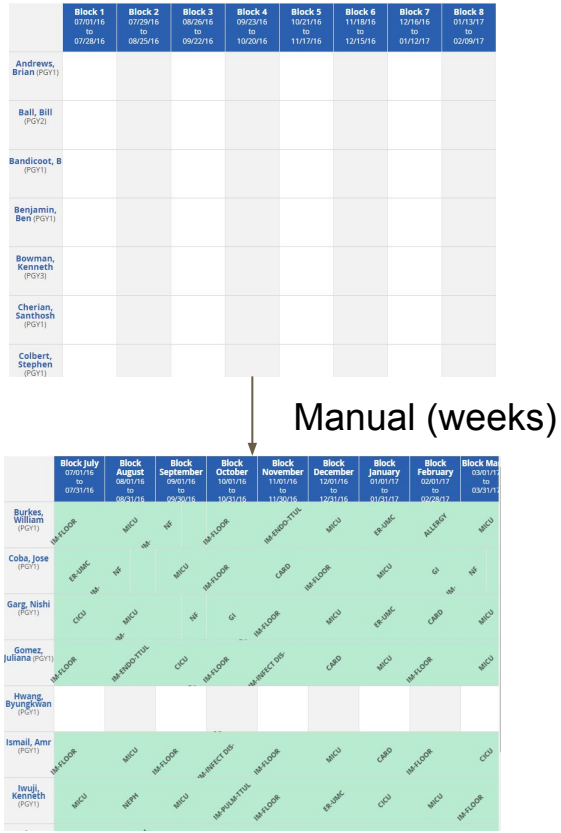
Medtrics Automated Scheduling

Jasper Ding, AC Li,
Son Pham, Tung Phan

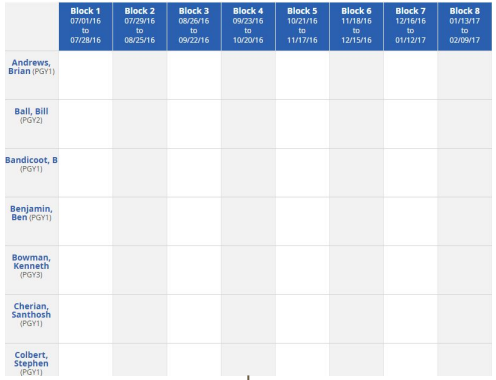
Before



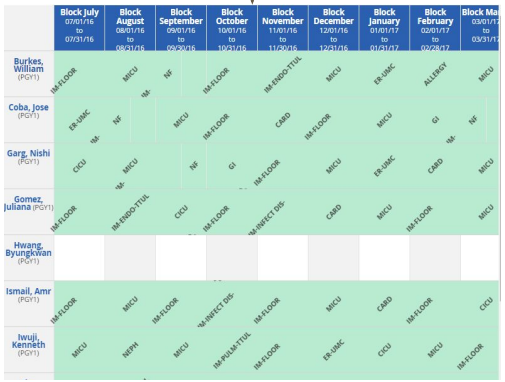
Background / Motivation

- Previously takes coordinators up to 6 weeks
- No scalable automated scheduling solution
- No helpful and user-friendly tool to assist the scheduling process

