Math 480 - Course Project

Creating a schedule for "Chaos Week"

Derek Rhodehamel
Justin Zecak
Davis Zvejnieks

1 Abstract

MathILy is a mathematics summer camp in Philadelphia. Every summer in between regularly scheduled classes is an impromptu week-long set of classes decided by student interests. MathILy needs a way to quickly create an assignment of students and teachers to classes, and classes to timeslots, so that are no scheduling conflicts.

2 Problem description

MathILy is a summer math camp for high school students. In the middle of the camp there is a special week called the Week of Chaos. During the Week of Chaos a number of week-long classes are taught during some number of time slots (typically five). Each camper is scheduled for a class during each of the time slots, and each of the teachers at the camp is scheduled to teach no more than 4 of these classes. Before the week of chaos, students are asked to fill out a survey where they rate their levels of interest in the various classes on a scale of 0-3. Teachers are asked which classes they are qualified to teach and which classes they would most like to teach. Then the camp administrators have to create a Chaos Week schedule based on student and teacher preferences. Our goal for this project is to create a method to schedule based on an artificial set of student and teacher preferences which maximizes that satisfaction of student preferences, such that students are never scheduled for the multiple classes during a given time slot, every student is assigned to a class during each time slot, and the teaching load is evenly distributed between the teachers.

3 Our approach

At root, the problem is to find an optimal 3-matching. Unfortunately, finding a 3-or-more-matching is proven to be NP-complete, so a polynomial time algorithm is not known at this time. Thus we chose to formulate the problem as a linear program (LP), since this method is both flexible and relatively scalable.

The end result must contain all integer values, for example, we cannot assign a student to half of a class. Thus the problem is an integer programming (IP) problem, and we'll use an LP solution as a basis for our IP problem. To approach this problem we chose to implement the GNU Linear Programming Kit (GLPK) to solve the IP, as it is free to use with no license requirements or other restrictions.

Because there is no data yet for student preferences of 2016, we used randomized data fitting within the constraints and estimations of Dr. Ostroff. This is done to show the feasibility of using this approach for data sets larger than the 2015 data. While our approach is built on the requirements of MathILy's chaos week, our algorithm has been generalized to accept a variable amount for the scheduling requirements. This allows this method of scheduling to extend beyond just MathILy's Week of Chaos.

3.1 Variables

Let x represent student assignment into n classes. y represents instructor assignment into n classes. z represents time slots assignment of n classes. We set up the variables to be adjustable depending on the requirements in different circumstances.

$$h \in \{1, 2, ..., l\}\text{, where } l \text{ is the number of instructors}$$

$$i \in \{1, 2, ..., m\}\text{, where } m \text{ is the number of students}$$

$$j \in \{1, 2, ..., n\} \text{ where } n \text{ is the number of classes}$$

$$t \in \{1, 2, ..., s\} \text{ where } s \text{ is the number of timeslots}$$

$$x_{ij} = 1 \text{ if student } i \text{ is assigned to class } j\text{, 0 otherwise}$$

$$y_{hj} = 1 \text{ if instructor } h \text{ is assigned to class } j\text{, 0 otherwise}$$

$$z_{tj} = 1 \text{ if class } j \text{ is assigned to timeslot } t\text{, 0 otherwise}$$

We also incorporate the student preference matrix P and instructor eligibility matrix E in calculations as we seek to maximize their preferences and eligibility.

$P = (p_{ij})$, where p_{ij} represents preference rating of student i for class j

$E = (e_{hj})$, where e_{hj} represents eligibility rating of instructor h for class j

$F = (f_{ij})$, where $f_{ij} = 1$ if student$i$ is forced to take for class j, 0 otherwise

For the sample problem of 2016:

$$l = 5, m = 24, n = 15, s = 5$$

Our objective function is the sum of student ratings for their assigned classes plus the sum of teacher ratings for the classes they are assigned to teach.

