Automated Scheduling of Residency Programs (Medtrics)

Jasper Ding, AC Li, Son Pham, Tung Phan

Computer Science department, Bucknell University, Lewisburg, PA

BACKGROUND

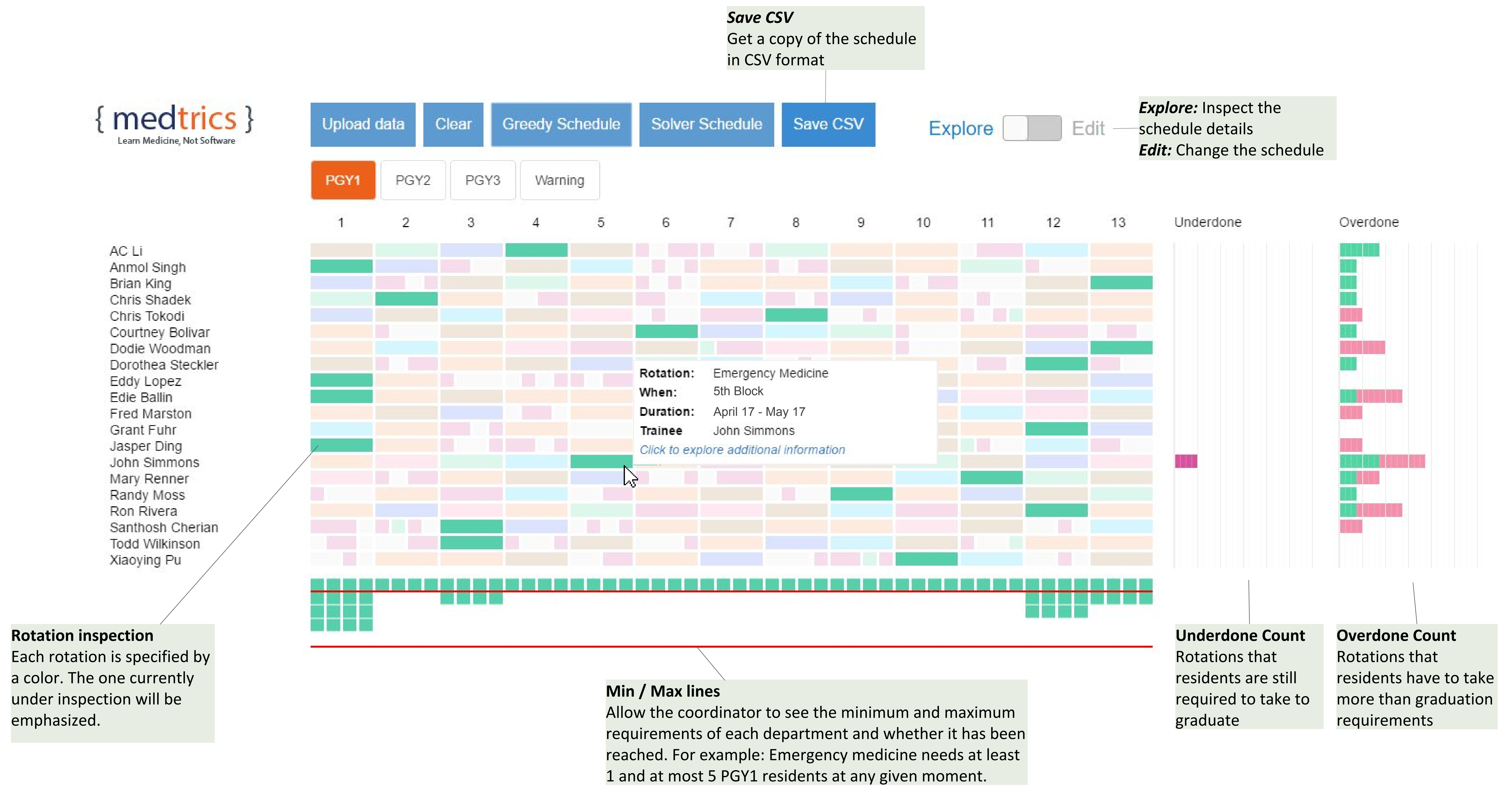
The client for our senior design project is **Medtrics**, a medical residency management software company based in Lewisburg, PA. In more than 15 institutions supported by Medtrics, there is a crucial task still performed manually and takes up to six weeks of mental work - rotation scheduling. Although this problem has been studied in academic researches and some individual hospitals, there has not been an attempt to develop a scalable and user-friendly system to work with multiple institutions.

