Miami University

Supply-Chain Optimization

Jon Kosir
Kediqi Han
Jinxin Liu
Huaiyu Zhang
Drew Sanders

December 11, 2015

Abstract

Combining our own methods with research conducted by Harvard University, we are creating a new technical platform for the University of Cincinnati Medical Center (UCMC) with a focus on optimizing hospital resources. This platform – the first of its kind – will enclose stochastic modeling techniques in a user friendly software package. This package is what we will be delivering as a system capable of interfacing with a variety of software and hardware.

Table of Contents:

Introduction

The national healthcare system is currently facing a resource crisis, with efficient patient care at its center. Assuming patient admission is possible (given available hospital beds) misallocation of resources can prevent patients from receiving care and being discharged in a timely manner. This imbues increased cost, and decreases potential revenue.

A common response has been to expand facilities to board more patients. This exacerbates the problem, increasing pressure on the existing bottleneck – the patient care time frame. Without receiving care the patient population expands, leading to patient diversion to other hospitals. This compounds the inefficiencies described above.

The proposed solution is to maximize efficiency of current patient care resources; efficiently caring for patients will allow them to leave the hospital sooner, freeing up resources for other patients and simultaneously maximizing revenue. Current research supports this approach, but it has drawbacks – namely, the intensive calculations. The complexity of this system requires heavy processing, leading to an unacceptable expenditure of time and/or money.

The challenge of this project is to create an optimization program that is efficient in its use of computer resources. We are not using a computer cluster nor hardware designed for heavy processing, instead designing a system capable of functioning on a consumer grade computer with stock settings.

Literature Review

The process of optimizing hospital resources is known in the literature as variability methodology. This methodology analyzes two sources of variability – artificial and natural. Artificial variability is inefficiency produced by the system itself; in other words, controllable inefficiency. Natural variability is that which cannot be predicted or controlled, i.e. patient reliability, demand for emergency services, etc. Variability methodology focuses on maximizing elimination of artificial variability, and accounting for a maximum of natural variability through the design of a system that will accommodate it.

Current research places high priority on the optimizing the scheduling of emergency surgeries. Existing healthcare systems are faced with two choices in the case of emergency services – either delay scheduled surgeries, or wait for an open operating room. Both solutions to this problem lead to a lengthened patient wait time, and subsequent patient and provider dissatisfaction. Additionally, these delays have the ability to become life threatening. These emergencies must be accommodated to maximize quality of care.

Quality of care is not limited to a successful implementation of medical services; rather, it is defined by a spectrum of variables. Of these variables, patient wait time has been identified as the most influential in determining patient satisfaction. By minimizing these wait times, not only are medical services being administered efficiently but overall satisfaction with the system is improved through enhanced trust from patients to their provider. Conversely, longer wait times lead to prolonging of patient anxiety and hunger, the latter of which can be a significant factor when considering pediatric surgeries.

In an effort to offset these costs, hospitals will often push providers into starting a surgery at the earliest possible time. This push can lead to hospitals and/or providers cutting corners, potentially placing a surgery in a less than optimal room or rushing a surgery in an attempt to return to the original schedule. Such pressure leads to mistakes in the system, which are less forgivable in the medical profession than almost any other industry and also have the potential to incur massive cost to the hospital and provider.

Surgical delays can often lead to longer working hours for staff. This is known as overtime, and imbues increased costs in both finances and efficiency. Hospitals view overtime as

a tool to combat inefficiency, but because it cannot be predicted it often leads to feelings of anxiety among workers, most notably fear of making mistakes. These mistakes can themselves cause delays and longer working days. Additionally, shifts of 12 hours or more are strongly correlated with an increased number of mistakes among workers. Overtime leads to decrease in optimal working conditions across the board; from patient and provider satisfaction to hospital operating cost. To date, an estimated 98,000 patients per year die due to medication errors. Nurses intercept an estimated 86% of these medication errors, and there is a correlation between patient complications and time nurses spend with the patients.

Design Process

A. Data Collection

We have collected/built the following database structure:

- Identification number: All data for an arbitrary procedure will be flagged by this number, giving us a structure by which to query the database.
- Operating Room Location: Our model makes the assumption that an arbitrary operating room is maintained for only a subset of procedure types, and thus the room must be assigned based on the requested procedure.
- Case Urgency: A trauma procedure must be given higher prioritization than non-trauma procedures, and thus our algorithm considers them first.
- Primary Procedure Type: The algorithm must know what probability distribution to retrieve when scheduling a time-frame, and these are organized by procedure type.
- Time Setup Starts/Finishes: The first of three phases that our algorithm analyzes when recommending a time-frame. Following the last procedure's cleanup, the room equipment and tools must be laid out for the incoming procedure.
- Time Patient Enters/Leaves: The second (and main portion) of three phases that our algorithm analyzes when recommending a time frame. This encompasses all elements of the procedure from the moment the patient enters the room to the time he/she leaves.
- Time Cleanup Starts/Finishes: The final phase that our algorithm analyzes when recommending a time-frame. Once the procedure is finished, time must be allocated to clean the operating room and turn it over for the next procedure.