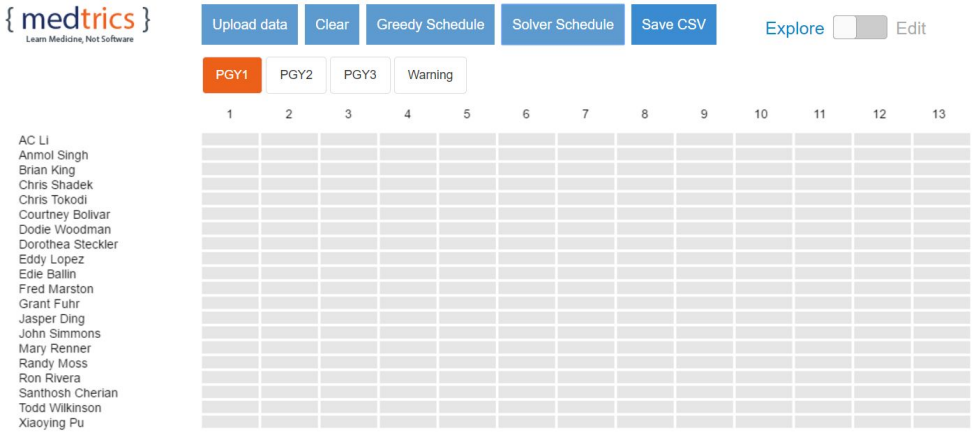
Before



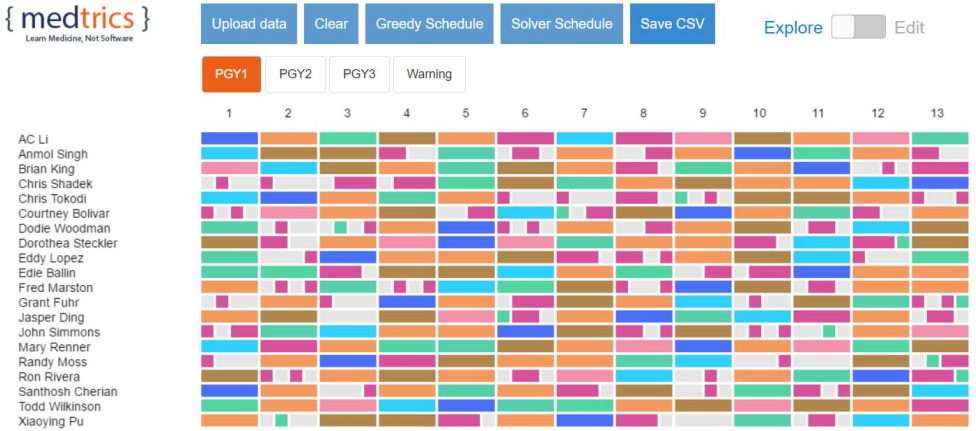
Manual (weeks)



After



Automatic (Seconds)



Problem

Four main sets of requirements to satisfy:

1. Minimum/Maximum staffing requirements for **each department**.

2. Graduation requirements for **each resident.**

3. Other **specific heuristics** imposed by the Accreditation Council for Graduate Medical Education (ACGME): Vacations, Electives, Rotation Min/Max Length, Location, etc.

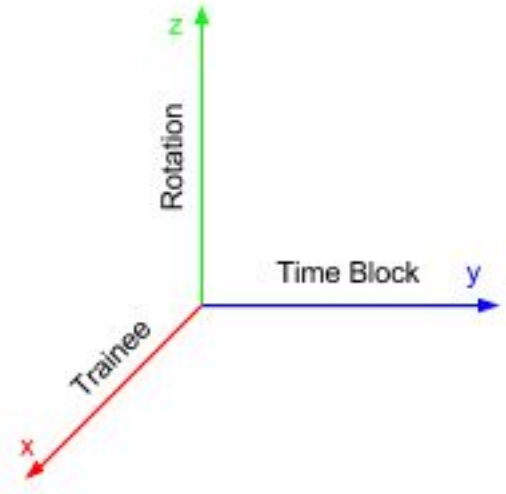
4. Student **preferences** over when to take rotations / Existing **partially filled schedules**

Solver Algorithm

Integer Programming (IP) Problem

A schedule → A set of points on 3D space with axes: **Trainee, Time, Rotation**

Example: Trainee 12, during Time Block 34, doing Rotation 5 can be thought of as point (12, 34, 5)



Python:

```
trainee = 12

block = 34

rotation = 5

schedule[trainee][block][rotation] = 1
```

Finding Best Schedule → Filling the 3D space with integer 0 or 1 optimally

Representing the requirements

Verbal requirements → Mathematical Constraints → IP Solver Constraints

Example: "A trainee can only do one thing at any given time"

→ For Trainee **i**, during Time Block **j**, across Rotation **k**'s, $0 \leq \sum_{k} schedule[i][j][k] \leq 1$

→ Python

```
for i in range(num_trainee):
    for j in range(num_block):
        constraint = solver.Constraint(0, 1)
        for k in range(num_rotation):
            constraint.SetCoefficient(attend_list[i][j][k], 1)
```

Problem Objective: Minimizing the number of blocks trainees have to take.

