exams after a clustering has been established. Given n clusters to be sequenced into n exam periods with σ_i students in cluster $i \in \{1, \ldots, n\}$, Fisher and Shier [1983] order the clusters by increasing number of students: $0 \leq \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_n$. Their initial solution to the exam scheduling problem is to sequence the clusters in the following manner:

$$\sigma_n \sigma_1 \sigma_{n-2} \sigma_3 \ldots \sigma_4 \sigma_{n-3} \sigma_2 \sigma_{n-1}. \tag{5.4}$$

They then attempt to improve the solution by swapping pairs of clusters. Their use of this technique on several contrived problems resulted in solutions that were on average 1.4% away from the optimal solution. We use the heuristics proposed by [Fisher and Shier 1983] to perform local searches after using our heuristics for clustering and sequencing. We also consider using (5.4) as a starting point for swapping clusters.

Balakrishman et al. [1992] use a modified TSP formulation to minimize consecutive exams excluding evening to morning occurrences. The authors formulate the problem as a network flow problem and solve a Lagrangrian relaxation [Balakrishman et al. 1992]. The primary sequencing references we use are [Wolsey 1998] and [Levchenkov et al. 2006]. The TSP formulation we extend from [Wolsey 1998] is given in Subsection 5.4.3. We finish this section with a description of the sequencing algorithm by [Levchenkov et al. 2006].

Levchenkov et al. [2006] formulated a two-phase binary linear program for sequencing exams at Cornell University. Cornell maps course times to exam times, in the same manner we propose. Their timetabling problem is smaller than those thus far described in that they have fewer course times than exam periods. Thus, they do not use any clustering methods to reduce the number of course times to the number of exam periods. Furthermore, they are able to schedule common exams and course times within the exam schedule with no overlap. Thus, their problem is to sequence the exams in a pleasing manner. To do this, the authors attempt to minimize the number of three exams in a day, two exams in a day, two consecutive exams in a day, three consecutive exams, and two consecutive exams. The formulation presented by [Levchenkov et al. 2006] is specific to a 21-period, 7-day exam schedule and includes constraints concerning the specific needs of the authors' university. The formulation, less the constraints specific to Cornell's needs, is given in Formulation 5.1 and Formulation 5.2.

The solution procedure employs a two-stage approach. The first stage attempts to minimize the number of three exams in a day, two exams in a day, and consecutive exams in a day by identifying triples of course times and/or common exams to be grouped together each day. *Common exams* are used for certain courses with sections across multiple course times that give students in all sections the final at a single, common time. A triple in this context is a set of three course times. The result of the first stage of the algorithm is seven unordered triples, with the center elements fixed. Minimizing the frequency of consecutive exams in a day fixes the inner element by eliminating the pair in the triple with the maximum common students. Figure 5.1, taken directly from [Levchenkov et al. 2006], gives an example of a triple with the number of common students between each pair labeled. Fixing the middle element reduces the number of two consecutive exams in a day from 100 to 90. This intuitive approach for minimizing the number of two consecutive exams in a day, RD2, is seen in the formulation in Formulation 5.1.

The second stage of the algorithm attempts to minimize the number of two consecutive exams (including overnight) and three consecutive exams (including overnight) by sequencing, and ordering forward or backward, the seven triples into the seven days of the exam schedule. The number of variables in the first stage is $C_3^{21} = 1,330$. CPLEX, software by

Formulation 5.1: Sequencing formulation by Levchenkov et al.: Stage 1

Goal: Minimize the weighted sum of 3 exams in a day, 2 exams in a day, and 2 consecutive exams in a day, by identifying the triples to be grouped together each day.
Preprocessing: Fix the inner cluster in each possible triple (by minimizing 2 exams in a row in a day).
Number of variables: $C_3^{21} = 1,330$

Notation, variable definitions
$E = \{1, 2, ..., 21\}$ is the set of clusters.
$T_{ij}^{(2)}$=number of students who have exams for courses in clusters i and $j \in E$.
$T_{ijk}^{(3)}$=number of students who have exams for courses in clusters i, j, and $k \in E$.
$x_{ijk} \in \{0, 1\}$ determines whether clusters within the triple $(i, j, k) \in T = \left\{(i, j, k) \in E^3 : i < j < k\right\}$ are scheduled on the same day.

Performance measures:
• 3 in a day $D3 = \sum_{(i,j,k)\in T} x_{ijk} \left(T_{ijk}^{(3)}\right)$
• 2 in a day $D2 = \sum_{(i,j,k)\in T} x_{ijk} \left(T_{ij}^{(2)} + T_{ik}^{(2)} + T_{jk}^{(2)}\right)$
• 2 consecutive in a day $RD2 = \sum_{(i,j,k)\in T} x_{ijk} \left(T_{ij}^{(2)} + T_{ik}^{(2)} + T_{jk}^{(2)} - \max\left\{T_{ij}^{(2)}, T_{ik}^{(2)}, T_{jk}^{(2)}\right\}\right)$
$\Rightarrow$ the middle cluster is fixed

Problem Formulation:
$\min \sum_{(i,j,k)\in T} \{w_1 D3 + w_2 D2 + w_3 RD2\}$
s.t. $\sum_{m\in(i,j,k)\in T} x_{ijk} = 1 \ \forall m \in E$
where w_1, w_2, w_3 are weights determined by the decision maker.

Let $S1$ = the solution set.

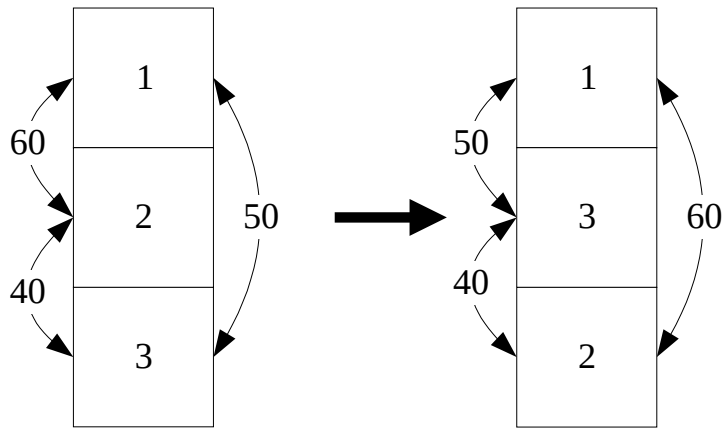


Figure 5.1: Fixing the middle element of a triple. *This diagram shows that to fix the middle element of a triple, we separate the pair of elements with the maximum number of common students.*

Formulation 5.2: Sequencing formulation by Levchenkov et al.: Stage 2

Goal: Minimize the weighted sum of 2 consecutive exams (including overnight) and 3 consecutive exams (including overnight) by sequencing and ordering the seven triples given in S1.
Number of variables: $7! \cdot 2^7 = 645,120$

Notation, variable definitions
$y_i \in \{0,1\}$, $1 \leq i \leq 645,120$, represents a permutation of S1, with each triple ordered forward or backward.
$Y = \{y_1, \cdots, y_{645,120}\}$ is the set of permutations of S1, with each triple ordered forward or backward.

Performance measures:
• 2 consecutive $R2_i$ for permutation i
• 3 consecutive $R3_i$ for permutation i

Problem Formulation:
$\min \sum_{i=1}^{645,120} (v_1 R2_i + v_2 R3_i) y_i$
s.t. $y_i \in Y$
where v_1 and v_2 are weights determined by the decision maker.

Let y^* be the optimal solution.

ILOG for solving linear, mixed integer, quadratic and mixed integer quadratic programs, can process 1,330 variables in a binary program easily [CPLEX 2006]. The second stage is solved by complete enumeration of the permutations and orderings of the seven triples. That is, each day must have one of the triples, no repeats, which yields 7! possibilities. Because each triple can be ordered forward or backward, the result is $7! \cdot 2^7 = 645,120$ possible permutations. That number, also shown as the number of variables in the second stage of the algorithm in Formulation 5.2, is misleading because the constraints have been taken away from the formulation, allowing more permutations to be feasible than would be in practice. Levchenkov et al. [2006] report that for their specific problem, the number of permutations was fewer than 100,000.

5.3 Data

The first step of this study involved gathering data on all enrolled students and all courses taught at Clemson University during the fall and spring semesters of 2003-2006. We pre-

processed this data using Microsoft Access and its SQL capabilities. We narrowed our study to courses and students with final exams held during finals week. This excluded labs and courses without final exams. Administrative policy at CU is to schedule variations of standard class times at the same exam time. For example a class that meets at eight o'clock Monday, Wednesday, Friday (8:00MWF) during the semester will be mapped to the same exam time as a class that meets at eight o'clock Monday, Wednesday (8:00MW). Henceforth we will use 8:00MWF to refer to all courses whose first meeting time of the week is 8:00 Monday, Wednesday, or Friday and refer to this as a *pre-grouped* course time. This mapping policy may cause exam conflicts to be built into the examination timetable. If students are enrolled in a course that meets 8:00 Monday and a different course that meets at 8:00 Wednesday, then those students will have a conflict in their exam schedule because both of those course meeting times will be mapped to the same exam period. For fall 2004, spring 2005, fall 2005, and spring 2006 there were 178, 167, 328, and 460 built-in conflicts, respectively. However, as many as 72% of these conflicts are associated with one department, which schedules their own finals.

Current policy at CU is that classes not scheduled to begin at a standard class time do not receive an exam assignment. In keeping with this policy, we narrowed our study to only those courses beginning at standard class times, which make up more than 96% of the courses during finals week. Finally, some classes at CU are mapped to exam times not by the time at which they meet during the semester, but by the course title. These exams are *common* exams. Currently, there are six departments giving common exams: Accounting, Chemistry, Communications, Experimental Statistics, Mathematical Sciences, and Physics. Because the common exams given by the Mathematical Sciences department are taken by nearly twice the number of students of any other exam, we split it into two exams consisting of non-engineering courses, refered to as MTHSC1, and engineering courses, refered to as MTHSC2. In our results, we abbreviate the names of the common exams to ACCT, CH, COMM, EX ST, MTHSC1, MTHSC2, and PHYS. Thus, the scope of our study consists of courses at CU offered during the spring and fall semesters of 2003-2006 with finals and standard starting class times or common exams. As many universities use standard class times and also give common exams, the input data is appropriately general. To appreciate

the size of this problem, note that in fall 2005, there were 61,974 exams scheduled for 17,284 students at CU, and there were 98 individual course times or common exams and 32 pre-grouped course times and common exams.

This data gives the reader a sense of the scope of the examination timetabling problem. We will present algorithms to solve the problem in general in the next section. The specific results for the alternative timetables created using our algorithms for CU are given in Section 5.5.

5.4 Timetabling Algorithms

The approaches to the examination timetabling problem we present in this section are a multi-criteria quadratic program using binary variables and heuristics that break the problem into several steps where individual criteria are considered at each step. We discuss the formulation of the multi-criteria approach in Subsection 5.4.1. Our timetabling algorithms using the second approach have two phases that include clustering and sequencing. We use two distinct clustering methods in the first phase. These methods and the exact formulation for the clustering are discussed in Subsection 5.4.2. In Subsection 5.4.3 we discuss the sequencing phase of the algorithm and present two formulations for sequencing. The implementation of the algorithms and several approaches to finding representative timetables are discussed in Subsection 5.4.4.

5.4.1 Multi-criteria Formulation

To solve the exam timetabling problem in one step, we define a multi-criteria program that encapsulates the criteria and constraints of the problem. Suppose we are interested in minimizing the number of conflicts and the number of two consecutive exams. We may add additional constraints after the initial formulation. We define a multi-criteria quadratic program with binary variables to represent the exam timetabling problem as follows. First recall the notation from Section 5.1 and equations (5.1) and (5.2) in particular. Suppose we have c courses, $I = \{1, \ldots, c\}$, and we want to schedule $e < c$ exam periods, $J = \{1, \ldots, e\}$. The vector x is a $(c \cdot e)$-by-1 vector where $x_m = \hat{x}_{ij}$ with $m = (i-1)e + j$, $m \in \{1, \ldots, c \cdot e\}$,

and $\hat{x}_{ij} = \{1$ if course time i is assigned to exam period j; 0 otherwise$\}$, $i \in I$, $j \in J$. The coefficient matrices Q^1 and Q^2 are given by $Q^1_{mn} = \widehat{Q^1}_{(ij)(kl)}$ and $Q^2_{mn} = \widehat{Q^2}_{(ij)(kl)}$, respectively, where $m = (i-1)e + j$, $n = (k-1)e + l$, and $m, n \in \{1, \ldots, c \cdot e\}$. We define $\widehat{Q^1} = \left[\widehat{Q^1}_{(ij)(kl)}\right]$ by

$$\widehat{Q^1}_{(ij)(kl)} = \begin{cases} f_1(d_{ik}) & \text{if} \quad j = l \text{ and } (ij) \prec (kl) \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

and $\widehat{Q^2} = \left[\widehat{Q^2}_{(ij)(kl)}\right]$ by

$$\widehat{Q^2}_{(ij)(kl)} = \begin{cases} f_2(d_{ik}) & \text{if} \quad |j-l| = 1 \text{ and } (ij) \prec (kl) \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

where $i, k \in I$, $j, l \in J$; the distance matrix $D = [d_{ik}]$ where d_{ik} = the number of students in both course times i and k; f_1, is a penalty function for conflicts; f_2, is a penalty function for consecutive exams, and $(ij) \prec (kl)$ means that $(i-1)e + j < (k-1)e + l$. Then the multi-criteria formulation is given as follows:

$$\begin{aligned} \min \quad & F(x) = \left[x'Q^1x, x'Q^2x\right] \\ \text{s.t.} \quad & \textstyle\sum_{j \in J} \hat{x}_{ij} = 1, i \in I \\ & \hat{x}_{ij} \text{ binary } \forall i \in I, \forall j \in J. \end{aligned} \tag{5.7}$$

We can reduce this multi-criteria program into a single objective quadratic program. Define the coefficient matrix Q by $Q_{mn} = \hat{Q}_{(ij)(kl)}$ where m is defined as above. We define $\hat{Q} = \left[\hat{Q}_{(ij)(kl)}\right]$ by

$$\hat{Q}_{(ij)(kl)} = \begin{cases} f_1(d_{ik}) & \text{if} \quad j = l \text{ and } (ij) \prec (kl) \\ f_2(d_{ik}) & \text{if} \quad |j-l| = 1 \text{ and } (ij) \prec (kl) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

where the indices have the same meaning as above. Using this coefficient matrix, the single objective QP is formulated as in (5.1).