B. Algorithm Development

Our algorithmic solution to this problem has been developed using stochastic modeling techniques and implemented through Matlab. The foundation of the algorithm lies in queueing theory, as the project's main goal is that the utilization rate for an arbitrary operating room must remain high while simultaneously minimizing delay times and maximizing throughput. The in-depth process of how the algorithm functions is described below.

Once the algorithm is initiated from an outside source, it starts by requesting all data currently logged in the database. All entries with missing data (signaling that they have yet to be scheduled) are set aside and only complete entries are taken into initial consideration. This data is then filtered into subsets, organized by procedure type but containing all data attached to the arbitrary event. This process is done through string matching; all database events with sufficiently matching procedure types (they do not have to be identical, for example two procedures "Amputation Leg Above Knee" and

"Amputation Leg Below Knee") are grouped into a single subset. Filtering all database entries this way results in matrices of varying sizes, which are then used to return detailed statistics of their respective procedure types.

Given an arbitrary subset of data, the median and standard deviation for procedure, setup, and cleanup times are found. The median was deliberately used because the accuracy of the hospital's records caused issues while attempting to return the mean. For example, the vast majority of cleanup times averaged approximately ten minutes, but several isolated events took well over an hour and thereby misrepresented the true averages. Taking the accurate statistics and modeling each procedure duration as a Gaussian function, the probability distribution is then generated. Negative values on the x-axis (corresponding to negative amounts of time) are truncated from the curve, as it is not realistic that a procedure could be scheduled for a negative amount of time.

Using the probability distribution, the algorithm then simulates scheduling the procedure for various amounts of time; starting with a minimum of one minute and simulating five minute intervals up to the maximum x-value of the probability distribution. At the conclusion of each simulation two pieces of data are recorded with respect to the maximum x-value of the probability distribution; that is, the total delay introduced when a procedure needs more time than it is scheduled for, and the total time returned to the schedule by successfully completing a procedure within a shorter time-frame than its full probability distribution describes. Originally, these two elements were meant to be set as a ratio of the former over the latter. After analyzing the results, the plot of total delay versus time was observed as a negative exponential function approaching infinity near the origin and approaching zero as the function inputs approached infinity. This value became functionally useless when attempting to use in any form of ratio, as it set the ratio to either zero or infinity depending on how the ratio was formed. Instead, the observation was made that the peak value from the plot of total return versus time lead the median of the probability distribution by a small margin. The simulated time-frame corresponding to the intersection of the probability distribution and this max value is then returned as the optimal scheduled time and is used to populate a lookup table containing procedure types, operating room assignments, and time-frames.

Using this lookup table, the algorithm can now recommend all scheduling details for any procedure within its memory. Taking the incomplete entries from the initial data import described above, the algorithm first references the lookup table then runs through a set of conditions prior to placing the procedure in the schedule.

Given that previously scheduled surgeries could have ended earlier than predicted, or been cancelled for a variety of reasons (for example, patient reliability), it is vital that the algorithm analyze the existing schedule for openings. It references all instances for the same assigned operating room that have not yet occurred, then looks for gaps larger than the total time required to complete the procedure in question. If such a gap is found, the procedure is then assigned to fill the opening. If it is not found, but such a gap does still exist, it will bump the rest of the schedule provided that the delay it causes is sufficiently small (must cause a delay of less than 10% of the procedure time in question). If none of the above are true, the algorithm defaults to scheduling the procedure onto the end of the existing schedule.

C. Server/Database Development

The server in this project is the foundation. It allows all of our information to be located in one spot, as well as creates an easy place for everything to communicate. We used a simple LAMP (Linux Apache MySQL, and PHP) server with Ubuntu as our Linux operating system. The server also has Matlab installed on it. This allows us to run our most important and fundamental part of our project, our Matlab scripts, via the command

line.

The MySQL database is very simple and small in scale. In the scope of this part of the project (the work is being continued the size of the database in terms of tables is growing) the MySQL database contains two tables. One of these tables contains all the information that is obtained for the users. The second table contains all of the data the hospital provided us and the information needed for our Matlab algorithms.

The server contains our PHP scripts which is used to begin processes as well as do the needed steps in between. All of the processes that the PHP scripts do is explained in the next sub section, iOS development.

