

Dillon Gym Shift Coverage System: Optimizing Using Queues and Response Probability

Andrew Lin

Adviser: Mark Braverman

Abstract

Dillon Gym is Princeton University's campus recreation center. In order to keep the doors open, the front desk is staffed at all times, usually with 2 student employees. With a relatively large number of employees, small regular hours a week per student, and constant need for shift coverage, this system is ripe for optimization. I propose a new system for covering shifts using a randomized queue of employees and their response probabilities to reduce the number of emails that employees receive, provide them a larger time frame to accept a shift, and create a fairer and less competitive system for shift coverage. I also provide proof of concept for a potential user-friendly implementation that functions solely on Google Forms and email from the employee's side.

1. Introduction

1.1. Historical Context and Background

Rebuilt in 1947 [2] after the original gym burned down in a fire, the gym's iconic tower offer students a respite from the rigors of academics. Access to the facility is granted through a single entrance on the north side of the gym, as circled in red in Figure 1. The facilities staff of Dillon Gym are in charge of granting access to the building, making sales, and serving as an access point to the other staff who work in Dillon Gymnasium.

The staff has grown significantly over the past few years, shifting from a model of long shifts worked by non-student employees to many short shifts worked by student employees. As a result, there are far more shifts per week. In the spring semester of 2020, 63 different employees had weekly shifts at the front desk alone, with additional shifts at the fitness center located at the

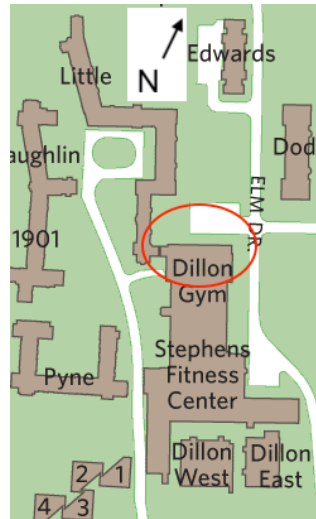


Figure 1: Map of Campus. The Red Circle Indicates the Front Entrance.

[1]

Stephens Fitness Center inside of Dillon Gym. Employees often have conflicts with their weekly shift because of the constant change of a college student's schedule, and approximately 300 shifts were covered in the fall semester of 2019.

1.2. Current System and Motivation

The current system for shift coverage is a first-come-first serve system run completely through an email listserv. The current flow of shift coverage is as follows:

1. Employee A has a conflict with their regularly scheduled shift.
2. Employee A sends a mass email through the listserv to the entire Dillon Gym Staff with the date and time of their shift that needs to be covered.
3. One of the employees (we will call them Employee B) wants to cover the shift, and emails Employee A that they can cover the shift.
4. Employee A receives the email from Employee B, and then sends out another mass email through the listserv to the entire Dillon Gym Staff saying that the shift is now covered by Employee B.
5. If another employee (we will call them Employee C) also wants to cover the shift, they will also email Employee A.

6. If Employee A receives emails from multiple people who want to cover the shift, they select the person who emailed them first to get the shift.

There are multiple undesirable qualities with the current system. Under the current system, the roughly 300 shifts covered in the fall of 2019 generates at least 600 emails to every person's inbox every semester, even if they are not interested in taking a shift: one to announce that the shift is open, and one to announce that someone has filled the shift.

Often, Employee A will receive multiple requests from students such as Employee B and C to cover their shift before they can send out the mass email that their shift has been covered as described in Step 4. It is inconvenient to be Employee C and send an email requesting a shift, only to be notified later that Employee B sent an email first, earning them the shift before Employee C.

In addition, with such a large number of employees, shifts are often filled within minutes, incentivizing students to check their emails in class and creating a competitive culture around shift coverage, which goes against the Dillon Gym Facilities Staff mission and goal to have Dillon Gym be a stress-free environment for people to relax and unwind from the daily rigors of academic life.

1.3. Goal

My Goal is to design and implement a mechanism that alleviates the undesirable qualities of the current system. The four main desirable parameters are as follows:

- Minimize the total number of emails sent
- Maximize an employee's time to respond to an available shift
- Be considered fair by employees and employers
- Ensure all shifts are covered by the time they occur

It also must immediately notify students when a shift has been taken so students don't continue to volunteer for shifts that have already been taken.

In addition, from end-of-year interviews with the Coordinator of Recreational Facilities and Operations Mike Mix, students considered using an outside app to be one of the most undesirable

qualities of a system, so this mechanism must be conducted purely through email and easily accessible links from the side of the employees.

1.4. Approach and Implementation Summary

There exist certain tradeoffs in achieving the desirable qualities of a mechanism listed in Section 1.3. Decreasing the number of emails sent increases the average response time for fulfilling a shift, which are both desirable. However, it also causes shifts to take longer to cover, an undesirable quality of the system. Also, decreasing the number of emails sent poses a potential problem to fairness, because not everyone is notified about every shift.