The formulation given in (5.1) is NP-hard to solve as written. Furthermore, since the objective function is not convex, most commercial quadratic programming software does not guarantee convergence to the global optimum, possibly becoming trapped at a local optimum. To solve the latter problem, we may use the special structure of the program to re-write it as an integer program with linear objective function as follows. Using the notation from above, let $y_p = \hat{y}_{mn} = \tilde{y}_{(ij)(kl)} = \{1 \text{ if } \hat{x}_{ij} = \hat{x}_{kl} = 1;\ 0 \text{ otherwise}\}$, where $i, k \in I$ and $j, l \in J$ such that $i < k$ and $|j - l| \leq 1$. Then y is a $c(c-1)(3e-2)/2$ by 1 column vector with p defined as follows:

$$p = \begin{cases} (i-1)\sigma_i + 2(k-i-1) + l & \text{if} \quad j = 1 \\ (i-1)\sigma_i + 2(c-i) + 3(k-i-1) + (l-j+2) & \text{if} \quad j \notin \{1, e\} \\ (i-1)\sigma_i + 2(c-i) + 3(e-2)(c-i) + 2(k-i-1) + l - (e-2) & \text{if} \quad j = e \end{cases} \tag{5.9}$$

where $\sigma_i = \sum_{q=1}^{i-1}(3e-2)(c-q) = (i-1)(3e-2)(c-i/2)$. The intuition behind these indices is as follows. When course time i is mapped to exam period $j = 1$, the only other mappings that produce conflicts or consecutive exams associated with course time i are when course time k is mapped to exam period $l = \{1, 2\}$. At the other end of the schedule, when course time i is mapped to exam period $j = e$, the only other mappings that produce conflicts or consecutive exams associated with course time i are when course time k is mapped to exam period $l = \{e-1, e\}$. When course time i is mapped to exam period $j \notin \{1, e\}$, the only other mappings that produce conflicts or consecutive exams associated with course time i are when course time k is mapped to exam period $l \in \{j-1, j, j+1\}$. In order to avoid redundant variables (i.e. $y_{(11)(21)}$ and $y_{(21)(11)}$) we enforce the restrictions $i < k$ and $|j - l| \leq 1$. This results in a decreasing number of variables with index i as i increases, explaining the function σ_i. The vector y has $c(c-1)(3e-2)/2$ additional binary variables. Recall there are already $c \cdot e$ binary variables given in the quadratic formulation. Formulation 5.3 defines the cost vector c and presents the integer linear program (ILP) formulation of the multi-criteria program.

The formulation given in (5.11) generates many variables. For example, when mapping $c = 32$ course times to $e = 20$ exam periods, the number of variables is $c(c-1)(3e-2)/2 +$

Formulation 5.3: Scalarized multi-criteria binary linear final examination timetabling formulation.

Define the cost vector s for all $i < k$ and $|j - l| \leq 1$ as follows:

$$s_p = \hat{s}_{mn} = \tilde{s}_{(ij)(kl)} = \begin{cases} f_1(d_{ik}) & \text{if} \quad j = l \\ f_2(d_{ik}) & \text{if} \quad |j - l| = 1. \end{cases} \tag{5.10}$$

Then, the LP representing the multi-criteria final exam timetabling problems is given by

$$\begin{array}{ll} \min & sy \\ \text{s.t.} & \sum_{j \in J} x_{ij} = 1, i \in I \\ & y_{(ij)(kl)} \leq x_{ij} \\ & y_{(ij)(kl)} \leq x_{kl} \\ & y_{(ij)(kl)} \geq x_{ij} + x_{kl} - 1 \\ & x_{ij} \text{ binary } \forall i \in I, \forall j \in J \\ & y_{(ij)(kl)} \text{ binary where } i, k \in I, \text{ and } j, l \in J : i < k \text{ and } |j - l| \leq 1. \end{array} \tag{5.11}$$

$c \cdot e = 29,408$. The number of variables is large enough that insufficient memory problems can be encountered using commercial QP solvers. Even if memory is not a problem, the amount of time to solve (5.11) is large as demonstrated by the computational results in Section 5.5.

The difficulties in the implementation of the exam timetabling problem when formulated as a QP or an ILP lead us to split at the problem into segments. The algorithms we develop for the exam timetabling problem build timetables in two phases. In the first phase, we group course times into a number of groups equal to the specified number of exam periods. In this phase, we attempt to minimize conflicts resulting from the groupings. In the second phase, we sequence the groups in order to minimize the number of consecutive exams. We present the results of this study in the following three subsections.

5.4.2 Clustering Algorithms

In the first phase of our exam timetabling algorithms, we group course times into a number of groups equal to the specified number of exam periods. Theoretically, we can find an optimal clustering by using the formulation given in (5.1), where J is the set of clusters, D_{ij} is the number of conflicts generated by mapping course times i and j to the same cluster, and $\hat{x}_{ij} = \{1$ if course time i is assigned to cluster j; 0 otherwise$\}$. Again, there are several

problems with this formulation; it is NP-hard to solve and QP solvers may not converge to the global optimal solution because the objective function is not necessarily convex. If re-written as an ILP, it is highly degenerate because the ordering of the clusters is irrelevant. In addition, computer memory requirements may prevent a solution to the IP from being found. Furthermore, because the goal is to find reusable examination timetables using historic data, the optimal solution to the clustering is not necessarily the optimal clustering for the new timetable. Thus, we develop two different clustering heuristics to perform this grouping; one is a hierarchical agglomerative algorithm and the other is a partitioning algorithm. Both algorithms use the 1-norm to measure distance. For a vector $x = (x_1, x_2, ..., x_n)$ the 1-norm is defined as

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| .$$

We use the 1-norm because when a third course time, C, is grouped with two course times, A and B, we incur the conflicts between A and B as well as the conflicts between C and A and between C and B. That is, the distance between two clusters X and Y is the sum

$$D(X, Y) = \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} d(x_i, y_j) \tag{5.12}$$

where n_X and n_Y indicate the number of elements in clusters X and Y, respectively; $x_i \in X$, $y_j \in Y$; and $d(x_i, y_j)$ is the number of students taking classes in both course times x_i and y_i.

Hierarchical clustering methods construct groupings based on previously constructed groupings. These methods can be *agglomerative* or *divisive*. Agglomerative methods begin with each element as a singleton and successively join elements to create fewer, larger clusters. Divisive methods begin with large clusters and successively divide clusters to create more, smaller clusters. The tree structure obtained from hierarchical clusterings is known as a *dendrogram*. Figure 5.2 shows a dendrogram for clustering three elements. The arrows at the top and bottom show the difference between agglomerative and divisive methods.

We develop a hierarchical agglomerative clustering algorithm for the exam timetabling problem. The algorithm starts with all variations of course times as distinct clusters. It finds

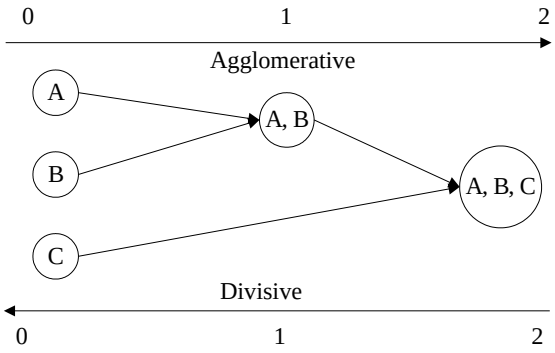


Figure 5.2: Depiction of agglomerative and divisive clustering methods.

the minimum distance between two distinct course times and joins the times. We employ an additional data structure to keep track of which course times are clustered. The algorithm uses the distance matrix of conflicts between course times as described previously. During the algorithm when two course times i and j, $i < j$, are clustered, row i and column i are updated to reflect the join as follows: $D_{ik} = D_{ik} + D_{jk}$, and $D_{ki} = D_{ki} + D_{kj}$ for all k. That is, the data for course time i now reflects the conflicts of both course time i and course time j with every other course time. Row j and column j can be deleted from the matrix. This procedure repeats until the desired number of exam periods is reached. Note that there could be ties for the value of the minimum distance. Ties are broken by choosing the first minimum element in the matrix, where first is the element with the lowest sum of row and column indices. In case of a tie for the sum, the element with the lowest row index is used. The algorithm is summarized in Algorithm 5.4.

The order in which the course times are joined may affect the final clustering. In addition the order may affect the number of conflicts that the final clustering yields. Table 5.5 and Figure 5.3 shows an example of a clustering that is not invariant under the initial ordering of the elements. The highlighted number in each matrix is the minimum D_{ij}; elements i and j will be clustered in the next step. Clustering 1 results in two clusters—course times 1 and 2 and course times 3 and 4—with a total six conflicts. Clustering 2 results in two clusters—course times 2 and 3 and course times 1 and 4—with a total of 4 conflicts.

To address this ordering problem, we run our clustering algorithm repeatedly, each time using a different initial order of the course times, until pre-set criteria, such as a maximum

Algorithm 5.4: Hierarchical, agglomerative clustering algorithm.

Input:

1. Symmetric matrix D of distances between elements.
2. Number n of elements.
3. Array A_{nx1} for recording the mapping of elements to clusters. Initially, $A(i) = i$.
4. Number k of desired clusters.
5. Number of conflicts, $c = 0$ at start.

Algorithm:

1. Find minimum, non-diagonal element D_{ij}, $i < j$.
2. Join elements i and j by updating D as follows: $D_{im} = D_{im}+D_{jm}$, and $D_{mi} = D_{mi}+D_{mj}$ for all m.
3. Delete row and column j from D.
4. Update $c = c + D_{ij}$.
5. Update $n = n - 1$.
6. Update $A(j) = i$.
7. If $n = k$, stop. If $n > k$, repeat 1-7.

Output:

1. Array of elements mapped to one of k clusters.
2. Number of conflicts, c, generated by clustering.

Clustering 1	$\begin{bmatrix} 0 & 1 & 6 & 3 \\ 1 & 0 & 1 & 4 \\ 6 & 1 & 0 & 5 \\ 3 & 4 & 5 & 0 \end{bmatrix}$	$\Rightarrow$	$\begin{bmatrix} 1 & 7 & 7 \\ 7 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix}$	$\Rightarrow$ $\begin{bmatrix} 1 & 14 \\ 14 & 5 \end{bmatrix}$
Conflicts:	0		1	6
Clustering 2	$\begin{bmatrix} 0 & 1 & 6 & 3 \\ 1 & 0 & 1 & 4 \\ 6 & 1 & 0 & 5 \\ 3 & 4 & 5 & 0 \end{bmatrix}$	$\Rightarrow$	$\begin{bmatrix} 0 & 7 & 3 \\ 7 & 1 & 9 \\ 3 & 9 & 0 \end{bmatrix}$	$\Rightarrow$ $\begin{bmatrix} 3 & 16 \\ 16 & 1 \end{bmatrix}$
Conflicts:	0		1	4

Table 5.5: Example of clustering not invariant under ordering.

