Stochastic Surgery Scheduling with Multiple Operating Rooms

Kabir Chandrasekher and Lev Tauz **Joint Work with: Jean Walrand, Rahul Jain, Ramtin Pedarsani, Rhonda Righter**

Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA

Aim

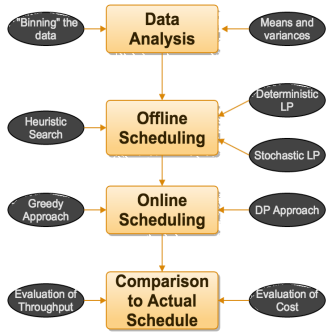
- To improve upon the current scheduling scheme used for surgeries by hospitals

Background

- The scheduling of surgeries is currently done manually in many hospitals
- Hospital administrators use empirical knowledge to determine the amount of time to allot
- They use block scheduling in which certain ORs are "blocked off" for certain departments and then schedule within these blocks
- Large amounts of idle time are built into the schedule to compensate for unexpected delays
- Three major sources of preventable cost: idle time in operating rooms, surgery overtime, and block overflow

Approach

- We use data from a year of surgeries to determine statistics and distributions
- The scheduler is split into two independent stages
- Stage 1: A linear programming offline approach to determine a schedule for new cases
- Stage 2: An online approach to fine-tune the schedule as real-time data arrives



Main Problems

- The combinatorial nature of the orderings of the surgeries provides an inescapable source of complexity
- The large number of variables causes the state space of the dynamic program to explode and makes iterative re-ordering approaches time inefficient

Deterministic LP Model

- Constraints: I cases, J ORs, K surgeons, D days, $[0,T]$ interval of time.
- Basic Variables: $x_{ijdt} = \begin{cases} 1 & \text{: Case } i \text{ is scheduled in OR } j \text{ on day } d \text{ at time } t \\ 0 & \text{: Case } i \text{ not scheduled in OR } j \text{ on day } d \text{ at time } t \end{cases}$
- $A_{ik} = \begin{cases} 1 & \text{Case } i \text{ assigned to surgeon } k \\ 0 & \text{Case } i \text{ not assigned to surgeon } k \end{cases}$, $B_{jkdt} = \begin{cases} 1 & \text{Surgeon } k\text{'s department assigned to OR } j \\ 0 & \text{Surgeon } k\text{'s department not assigned to OR } j \end{cases}$
- An indicator for case i being performed In OR j on day d at time t:

$$y_{ijdt} = \sum_{s=1}^{t} x_{ijds} - \sum_{s=1}^{t-D_i} x_{ijds}$$

- Overtime Variable: $p_i = \sum_{i,d,j,t>T} y_{ijdt}$
- Idle time Variable: $q_{j,d} = T - \sum_{i,t\leq T} y_{ijdt}$
- Block Overflow Variable: $r_{j,k,d} = T_{j,k,d} - \sum_{i,t\leq T_{j,k,d}-D_i} (t+D_i) x_{ijdt} A_{ik} B_{jkt}$
- Objective: $\min \alpha \sum_i p_i + \beta \sum_{j,d} q_{j,d} + \gamma \sum_{j,k,d} r_{j,k,d}$

Stochastic LP Model and Dealing with Stochastic Infeasibility

- Same constraint and objective as deterministic model with an added sample parameter
- Empirically shown to be computationally infeasible as the size of the linear program's variable matrix increases exponentially with the number of samples
- A key result as that the stochastic linear program model is computationally infeasible
- To deal with this, we propose that a near-optimal solution may be of the form $\mu + \alpha\sigma$
- To find the optimal value within this class of solutions, we use a local search to determine α
- We fix an order using the deterministic LP with the current α value and use hill climbing by perturbing its value by some epsilon, terminating when the α value stagnates

Scheduling Software Overview

Abstraction Layer #1: Raw Data
- Client data provided by the hospital

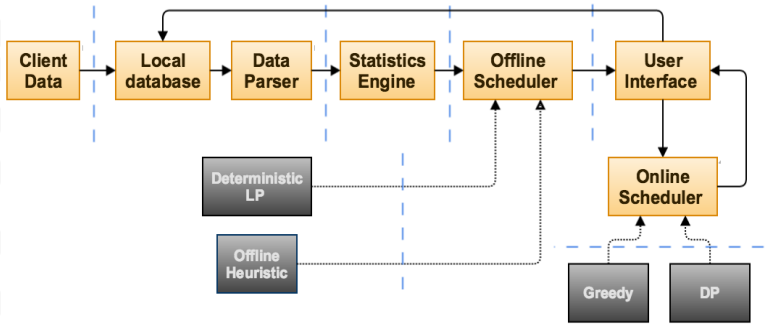
Abstraction Layer #2: Data Parser
- Client data stored in MySQL database
- Data presented to the next abstraction layer in a uniform format regardless of input

Abstraction Layer #3: Statistics Engine
- Currently calculates means and variances associated with various groupings
- In the future will include machine learning to infer the distributions associated with such groups

Abstraction Layer #4: Offline Scheduler
- Supports three offline scheduling algorithms
- Default scheduler is deterministic with means as durations
- Stochastic LP available
- Local search available

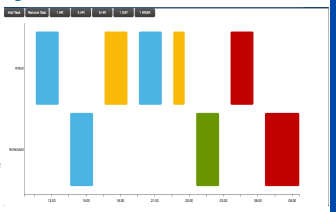
Abstraction Layer #5: User Interface
- Simple web application built utilizing Django
- Allows for input of cases to be scheduled
- Outputs schedule and gantt chart using d3.js that highlights critical sections

Abstraction Layer #6: Online Scheduler
- Current model uses a greedy scheduler
- Dynamic programming model also available
- Used to warn patients ahead of time and reduce patient waiting time



Software Internals and Adoption

- Backend written in python and is a web app using django with an output using d3.js.
- Our hope is for hospital administrators to adopt the usage of this system to augment their predictions
- For this purpose, it is one of our major goals to make an interactive and easy to use interface
- A sample visualization is pictured to the right



Conclusions

- The throughput of the system can be significantly improved even with naïve scheduling by means
- A stochastic linear programming approach is computationally infeasible
- Offline scheduling provides marked cost improvement over manual scheduling

Further Work

- It remains to be seen how to optimally determine the throughput of the system without dropping any cases from being scheduled
- The completion of the full-stack scheduling system will allow for ease of testing and visualization of various methods and provide more systematic data collection
- We would like to see the effects of both offline and online programming in tandem
- Incorporation of PACU constraints and examining the creation of block schedules provide additional challenges