After identifying the desirable and undesirable qualities of a mechanism and inherent tradeoffs between qualities, some potential mechanisms that seemed to fulfil the qualities were brainstormed and sketched up. Running simulations on the mechanism with different categories of employees helped provide a numerical basis and support for different mechanisms. By analyzing the systems from both a simulated numerical basis and a position of common-sense reasoning, a mechanism was finally decided upon.

I propose a mechanism using a randomized queue of employees and their response probabilities to reduce the number of emails that employees receive, get shifts consistently filled before they occur, increase the window of response time, and provide a numerically fair implementation.

My implementation seeks to be fully usable by employees through one Google Form and email. I use Google App Scripts, Google Forms, Gmail, Heroku Cloud, PostgreSQL, Flask, and Python to implement the logic for my mechanism, described in detail in Section 5.

My end result is a fully-functional, optimized mechanism of a queue of employees and probabilities that appears very similar to the current system to the employees, decreasing the number of emails they receive while minimizing the learning curve on the employee side.

1.5. Paper Structure

This paper's organization will largely follow the chronological order of coming up with and finalizing a more efficient mechanism for Dillon Gym Shift Coverage. The approach outlines two

potential systems that had desirable properties and contains the logic behind their creation. The results section contains an analysis of the performance of the mechanisms and discusses their pros and cons. The conclusion contains concluding remarks about deciding on the final mechanism with queues and response probabilities, an analysis on why it is better than the current system and the other potential new system, and potential future work ideas. Because the results and conclusion of the mechanism show why the mechanism was finally adopted as the mechanism to use in the Dillon Gym Shift Coverage System, the physical implementation of the new mechanism could not be done until the mechanism was decided up and finalized. Therefore the physical implementation of the system using our own hosted servers, email, and Google Forms will appear in this paper after the results and conclusion.

2. Approach

After brainstorming systems, two mechanisms stood out as being much better than the current system. Both systems could be implemented using email because an outside app is considered highly undesirable by the employees. Mechanism 1 attempted to assign a type of currency to every employee based on how many times an employee entered auctions to gain a shift. Using this currency, every open shift would be auctioned off to the employee with the most currency. Mechanism 2 maintained the first-come-first-serve format of the current system, but optimized the number of emails sent by analyzing the response probabilities of employees.

Comparing these two systems, Mechanism 2 was determined to align better with the desirable properties listed in Section 1.3. Both mechanisms are described below to illustrate the considerations and comparisons that eventually led to the adoption of Mechanism 2 as the final mechanism.

2.1. Mechanism 1: Response-Based Currency

This mechanism assigns every employee a currency, initialized to 0. If Employee A had a conflict with their weekly shift, they would put it up for auction to the mechanism. This mechanism would then send an email to the top n employees with the highest currency, letting them know that a shift

was available. If there are multiple people who all have the same currency such that there is no exact top n , the mechanism randomly selects people. These employees would have a time interval t to decide whether they wanted the shift and then enroll in the auction. Out of these n employees, k would respond and enroll in the auction. These k employees would then have their currency incremented by 1. Out of the k employees who enrolled in the auction, the one with the highest currency would win the auction for the open shift and have their currency reset to 0. If some of the k employees are tied with the same amount of currency, the system randomly selects one. Figure 2 shows a simplified version of this system in action to show how it works.

The number n is a parameter that can be adjusted based on how the system functions and can be set to get the optimal number k out of the n people to actually respond. If n is large, the number of responses k will be larger, and people will have higher currencies and the auctions will contain more people, sending more emails overall. If n is small, then most people will have currency of 0 at all times, and the auctions will often only contain one or two people, and sometimes 0 responses, sending fewer emails, but occasionally having to repeat if none of the n people enroll in an auction, filling shifts slower. The main idea behind this system is that people who respond more will have more currency over the course of the semester and therefore win more auctions.

2.2. Mechanism 2: Queues and Response Probability

This mechanism is a first-come-first-serve mechanism that subdivides the population of employees into smaller groups dynamically. If Employee A had a conflict with their weekly shift, they would put it up for auction to the mechanism. The mechanism creates a randomized queue of employees. The mechanism keeps track of the response rate of every employee, which we will call r . The mechanism will select the first n employees in the queue such that their probabilities of response r add up to some low probability p . They are then given a time window t to respond to the email. If one of the n employees responds in the time window t , they win the shift because it is first-come-first-serve, the shift is filled, and the other n employees are notified.

