
Medtrics Automated Scheduling

— Jasper Ding, AC Li,
Son Pham, Tung Phan —

Before

	Block 1 07/01/16 to 07/28/16	Block 2 07/29/16 to 08/25/16	Block 3 08/26/16 to 09/22/16	Block 4 09/23/16 to 10/20/16	Block 5 10/21/16 to 11/17/16	Block 6 11/18/16 to 12/15/16	Block 7 12/16/16 to 01/12/17	Block 8 01/13/17 to 02/09/17
Andrews, Brian (PGY1)								
Ball, Bill (PGY2)								
Bandicoot, B (PGY1)								
Benjamin, Ben (PGY1)								
Bowman, Kenneth (PGY3)								
Cherian, Santhosh (PGY1)								
Colbert, Stephen (PGY1)								

Manual (weeks)

	Block July 07/01/16 to 07/31/16	Block August 08/01/16 to 08/31/16	Block September 09/01/16 to 09/30/16	Block October 10/01/16 to 10/31/16	Block November 11/01/16 to 11/30/16	Block December 12/01/16 to 12/31/16	Block January 01/01/17 to 01/31/17	Block February 02/01/17 to 02/28/17	Block March 03/01/17 to 03/31/17
Burkes, William (PGY1)	IM-FLOOR	MCU	NR	IM-FLOOR	IM-RECT TBL	MCU	ER-UMC	ALLIED	MCU
Coba, Jose (PGY1)	ER-UMC	NR	NR	IM-FLOOR	CARD	IM-FLOOR	MCU	GI	NR
Garg, Nishi (PGY1)	CCU	NR	MCU	NR	GI	IM-FLOOR	MCU	ER-UMC	CARD
Gomez, Juliana (PGY1)	IM-FLOOR	IM-RECT TBL	CCU	IM-FLOOR	IM-RECT TBL	CARD	MCU	IM-FLOOR	MCU
Hwang, Byungkwan (PGY1)									
Ismail, Amr (PGY1)	IM-FLOOR	MCU	IM-FLOOR	IM-RECT TBL	IM-FLOOR	MCU	CARD	IM-FLOOR	CCU
Iwaji, Kenneth (PGY1)	MCU	NR	MCU	IM-FLOOR	IM-FLOOR	ER-UMC	CCU	MCU	IM-FLOOR

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

	Block 1 07/01/16 to 07/28/16	Block 2 07/29/16 to 08/25/16	Block 3 08/26/16 to 09/22/16	Block 4 09/23/16 to 10/20/16	Block 5 10/21/16 to 11/17/16	Block 6 11/18/16 to 12/15/16	Block 7 12/16/16 to 01/19/17	Block 8 01/19/17 to 02/09/17
Andrews, Brian (PG1)								
Ball, Bill (PG2)								
Bandicoot, B (PG1)								
Benjamin, Ben (PG1)								
Bowman, Kenneth (PG3)								
Cherian, Santhosh (PG1)								
Colbert, Stephen (PG1)								

Manual (weeks)

	Block July 07/01/16 to 07/31/16	Block August 08/01/16 to 08/31/16	Block September 09/01/16 to 09/30/16	Block October 10/01/16 to 10/31/16	Block November 11/01/16 to 11/30/16	Block December 12/01/16 to 12/31/16	Block January 01/01/17 to 01/31/17	Block February 02/01/17 to 02/28/17	Block March 03/01/17 to 03/31/17
Burkes, William (POT)	IM-FLOOR	MCU	HF	IM-FLOOR	IM-BBQ-DINING	MCU	ER-BMC	ALL-EXT	
Coba, Jose (POT)	ER-BMC	HF	MCU	IM-FLOOR	CARD	IM-FLOOR	MCU	GI	HF
Garg, Nishi (POT)	CCU	HF	MCU		GI	IM-FLOOR	MCU	ER-BMC	CARD
Gomez, Juliana (POT)		HF	MCU			IM-FLOOR	MCU	ER-BMC	CARD
	IM-FLOOR		CCU	IM-FLOOR	IM-WEST DIN	CARD	MCU	IM-FLOOR	MCU
Hwang, Byungkwon (POT)									
Ismail, Amr (POT)	IM-FLOOR	MCU	IM-FLOOR	IM-WEST DIN	IM-FLOOR	MCU	CARD	IM-FLOOR	CCU
Iwuli, Kenneth (POT)	MCU	HF/H	MCU	IM-PULVITALL	IM-FLOOR	ER-BMC	CCU	MCU	IM-FLOOR

After

[illegible]

Automatic (Seconds)

[illegible]

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

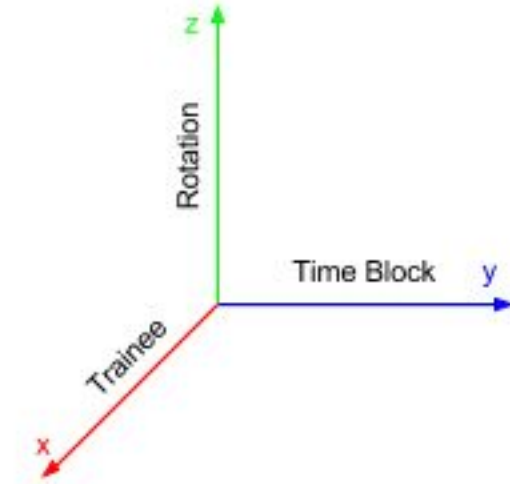
A schedule → A set of points on 3D space with axes: **Trainee, Time, Rotation**

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, $0 \leq \sum_k \text{schedule}[i][j][k] \leq 1$

→ 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.