PROBLEM

The scheduling problem involves assigning **residents** to different departments in a hospital (**rotations**). There are four main sets of requirements to satisfy:

1. Min/max manpower requirements for each department.
2. Graduation requirements for each resident.
3. Other specific heuristics imposed by the Accreditation Council for Graduate Medical Education (ACGME) such as Vacations, Electives, Rotation Length, Location, etc.
4. Student preferences over when to take rotations / Existing partially filled schedules

EXAMPLE PROBLEM



Rotation: Emergency Medicine
When: 5th Block
Duration: April 17 - May 17
Trainee John Simmons
Click to explore additional information

Save CSV
Get a copy of the schedule in CSV format

Explore: Inspect the schedule details
Edit: Change the schedule

Rotation inspection
Each rotation is specified by a color. The one currently under inspection will be emphasized.

Min / Max lines
Allow the coordinator to see the minimum and maximum requirements of each department and whether it has been reached. For example: Emergency medicine needs at least 1 and at most 5 PGY1 residents at any given moment.

Underdone Count
Rotations that residents are still required to take to graduate

Overdone Count
Rotations that residents have to take more than graduation requirements

PROBLEM SPECIFICITY

- Residents: 120 (40 in each PGY level)
- Rotations: 7
- Rotation constraints: Minimum and maximum number of PGY1, PGY2, PGY3 as well as any combinations between the 3 roles.
- 1 type of rotations can be broken apart to half blocks and quarter blocks