If none of the n employees respond in time t , then the next k employees in the queue are

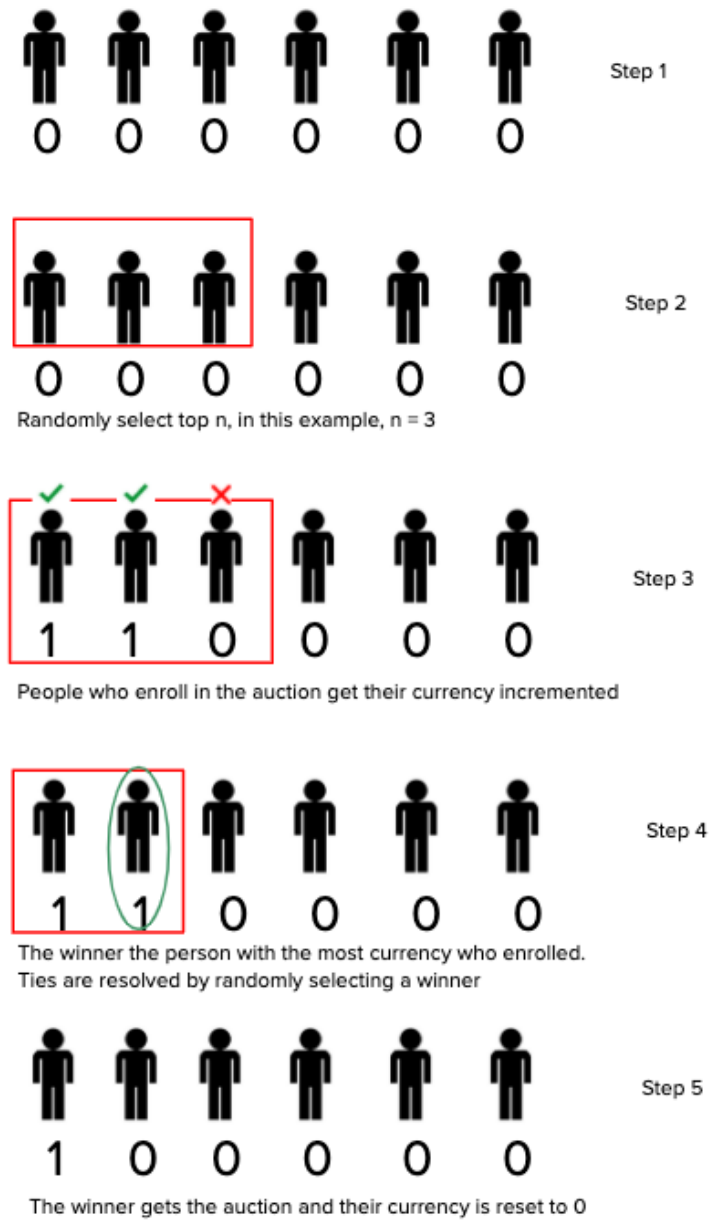


Figure 2: A single shift covered by Mechanism 1.

selected such that their probabilities of response r added to the probabilities of response of the first n employees sum to the probability $2p$. Now both the groups of n employees and k employees again have time t to respond to the email. If one of the $n + k$ employees responds, they win the shift because it is first-come-first-serve, the shift is filled, and the other $n + k$ employees are notified.

If none of the $n + k$ employees respond in time t , this process will repeat with probability such that the probabilities of response r add to $2^z p$, where z is the round number, starting at 0.

Each of these employees is initialized to have a probability of .1 for the first 5 times through the queue of students so there can be a large enough sample size so that their response probabilities can be accurately determined. This number is rather arbitrary, but setting p to be 0.5 will select 5 students in the first round, 10 in the second, 20 in the third, 40 in the fourth, 80 in the fifth, roughly covering the entire staff in time period $5t$. These step sizes seem reasonable given the size of the staff at Dillon Gym.

The question arises: what happens if an employee in a round responds very quickly? Does that affect the response probabilities of all the other employees in that round and cause them to wait to go through the entire queue again to be selected? This seems "unfair" and makes it far more desirable to be in an earlier round than a later round, which is determined by randomness. Therefore, if an employee in a round responds before the window t is finished, then everyone from that round besides the winner stays at the front of the queue.

In addition, the queue is re-randomized every time through it so that an employee isn't stuck next to the same people every time, as some people respond with much higher rates than others.

Deciding the duration of time period t is another parameter that must be set during for this system. To simplify the system, we will set t to be 1 hour, because this gives students enough time so that they don't have to check their emails during class, as most classes are 50 minutes long.

3. Results

3.1. Mechanism 1: Response-Based Currency

I ran two simulations on Mechanism 1, one where everyone had an equal response probability of 0.1 where $n = 20$ and one where $n = 50$. I used 100 students to run the simulation, and simulated for 1000 times, even numbers that were close to the actual numbers of students and shifts for an entire year.

Figure 3 shows the results of running Mechanism 1 100 times, where $n = 20$. The average number of shifts won per person is 10. The standard deviation was undesirable at 2.659, but it only sent 202 emails to every person.