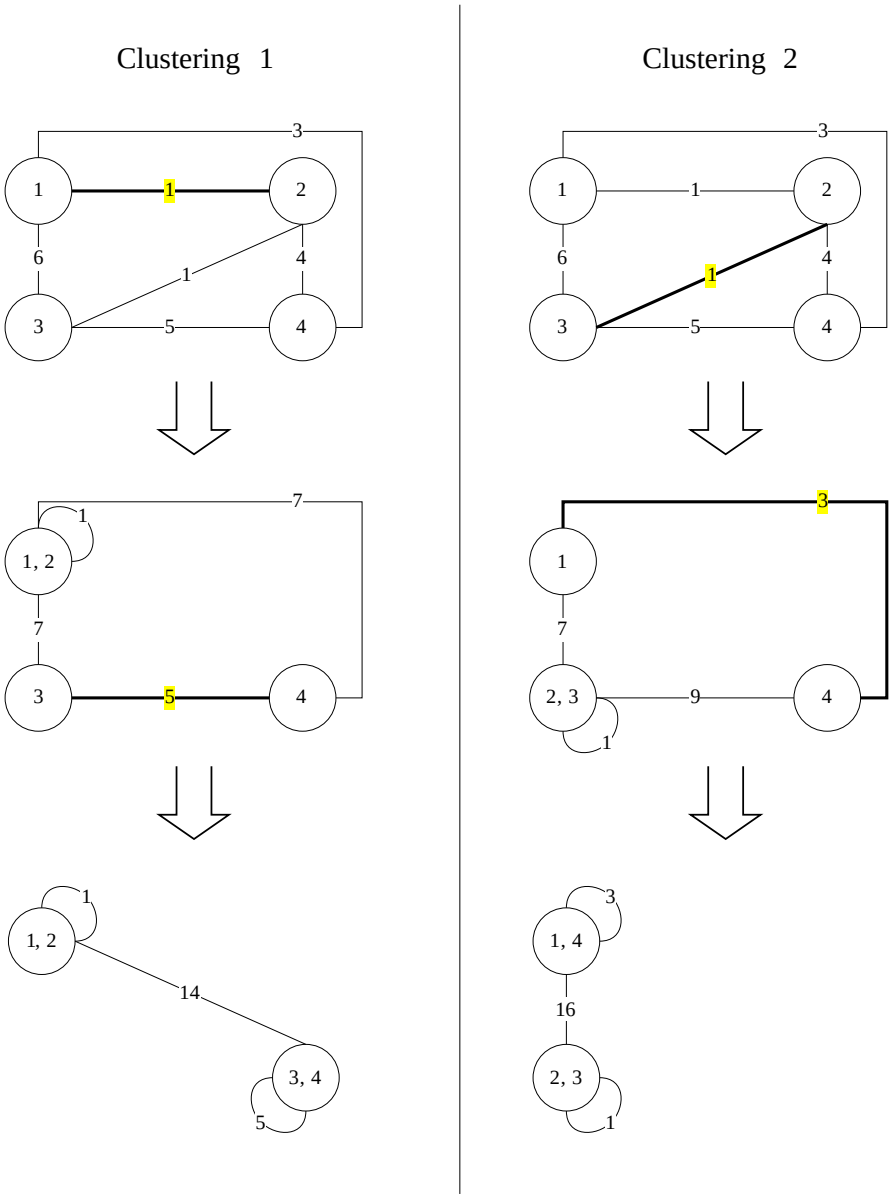


Figure 5.3: Example of clustering not invariant under ordering.

number of iterations, has been achieved. For simplicity in Table 5.5, we begin with the same distance matrix for both clusterings. However, our algorithm rearranges the ordering of the elements for repeated runs and uses the first minimum element rule for joining elements.

Our hierarchical algorithm is similar to the compression algorithm described by Mehta [1981]. However, Mehta proposed to use his compression algorithm only as an addition when graph-theoretic methods failed to find an appropriately small chromatic number of the graph created by the distance matrix. He also did not note or address the dependence of the final solution on the initial ordering. Our hierarchical agglomerative algorithm is the same as the joining algorithm described in chapter twelve of *Clustering Algorithms* [Hartigan 1975]. Hartigan [1975] claims but does not prove that the clustering produced by the hierarchical agglomerative algorithm is invariant under initial ordering if all distances computed during the algorithm are distinct. We pose this assertion as Proposition 5.4.1 and prove it below.

Proposition 5.4.1. *The clustering produced by the hierarchical agglomerative algorithm is invariant under initial ordering of the elements to be clustered if all distances computed during the algorithm are distinct.*

Proof. Assume all distances computed at each step of the algorithm are distinct. Suppose, by way of contradiction, that two distinct initial orderings, σ_1 and σ_2, with distance matrices D^1 and D^2, respectively, produce two distinct clusterings. Then, during at least one step, σ_1 joins elements i and j while σ_2 joins elements k and l where $\{i, j\} \neq \{k, l\}$. Let step α be the first in which σ_1 and σ_2 join different elements. If $D^1_{ij} < D^2_{kl}$, then σ_2 joins i and j rather than k and l. If $D^1_{ij} > D^2_{kl}$, then σ_1 joins k and l rather than i and j. Since $D^1_{ij} \neq D^2_{kl}$ through step α, and because at step α, σ_1 and σ_2 join different elements, we must have $D^1_{ij} = D^2_{kl}$, which is a contradiction to the assumption that all distances computed at each step of the algorithm are distinct. Thus, the proposition holds. □

The other clustering algorithm we consider uses a partitioning method. Partition clustering algorithms work differently from hierarchical methods in that they do not use previously constructed clusters to generate new clusters. Some partitioning algorithms specify a minimum radius and construct clusters that fit within that radius [Hartigan 1975]. Other

partitioning algorithms, including k-means and partitioning around medoids, find a specified number of representative elements and then cluster around these elements [Hartigan 1975; Kaufman and Rousseeuw 1990; Rencher 1995].

A k-means clustering algorithm first selects k representative elements either randomly or by some specified criteria. It then assigns each remaining element to the closest representative element using a specified metric. When a cluster has more than one element, the representative element is replaced by the *centroid* of the cluster, which is the mean of the elements in the cluster. The algorithm examines each element and reassigns it to another cluster if it is closer to the centroid of another cluster. Note that centroids must be recalculated after each reassignment. The k-means clustering algorithm iterates between reassignment and recalculation until no more reassignments can be made or until some other specified metric for convergence has been achieved [Rencher 1995].

The clustering algorithm known as partitioning around medoids (PAM) was developed by Kaufman and Rousseeuw [1987]. The authors define a *medoid* as "that object of the cluster for which the average dissimilarity to all the objects of the cluster is minimal" [Kaufman and Rousseeuw 1990]. To our knowledge, the algorithm has never been used in examination timetabling. The algorithm is similar to the k-means algorithm in that it first finds k representative elements and then clusters the remaining elements around the k medoids based on the distance to the medoid [Kaufman and Rousseeuw 1990]. At this point the PAM algorithm diverges from the k-means algorithm. The medoid never becomes the mean of the cluster; it remains the representative element of the cluster—similar to the median of a set, suggesting the name medoid. The PAM algorithm calculates the sum of the dissimilarities in each cluster from the representative medoid. That is, it calculates the sum

$$S = \sum_{i=1}^{k} \sum_{j=1}^{n_k} \left| x_j - m_i \right| \tag{5.13}$$

where k is the number of medoids, n_k is the number of elements in each cluster, $|.|$ is the distance metric, and m_i, $1 \leq i \leq k$, are the medoids. In order to improve the clustering, the algorithm chooses each medoid, i, in turn and each element that is not a medoid, j, and examines the change to S if their roles are interchanged. That is, it calculates S' as if j

was a medoid and i was a non-medoid. Thus, S' reflects the reassignment of elements to the clusters of the closest medoid when j is a medoid and i is not. Then, the pair i,j with the minimal S' is chosen. If $S - S'$ is positive, the two elements are interchanged, and the algorithm continues. If $S - S'$ is negative or zero, the objective cannot be decreased by swapping and thus the algorithm ends. Because all possible interchanges are considered, Kaufman and Rousseeuw [1990] note that the algorithm is invariant under the initial ordering except when the distances between multiple objects are tied. This means that the optimal value of the objective function will not change despite permutations to the initial ordering. However, the clustering produced by the algorithm could be different depending on the initial ordering if there exist more than one clustering with the optimal objective function value.

The PAM method does not suit the needs of clustering course times to exam periods because it does not take into account the cumulative nature of joining course times together. Recall from (5.12) that when a third course, C, is grouped with two courses, A and B, we incur the conflicts between A and B as well as the conflicts between C and A and between C and B. Thus, we use a modified PAM algorithm for our second clustering algorithm. We begin by choosing k representative elements using the method used in the PAM algorithm. The method for choosing these medoids is outlined in [Kaufman and Rousseeuw 1990] and is reproduced in Algorithm 5.5. Note that in selecting the k representative elements, the objective function is the sum of the dissimilarities between each element and the nearest representative object.

If we let s be the sum of the dissimilarities from each object to its nearest medoid after the k medoids have been found, but before the algorithm has continued, we note that the value of the sum s is not invariant under the initial ordering of the elements to be clustered, even if all distances are distinct. As an example, suppose we are clustering four elements, e_1, e_2, e_3, and e_4 into two clusters. Let the elements have the distance matrix

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 6 & 2 \\ 1 & 0 & 5 & 3 \\ 6 & 5 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{bmatrix}.$$

Algorithm 5.5: Part I of Partitioning Around Medoids (PAM) Algorithm.

The first object is the one for which the sum of the dissimilarities to all other objects is as small as possible. This object is most centrally located in the set of objects. Subsequently, at each step another object is selected. This object is the one which decreases the objective function as much as possible. To find this object, the following steps are carried out:

1. Consider each object i which has not yet been selected.
2. Consider each other non selected object j and calculate the difference between its dissimilarity D_j with the most similar previously selected object, and its dissimilarity $d(j, i)$ with object i.
3. If this difference is positive, object j will contribute to the decision to select object i. Therefore, we calculate $C_{ji} = \max(D_j - d(j, i), 0)$.
4. Calculate the total gain obtained by selecting object i: $\sum_j C_{ji}$
5. Choose the not yet selected object i which maximizes$_i \sum_j C_{ji}$

This process is continued until k objects have been found.

The sum of the dissimilarities to all other elements for each element is simply the row sum representing that element. This value is nine for e_1, e_2, and e_4. It is fifteen for element e_3. Thus, we may choose element e_1, e_2, or e_4 as the element with the minimum sum of dissimilarities to all other elements. Table 5.6 shows the calculations and results from choosing e_1 or e_2 as the first representative element. The results show s is not invariant under the initial ordering.

Our algorithm diverges from the PAM algorithm after we find the initial k representative objects. Instead of labeling the representative objects medoids as in PAM, we will define them as *cumoids*, which represent the cumulative distance from every element in the cluster. Thus, we designate the algorithm we use as partitioning around cumoids (PAC). At this point we cluster the remaining elements around the k cumoids, using a different metric than is used in the PAM algorithm. As in our hierarchical clustering algorithm, we use the 1-norm to find the distance between a cluster and an element. That is, we cluster the remaining elements around the closest cumoids using the distance metric

$$D(X, y) = \sum_{i=1}^{n_X} d(x_i, y)$$

where n_X indicates the number of elements in cluster X; $x_i \in X$; y is the element yet to be

The first representative element chosen is e_1 with $s = 9$.	
$i = e_2$	$j = e_2$: $D_{e_2} = 1, d(e_2, e_2) = 0 \Rightarrow C_{e_2e_2} = max\{1 - 0, 0\} = 1$
	$j = e_3$: $D_{e_3} = 6, d(e_3, e_2) = 5 \Rightarrow C_{e_3e_2} = max\{6 - 5, 0\} = 1$
	$j = e_4$: $D_{e_4} = 2, d(e_4, e_2) = 3 \Rightarrow C_{e_4e_2} = max\{2 - 3, 0\} = 0$
	$\Rightarrow \sum_j C_{je_2} = 2$
$i = e_3$	$j = e_2$: $D_{e_2} = 1, d(e_2, e_3) = 5 \Rightarrow C_{e_2e_3} = max\{1 - 5, 0\} = 0$
	$j = e_3$: $D_{e_3} = 6, d(e_3, e_3) = 0 \Rightarrow C_{e_3e_3} = max\{6 - 0, 0\} = 6$
	$j = e_4$: $D_{e_4} = 2, d(e_4, e_3) = 4 \Rightarrow C_{e_4e_3} = max\{2 - 4, 0\} = 0$
	$\Rightarrow \sum_j C_{je_3} = 6$
$i = e_4$	$j = e_2$: $D_{e_2} = 1, d(e_2, e_4) = 3 \Rightarrow C_{e_2e_4} = max\{1 - 3, 0\} = 0$
	$j = e_3$: $D_{e_3} = 6, d(e_3, e_4) = 4 \Rightarrow C_{e_3e_4} = max\{6 - 4, 0\} = 2$
	$j = e_4$: $D_{e_4} = 2, d(e_4, e_4) = 0 \Rightarrow C_{e_4e_4} = max\{2 - 0, 0\} = 2$
	$\Rightarrow \sum_j C_{je_4} = 4$
$\Rightarrow$ the second representative element chosen is e_3 resulting in $s = 9 - 6 = 3$.	

The first representative element chosen is e_2 with $s = 9$.	
$i = e_1$	$j = e_1$: $D_{e_1} = 1, d(e_1, e_1) = 0 \Rightarrow C_{e_1e_1} = max\{1 - 0, 0\} = 1$
	$j = e_3$: $D_{e_3} = 5, d(e_3, e_1) = 6 \Rightarrow C_{e_3e_1} = max\{5 - 6, 0\} = 0$
	$j = e_4$: $D_{e_4} = 3, d(e_4, e_1) = 2 \Rightarrow C_{e_4e_1} = max\{3 - 2, 0\} = 1$
	$\Rightarrow \sum_j C_{je_2} = 2$
$i = e_3$	$j = e_1$: $D_{e_1} = 1, d(e_1, e_3) = 6 \Rightarrow C_{e_1e_3} = max\{1 - 6, 0\} = 0$
	$j = e_3$: $D_{e_3} = 5, d(e_3, e_3) = 0 \Rightarrow C_{e_3e_3} = max\{5 - 0, 0\} = 5$
	$j = e_4$: $D_{e_4} = 3, d(e_4, e_3) = 4 \Rightarrow C_{e_4e_3} = max\{3 - 4, 0\} = 0$
	$\Rightarrow \sum_j C_{je_3} = 5$
$i = e_4$	$j = e_1$: $D_{e_1} = 1, d(e_1, e_4) = 2 \Rightarrow C_{e_1e_4} = max\{1 - 2, 0\} = 0$
	$j = e_3$: $D_{e_3} = 5, d(e_3, e_4) = 4 \Rightarrow C_{e_3e_4} = max\{5 - 4, 0\} = 1$
	$j = e_4$: $D_{e_4} = 3, d(e_4, e_4) = 0 \Rightarrow C_{e_4e_4} = max\{3 - 0, 0\} = 3$
	$\Rightarrow \sum_j C_{je_4} = 4$
$\Rightarrow$ the second representative element chosen is e_3 resulting in $s = 9 - 5 = 4$.	