Then lastly you have the web interface side of the server. This section of the server contains PHP and html files. These files create a web site portal which is accessible over the Internet. Currently this project, due to limitations, does not have a specific web site name. Another limitation is that it does not have the ability to be accessed outside of Miami's network. This could be easily fixed when need be, and when the project is further out of the development stages. This website portal is used for the administrative aspect of our project. Through this portal the user who has administrative access is able to add data to specified tables in our database. This will be talked about more in detail in our Website Development subsection below.

D. iOS Development

The iOS application which has been developed for this project is very simple in structure. It was coded in Apple's proprietary language, Swift 2.0, and consists of two very basic screens with very few fields and buttons.

The first screen is the login screen where a user inputs his/her username and password (Picture 1.1 in the appendices). The application then checks if the user sent a valid username and matching password. It does this by sending the information to our server, which then runs a PHP script to verify the username exists in the database and that the password provided matches. If the password does not match the username or the username simply does not exist the PHP script sends back a failure to the application. When this failure is received the application displays a warning pop-up letting the user know the information they inputed is incorrect (Picture 1.2 in the appendices). If the username exists and the password matches the provided username the PHP script returns a success back to the application and the application sends the user to the next, main, screen.

This main screen consists of our two most important aspects of this application (Picture 1.3 in the appendices). First it has the optimization button. This button runs our optimization Matlab script on the server, this script is explained more in detail under the Algorithm Development portion of this paper. It runs our optimization script by running a PHP script which is located on our server. Not only does this script run our optimization script but it also runs the intermediate processes needed. These processes are as followed: First it creates a CSV file from the information located in our table in our database. Next it converts this CSV file to a XLS file. This XLS file is read by our Matlab program, which the PHP script initiates. When the Matlab script is done running the Matlab program returns a CSV file and the Matlab program is closed. The PHP script then takes the CSV and imports and replaces our old data with the new data in the database table. This is all happening in the background on the server. The user's application is not effected by this process. As the button is pressed the application creates a warning dialogue and displays it. This warning simply tells the user that the optimization will take approximately ten to fifteen minutes and that he/she should wait that long to view the data our main algorithm provides (Picture 1.4 in the appendices).

The other button on this page is our view data button. This button runs a separate

PHP script which runs our main Matlab script. This script does the all the processes stated in the paragraph above which precede the Matlab script running. The difference is in the processes run after the Matlab program is closed. This is where the CSV file outputted by the program is converted to a JSON file. The information in the JSON file is grabbed by the iOS application. As all the above happens the application switches to a table view screen. The data that the application grabbed from the JSON is then displayed in this table view (Picture 1.5 in the appendices). Since this screen is separate from the main screen, it also consists of a back button. When this button is pressed it takes us back to the main screen.

Lastly a very simple logout button is located in the upper rate had corner of the main screen. This button is self explanatory, as it simply logs the user out and takes them back to the login screen.

E. Website Development

The website is based on LAMP web framework, and it is one of the most popular framework internationally. LAMP means Linux operating system + Apache HTTP server + MySQL database + PHP programming language. LAMP also has many advantages such like open-source, highly secure, low cost and can run on many platforms. In our website, it has a login page where doctors, nurses and patients are able to register. There are five tabs after users logged into the website, the five tabs including Schedule, Doctor info, Nurse info, Patient info and Operating room info. In Schedule tab, it shows the information for each OR Case, such as IDs of Doctors, Nurses and patient, Room location, Surgery type, time and so on. Users can add and delete cases to the table as they want by clicking the add and delete button. In the Doctor info tab, it includes doctor ID, Doctor First and Last name. In the Nurse info page, there are Nurse ID and nurse first name and last name. In the patient info page, there are patient ID, patient first, last name and procedure type. At last in the OR room page, users can see OR room ID, and room location. The use of the website is basically showing the information of the database and allowing doctors and nurses to modify the data, so that the algorithm part can do optimization using the those data.

Alternative Solutions

This project is a fairly specific and new concept which is still in growth. Many of the items which we could call an alternative solution could very much be added later in the development process. Still, we did go a specific route for certain reasons.

When beginning this process we did in debt research of this area to see what other people have done thus far. A couple studies had done similar projects in this field of interest, though instead of having a system like ours they decided to use excel. We very much could have done the same and used excel to come up with our results. Still, the route we decided to go we felt was newer and more robust. Using Matlab instead of excel allowed us to have much higher computation power. Also our system is able to be updated in real time, always changing our data instead of having one static input that you must follow. Therefor our data can always be updated and new results can be provided. With following this path we were able to be more unique and, in our mind, provide a better system.