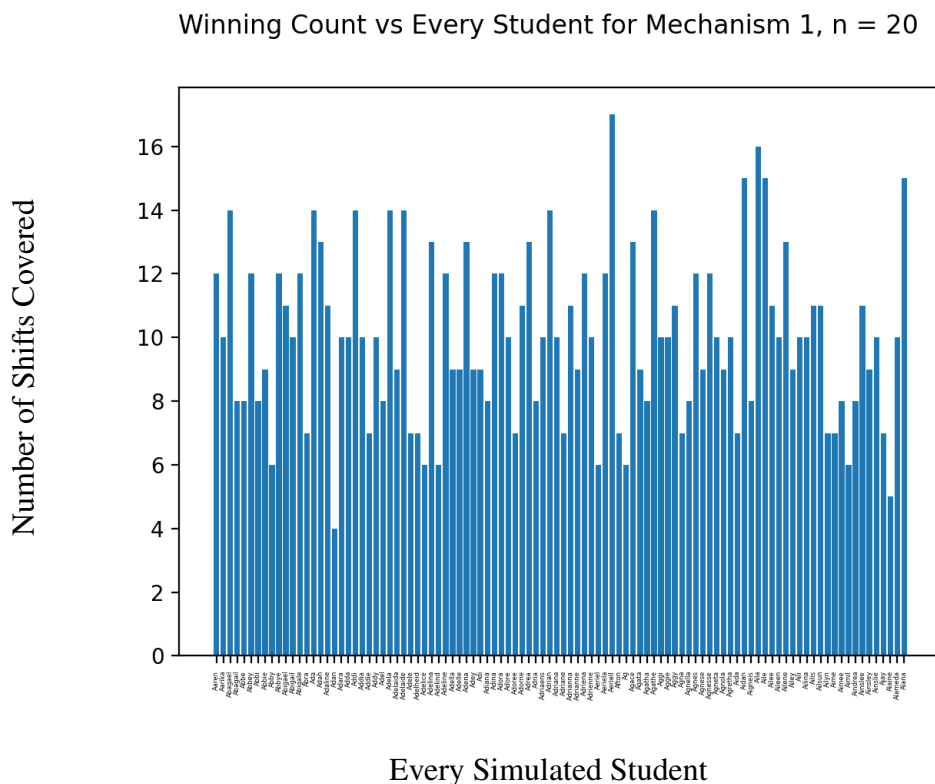


Figure 3

Figure 4 shows the results of running Mechanism 1 100 times, where $n = 50$. The average number of shifts won per person is still 10. The standard deviation was better than when $n = 20$ at 1.735, but it sent 500 emails to every person.

This shows the tradeoff between fairness and the number of emails we send for Mechanism 1. Sending more emails makes it more fair, but we are trying to minimize the number of emails. Therefore, the essential tradeoff in this system is between fairness and number of emails. Unfortunately, these are the two most desirable properties of the mechanism we are trying to create.