Table 5.6: Example of representative elements not invariant under ordering.

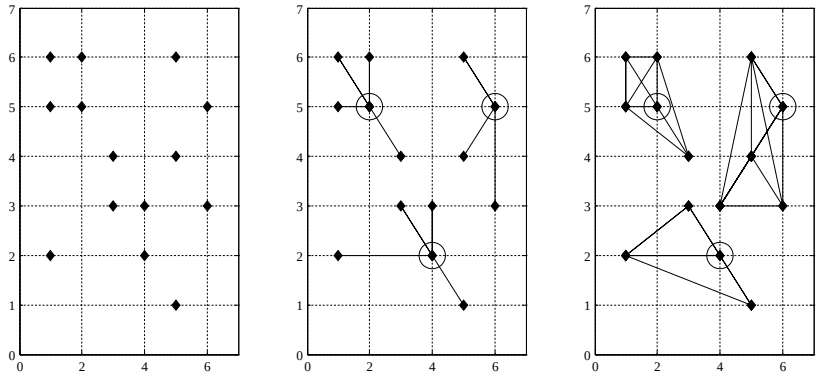


Figure 5.4: Two dimensional example of PAC clustering algorithm. *This diagram shows the steps of the PAC algorithm—from elements, to representative elements, to clusters. Note that the clusters defined by the representative elements are not necessarily the final clusters.*

clustered; and $d(x_i, y)$ is the overlap of students between periods x_i and y. See the example of the PAC clustering algorithm given in Figure 5.4.

Like the PAM algorithm, the PAC algorithm calculates the sum of the dissimilarities over all clusters. The formula for this calculation is

$$T = \sum_{i=1}^{k} D(X_i, X_i) = \sum_{i=1}^{k} \sum_{j=1}^{n_{X_i}} \sum_{l=j+1}^{n_{X_i}} d(x_j, x_l)$$

where k is the number of cumoids, X_i represents the clusters, n_{X_i} is the number of elements in each cluster, and $d(x_i, y_i)$ is the overlap of students between periods x_i and y_i. In order to improve the clustering, the PAC algorithm chooses one cumoid and one element that is not a cumoid and examines the change to T if their roles are interchanged. That is, it calculates T' as if the non-cumoid was a cumoid and the cumoid was a non-cumoid. Thus, T' reflects the reassignment of elements to the clusters of the closest cumoid. If $T - T'$ is positive, the two elements are interchanged. Unlike the PAM algorithm which converges to the optimal solution after iterating through all possible clusters, the PAC algorithm does not guarantee an optimal solution. Due to the nature of the 1-norm, the order in which elements are assigned to clusters may affect the final clustering and number of conflicts the clustering yields. That is, the results of the PAC algorithm are not invariant under the initial ordering. Furthermore, unlike the hierarchical agglomerative clustering algorithm,

the objective function value obtained using the PAC algorithm is not invariant under initial ordering even when all the distances computed during the algorithm are distinct. As an example where the initial ordering matters to the outcome of the PAC algorithm when the distances are all distinct, suppose we are clustering five elements e_1, e_2, e_3, e_4, and e_5 into two clusters. Let the elements have the distance matrix

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 5 & 6 & 8 \\ 2 & 5 & 0 & 7 & 10 \\ 3 & 6 & 7 & 0 & 9 \\ 4 & 8 & 10 & 9 & 0 \end{bmatrix}.$$

When the elements are ordered e_1, e_2, e_3, e_4, e_5, the clustering is $\{e_1, e_2, e_3\}$, $\{e_4, e_5\}$ and the objective function value is seventeen; when they are ordered e_4, e_2, e_3, e_1, e_5, the clustering is $\{e_1, e_2, e_4\}$, $\{e_3, e_5\}$ and the objective function value is twenty. Thus, as in our hierarchical algorithm, we randomly order the elements and run the algorithm repeatedly until a convergence metric has been achieved. A summary of the partitioning around cumoids algorithm is given in Algorithm 5.6.

Figure 5.5 shows the difference between the K-means, PAM, and PAC clustering algorithms using Euclidean distances in two dimensions. Each algorithm yields one cluster over four points, but the cost associated with the clustering is different in each case as shown by the lines drawn in each diagram. In this simple example there is only one cluster of all points. The algorithms may find different clusterings when more than one cluster is sought.

Empirical results comparing the two clustering algorithms and the optimal solution obtained by (5.11) are given in Section 5.5. The next subsection describes the second phase of our final examination timetabling procedure: the sequencing phase.

5.4.3 Sequencing

After we cluster the course times and thus determine which course times will be grouped into the same exam period, we determine a sequence of the clusters that minimizes the number of consecutive exams. This sequencing is the second phase of our examination timetabling algorithm. We formulate this phase of the problem in one of two ways. The

Algorithm 5.6: PAC clustering algorithm.

Input:

1. Symmetric matrix D of distances between elements.
2. Number n of elements.
3. Empty array A_{nx1} for recording the mapping of elements to clusters.
4. Number k of desired clusters.
5. Number of conflicts, $S = 0$ at start.
6. Index $i = 0$.
7. Let D_{copy} be a copy of D.

Algorithm:

1. Find k representative elements.
   (a) Use the partitioning around medoids approach using D.
   (b) Update $i = i + 1$ for each addition of a representative element j.
   (c) Update $A(j) = i$.
   (d) Let $B = A$.
2. Cluster remaining elements around representative elements.
   (a) Starting with the first non-representative element i, find the nearest representative element j.
   (b) Join i and j by updating D as follows: $D_{im} = D_{im} + D_{jm}$, and $D_{mi} = D_{mi} + D_{mj}$ for all m.
   (c) Delete row and column j from D.
   (d) Update $A(i) = A(j)$.
   (e) Update $S = S + D_{ij}$.
   (f) Repeat (a)-(e) until all non-representative elements have been clustered.
3. Swap non-representative elements with representative elements one at a time. Repeat 2.
   (a) For each representative element i and each non-representative element j, let $B(j) = B(i)$ and $B(i)$ be null. That is, let element j be a representative element and element i be non-representative.
   (b) Repeat step 2 with $D = D_{copy}$, $A = B$, and $S = S'$.
   (c) If $S - S' > 0$, record $C = B$ and $c = S'$. Else, record $C = A$ and $c = S$.

Output:

1. Array of elements, C mapped to one of k clusters.
2. Number of conflicts, c, generated by clustering.

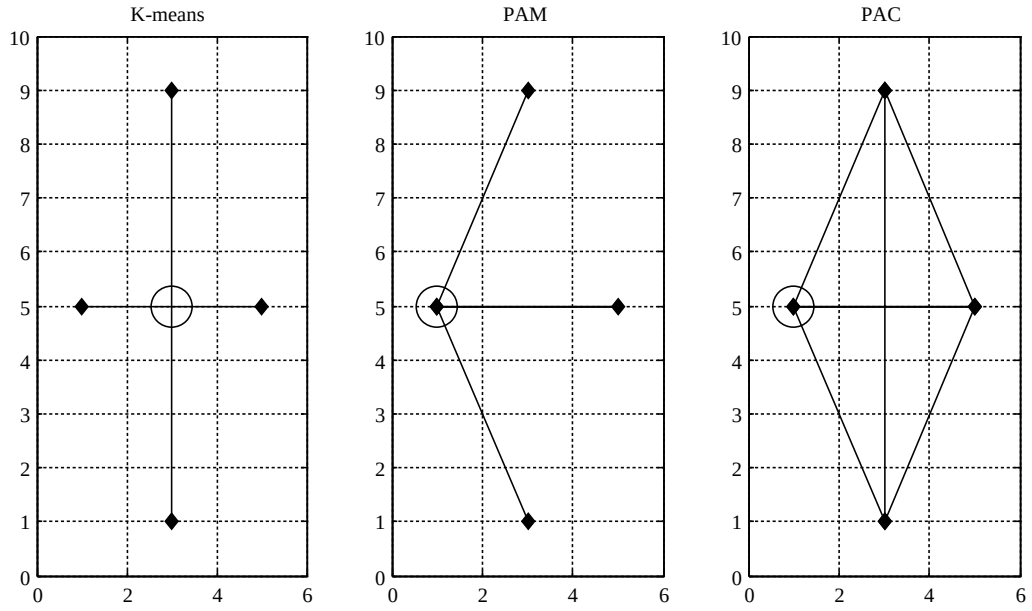


Figure 5.5: Comparison of the K-means, PAM, and PAC clustering algorithms. *This diagram shows the results of clustering four points into one cluster using the K-means, PAM, and PAC clustering algorithms, respectively.*

first is formulating the sequencing problem as a symmetric traveling salesman problem (STSP) where the distance between two clusters is the number of two consecutive exams that would result in scheduling the two clusters in consecutive exam periods. The problem is to find a cycle containing every cluster that minimizes the cost. Several researchers [Mehta 1981; Fisher and Shier 1983; Carter 1986] have suggested using a STSP formulation for sequencing clusters similar to what we propose here. The STSP is given in Formulation 5.7 as it is given in [Wolsey 1998], but we change the variable definitions to match the particulars of sequencing clusters.

Note that constraints (a) and (b) require that each cluster be placed in the sequence only once. Constraint (c) eliminates sub tours so that the sequence is continuous. The STSP is another famous NP-hard problem [West 2001]. Because the instances we are working with have few clusters—typically fewer than thirty—we used a reliable TSP solver that solves small TSP problems to optimality. The TSP solver we use is called Concorde and can be downloaded for free at www.tsp.gatech.edu/concorde/index.html [Concorde 2005].

In order to solve the STSP, we construct the symmetric matrix C where C_{ij} is the number of consecutive exams resulting from scheduling clusters i and j in subsequent exam periods.

Formulation 5.7: STSP formulation of the sequencing problem.

$$
\begin{array}{lll}
\min & \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij} x_{ij} & \\
\text{s.t.} & \sum_{j:j\neq i} x_{ij} = 1, i = 1, \ldots, k & \text{(a)} \\
& \sum_{i:i\neq j} x_{ij} = 1, j = 1, \ldots, k & \text{(b)} \\
& \sum_{i:i\in S} \sum_{i:j\notin S} x_{ij} \geq 1, S \subset N, S \neq \emptyset & \text{(c)} \\
& x_{ij} \text{ binary for } i = 1, \ldots, k,\ j = 1, \ldots, k &
\end{array}
\tag{5.14}
$$

where

$$
x_{ij} = \begin{cases} 1 & \text{if clusters } i \text{ and } j \text{ are scheduled subsequently} \\ 0 & \text{otherwise,} \end{cases}
$$

c_{ij} is the cost of scheduling clusters i and j subsequently,
and k is the number of clusters (and also the number of exam periods).

Note that the diagonal entries are equal to zero because the same exam cannot be scheduled in two periods. We use Concorde to find the optimal cycle of clusters. [Fisher and Shier 1983] suggest including a dummy cluster with zero distance to all other clusters such that the result of the TSP algorithm is a path instead of a cycle. We considered this technique and tested several examples using it. However, our final timetabling algorithm does not use this technique because we want to give decision makers more flexibility in where to begin the sequence.

The final decisions in the exam timetabling problem are choosing an edge on which to break the cycle and choosing a direction for the sequence. See the first diagram in Figure 5.6 for an example cycle of twelve clusters where the number of exam periods per day is three. In general, suppose that k is the number of exam periods to be scheduled each day of the timetable—typically three or four. Removing an edge of the cycle fixes the performance measures consecutive exams and multiple exams in a day. One strategy for which edge to choose is to remove the cycle edge of maximum value. This decreases the total number of consecutive exams by the greatest amount, but does not directly take into account the number of multiple exams in a day. Because breaking the cycle on every kth edge results in the same number of multiple exams in a day, within the cycle edges there are k equivalence classes with respect to multiple exams in a day. Instead of breaking on the edge of maximum value, another strategy is to break the cycle on an edge from the equivalence class with the fewest number of multiple exams in a day. For example, Figure

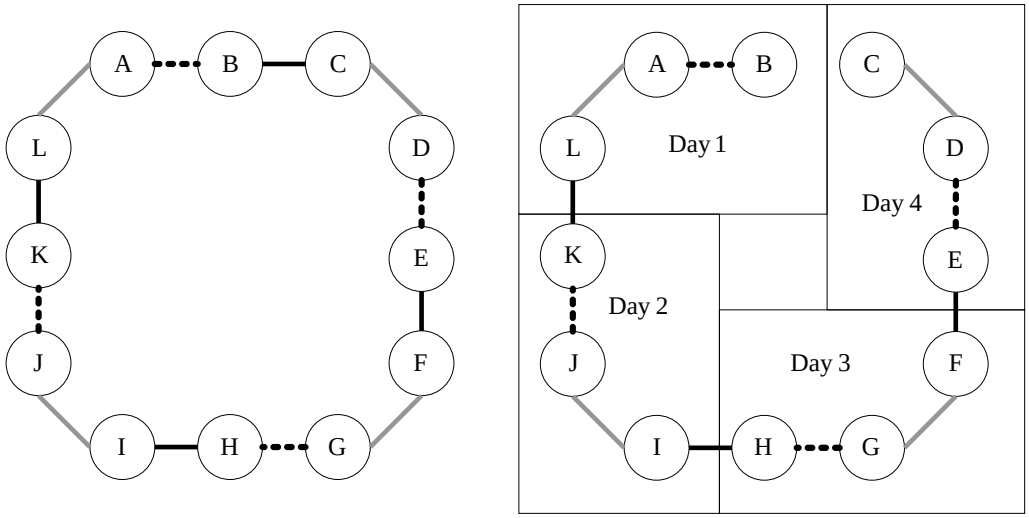


Figure 5.6: Example of TSP solution and final timetable. *This diagram shows the step from the TSP solution to the final timetable.*

5.6 has three equivalence classes, denoted by gray, black, and black dotted lines. Breaking the cycle on the edge between B and C would result in a timetable with the same number of three exams in a day as breaking on the edge between H and I. A final strategy we suggest is breaking the sequence so that certain exams are strategically placed. For example, if we require exam L in Figure 5.6 to occur on the first day of exam week, we have limited the set of edges for which we can break the cycle. These strategies can be used simultaneously. For example, we can use the first two by choosing the equivalence class with the fewest number of multiple exams in a day and then breaking on the edge of maximum value within the equivalence class.