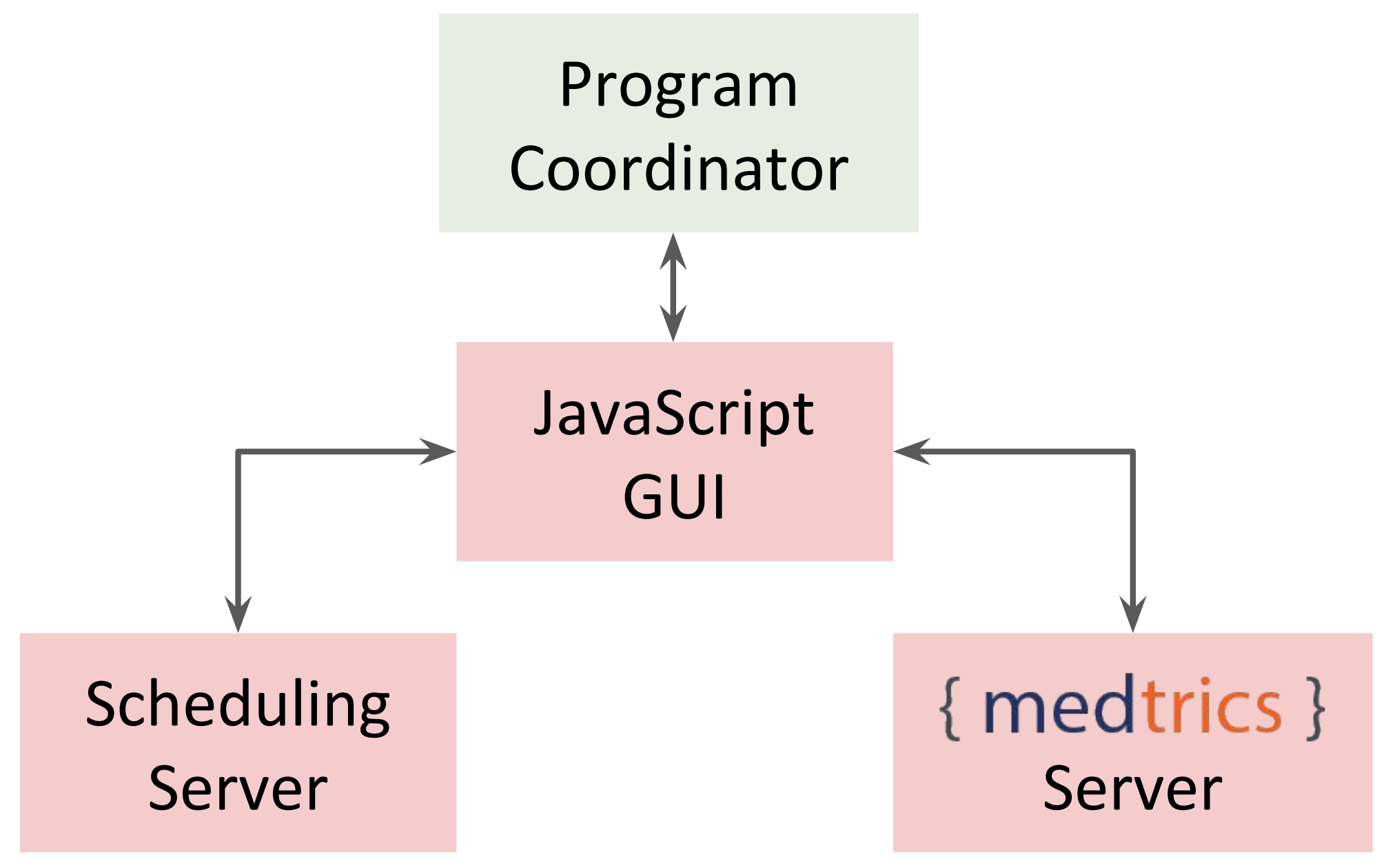
GREEDY ALGORITHM RESULTS

- Scheduling time: 0.12s
- Total underdone: 3.5 rotations (0.03 / student)
- Total overdone: 0 rotations (0 / student)
- Understaffed/Overstaffed rotations: 0

SOLVER ALGORITHM RESULTS

- Solver uses 49920 variables, 10112 constraints
- Scheduling time: 7.28s
- Total underdone: 0
- Total overdone: 0
- Understaffed/Overstaffed rotations: 0

DATA INTERFACE



The program coordinator will interact with our JavaScript Graphical User Interface (GUI), which allow them to pre-fill some rotations according to outside constraint. The coordinator can then press the "Schedule" button which will send the information to the Scheduling Server. Upon completion, the Scheduling Server will send back a full schedule that respect the partially filled schedule, and requirements of each department/resident. The coordinator will once again have a chance to modify the schedule for the last time before sending the schedule back to Medtrics standard platform.

USER INTERFACE

We use Pixi.js JavaScript Library that adopts WebGL to draw data visualizations efficiently to help users better understand the schedule generated by the two scheduling algorithms. The visualization also has interactivity that allows user to both explore and edit. There are also buttons to let users upload prefilled schedule, pick one of the algorithms, and save the schedule as .csv files.

Explore

- Users can view the minimum and maximum requirements for each rotation
- Users can view how each resident is doing regarding graduation requirements

Edit

- Users can pre-fill the schedule before running the automated scheduling algorithms
- Users can make adjustments to the rotation blocks on the resulting schedule generated by the algorithms to meet particular requirements

GREEDY ALGORITHM

Our greedy algorithm follows **four main steps:** [1]

1. If there exists a rotation in some period where teaching service demands (rotation min) have not been satisfied, then find residents who still have unsatisfied graduation requirements in that rotation, and assign the resident. Repeat until no more residents can be assigned.

2. If no resident exists with unsatisfied graduation requirements for a particular rotation with unsatisfied teaching service demands in some time block, find an unassigned resident, and assign to this rotation/time block.

3. If there are residents with unsatisfied graduation requirements for any rotation, choose an available period to assign them to that rotation.