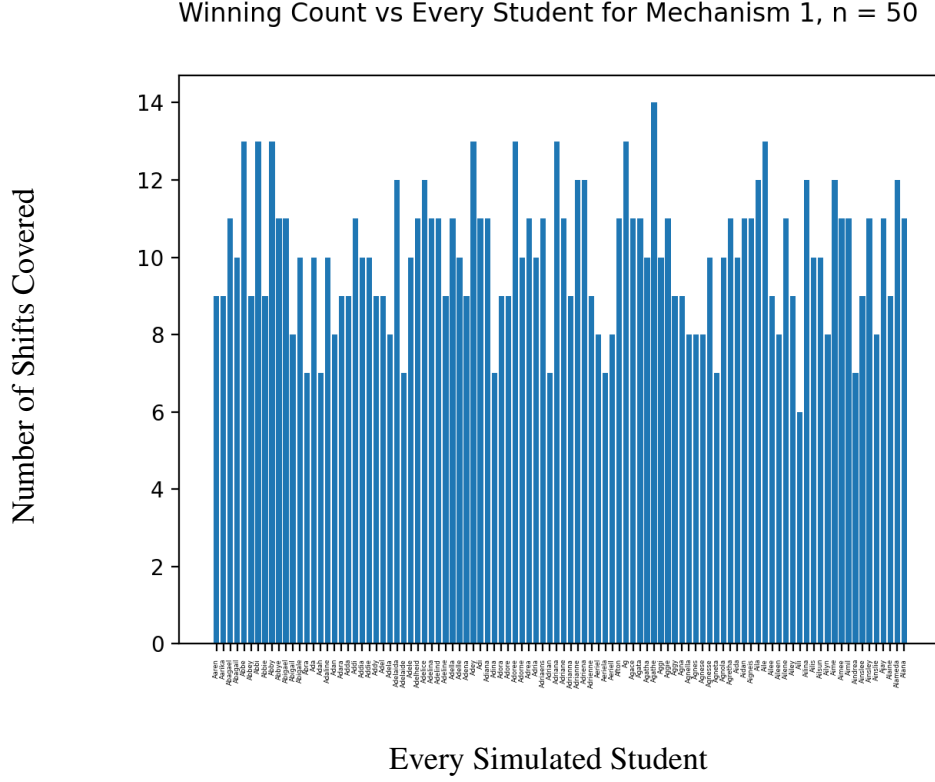


Figure 4

3.2. Mechanism 2: Queues and Response Probability

Analyzing Mechanism 2 from a numerical standpoint is more difficult. First come first serve is considered "fair" because people all receive the email at the same time and it is a fair race to see who responds first. However, the numerical distribution of shifts is quite unfair from a statistical perspective because it all depends on how fast a person can email in.

However, comparing Mechanism 2 to a general first-come-first-serve selection process will provide a standard for comparison. Simulating a regular first-come-first-serve shift auction with 1 type of employee for 100 employees and 1000 shifts, we get that the average number of shifts per person is 10, and the standard deviation is 3.185.

Simulating a similar environment using Mechanism 2 with 1 type of employee, where $p = 0.5$, there are 100 employees and 1000 shifts, we still get the average number of shifts per person is 10, and the standard deviation is 2.95. Although these simulations are simplified, this shows that Mechanism 2 will provide a slightly more even distribution of shifts than first-come-first-

serve. In addition, the main complaint with the current first-come-first-serve system is the lack of time available to sign up for shifts; indeed, according to Coordinator of Recreational Facilities and Operations Mike Mix, most students view the current system as fair.

Therefore, since Mechanism 2 uses a queue to split the staff into smaller first-come-first-serve auctions, students will likely consider this method just as fair as the larger first-come-first-serve system. Mechanism 2 also provides a more even numerical spread than the current system, which means that it is actually more fair than a system that is already considered fair.

The amount of emails that the system sends is dependent upon response rate and the parameter p . If the response rate of students is lower, then the amount of emails that have to be sent will rise. This intuitively makes sense because it will take more emails to find a person willing to cover a shift if less people are interested in covering a shift. If the parameter p is low, then the number of students in each round will be lower, therefore causing the auction to take longer. If the time parameter t for each round is higher, it will increase the probability that students respond per round because more students will check their emails, but increase the total time it takes to cover a shift.

These three tradeoffs are much better for our system than Mechanism 1's tradeoffs. Students are incentivized to respond to the auction because most students want to pick up additional shifts. Therefore, they are incentivized to respond more often, causing the system to get less emails. The system doesn't need to minimize the time it takes for a shift to be covered. Therefore, we can err on setting p to be a little too low because it increases at a speed $2^z p$ where z is the round number. It will increase exponentially until the probability that a shift will be taken is very high after a few rounds. We will fix t to be 1 hour for now because it will allow students enough time to check their email between classes. With this set to 1 hour, if p starts at some number very low such as 0.25, after 5 hours there will be a probability 4 that a shift will be filled, meaning four people would want to take the shift.

Simulating with both low-end and high-end probabilities for responses for 100 students and 1000 shifts, we can get a rough idea of how many emails the system will use. Simulating with

an average response probability of 1 in 10, students averaged 58.73 received emails. Simulating with an average response probability of 1 in 20, students averaged 91.83 received emails. Both the high and low estimates are significantly lower than the 1000 emails that each student would receive for the current first-come-first-serve system. This mechanism lowers the number of emails received by a factor of 10 using conservative estimates.

Each student will have 1 hour to respond to the email and put their name in for the shift. As mentioned in Section 2.2, If another employee in the probability group of Employee A responds before Employee A, then Student A is still at the front of the queue for the next auction. In this way, employees are guaranteed at least one hour (or time period t) to respond to get a shift every cycle through the student queue. This guarantee is a far larger time guarantee than the current system, which provides no time guarantees for students to consider their schedules and preferences.

4. Conclusions

Mechanism 1 worked very well to maximize an employee's time to respond to an available shift. Using auctions and assigning students a type of currency based on how many auctions they entered, it successfully provided more shifts to the students who entered more auctions. However, this mechanism contained a fundamental tradeoff between two desirable qualities: fairness and total number of emails. In order for the system to function completely fairly and not prioritize some students over others, the mechanism needed to email every student. When the mechanism selects sub-sections of the population, it minimizes the number of emails but provides an unfair, albeit random, advantage to the students that the mechanism selects first for the auction. The fewer emails the mechanism sent, the more unfair the advantage it gave certain students, allowing some students to accrue "currency" at a faster rate than their peers, even though all students bid with identical traits.