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} p_{ij} + \sum_{h=1}^{l} \sum_{j=1}^{n} y_{ij} e_{ij}$$

We seek to maximize this function subject to the follow sets of constraints:

3.2 Classes

3.2.1 Classes Taken

Each student must be enrolled in 5 classes during Chaos Week.

$$\forall i \in \{1, 2, ..., m\}, \sum_{j=1}^{n} x_{ij} = 5$$

3.2.2 Class Size

Each class should have around 7 students at most. We broke this up into two constraints, by setting a class size minimum and maximum. We found that without setting a minimum, class distribution was uneven. Some classes had only two or three students, and this violated the needs of Dr. Ostroff.

$$\forall j \in \{1, 2, ..., n\}, \sum_{i=1}^{m} x_{ij} \leq 8$$

$$\forall j \in \{1, 2, ..., n\}, \sum_{i=1}^{m} x_{ij} \geq 5$$

3.2.3 Instructors per class limit

First, each class should only have one instructor. While the 2015 data showed that Dr. Ostroff taught a class with another instructor, we decided this would add unnecessary complexity for the algorithm. We suggest that, if another teacher is not scheduled at that time, he or she may co-teach.

$$\forall j \in \{1, 2, ..., n\}, \sum_{h=1}^{l} y_{hj} = 1$$

3.3 Instructor eligibility

The camp instructors provide information on which classes they are qualified to teach in addition to which classes they prefer to teach. It is a hard constraint that a teacher is never scheduled to teach a class that they are not qualified to teach. Thus for each class we add constraints stating that unqualified teachers cannot teach that class.

Eligibility for instructors is represented by the matrix E. This can be a matrix containing only binary numbers, or integer or real numbers. Using non-binary numbers may be useful in ranking instructors indicating possible back up instructors.

$$\forall j \in \{1, 2, ..., n\}, \sum_{h=1}^{l} y_{hj} e_{hj} \geq 0$$

3.4 Overrides

An interesting feature of our problem is that, even though the goal is to make a schedule that maximizes student preferences, instructors will frequently override student preferences. This is because students usually do not have a clear idea of what goes on in the various classes at the time when they take the survey. If the instructors feel that a student underrated a class that fits well with her general interests, or ought to be in a class because it addresses a hole in her skill set, they will simply place the student in the relevant class regardless of their preferences. In our model, overrides like these are added to the initial constraints.

Overrides constraints are calculated on a per student basis, reducing the number of constraints.

$$\text{if } f_{ij} = 1, \text{ then } x_{ij} = 1$$

3.5 Time constraints

Each day there is a fixed number of time slots during which classes can be scheduled. In our artificial data set we set this number of time slots to 5. Clearly we cannot schedule teachers to teach multiple classes during a single time-slot or schedule students to take multiple classes during a single slot. We formulated these constraints in the obvious way: the sum of classes assigned to each teacher during each time-slot must be less than or equal to 1, and for students this sum must be equal to one, since the students must take exactly one class during every time slot.

3.5.1 Number of classes per timeslot

This depends on the scheduling decided ahead of time by the program's organizers. With 15 classes and 5 timeslots, we create a constraint such that only 3 classes can occupy any given timeslot.

$$\forall t \in \{1, 2, ..., s\}, \sum_{j=1}^{n} z_{tj} = 3$$

3.5.2 Each class is assigned to only one timeslot

Each class should only occur once in the schedule. This ensures that the algorithm does not disregard the time conflicts.

$$\forall j \in \{1, 2, ..., n\}, \sum_{t=1}^{s} z_{tj} = 1$$

3.6 Student time conflicts

To prevent students being assigned to classes that occur in the same timeslot, we must add a large number of constraints per student. For any given selection of three classes (there are $\binom{15}{3} = 455$) the total of the binary variables representing timeslots and student enrollment must be less than 1 plus the number of simultaneous classes.

For example, consider timeslot 1, student 2:

$$t_{13} + t_{14} + t_{15} + x_{23} + x_{24} + x_{25}$$

Student 2 can be enrolled in classes 2, 3 and 4 as long as either only one of those classes are scheduled for timeslot 1, or none of them are. The issue

arises when the total is above 4, meaning there is a time conflict.