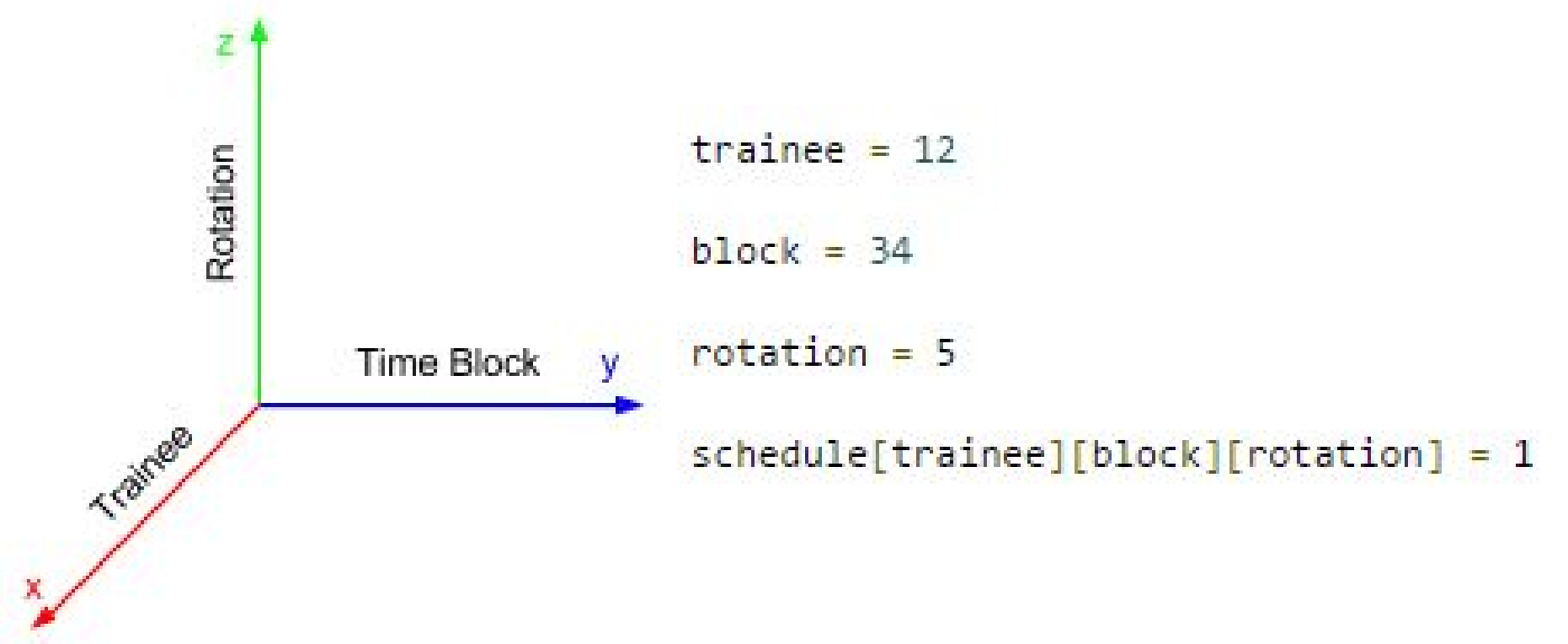
4. If there are still residents with unsatisfied graduation requirements for any rotation, choose a period occupied by vacation to replace them to that rotation.

SOLVER ALGORITHM

We translate the rotation scheduling problem into a **Integer Programming (IP)** Problem, and used Google or-tools linear solver wrapper to interact with the Coin-or Branch and Cut (CBC) Solver. A schedule is represented by a set of points on 3D space with axes: Resident, Time, Rotation. The Solver's objective is set to minimizing the number of blocks residents have to take. Each rotation demand, graduation requirement, etc. is translated into a mathematical constraint and then inputted into the solver. For example, **"A resident can only do one thing at any given time"** is translated to.

For resident **i**, during Time Block **j**, across Rotation **k**'s

$$0 \le \sum_{k} schedule[i][j][k] \le 1$$



```
trainee = 12
block = 34
rotation = 5
schedule[trainee][block][rotation] = 1
```

ACKNOWLEDGEMENTS

The completion of this project could not have been possible without the assistance of many people whose names may not all be enumerated. Their contributions are sincerely appreciated and gratefully acknowledged. However, the team would like to express our deep appreciation to the following:
Our client **Medtrics** for giving us the opportunity to execute this project as well as their constant support along the journey.
Bucknell professors **Mihai Banciu** and **Matt Bailey** for the early guidance with Integer Programming.
Professor **Brian King** for his valuable software engineering advices and guidance towards crucial engineering decisions.

REFERENCES

1. Guo, Jiayi, David R. Morrison, Sheldon H. Jacobson, and Janet A. Jokela. "Complexity results for the basic residency scheduling problem." Journal of Scheduling 17.3 (2013): 211-23. Web.