Once an edge has been removed from the sequence, we have an ordering of exam periods. This ordering will preserve the number of consecutive exams and the number of multiple exams in a day whether it is enforced directly or in reverse. These final ordering decisions are often left to policy makers.

Recall from Section 5.2 that [Fisher and Shier 1983] develop a heuristic for solving the sequencing problem of the examination timetabling problem. Because we have the ability to solve the problem to optimality using Concorde, we do not use their initial solution to the problem given in (5.4). Furthermore, our TSP sequencing methodology includes more

criteria than just consecutive exams. However, in cases where other constraints were present such that we could not use the optimal sequence, we did utilize (5.4) as a starting solution.

The second formulation for sequencing the exam clusters is an extension of the work of Levchenkov et al. [2006]. Recall from Section 5.2 that the work on the Cornell timetable does not include a clustering phase as the number of course times is fewer than the number of exam periods. Also recall that the Cornell formulation is a two-stage sequencing problem specifically designed for a 7-day, 21-period exam schedule. When we are working on a 7-day, 21-period exam schedule, we use the formulation given by [Levchenkov et al. 2006] directly on the established clusterings with two exceptions. First, we remove the specific constraints used for the Cornell timetable and add our own specific constraints. Second, instead of limiting our search to the $C_3^{21} = 1,330$ unordered triples and fixing the middle element, we examine all $P_3^{21} = 7,980$ ordered triples. This widens our feasible region and allows us to consider more permutations of the set of clusters in stage one.

We are interested also in a 5-day, 20-period exam schedule, so we extend the Cornell formulation to accommodate an extra period per day. We define a quadruple here as a set of four clusters of course times. We examine $C_4^{20} = 4,845$ unordered quadruples and consider more criteria in each phase because of the fourth cluster being added. Because of the great number of variables and limited computing resources, examining the $P_4^{20} = 116,280$ ordered quadruples is impractical. In addition to the criteria examined in the 7-day, 21-period formulation, we also consider four exams in a day and three consecutive exams in a day. The complete algorithm, less the specific constraints, is given in Formulations 5.8 and 5.9.

Our extension of the formulation given by [Levchenkov et al. 2006] remains a two-stage approach. The first stage attempts to minimize the number of four exams in a day, three exams in a day, two exams in a day, and three consecutive exams in a day by identifying the quadruples to be grouped together each day. The first three criteria do not affect the ordering of the clusters in each quadruple. To minimize the fourth criterion, we fix the inner two clusters, but do not order them. Similar to the Cornell formulation for fixing the center element of a triple, to fix the two inner clusters and evaluate the number of three consecutive exams in a day, we add the number of common students in all $C_3^4 = 4$ triples

Formulation 5.8: Extension of sequencing formulation by Levchenkov et al.: Stage 1

Goal: Minimize the weighted sum of 4 exams in a day, 3 exams in a day, 3 consecutive exams in a day, and 2 exams in a day by identifying the quadruples to be grouped together each day.
Preprocessing: Fix the inner two clusters in each possible quadruple (by minimizing 3 consecutive exams in a day).
Number of variables: $C_4^{20} = 4,845$

Notation, variable definitions
$E = \{1, 2, ..., 20\}$ is the set of clusters.
$T_{ij}^{(2)}$=number of students who have exams for courses in clusters i and $j \in E$.
$T_{ijk}^{(3)}$=number of students who have exams for courses in clusters i, j, and $k \in E$.
$T_{ijkl}^{(4)}$=number of students who have exams for courses in clusters i, j, k, and $l \in E$.
$x_{ijkl} \in \{0, 1\}$ determines whether clusters within the quadruple
$(i, j, k, l) \in T = \left\{(i, j, k, l) \in E^4 : i < j < k < l\right\}$ are scheduled on the same day.

Performance measures:
• 4 in a day $D4 = \sum_{(i,j,k,l)\in T} x_{ijkl} T_{ijkl}^{(4)}$
• 3 in a day $D3 = \sum_{(i,j,k,l)\in T} x_{ijkl} \left(T_{ijk}^{(3)} + T_{ikl}^{(3)} + T_{ijl}^{(3)} + T_{jkl}^{(3)}\right)$
• 3 consecutive in a day $RD3 = \sum_{(i,j,k,l)\in T} x_{ijkl} \left(T_{ijk}^{(3)} + T_{ikl}^{(3)} + T_{ijl}^{(3)} + T_{jkl}^{(3)} - \max\left\{T_{ijk}^{(3)}, T_{ikl}^{(3)}, T_{ijl}^{(3)}, T_{jkl}^{(3)}\right\}\right.$
$\left. - \max\left\{\left\{T_{ijk}^{(3)}, T_{ikl}^{(3)}, T_{ijl}^{(3)}, T_{jkl}^{(3)}\right\} \setminus \max\left\{T_{ijk}^{(3)}, T_{ikl}^{(3)}, T_{ijl}^{(3)}, T_{jkl}^{(3)}\right\}\right\}\right)$
$\Rightarrow$ the middle two clusters are fixed, but not ordered.
• 2 in a day $D2 = \sum_{(i,j,k,l)\in T} x_{ijkl} \left(T_{ij}^{(2)} + T_{ik}^{(2)} + T_{il}^{(2)} + T_{jk}^{(2)} + T_{jl}^{(2)} + T_{kl}^{(2)}\right)$

Problem Formulation:
$\min \sum_{(i,j,k,l)\in T} \{w_1 D4 + w_2 D3 + w_3 RD3 + w_4 D2\}$
s.t. $\sum_{m\in(i,j,k,l)\in T} x_{ijkl} = 1 \ \forall m \in E$
where w_1, w_2, w_3, and w_4 are weights determined by the decision maker.

Let $S1$ = the solution set.

Formulation 5.9: Extension of sequencing formulation by Levchenkov et al.: Stage 2

Goal: Minimize the weighted sum of 2 consecutive exams in a day, 2 consecutive exams (including overnight), and 3 consecutive exams (including overnight) by sequencing and ordering the five quadruples given in S1. Note that each quadruple can be ordered in one of four ways: center elements forward or backward, end elements forward or backward.
Number of variables: $5! \cdot 4^5 = 122,880$

Notation, variable definitions
$y_i \in \{0, 1\}$, $1 \leq i \leq 122,880$, represents a permutation of S1, with each quadruple ordered one of four ways as noted.
$Y = \{y_1, \cdots, y_{122,880}\}$ is the set of permutations of S1, with each quadruple ordered one of four ways as noted.

Performance measures:
• 2 consecutive in a day $RD2_i$ for permutation i
• 2 consecutive $R2_i$ for permutation i
• 3 consecutive $R3_i$ for permutation i

Problem Formulation:
$\min \sum_{i=1}^{122,880} (v_1 RD2_i + v_2 R2_i + v_3 R3_i) y_i$
s.t. $y_i \in Y$
where v_1, v_2, and v_3 are weights determined by the decision maker.

Let y^* be the optimal solution.

in the quadruple and then subtract the values from the two triples with the most common students. An example of these calculations is given in Figure 5.7. Also note the formulation of the criterion three consecutive exams in a day, RD3, in Formulation 5.8.

The result of the first stage is five unordered quadruples, with the two center elements fixed. The second stage attempts to minimize the number of two consecutive exams in a day, two consecutive exams (including overnight), and three consecutive exams (including overnight) by sequencing and ordering the five quadruples. Note that each quadruple can be ordered in one of four ways: center elements forward or backward, end elements forward or backward. Thus, the number of permutations of the five quadruples is $5! \cdot 4^5 = 122,880$. Although this number is manageable, in practice the number is smaller due to specific constraints making many permutations infeasible. Similar to the original algorithm by [Levchenkov et al. 2006], our second stage is solved also by complete enumeration of the permutations and orderings of the five triples.

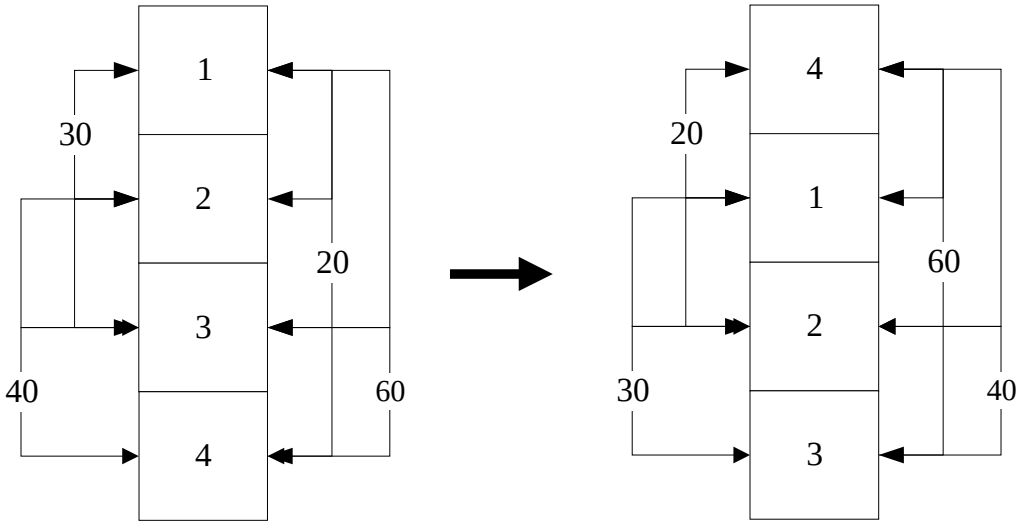


Figure 5.7: Fixing the two middle elements of a quadruple. *This diagram shows that to fix the two middle elements of a triple, we separate the two triples with the highest numbers of common students.*

We use Fisher and Shier's [1983] concept of swapping clusters within the timetable to search for a local optimal solution about the current timetable, S, at which we have arrived thus far (after the clustering and sequencing phase). The authors outline their procedure, which they name the Examination Scheduling Problem (ESP) Heuristic Procedure, in [Fisher and Shier 1983], and we reproduce it in Algorithm 5.10 changing only notation. Note that n is the number of exam periods.

Formulation 5.10: Examination Scheduling Problem (ESP) Heuristic Procedure

---

ESP Heuristic Procedure

1. Calculate the number of consecutive exams of the timetable S. Let this be $f(S)$.
2. Set $m = 0$, $i = 0$.
3. Set $i = i + 1$. If $i \geq n$, set $i = 1$.
4. Set $j = i + 1$.
5. If $j > n$ go to Step 3.
6. Set $m = m + 1$. If $m > n(n-1)/2$, then STOP. Let S' be the timetable with periods i and j exchanged. If $f(S) - f(S') > 0$ then interchange periods i and j in S, and set $m = 0$.
7. Set $j = j + 1$. Go to Step 5.

The results of using this heuristic are presented in Section 5.5. Unlike the problem discussed in [Fisher and Shier 1983], many institutions have more course times than exam periods and must cluster the course times into the number of exam periods before sequencing. For this reason, we extended the ESP heuristic procedure of Fisher and Shier to swap elements within clusters in order to minimize the number of consecutive exams. This swapping changes the clustering and the sequencing of the examination timetable. The results of this extension are presented in Section 5.5.

We also considered moving elements from one cluster to another, but not swapping, since some clusters contain more elements than others. For example, one cluster might have course times 3:00MWF and 3:30MW while another has only 10:10MWF. Instead of enforcing swapping, which ensures that the number of elements in each cluster remains fixed, we considered moving elements from one cluster to another. Without further restrictions, this would result in every other exam period being empty as the empty periods ensure a minimum number of consecutive exams. Because we are satisfied with the quality of the solutions up to this point, we leave this potential heuristic for future research.

Empirical results comparing the two sequencing algorithms are given in Section 5.5.