Let:

$$S_3 = \text{the set of all 3-size combinations of classes}$$
$$c \in S_3, c = \{j_1, j_2, j_3\}$$

$$\forall t \in \{1, 2, ..., s\}, \forall c \in S_3, \forall i \in \{1, 2, ..., m\} \sum_{b}^{3} z_{tj_b} + x_{ij_b} < 4$$

Because this creates 455 constraints for each student, in each timeslot, it adds a lot of time complexity. This method creates 54,600 constraints to the problem.

3.6.1 Instructor time conflicts

A similar method is used to prevent time conflicts for instructors. This adds 11,375 constraints.

$$\forall t \in \{1, 2, ..., s\}, \forall c \in S_3, \forall h \in \{1, 2, ..., l\} \sum_{b}^{3} z_{tj_b} + y_{ij_b} < 4$$

In the end this amounts to a very large number of constraints, and it still takes quite a while to find the optimal solution. Thankfully, given the nature of our problem it is not actually necessary to find the optimal solution, since the preference function we take as our objective function is essentially just a proxy for subjective and approximate measures of student and teacher preference. Therefore our model allows users to vary how long they want to let the linear program solver run, rather than waiting for the solver to run to completion. In a matter of minutes the user can compute a solution which is very good albeit mathematically suboptimal, but which will be optimal or close to optimal from the practical standpoint of students and schedulers.

To substantiate this claim we ran our model for just one minute using the preference data from 2015, and arrived at a schedule which was nearly identical to the actual schedule chosen by the MathILy instructors. The few discrepancies are most likely due to the overrides for 2015, which we do not have access to.

It should be noted that if the user does want to compute the mathematically optimal schedule, this can be done in a matter of hours for small to

medium groups of students. That is enough time to be inconvenient, but it is still an improvement over finding a matching of a graph or hyper-graph by hand.

4 Simplification

While this problem does not require a significant number of simplifications, some assumptions must be made about the data in order to satisfy our solutions. One simplification is that we must assume is that the preferences of teachers are evenly distributed over the total collection of classes to be taught. We also determined a maximum class size of 8 students, and minimum class size of 5 students, based on last years data. Additionally, our method doesn't determine if two or more teachers instruct a class, as was found in 2015. Our initial tests are also based on a small number of randomly generated student and teacher preferences. Thus, another simplification, is that computation times are based on these artificial ratings.

5 Mathematical Solution

Depending on the size of the data set, finding the assignment of classes with the maximum total preference could take days. We recommend either setting a time limit for the LP relaxation of the IP, or an integrality gap threshold. The integrality gap is the ratio between the relaxed LP optimal solution and the IP solution. Something interesting to note is that the optimal solution of the IP without any time constraints, is the same value of the optimal LP solution. That means whatever the integrality gap is, it represents the ratio of satisfaction given the ideal schedule where every student receives his most preferred class.

The solution to our artificial data set for 2016 was found in seven minutes was the first basic integer feasible solution found. Its maximum preference total of students and teachers assigned was 426 with an integrality gap of 11.7%. After letting the algorithm run for exactly three hours, this was improved to a maximum of 443 with an integrality gap of 7.4%. That is, the schedule created by the limits of time, we were within 7.4% of matching the ideal preferences of students and teachers.

We chose GLPK to solve the integer programming problem, not only for its lack of restrictions in public or commercial applications, but for its flexibility in setting solver parameters. Different optimization parameters

were tested for their speed in producing feasible solutions.

We found the following essential to produce the fastest results:

- Branching by pseudo-cost heuristic
- Backtracking by best prediction heuristic
- Cuts by mixed integer rounding

6 Results

We were able to find a viable solution to this problem using Linear Programming in approximately 7 minutes. Overall this is a much better time frame than the previous time frame used by our community partner which was reported as a 2 hour process. It is to be noted that while the initial solution found by this Linear Program is not the most optimal, it does satisfy all of the given constraints. By letting the program run for longer period of time, the assurance of a well formulated solution goes up. The schedule found for the data set after running the algorithm for three hours is discussed further below.