→ $min \sum_{i,j,k} schedule[i][j][k]$

→

```
objective = solver.Objective()
for i in range(num_trainee):
    for j in range(num_block):
        for k in range(num_rotation):
            objective.SetCoefficient(attend_list[i][j][k], 1)
objective.SetMinimization()
```

Actually Solving The Problem

Wrapper Interface Python API: **Google or-tools**

https://developers.google.com/optimization/

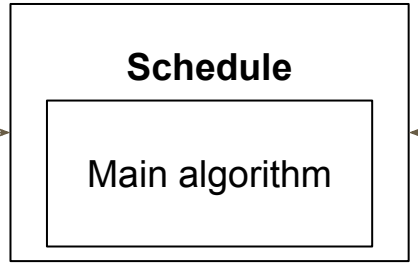
Third Party Open Source Solver: **Coin-or Branch and Cut (CBC) Solver**

https://projects.coin-or.org/Cbc

Performance:

A scheduling problem with **120 trainees, 7 rotations in 52 week period** was translated into a IP problem with **49920 variables, 10112 constraints** and takes up to **5.6 seconds** to solve.

Greedy Algorithm (Python)



Trainee

Main attributes	Type
requirement	dictionary
limit	dictionary
block	array(rotation Obj.)
role	string

Rotation

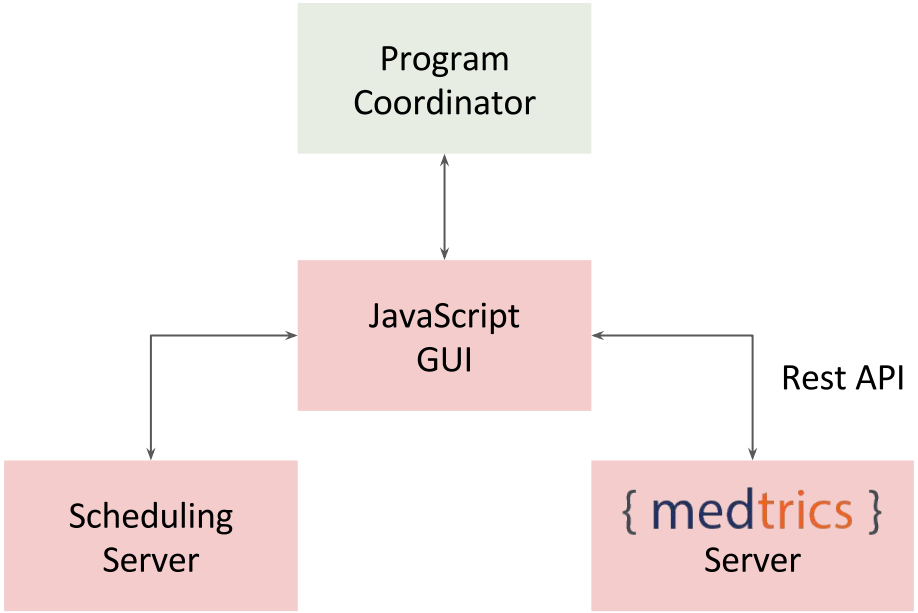
Main attributes	Type
min	int
max	int
min_block_length	int
id	int

Four big steps

	Condition		Action
1.	If a rotation exists in some period with unsatisfied min, and meanwhile, there exist residents with unsatisfied educational requirements for that rotation.	→	Choose the residents cover the rotation for that period.
2.	If no resident exists with unsatisfied educational requirements for a particular rotation that has unsatisfied teaching service demands in some period.	→	Choose an unassigned resident to assign to this rotation.
3.	If residents exist with unsatisfied educational requirements for some rotations.	→	Choose a period to assign them to that rotation.
4.	If residents exist with unsatisfied educational requirements for some rotations, after doing step 3.	→	Choose a period occupied by vacation to replace them to that rotation.

Source : 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.

Data Interfacing



GUI

- **HTML 5**, **JavaScript** and **CSS 3**
- **jQuery** for interaction with HTML
- **Pixi.js** library to draw data visualizations
- **qtip2.js** for tooltip balloons
- **Magnific-Popup.js** for the loader

Demo

Acknowledgements

Client

Santhosh Cherian

Chris Tokodi

Jon Davis

Matthew Terry

Thanks

Prof. Mihai Banciu

Prof. Matthew Bailey

Department of Computer Science

Bucknell School of Engineering

SPECIAL **Thanks**

Prof. Brian King

Questions?