5.4.4 Implementation

In implementing the exact ILP formulation for clustering, we used CPLEX with mixed results [CPLEX 2006]. We used CPLEX version 10.0 on a Sun V440 with 16GB RAM, 4x1.6GHz CPU, running Solaris 10. CPLEX was the only commercial software we found able to manage the large problems—MATLAB does not guarantee convergence of the QP and requires too much memory for the IP; SAS/OR was not able to solve the QP as of this writing; the COIN/OR QP solver Bonmin and the CPLEX QP solver also does not guarantee convergence of non-convex QP's [MATLAB 2006; Bonami et al. 2007; SAS/OR 2003]. The COIN/OR QP solver Bonmin (Basic Open-source Nonlinear Mixed INteger programming) can be downloaded for free at https://projects.coin-or.org/Bonmin. The algorithms used by Bonmin are discussed in [Bonami et al. 2005]. The computation time for CPLEX to solve the ILP was extremely long. The minimum run time was more than seven days. The implementation of the clustering heuristics we developed was more efficient in terms

of memory usage and execution time. We programmed both of our clustering algorithms using MATLAB, and execution time for a run on one semester's worth of data was a matter of seconds.

Another implementation issue is the amount of data to use. We use several strategies for determining the best clusterings shown in Table 5.7 including: (1)-(2) using the average distance matrix from the previous four (six) semesters and (3)-(4) running the clustering algorithm on each of the past four (six) semesters, pregrouping the clusterings found repeatedly in more than a fixed percentage of the semesters and then running again on the average of the past four (six) semesters. We use these strategies to find the best four clusterings based on the number of conflicts for each of the past six semesters. We then run the sequencing phase. We compare each timetable based on conflicts, two consecutive exams, and multiple exams in a day for each of the past (four) six semesters. If this comparison reveals one timetable that dominates all the others in all aspects, the choice is obvious. However, the final decision is often left to qualitative considerations such as placement of common exams and large enrollment course times.

Adding constraints to the heuristic timetabling methods we have presented is possible and was necessary to complete the timetabling solutions for the empirical section of this study. For example, if two course times are required to have the same exam time, then the course times are pre-grouped prior to running the clustering algorithm. If certain course times are required to have exam times in the evening while others must meet during the day, two simultaneous clusterings can be done for each set of exam times. If certain course times are required to have exam times within a subset of periods during exam week, this constraint can be handled in the sequencing formulations. In particular, we can use a special structure of the local search procedure given in Algorithm 5.10 in which the swapping must

Strategy	Data	Description
1	4 semesters	Average distance matrix for clustering and sequencing.
2	6 semesters	
3	4 semesters	Individual clustering; pre-group repeated clusterings and run on average; average sequencing.
4	6 semesters	

Table 5.7: Four strategies for using historical data in the exam timetabling problem.

occur only in subsets of exam periods to ensure that evening courses are mapped to evening exams, common exams occur within the correct periods, etc.

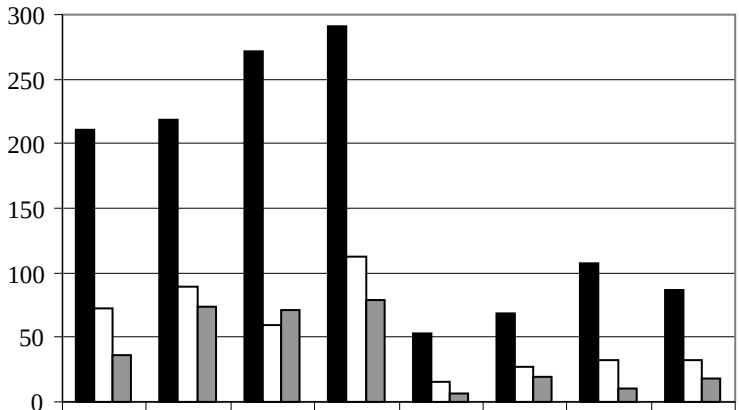
5.5 Results

In this section we present results for our examination timetabling algorithms tested on Clemson University data. First we present the clustering results comparing the PAC and the hierarchical clustering algorithms. We also examine the clustering results from the exact formulation programmed as an ILP and a QP. We then consider results from the two sequencing formulations presented previously: the STSP and the extension of the Levchenkov et al. formulation. We also present local search results using the swapping heuristic proposed by Fisher and Shier [Fisher and Shier 1983]. Finally, we present the proposed timetables for CU. We compare the statistics from these timetables to the current timetable.

Clustering results

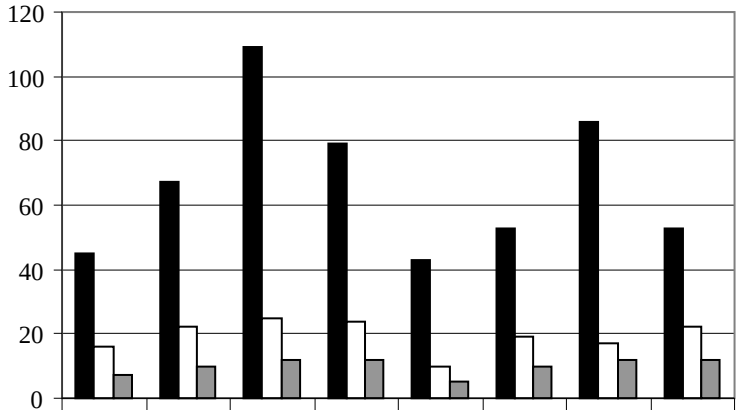
Both the hierarchical and the PAC clustering algorithms perform well on CU data. The results for some of the clusterings are shown in Figures 5.8 and 5.9. Figure 5.8 shows that the PAC method yields significantly fewer conflicts than the hierarchical method when used on the individual course times prior to pregrouping. However, after the data has been pregrouped into fewer clusters, the PAC method and the hierarchical method perform well, but the PAC method still dominates the hierarchical method (Figure 5.9).

Figure 5.10 shows the solutions for the exact (ILP) formulation, as given in Formulation (5.11) with $f_1(d_{ik}) = d_{ik}$ and $f_2(d_{ik}) = 0$, compared to the PAC and hierarchical methods when clustering 32 elements into 18 exam periods. Recall that pre-grouping the course times into groups prior to clustering yields built-in conflicts. For fall 2004, spring 2005, fall 2005, and spring 2006 there were 178, 167, 328, and 460 built-in conflicts, respectively. We are only counting the number of conflicts generated by the clustering algorithms, not the built-in conflicts. For this problem, the ILP has $32 \cdot 18 + 18 \cdot \sum_{i=1}^{31} i = 9,504$ binary variables. We use CPLEX to run the program. We also try solving the exact QP formulation as given in (5.1) using the COIN-OR solver Bonmin. The QP has $32 \cdot 18 = 576$ binary variables. We



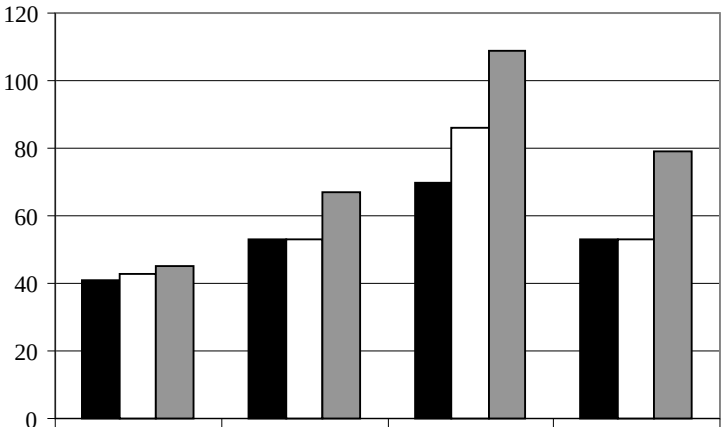
	F 2004	S 2005	F 2005	S 2006	F 2004	S 2005	F 2005	S 2006
	Hierarchical				Partitioning			
■ 18 period	211	218	271	291	53	68	107	86
□ 20 period	73	89	59	112	16	27	32	32
■ 21 period	36	74	71	79	7	19	10	18

Figure 5.8: Conflicts generated by two clustering methods on 98 elements.



	F 2004	S 2005	F 2005	S 2006	F 2004	S 2005	F 2005	S 2006
	Hierarchical				Partitioning			
■ 18 period	45	67	109	79	43	53	86	53
□ 20 period	16	22	25	24	10	19	17	22
■ 21 period	7	10	12	12	5	10	12	12

Figure 5.9: Conflicts generated by two clustering methods on 32 pre-grouped elements.



	Fall 2004	Spring 2005	Fall 2005	Spring 2006
ILP	41	53	70	53
Partitioning	43	53	86	53
Hierarchical	45	67	109	79

Figure 5.10: Conflicts generated by three clustering methods on 32 pre-grouped elements into 18 clusters.

use the NEOS server, found at http://neos.mcs.anl.gov/neos/, for running Bonmin [Czyzyk et al. 1998; Dolan 2001; Gropp and Moré 1997]. The running times were relatively short, but the solution quality was extremely bad, 4,921, 4,228, 4,769, and 4,674 conflicts for fall 2004, spring 2005, fall 2005, and spring 2006, respectively. Bonmin does not guarantee convergence for non-convex objective functions, and our formulations are non-convex.

Figure 5.10 shows that the PAC method finds the optimal solution for two of the four problems. Both heuristics find solutions close to the optimal solutions in much less time. The reason that we have few results on the exact formulation is the length of time it takes to arrive at a solution: a minimum of seven-days. Many problems continued to run in CPLEX for more than 79 days before we ended the programs without an integer solution. The run-time for our heuristic methods is seconds even taking into account random orderings and multiple runs. These results are typical for the 20-period and 21-period clusterings as well.

Sequencing results

To compare the two sequencing strategies—the TSP formulation versus the formulation by [Levchenkov et al. 2006]—we complete the clustering phase of our exam timetabling algorithm and sequence the established clustering using both strategies. Table 5.8 shows

Performance measures (exams)	Extended Levchenkov et al. formulation	STSP formulation
4 in a day	16	16
4 consecutive (including overnight)	58	50
3 in a day	912	1,023
3 consecutive	569	353
3 consecutive (including overnight)	735	655
2 in a day	11,232	11,207
2 consecutive	7,073	4,311
2 consecutive (including overnight)	7,523	5,205

Table 5.8: Comparison of average sequencing results over fall 2004 to spring 2006 using the STSP or the extended Levchenkov et al. formulation.

the average criteria over the fall and spring semesters between 2004 and 2006 when twenty clusters are sequenced using the STSP or the extended Levchenkov et al. formulation. Both formulations perform well, but the STSP formulation has significantly lower values for two consecutive exams and two consecutive exams including overnight. This is due to the restricted number of permutations considered by the extended Levchenkov et al. formulation. Because of the two phase approach of that formulation and the fixing of the quadruples in the first phase, the second phase has a limited region in which to search for an optimal solution.

Local search results

When we have constraints that do not allow us to use the optimal TSP sequencing solution, we use an extension of the ESP heuristic procedure by Fisher and Shier [1983] to perform local searches after completing our exam timetabling algorithms in which the objective is to minimize consecutive exams. Instead of allowing the heuristic freedom to swap any pair of clusters, we restrict the heuristic to swapping clusters within two sets of four mutually exclusive groups. The groups here are particular to a five-day, twenty-period schedule. The first set of groups are (1) clusters mapped to the evening periods (last period) of the first three days, (2) clusters mapped to the evening periods of the last two days, (3) clusters mapped to the day periods (first three periods) of the first three days of exam week,

and (4) clusters mapped to the day periods of the last two days of exam week. The second set of groups are (5) clusters with common exams and evening course times, (6) clusters with common exams and no evening course times, (7) all other clusters with evening course times, and (8) all other clusters. These groups reflect the restrictions Clemson has placed on its exam timetable, including evening course times must be mapped to evening exam periods and common exams must occur within the first three days of exam week. Elements within each group may be swapped. However, we have to keep track of the exchanges because swapping within one group may cause another set of groups to change. For example, if we exchange a pair of clusters in set (1), sets (6) and (5) may be changed. If we exchange a pair of clusters in set (8), sets (3) and (4) may be changed.

Because the number of course times exceeds the number of exam periods for all of our timetables, many exam periods have multiple course times mapped to them. We will refer to course times as elements of a cluster for the present discussion. Thus, many clusters contain multiple elements. To expand our search, we consider swapping elements instead of entire clusters. This gives us more freedom to explore the decision space. Note that exchanging elements results in a new clustering and sequencing, whereas swapping clusters allowed the clustering to remain fixed. Swapping elements is more challenging than swapping clusters because, for example, an element that is not a common element or an evening element may be in a cluster with those elements. This non-special element is allowed to be mapped to any exam period. However, the common element or the evening element still has restrictions about where it can be mapped. We restrict element swapping using the same two sets of groups outlined above. This restricts the decision space such that it is not possible for non-special elements in clusters with common or evening elements to swap out of clusters with common or evening elements. For example, given a starting timetable with the element (course time) 12:20MWF mapped to the twelfth exam period of a twenty-period, five-day exam timetable, that element may be swapped to clusters four, eight, sixteen, and twenty. However, under our swapping restriction, it can not be mapped to any exam periods during the day (the first three periods) even though it is not an evening course time. Because this is simply a local search meant to make marginal improvements in the number of consecutive exams after our timetabling algorithms are completed, we do not mean for

Performance measures (exams)	Proposed 5 day Timetable		
	No swapping	Cluster Swapping	Element Swapping
conflicts	346	346	501
4 in a day	16	18	19
4 consecutive	50	61	47
(including overnight)			
3 in a day	1,023	1,119	1,053
3 consecutive	353	376	369
3 consecutive	655	628	617
(including overnight)			
2 in a day	11,207	10,972	10,754
2 consecutive	4,311	4,303	4,276
2 consecutive	5,205	5,025	4,858
(including overnight)			

Table 5.9: Comparison of average performance measures over fall 2004 to spring 2006 for three alternatives of the proposed 20-period, 5-day timetable.

it to be exhaustive. Complete swapping may be another area to examine in future research along with element moving.