6.1 Teacher Assignments by Eligibility

Table 1 on page 9 refers to the data generated for the sample data in 2016. For a total of five teachers, each had an eligibility score to teach the classes. It was assumed that there was an even distribution of classes each teacher could instruct. Note that when a cell is colored green it corresponds to that teacher being scheduled to teach that class, as shown in 2 on page 9, red refers to classes a teacher could teach, but was not scheduled for. It's important to note, that no teacher was assigned a class he or she could not teach, i.e., no cell with the entry of 0 is colored green. This satisfies the hard constraint of each class being instructed by an eligible teacher.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Teacher a	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1
Teacher b	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
Teacher c	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0
Teacher d	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0
Teacher e	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0

Table 1: Teacher eligibility to teach a given class. Entry is 1 if the teacher in the row can teach the class in the column

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Teacher a	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1
Teacher b	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
Teacher c	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0
Teacher d	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0
Teacher e	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0

Table 2: Teacher assignment for the 15 classes. The entry is 1 if a teacher is scheduled to teach the class in the column.

6.2 Student Assignments

Student assignments to classes as a result of the LP solution are shown in 3 on page 10. This is the raw result from the algorithm we used. While the data in this form is not best suited for a printed schedule, we can check that each student is indeed assigned 5 classes, and each class has a maximum of 8 students. Both of these constraints were set in the LP and they meet the requirements given by Dr. Ostroff.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Student A	1	0	1	0	1	1	0	0	0	0	1	0	0	0	0
Student B	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0
Student C	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0
Student D	0	0	1	0	0	1	0	1	1	0	0	0	0	0	1
Student E	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1
Student F	0	0	1	1	1	0	0	1	0	0	0	0	0	1	0
Student G	1	1	0	0	1	0	1	0	0	1	0	0	0	0	0
Student H	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0
Student I	0	0	0	1	0	0	1	0	0	0	1	1	1	0	0
Student J	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0
Student K	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0
Student L	0	1	0	0	1	1	0	0	1	0	0	0	1	0	0
Student M	1	1	0	0	0	0	1	0	0	0	0	0	1	0	1
Student N	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0
Student O	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1
Student P	0	0	1	0	1	1	0	1	1	0	0	0	0	0	0
Student Q	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0
Student R	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0
Student S	1	1	0	0	0	1	0	0	0	1	0	1	0	0	0
Student T	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1
Student U	0	0	0	1	0	0	1	0	0	1	1	1	0	0	0
Student V	0	0	0	1	0	1	0	1	0	0	0	0	1	0	1
Student W	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1
Student X	0	1	1	1	0	1	0	0	0	0	0	0	0	0	1

Table 3: Student assignments; The entry of a cell is 1 if the student of that row is assigned to the column of that class.

6.3 Time Slots

Organizing the data by classes, with their respective teachers and students is the end result that Dr. Ostroff and the MathILy staff would need for "the Week of Chaos." This is shown in table 4 on page 11.

As a verification step, table 5 on page 11 is sorted by time slot. Then within each three classes that share a time slot, we see that no student is assigned to a class at the same time-slot, nor is any teacher.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TimeSlot	4	5	2	4	1	3	3	5	4	2	5	1	2	3	1
Teacher	d	c	c	a	e	d	b	b	c	e	a	d	d	e	a
Student	A	G	A	F	A	A	B	D	D	G	A	B	C	F	D
Student	B	H	B	H	C	D	C	F	E	J	B	I	E	H	E
Student	C	J	D	I	F	L	E	N	L	Q	C	J	I	J	M
Student	G	L	F	J	G	P	G	O	N	R	E	N	L	N	O
Student	K	M	H	R	H	Q	I	P	O	S	I	Q	M	O	T
Student	M	R	K	U	K	S	K	T	P	T	K	R	N	R	V
Student	Q	S	P	V	L	V	M	V	T	U	Q	S	O	T	W
Student	S	X	X	X	P	X	U	W	W	W	U	U	V	W	X