This fundamental tradeoff between two desired qualities made the implementation of Mechanism 1 choose between prioritizing fairness or minimizing the number of emails. While still a significant improvement from the current first-come-first-serve system in the time allowed

for students to respond to an available shift, the number emails could not be optimized without compromising the fairness of the system.

Mechanism 2 gives employees less time to respond than Mechanism 1 and still makes shift coverage be a first-come-first-serve race. However, Mechanism 2 sends emails to rounds of users based on the probability that the shift will be filled in a current round. By setting that probability for the first two rounds to be less than 1, this system provides employees in the first couple of rounds the opportunity to accept a shift without having to worry too much about racing their peers to accept a shift. In the case that another employee responds quickly before the other employees in their round can see the email, those employees remain at the front of the queue.

Through this approach, Mechanism 2 keeps first-come-first-serve as non-competitive as possible, maintaining the fairness of first-come-first-serve while allowing the number of emails to shrink and giving students 1 hour (or time t) to accept a shift. This mechanism fulfils the goals the new Dillon Shift Coverage System more effectively than Mechanism 1. Instead of pitting two of the goals against each other in a fundamental tradeoff like Mechanism 1, Mechanism 2 separates fairness from the number of emails that users receive. The time between rounds increases the total amount of time it takes for a shift to be covered, but covering shifts quickly is not a priority for the system. As long as all shifts are covered before they occur, which Mechanism 2 fulfils, covering shifts quickly does not matter.

Mechanism 2 decreases the amount of emails sent to each user by a factor of 10 using the conservative estimate that users will accept 1/20 of the shifts that they are notified about. It also guarantees each user will see a shift for at least 1 hour (or time t) every cycle through the queue of employees, providing a guaranteed amount of time to respond to an email to accept a shift that the previous system did not have. From simulations, it is statistically more fair than first-come-first-serve with a smaller standard deviation of 2.95 in number of shifts that employees receive compared to 3.185 in first-come-first-serve. In addition, by keeping the system as a variation on first-come-first-serve, it remains familiar to employees who already considered the previous system as fair.

Mechanism 2 strictly dominates the current system in three of the goal parameters: number of emails sent, fairness, and allowed response time. It fills shifts slower than the current system, but still fills them in adequate time, which fulfills the final goal parameter. Therefore, Mechanism 2 provides a better mechanism for optimizing the Dillon Gym Shift Coverage System than first-come-first-serve and Mechanism 1.

4.1. Future Work

This parameter t can be further optimized depending on the amount of time between when a shift is put up for coverage and the time the shift will occur. Throughout the course of this paper, the time for rounds of Mechanism 2 has been set for 1 hour because of the length of most courses at Princeton. However, consider a shift that needs to be filled in 1 hour. Mechanism two does not account for this type of last minute change, and only guarantees that shifts will be filled 1 hour with a probability of 50% because we set p to be 0.5 initially.

The mechanism can be further optimized by adjusting t depending on the amount of time between when a shift is put up for coverage and the date it occurs. For instance, if an employee auctions a shift for coverage that occurs in 30 days, time t could even be set to 24 hours. That way, each employee in that round would have 24 hours to see the email before new employees are added to the auction if it is not filled.

If a shift needs to be filled urgently, t can be adjusted to a single minute or even set to 0, emailing every employee like the old system's first-come-first-serve. This would provide the same guarantees that every shift is filled as the old system.

One potential issue of dynamically adjusting t is a change to the fairness guarantees that the system provides with time $t = 1$ hour. Since t would adjust every time a shift was put up for shift auction, each employee's time guarantee for how long they had to answer a shift would change with the t for the shift auction they were given. While this would optimize the system to give employees the most time to respond to emails, it would randomly favor the employees at the front of the queue with auctions that had a larger t . This could be considered unfair by other employees, and must be

further researched and possibly implemented in comparison to a mechanism with a set t .

5. Implementation

Implementing the system with a desirable user interface required email to be the main source of communication to and from the employees. There are three main parts to the implementation of Mechanism 2.

Part 1 is creating a platform for users to put up their shifts for auction. Part 2 is implementing the Mechanism on a server. Part 3 is communicating to the other employees that shifts are available and recording responses and notifying all employees who were emailed that a shift was open when that shift is no longer available.

Figure 5 provides a graphical flow diagram of the implementation of Mechanism 2.