Table 5.9 presents the performance measures for three alternatives of the proposed twenty-period, five-day timetable. The three alternatives are the proposed five-day timetable as given by our timetabling algorithms with no swapping, with cluster swapping, and with element swapping. The element swapping alternative significantly increases the average number of conflicts. This is the drawback of allowing for more freedom in exploring the search space. The cluster swapping alternative shows slightly lower numbers of three consecutive exams including overnight, two exams in a day, two consecutive exams, and two consecutive exams including overnight. However, the numbers are not much improved over the proposed five-day schedule with no swapping and are worse in several instances.

Final results

We measure the timetabling criteria for the current exam schedule at CU and present the results in Table 5.10 and Figure 5.11. The current schedule is a seven-day, twenty-one period schedule that runs from Saturday to Saturday, skipping Sunday. Figure 5.11 shows the average the number of students taking exams each period for the semesters fall 2004–spring 2006. Table 5.10 shows the performance measures for each semester and

Performance measures (exams)	Fall 2004	Spring 2005	Fall 2005	Spring 2006	Average
conflicts	239	230	437	588	374
4 in a day	0	0	0	0	0
4 consecutive	121	121	126	130	125
(including overnight)					
3 in a day	339	301	404	319	341
3 consecutive	339	301	404	319	341
3 consecutive	895	823	892	783	848
(including overnight)					
2 in a day	8,230	7,840	8,485	7,510	8,016
2 consecutive	5,850	5,465	5,679	5,140	5,534
2 consecutive	6,801	6,512	6,691	6,220	6,556
(including overnight)					

Table 5.10: Summary of the current 7-day, 21-period exam schedule at CU.

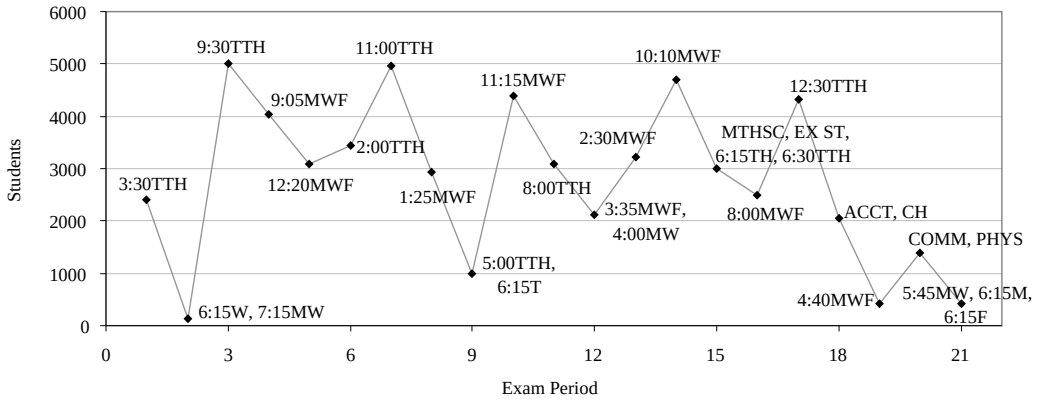


Figure 5.11: Current 7-day, 21-period exam schedule.

the average of all semesters presented. These numbers serve as a baseline for comparison against our proposed timetables.

In the current schedule, the last exam period is also used as a conflict-resolution period, which is a pre-specified period designed for students to take exams in the event that they have a conflict in their exam schedule. However, the current schedule has course times mapped to the conflict-resolution period. Thus, conflicts with the conflict-resolution period could occur. Current policy allows a student with three exams in a day to move an exam. If no other time can be found, the conflict-resolution period may be used. There is no empirical data showing how often the conflict-resolution period is used. Because it is the last period on the last day of exam week, we speculate that it is not widely used

Performance measures (exams)	Fall 2004	Spring 2005	Fall 2005	Spring 2006	Average
conflicts	234	225	397	529	346
4 in a day	15	14	13	20	16
4 consecutive	62	41	49	47	50
(including overnight)					
3 in a day	1,010	965	1,049	1,069	1,023
3 consecutive	384	337	368	322	353
3 consecutive	704	607	671	637	655
(including overnight)					
2 in a day	11,568	10,812	11,814	10,634	11,207
2 consecutive	4,595	4,025	4,767	3,857	4,311
2 consecutive	5,274	4,991	5,693	4,861	5,205
(including overnight)					

Table 5.11: Summary of the proposed 5-day, 20-period exam schedule.

even by students with conflicts. We do not include a conflict-resolution period in our proposed five-day, twenty-period timetable because it innately causes more conflicts. This is so because a conflict-resolution period takes the place of a regularly scheduled exam period, leaving fewer exam periods in which to fit the entire exam schedule. We do include a conflict-resolution period in our proposed twenty-one period schedule because the current twenty-one period schedule has one. If a conflict-resolution period is desired in our other timetables, we suggest that the exam period with the lowest number of exams is chosen.

Note that the common exams are given in the 15^{th}, 18^{th}, and 20^{th} exam periods in the current timetable. The average numbers of students taking scheduled exams on the first day and final day are around 7,500 and 2,250, respectively. The average number of students sitting for exams on any given day is around 8,400.

We created a five-day, twenty-period exam timetable because of the desires of students to eliminate the first Saturday of exam week (so that there is a reading day built into the schedule) and the desires of the faculty to eliminate the second Saturday of exam week (so that there is more time for grading before graduation grade collection deadlines). The twenty-period schedule consists of five days with four exams per day and no conflict-resolution period. Table 5.11 and Figure 5.12 show the relevant properties of the schedule had it been implemented during the fall and spring semesters of 2004–2006. This schedule is appealing because it eliminates both the first and last Saturdays of the current exam sched-

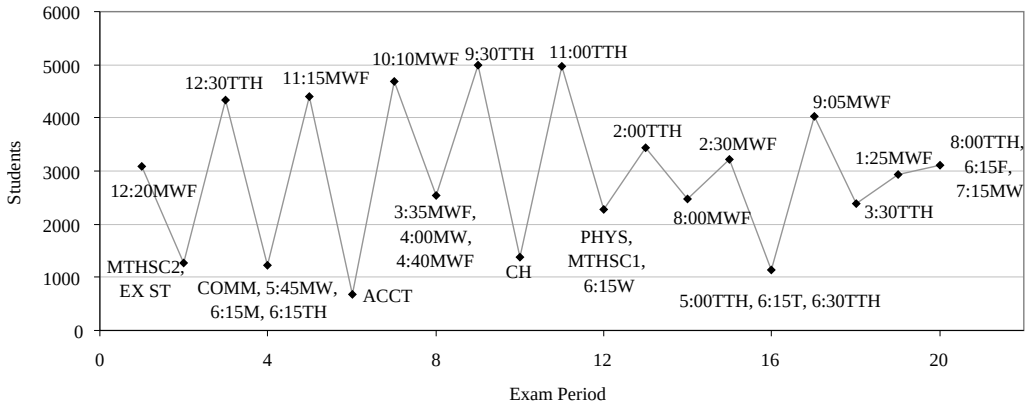


Figure 5.12: Proposed 5-day, 20-period exam schedule.

ule. The trade-off between eliminating both Saturdays is that the number of exams taken per day increases to around 11,700, which is about 7/5 of 8,400, or 40 percent greater. This multiplier is expected because we are compressing a seven-day schedule into five.

We also considered an eighteen and a twenty-one period exam schedule. The eighteen-period schedule would consist of six days with three three-hour exams per day and no conflict-resolution period running from Monday to Saturday. Because the schedule was compressed to eighteen periods, the number of conflicts increased and the number of exams held on the last day increased significantly. Thus, this schedule was abandoned as an option.

The twenty-one period exam schedule consists of seven days with three exams per day and a conflict-resolution period in the last exam period. This schedule is the same length as the current schedule, but has only twenty dedicated exam periods. Thus, the clustering is different from the current timetable. Because we cluster the course times into twenty periods, leaving one period open for the conflict-resolution period, we use the same clustering as we used in the twenty-period schedule. We re-sequence to minimize the number of conflicts, two consecutive exams, and three consecutive exams. Table 5.12 and Figure 5.13 show the relevant properties of the schedule had it been implemented during the fall and spring semesters of 2004–2006.

Notice that the number of students sitting for exams in the proposed schedules have a consistent high-low high-low property. This is due to efforts to minimize consecutive exams during the second phase of our algorithm. When there are an even number of exam

Performance measures (exams)	Fall 2004	Spring 2005	Fall 2005	Spring 2006	Average
conflicts	234	225	397	529	346
4 in a day	0	0	0	0	0
4 consecutive (including overnight)	105	99	116	118	110
3 in a day	332	471	374	434	403
3 consecutive	332	471	374	434	403
3 consecutive (including overnight)	845	877	1,030	813	891
2 in a day	8,482	8,064	8,739	7,947	8,308
2 consecutive	5,206	4,481	5,280	4,450	4,854
2 consecutive (including overnight)	5,711	5,152	5,974	5,207	5,511

Table 5.12: Summary of the proposed 7-day, 21-period exam schedule.

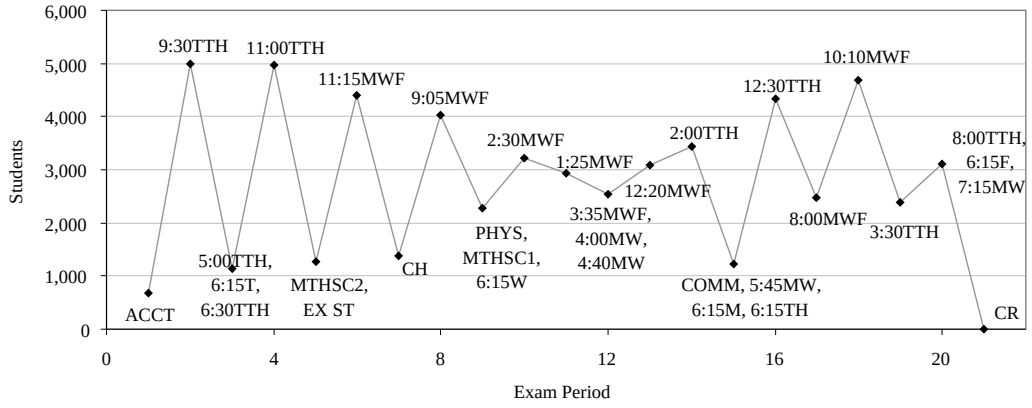


Figure 5.13: Proposed 7-day, 21-period exam schedule.

Performance measures (exams)	Current 21 period 7 day	Proposed 20 period 5 day	Proposed 21 period 7 day
conflicts	374	346	346
4 in a day	0	16	0
4 consecutive	125	50	110
(including overnight)			
3 in a day	341	1,023	403
3 consecutive	341	353	403
3 consecutive	848	655	891
(including overnight)			
2 in a day	8,016	11,207	8,308
2 consecutive	5,534	4,311	4,854
2 consecutive	6,556	5,205	5,511
(including overnight)			

Table 5.13: Comparison of current and proposed timetables.

periods each day of a timetable, this high-low, high-low property results in the number of students sitting for exams each day to be distributed more evenly than when each day has an odd number of periods. The number of exam periods per day may also affect other parts of a timetable. In particular, if certain clusters are required to be mapped to specific exam periods, problems may arise if the clusters represent to few or too many students. At CU, evening course times are mapped to evening or weekend exam periods. The four periods per day in the twenty-period schedule is convenient for this mapping because the evening course times typically have lower enrollment, so the schedule can have the high-low high-low property without any adjustment to the clustering. In the twenty-one period schedule, which has three exam periods per day, we could not have this high-low, high-low property without mapping some non-evening clusters to the evening exam periods. Because there are more exam periods to which we can map evening clusters than there are evening clusters, this was not a problem. However, it could be the case that clusterings have to be adjusted in order to comply with policy restrictions.

Table 5.13 shows a comparison of the statistics on the current and proposed exam schedules. In both proposed schedules, the number of conflicts and the number of two consecutive exams are less than in the current schedule. The proposed twenty-period schedule has significantly more occurrences of two and three exams in a day than either of the twenty-one

Block	Performance measures (exams)	Current 21 period 7 day	Proposed 20 period 5 day	Proposed 21 period 7 day
4 exam block	4 in a block	8	3	6
	3 in a block	238	192	242
	2 in a block	2,099	2,240	2,203
3 exam block	3 in a block	49	42	57
	2 in a block	1,145	1,243	1,188

Table 5.14: Comparison of spacing for the current and proposed timetables.

period timetables. This is expected and it is due to eliminating two days from the schedule and adding a fourth exam period to each day. Instead of an average of 8,400 exams per day in the current twenty-one period schedule, there are an average of 11,700 exams per day in the proposed twenty-period schedule. In the twenty-one period schedule, there is only one way in which to have three exams per day—they must be consecutive. In the twenty-period schedule, there are $C_3^4 = 4$ ways in which to have three exams per day. Thus we would expect the number of three exams in a day in the twenty-period timetable to be four times the number in the twenty-one period timetable. It is exactly that compared to the current timetables and less than that compared to the proposed twenty-one period timetable. In the twenty-one period schedule there are $C_2^3 = 3$ ways in which to have two exams in a day while there are $C_2^4 = 6$ ways in the twenty-period schedule. Thus, we would expect the number of two exams in a day in the twenty-period timetable to be twice the number in the twenty-one period timetable. In fact, we see that the number of two exams in a day in the twenty-period schedule is close to 7/5 of the number in the current and proposed twenty-one period timetable.