We could have very well used some sort of cluster computer or super computer, but instead we used a small desktop computer which would not have as much computational power. This route was followed because we wanted to make sure we did not have all the computation resources that a super computer would provide. We wanted to make sure that every hospital that would like could use our system. Also with limiting our computational power it made us have to

optimize our system to the best of our abilities, instead of creating a poor resource intensive algorithm.

We could have used a different server specifications. Instead we went what we felt was easiest and most people would be able to access. Everyone has the ability to access LAMP server software for free. This was huge in our decision to choose this specific type of server. Another reason we chose a LAMP server is because we have worked with one before and we were familiar with the system, making the work a little bit easier on us.

Another small thing we could have changed was to create an Android application instead of iOS. This route was not chose for any specific reason other then it's what we decided we wanted to develop for. Still, like stated above, development is still in process. Therefor the group is creating an Android application so the ability for a hospital to use these devices will be possible.

Lastly, some of the reason we took the routes we did and ended up with the project we have was due to limitations of data. When we started this project we had an entirely different view point of where this project would end up. We wanted to create a system that actually created a true block schedule for the hospital for that day of surgeries. Do to doctor patient confidentiality and other hospital policies the data we were able to obtain was not ideal to follow this path. Instead we had create a different system which still would help the scheduling process and optimize the downtime, but from the data we were able to obtain.

Therefor as you can see all of our alternative routes we chose were thought out and for a specific reason. Many small changes or well thought out plans led us in the direction of how our project ended as a whole. When there was multiple routes to follow research was done and the group as a entirety chose which route we saw fit for this specific project. Still, sometimes we did not have the choice in the path we took but instead a specific route had to be followed due to limitations.

Model Evaluation

Using our model to schedule a set of surgeries identical to the original dataset from UCMC, our algorithm saved approximately 3% in terms of scheduling time. Given that this dataset spanned a total of 5721 hours across all operating rooms, our algorithm freed up 172 hours; however, this estimate cannot be taken as the total impact of our model for reasons discussed below.

One of the considerations of this model is the handling of cancelled or rescheduled surgeries. The dataset from UCMC does not include cancelled surgeries, only successful surgeries and their time spans. Thus, these results conclude that our model is capable of improving the overall time frame by 3% given that no surgeries are rescheduled or cancelled.

The dataset from UCMC was the only controlled set of data available to us, and as such is the only set we could compare with our optimized model. So, although capable of dynamically scheduling procedures to compensate for cancellations, only the artificially inefficient aspects of our model could be fully evaluated with the naturally inefficient aspects merely implemented.

Ethical Implications

The purpose of this system is improving efficiency of using operating room. So it could reduce cost of hospital and manage resources better, such as doctors, nurses, patients, and operating rooms. However, there are some issues. The database has a lot of data from the users of this system, such as information of doctors, nurses, patients, and schedule of operating rooms. And these information includes some private data, like patient’s medical history. So it needs better protection of the stored data, like stronger secrecy system or encryption algorithm. For economic consideration, the system requires additional costs at least for a server with the database and the website, and an engineer maintaining the system. It also requires a simple

tutorial lecture to ensure users properly use the system and each user have to own an iOS device or an android device to use the system. As the system is used, hospital needs to deal with some unexpected situations. For example, it may requires a backup plan to ensure that operating rooms are still working when the system is done. And for the data stored in the database, the system requires that all data need to be correct, not duplicated, and updated immediately.

Conclusion

Hospital optimization systems is a very good topic. Improve the operating room efficiency not only saves thousands dollars per minute but saves people's life. Health care is important to everyone. Our algorithmic solution helps the hospital to improve the efficiency of operating room. Our system allows operating room to board more patient in unit time. Our project including five main parts, they are data collection, algorithm development, server/database development, iOS development, website development. And all those parts interact with each other and do optimization.

Future Work

This project is not a one year project. The future work Develop an Android application. Since we already have the IOS application, developing an android application is very straightforward. optimization the interface of IOS and android apps can also be done in the future. It can improve the algorithm to make the operating room arrangement more efficient. Improve website functions. For example, add a function of search for doctor, nurse and patient information. Transfer the data from school server to hospital server. Because hospital server is more safe and stable. Improve system security through data encryption. Doctor information and patient information will be encrypted. Set up a hardware designed for our system.

References

Veinott, Arthur. *Lectures in Supply-Chain Optimization*. Diss. Stanford, 2005. Stanford, California: 2005. Print.

Litvak E, Long MC. "Cost and Quality Under Managed Care: Irreconcilable Differences?" *American Journal of Managed Care*, 2000 3(3): 305-312.

Litvak E. "Optimizing Patient Flow by Managing its Variability." In Berman S. (ed.): *Front Office to Front Line: Essential Issues for Health Care Leaders. Oakbrook Terrace, IL: Joint Commission Resources,* 2005, pp. 91- 111.