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

→

```
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/>

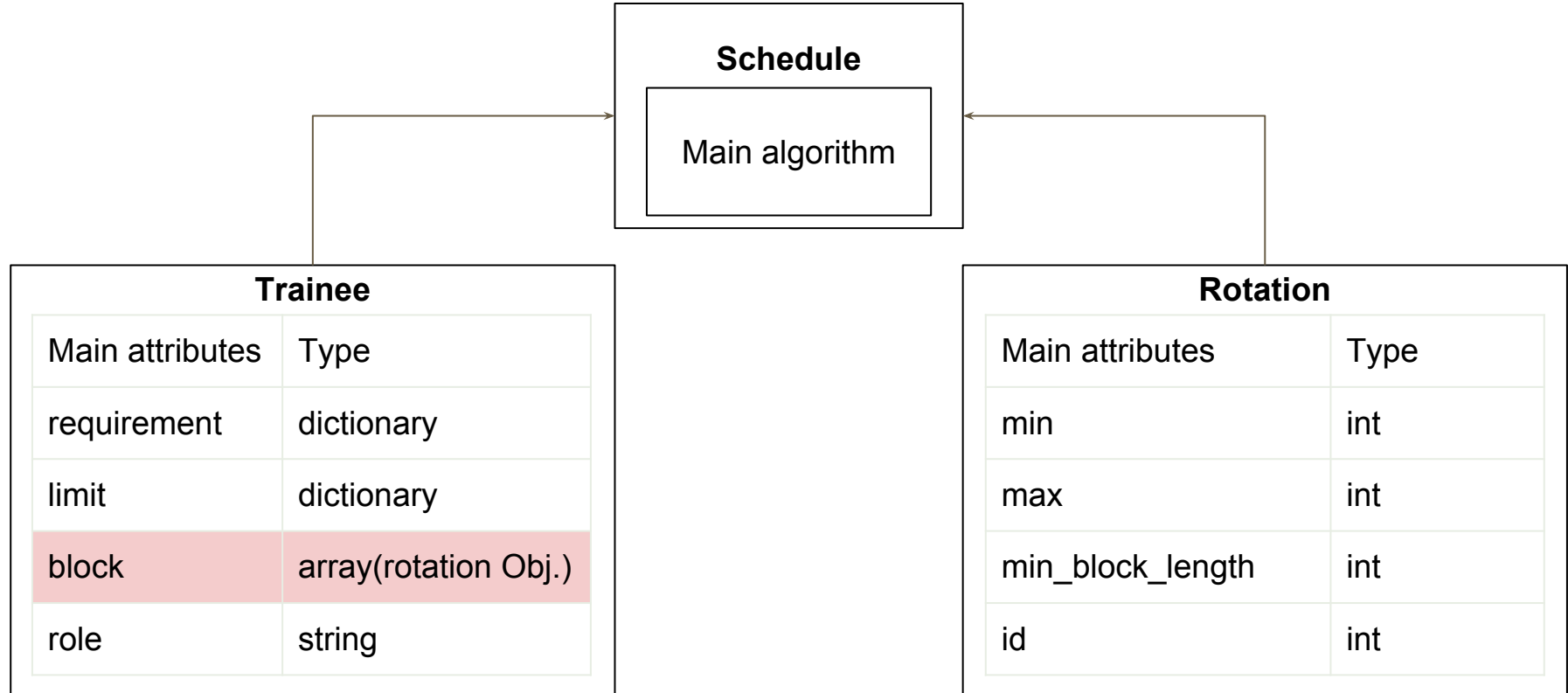
Third Party Open Source Solver: **Coin-or Branch and Cut (CBC) Solver**

<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)



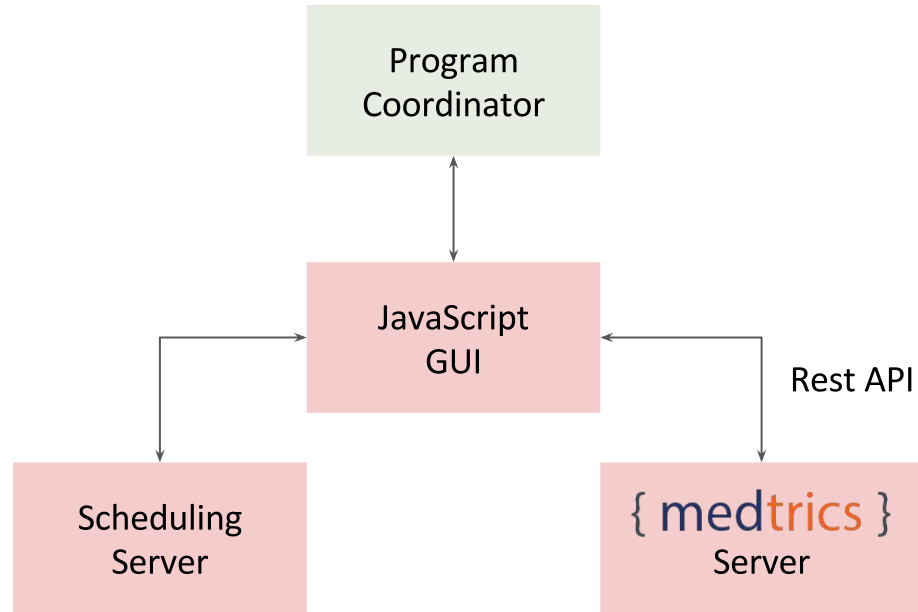
Four big steps

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. |

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?