The numbers in Table 5.13 are somewhat misleading in that they do not show a standard measure for the spread of the exams within the schedules. This is because two of the schedules have three exams per day while the five-day schedule has four exams per day. Table 5.14 shows the spread of the exams by considering blocks of exams in each schedule and counting the number of occurrences of multiple exams. We consider exam blocks of length three (three-exam blocks) and four (four-exam blocks) because these are the lengths of exam days in the various schedules. To find the spread in four-exam blocks in the twenty-one period, seven-day schedules, we compute the statistics on periods 1-4, 5-9, ..., 17-20

and then again 2-5, 6-10, . . .,18-21. We average the two to get the numbers in Table 5.14. Similarly, we take an average on three sets of periods for the eighteen period schedule. In a similar manner we find the spread in three-exam blocks in the twenty-period schedule. Table 5.14 shows that the proposed twenty-period schedule is spread out nearly as much as the current and proposed twenty-one period schedules even though it has more students on average in each exam period because. The proposed twenty-one period schedule has approximately commensurate spread as the current schedule. The proposed schedule has a dedicated conflict-resolution period, has the common exams nearer to the beginning of the week, and has fewer exams on the first day of exam week than the current schedule. Without these restrictions, the twenty-one period schedule could be the more spread out.

In summary, our timetabling algorithms have produced two alternative timetables of varying length. Each timetable surpassed at least some of the performance measures of the current timetable. Because our algorithms are straightforward, need very little run time, and produce quality results in comparison to the current timetable, they are desirable for creating reusable examination timetables.

Chapter 6

Conclusions

In this study we developed and presented procedures for modeling university populations, predicting course enrollments, allocating course seats to student groups, and timetabling final examinations. We briefly review the problems and our contributions.

Problem 1: Modeling University Student Populations

Modeling the student population of a university involves recording the number and characteristics of students in order to observe and predict changes in the population. We presented our procedure for building university population models and a specific model in Chapter 2. A main goal of this part of our study was to build a model for university enrollment that can be used for both long- and short-term projection. The results given in Sections 2.3.3, 2.3.4, and 2.4 show that we met that goal. The university enrollment model we presented uses an extended state space and the Markov property, rather than longitudinal data, to model student populations. The error rates for our projections are within the criteria we set, which reflects the error rates of variables exogenous to the model. The other main goal of this part of our study was to design a procedure for creating university population models. We met this goal in Section 2.3 where we described our procedure in detail, specifying when alternate decisions can be made to form different university population models. Testing the model's predictions as more information becomes available, using the model to develop strategies to adjust admission levels gradually in the face of changing retention and graduation rates or desired population changes, and extending the model to other colleges or universities are areas of further work.

Problem 2: Predicting Course Enrollments

Predicting course enrollment is an important part of building course schedules. We described our procedure for creating adaptive course prediction models in Chapter 3. We used

student characteristics to identify groups of undergraduates whose course enrollment rates are significantly different than the rest of the university population. Our model utilizes historical enrollment rates and current information to predict course enrollment for the coming semester. Our goal was to create models that make accurate predictions early in the course scheduling process and that can be updated as more information becomes available. Section 3.4 presented results that show we met this goal. The error rates for our course enrollment predictions were near or within the criteria we set. Areas for future work include continued prediction and testing, extension to other departments and universities, and model refinement as courses change.

Problem 3: Allocating Course Seats to Student Groups

Allocating course seats to specific student groups requires predicting the size of student groups, estimating the number of course seats needed by each group, and holding and releasing seats such that the correct number of seats are available to the specific groups. We developed a system that addresses these problems in Chapter 4. The course prediction model described in Chapter 3 aids in the system for allocating course seats to new students during summer registration sessions. Our goals were to estimate seat need accurately and develop a system that hedged our estimates without undermining the objective to give students similar enrollment choices no matter when they register. In meeting these goals, we addressed the concerns on how to estimate seat need each session, how to release seats among multiple course sections, and how to predict seat shortage and surpluses in Section 4.4. Our results presented in Section 4.5 show that we met our goals. There is great need to extend this work to other departments and universities.

Problem 4: Timetabling Final Examinations

In Chapter 5, we developed a two-phase heuristic procedure to build reusable final examination timetables. The first phase involves clustering course times into n groups, where n is the number of exam periods. The objective of this phase is to minimize the number of exam conflicts. We presented two distinct clustering heuristics for this phase: an established hierarchical technique and a novel partitioning technique. The objective of the second phase

is to minimize consecutive exams. We presented two methods for accomplishing this goal: formulating and solving the problem as a traveling salesman problem and formulating and solving the problem as an integer linear program. We finish the heuristic procedure with a local search for improved solutions.

Both the hierarchical and the PAC clustering algorithms produced relatively few conflicts when used on Clemson University data. We showed that the PAC algorithm achieves the optimal solution in several cases, and both heuristics we developed are near optimal in many cases. Both sequencing formulations performed well, but the STSP formulation had significantly lower values for two evaluation criteria. We showed that the heuristic procedure we developed yields timetables with fewer conflicts and fewer consecutive exams than the current timetable at Clemson University. Results are given in Section 5.5.

BIBLIOGRAPHY

Armstrong, D. F. and Nunley, C. W. 1981. Enrollment projection within a decision-making framework. *The Journal of Higher Education 52*, 295–309.

Balachandran, K. R. and Gerwin, D. 1973. Variable-work models for predicting course enrollments. *Operations Research 21*, 823–834.

Balakrishman, N., Lucena, A., and Wong, R. T. 1992. Scheduling examinations to reduce second-order conflicts. *Computers and Operations Research 19*, 353–361.

Black, J. 2004. *Essentials of Enrollment Management: Cases in the Field*. AACRAO, Washington, DC, USA.

Bonami, P., Belotti, P., Forrest, J. J., Ladanyi, L., Laird, C., Lee, J., Margot, F., and Waechter, A. 2007. Coin/or bonmin project. https://projects.coin-or.org/Bonmin.

Bonami, P., Biegler, L., Conn, A., Cornuejols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., N.Sawaya, and Waechter, A. 2005. An algorithmic framework for convex mixed integer nonlinear programs by ibm. *IBM Research Report RC23771*.

Brazziel, W. F. 1987. Forecasting older student enrollment: A cohort and participation rate model. *The Journal of Higher Education 58*, 223–231.

Brelaz, D. 1979. New methods to color the vertices of a graph. *Communications of the ACM 22,* 4, 251–256.

Burke, E. K., Elliman, D. G., and Weare, R. F. 1994. A university timetabling system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education 27,* 1, 1–18.

Burke, E. K. and Petrovic, S. 2002. Recent research directions in automated timetabling. *European Journal of Operational Research 140*, 266–280.

Campbell, L. H. 1975. A programming model for student enrollment planning. *Journal of the Australian Mathematical Society 19*, 30–39.

Carter, M. W. 1986. A survey of practical applications of examination timetabling algorithms. *Operations Research 34*, 193–202.

Carter, M. W., Laporte, G., and Lee, S. Y. 1996. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society 47*, 373–383.

Concorde 2005. Concorde traveling saleman problem solver. www.tsp.gatech.edu/concorde/index.html. Retrieved February 20, 2006.

CPLEX 2006. Cplex version 10.0. http://www.ilog.com/products/optimization/archive.cfm.

Cullen, C. G. 1990. *Matrices and Linear Transformations*, 2 ed. Dover, New York.

Czyzyk, J., Mesnier, M., and Moré, J. 1998. The neos server. *IEEE Journal on Computational Science and Engineering 5*, 68–75.

de Werra, D. 1985. An introduction to timetabling. *European Journal of Operational Research 19*, 98–110.

Dolan, E. 2001. The neos server 4.0 administrative guide. *Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory*.

Dooris, M. J., Kelley, J. M., and Trainer, J. F. 2004. *Successful strategic planning*. Jossey-Bass, San Francisco, CA, USA.

Ewell, P. 1995. *Student tracking: new techniques, new demands*. Jossey-Bass, San Francisco, CA, USA.

Fisher, J. G. and Shier, D. R. 1983. A heuristic procedure for large-scale examination scheduling problems. *Congressus Numerantium 39*, 399–409.

Glover, R. H. and Krotseng, M. V. 1993. *Developing executive information systems for higher education*. Jossey-Bass, San Francisco, CA, USA.

Gropp, W. and Moré, J. 1997. Optimization environments and the neos server. In *Approximation Theory and Optimization*. Cambridge University Press, 167–182.

Hartigan, J. A. 1975. *Clustering algorithms*. Wiley, New York, NY, USA.

Heiberger, R. M. 1993. Predicting next year's enrollment: Survival analysis of university enrollment histories. In *American Statistical Association, Social Statistical Section*. American Mathematical Society, 143–148.

Hopkins, D. S. and Massy, W. F. 1981. *Planning models for colleges and universities*. Stanford University Press, Stanford, CA, USA.

Karp, R. M. 1974. Complexity of computation. In *SIAM-AMS proceedings*. American Mathematical Society, Providence, RI, USA.

Kaufman, L. and Rousseeuw, P. 1987. *Statistical Data Analysis based on the L1 Norm*. Elsevier, North-Holland, Amsterdam, 405–416.

Kaufman, L. and Rousseeuw, P. 1990. *Finding groups in data : an introduction to cluster analysis*. Wiley series in probability and mathematical statistics. Wiley, New York, NY, USA.

Kulkarni, V. G. 1995. *Modeling and Analysis of Stochastic Systems*. Chapman and Hall Texts in Statistical Science Series. Chapman and Hall, London.

Laporte, G. and Desroches, S. 1984. Examination timetabling by computer. *Computers and Operations Research 11*, 351–360.

Levchenkov, D., Bland, R., and Shmoys, D. 2006. Scheduling final exams at cornell university. INFORMS 2006 Conference Presentation.

Lofti, V. and Cerveny, R. 1991. A final exam-scheduling package. *Journal of the Operational Research Society 42*, 205–216.

MARSHALL, K. T. 1973. A comparison of two personnel prediction models. *Operations Research 21*, 810–822.

MARSHALL, K. T. AND OLIVER, R. M. 1970. A constant-work model for student attendance and enrollment. *Operations Research 18*, 193–206.

MATLAB 2006. Matlab version 7.2. http://www.mathworks.com/products/matlab/.

MEHTA, N. K. 1981. The application of a graph coloring method to an examination scheduling problem. *Interfaces 11*, 57–65.

MURTAUGH, P. A., BURNS, L. D., AND SCHUSTER, J. 1999. Predicting the retention of university students. *Research in Higher Education 40*, 355–371.

PETROVIC, S. AND BURKE, E. 2004. University timetabling. In *Handbook of scheduling: algorithms, models, and performance analysis*, J. Y. T. Leung, Ed. Chapman & Hall/CRC, Boca Raton.

RENCHER, A. C. 1995. *Methods of multivariate analysis*. Wiley series in probability and mathematical statistics. Wiley, New York, NY, USA.

RODGERS, M. AND ZIMAR, H. 1993. *SEM Anthology*. AACRAO, Washington, DC, USA.

ROSS, S. M. 2002. *A First Course in Probability*, 6th ed. Prentice Hall, Upper Saddle River, New Jersey, USA.

SALLEY, C. D. 1979. Short-term enrollment forecasting for accurate budget planning. *The Journal of Higher Education 50*, 323–333.

SAS/OR 2003. Sas/or version 9.1. http://www.sas.com/.

SCHAERF, A. 1999. A survey of automated timetabling. *Artificial Intelligence Review 13*, 87–127.

SCHMID, C. F. AND SHANLEY, F. J. 1952. Techniques of forecasting university enrollment. *The Journal of Higher Education 23*, 483–488+502–503.

SEVIER, R. A. 2000. *Strategic Planning in Higher Education: Theory and Practice*. CASE Books.

SILVA, J. D. L., BURKE, E. K., AND PETROVIC, S. 2004. Multiobjective metaheuristics for scheduling and timetabling. In *Metaheuristics for Multiobjective Optimisation*, X. Gandibleux, M. Sevaux, K. Sorensen, and V. T'kindt, Eds. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin.

TITUS, M. A. 2004. An examination of the influence of institutional context on student persistence at 4-year colleges and universities: A multilevel approach. *Research in Higher Education 45*, 673–699.

T'KINDT, V. AND BILLAUT, J.-C. 2002. *Multicriteria Scheduling: Theory, Models, and Algorithms*. Springer-Verlag, Berlin.

WEILER, W. C. 1980. A model for short-term institutional enrollment forecasting. *The Journal of Higher Education 51*, 314–327.

West, D. 2001. *Introduction to Graph Theory*, 2 ed. Prentice-Hall, Upper Saddle River, NJ, USA.

White, G. and Chan, P. W. 1979. Toward the construction of optimal examination timetables. *INFOR 17*, 219–229.

Wolsey, L. A. 1998. *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York, NY, USA.