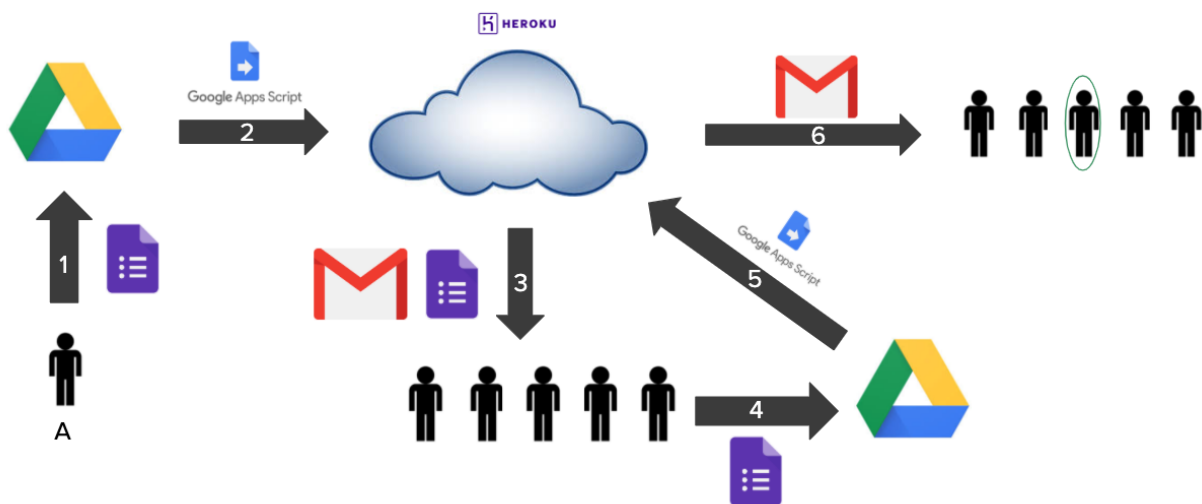


Figure 5: 1) Employee A uses a static Google Form to request coverage for their shift. 2) A Google App Script is triggered and takes the data from the form and sends it to my application on Heroku Cloud. 3) The Mechanism is implemented in Heroku Cloud and selects the first round of employees to email about the available shift. 4) One employee takes the open shift by filling out the Google Form in the email they receive. 5) Another Google App Script reads this response and sends it to my application on Heroku Cloud. 6) The system notifies all the people it originally emailed in step 3 that the shift has been covered.

5.1. Part 1: Getting Your Shift Covered

The current system for getting a shift covered is sending a mass email to the listserv saying the date and time of the shift that needs to be covered. Employees vehemently opposed the use of another outside application, citing that it is more annoying to check a separate application than it is to get lots of emails.

Therefore, I needed a standardized for users to enter information that was not on an application. Parsing emails is not standardized, so I decided to get user information into my system through Google form links sent over email. Currently Dillon Gym does most of their administrative work through Google Applications, using Google Forms to clock in and clock out, manage the lost and found, and do almost everything organizationally. I decided that to create a single Google Form titled "Shift Coverage Form" (see Figure 6) to serve as an employee endpoint for putting up a shift to be covered.

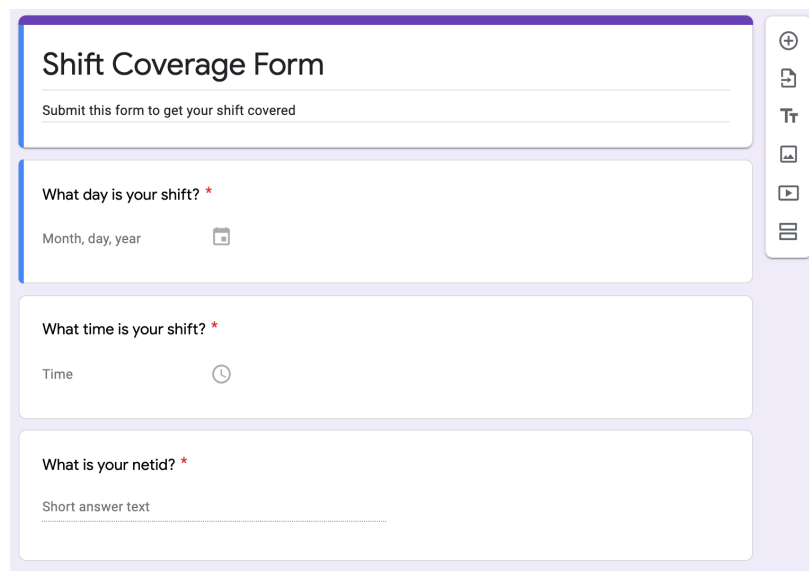
The image shows a Google Form titled "Shift Coverage Form". Below the title is a subtitle: "Submit this form to get your shift covered". The form contains three required questions, each marked with a red asterisk. The first question is "What day is your shift?" with a text input field labeled "Month, day, year" and a calendar icon. The second question is "What time is your shift?" with a text input field labeled "Time" and a clock icon. The third question is "What is your netid?" with a text input field labeled "Short answer text". On the right side of the form, there is a vertical toolbar with icons for adding, deleting, and duplicating questions, as well as a link icon.