Table 4: Class assignment with color coded time slot, teacher, and student

TimeSlot	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5
Class	5	12	15	3	10	13	6	7	14	1	4	9	2	8	11
Teacher	e	d	a	c	e	d	d	b	e	d	a	c	c	b	a
Student	A	B	D	A	G	C	A	B	F	A	F	D	G	D	A
Student	C	I	E	B	J	E	D	C	H	B	H	E	H	F	B
Student	F	J	M	D	Q	I	L	E	J	C	I	L	J	N	C
Student	G	N	O	F	R	L	P	G	N	G	J	N	L	O	E
Student	H	Q	T	H	S	M	Q	I	O	K	R	O	M	P	I
Student	K	R	V	K	T	N	S	K	R	M	U	P	R	T	K
Student	L	S	W	P	U	O	V	M	T	Q	V	T	S	V	Q
Student	P	U	X	X	W	V	X	U	W	S	X	W	X	W	U

Table 5: Class assignment sorted by time slot

7 Improvements

The exponential growth of time needed to find the optimal solution for this Linear Program is something that cannot be avoided. With an NP-Complete problem and the structure of LPs, we can only hope to improve the time required to find a viable solution. By determining a proper heuristic to prioritize possibly viable solutions would improve the run time of finding a solution within a certain time-frame. Another possible source of improvement would be a custom preference scale. Our current preferences were pre-determined by our community partner, however, if we were given adequate time we might be able to produce a preference scale which would lead to either more accurate or quicker run times when it comes to discovered solutions.

Additional improvements need to be made to meet some of the additional, soft constraints of the community partner. For instance, our community partner mentioned incorporating gender ratios to the class schedules which our solution does not currently address.

Another beneficial option would be to place teachers with as many unique students as possible, preferably so that each student and teacher is paired in at least one class. The ability to turn these additional constraints on or off would allow our program to explore several possible solutions for our community partner with relatively little effort on their part.

Finally any other optimization that could be made to our LP structure might be beneficial. If we could combine constraints that produce the same results or in some other way reduce our total number of constraints, we could speed up our program to make it more convenient when testing several data sets.

8 Conclusions

This project directly connected to the material learned in class, regarding the subject of Linear Programming. While more complex than those covered in the homework, the basic knowledge provided in class proved instrumental in guiding our decision making process. Originally we tried to formulate a solution using graph theory, finding matchings between a tripartite graph so find a possible solution. Upon further consideration however, we determined that th novelty provided by this approach was outweighed by the scalability and customization provided by Linear Programming. When researching similar problems many published papers advocated for the use of Linear

Programming over graphical methods.

9 Acknowledgments

First off we would like to thank Professor Sara Billey for being understanding when our previous two projects did not end up working out. The stumbles suffered throughout the quarter made progress difficult and uncertain, but Professor Billey was very accommodating and made sure that we kept moving forward. We would also like to thank Dr. Jonah Ostroff for providing this interesting problem for us so late in the quarter. While not as ambitious as our previous two problems, this scheduling problem seems genuinely useful while also providing value for future weeks of chaos to improve student and teacher experiences.

10 References

Yuk Hei Chan, a PhD candidate at the University of Maryland, wrote his masters thesis on linear programming relaxations of matching problems in On Linear Programming Relaxations of Hypergraph Matching. He discusses in his thesis the time complexity of finding k-matchings, integrality gap of various LP relaxations, and methods to improve the LP relaxations. Chan mentions a polynomial time approximation of k-matching, which could be applied for larger data sets. He also compares k-matching to other problems, such as set packing. This resource will be invaluable in searching for ways to improve schedule creation and adapt it to larger schedules.

Dr. Aldy Gunawan is a Research Scientist at the Living Analytics Research Center in Singapore. He has published many papers regarding the issue of scheduling construction. Specifically he has published a paper regarding Solving the Teacher Assignment Problem by Two Metaheuristics. In this paper he explores using the process of Simulated Annealing to find an optimal schedule for full-time and part-time teachers in a class schedule. He also has another paper A Genetic Algorithm for the Teacher Assignment Problem for a University in Indonesia which solves a similar problem but solves it using a genetic algorithm. Finally he has a third published paper, Solving the Teacher Assignment-Course Scheduling Problem by a Hybrid Algorithm in which he confronts another scheduling problem among teachers, classes, and students optimizes a solution based on Integer Programming. These papers provide a good basis since the problems addressed are similar to the problem of our community partner. However, they are not exact

solutions so we will have to formulate our own LP.