Figure 6: The Actual Shift Coverage Form

To put a shift for coverage, an employee needs to go to the form, enter in the day and time of their shift as well as their own netid. The netid is to ensure that employees aren't asked to cover their own shift. From there, the employee putting up a shift for coverage is done. The system will handle the rest.

5.2. Part 2: Implementing the Mechanism

I needed some way to get the information that employees entered into when they wanted their shifts covered into my Mechanism. Google provides developer API tools to make that process easier. I created a Google App Script, which is a JavaScript script run on Google Cloud that can be easily connected Google Forms on the same Google Account. This App Script is triggered whenever an employee puts their shift up for coverage using the Shift Coverage Form. Everytime an employee uses the Shift Coverage Form to put up their shift for coverage, a POST request containing the date, time, and netid of the employee is sent to my application hosted on Heroku Cloud.

The logic and implementation of Mechanism 2 is all done within this application on Heroku Cloud. I structured the application using Flask, and continuously store information about shifts that still need to be covered using PostgreSQL. I have a SQL table with every employee's netid, and update their response rates on this table.

This application handles the POST request sent from the Shift Coverage Form. Using the parameters from the POST request, My application then calls the API for a second Google App Script I created, which makes a separate Google Form that will eventually be emailed to employees so they can use it to sign up for the shift that needs to be covered. I store all pending shifts that haven't been covered yet in a SQL table called "pendingshifts," including the date, time, netid, and Google Form to record responses. It also contains a listed of emailed students' netids.

The first group of employees are selected from the queue such that their probabilities add up to p . These employees are emailed a link to the Google Form that was created and stored in the SQL database, and their names are added to the field of emailed students on the SQL database. If one of them responds, the auction is deleted from the SQL table "pendingshifts," and all the emailed students are emailed again, notifying them that the shift has been taken. If no one responds in 1 hour (or time t), then another group of students is emailed such that their probability adds to $2p$.

This is achieved through the use of cron jobs run through a separate worker on Heroku. This worker runs every minute, looping through all the current pending shifts, checking how much

time has elapsed since the start. If time t has elapsed, then this job calls the function to send more emails, specifying the probability that the sum of each person's probability adds up to.

5.3. Part 3: Emailing Students About Available Shifts

I use the SMTP_SSL protocol to email students. This protocol sends both unformatted and formatted text to successfully deliver the message to all configurations of inboxes. My mechanism delivers emails to tiers of students automatically by looping through the "pendingshifts" table and calling the block of code for emailing students when time t has elapsed. These emails contain a link to a Google Form to sign up, and also a line of text explaining how to volunteer to cover a shift.

After a student successfully covers a shift, a callback is triggered on the Heroku-hosted backend which initiates ending a pending shift. Another email is sent to every student who received an email for the shift. This email notifies them that the shift is no longer available. In addition, the Google Form for signing up for that shift is closed using the App Script API so that open forms don't clutter the Google Drive on which the mechanism is run. Finally, the pending shift is deleted from the "pendingshifts" SQL table and moved to a "completedshifts" table in the database.

5.4. The Details and Future Implementation Work

In my implementation, I tried to avoid fees for hosting my servers. The Flask-python backend is currently hosted on the Heroku Free tier. This tier has a 30 minute sleep session, meaning that the application will go to sleep 30 minutes after a user last pinged it. This poses no problems to employees putting their shifts up for auction, but caused the cron jobs to stop execution 30 minutes after a shift was put up for coverage if no other shifts were put up for coverage. To fix this, I used a service called caffine that pings my server every 30 minutes. This temporary fix works for now, but has a mandatory 6 hour sleep cycle at night. To fully implement this effectively, a new hosting service must be found, or Heroku must be upgraded to a paid tier.

5.5. Implementation Conclusion

This implementation allows the employees to access the Shift Coverage Mechanism in a simple way. While this method makes it extremely simple from an employee's perspective, it also hides some of the underlying complexity of the system. In the current implementation, users are unaware of their position in the queue and what round of the mechanism they are in when they get an email.

For now, I believe that hiding the intricacies of the mechanism is preferable. Employees will see the system as a simple combination of a Google Form to put up their shifts for coverage and less emails with longer response times than the previous system. This will allow the employees to adjust to the new system with ease and prevent new training and unnecessary adjustments. In combination with the new mechanism, this implementation will provide students with less emails and more time to respond, all with a similar feel to the old system using email.

6. Honor Pledge

I pledge my honor that this paper represents my own work in accordance with University regulations.

References

- [1] [Online]. Available: <https://i.pinimg.com/originals/eb/97/ed/eb97ed1b613a59cb8b2db996d0fda2b5.gif>
- [2] "Princeton campus rec homepage." [Online]. Available: <https://campusrec.princeton.edu/facilities-operations/locations/dillon-gym>