# Determining Rotation Matching Using Linear Sum Optimization

by **Roshan Lodha**

**TODO:**

* **Zotero the papers**

## Abstract

The assignment of rotation positions to third year medical students is a critical aspect of their development as medical professionals. However, the process can be challenging due to the limited number of available positions and the need to match students with clerkships that align with their interests and career goals. In this paper, we frame rotation order assignment as a cost problem and propose the use of a linear sum optimizer built to assign students to an optimal rotation order. Linear sum optimization is a mathematical method that can be used to solve the assignment problem, allowing for the optimal assignment of clerkship positions to students in a way that minimizes the cost. We believe that this approach has the potential to improve the clerkship order assignment process and enhance the learning experiences of medical students.

## Introduction (still under work)

**@TORI: The exact algorithm I used was the Jonker-Volgenant algorithm, which is an improvement on the Hungarian algorithm (both guarantee completeness and optimality → My algorithm runs in O(n^3) time while hungarian algorithm runs in O(n^4) time so mine is MUCH faster).**

**JVA:** [**https://link.springer.com/article/10.1007/bf02278710**](https://link.springer.com/article/10.1007/bf02278710) **(idk what language this is)**

**no init implementation of JVA: https://ieeexplore.ieee.org/document/7738348**

Third-year medical students complete a series of core rotations in internal medicine, surgery, pediatrics, obstetrics, gynecology, neurology, psychiatry, and family medicine. Elective rotations often require a prerequisite core rotation; for example, to enroll in the orthopedics elective, students must first complete the core surgery rotation. Thus, students tend to prefer the order in which they are assigned to these rotations.

Currently, rotation matching is done stochastically, posing a huge time cost for both students and faculty. Unequal student preferences in rotation order led to challenges in finding an optimal assignment for all students with schools employing varying approaches. Many schools use random assignment, which then leads students swapping to get their desired rotations and no input in the process. Other schools have employed rank-based preference matching which uses a lottery system to assign as many of the top ranks as possible (Renner et al., 1988). However, simply using rank-based preference matching assigns the problem of cost of an unfavorable preference away from the student. A recent study applied the Hungarian algorithm to rotation matching which resulted in a greater number of students receiving one of their three desired choices (Mclean et al., 2021). However, this implementation had several limitations including inferior time complexity and lack of student input in cost determination. Specifically, use of the Jonker-Volgenant algorithm in place of the Hungarian algorithm coupled with a variable and tileable matrix problem formulation allows for increased generalizability, speed, and student input. \*\*\*roshan do you have any add ons for limitations to their algorithm or how ours is different Thus, here, we propose a student-centered rotation assignment algorithm that finds the optimal student-order pairing. Notably, we allow students to choose the cost of an unfavorable assignment on an individual basis while ensuring an equal number of students are assigned to each order.

## Methods

### Problem Formulation and Encoding

We reframed the problem of optimal rotation order as a minimum-cost assignment problem. Each student has the option of rotation orders which they assigned a cost to. These costs were used to formulate an tall matrix. Each row was a student’s preference assignment for the rotations.

#### Determining Costs

Students were given “beans” and were asked to divide and assign these beans however they would like to the rotation orders. Null submissions were assigned beans to each rotation. All responses were scaled such that the sum of beans was exactly . Each rotation order’s assigned beans was then converted to a cost by subtracting it from . For example, if a rotation was assigned beans, its associated cost would be . Hyperparameter optimization was used to determine the optimal number of beans for a given application.

##### Linear Cost Alternative

In order to allow for broader generalizability, our algorithm allows for a ranked-preference based assignment optimizer coupled to a beans to rank conversion tool. In practice, this tool was not used (see discussion for further details).

### Algorithm Design

#### Matrix Padding

Linear sum optimization requires a wide or square matrix. Thus, we add phantom students with no rotation order preference until the number of rows is in moduli space . Subsequently, we tile the matrix to a width of resulting in a square matrix. The row order was randomly shuffled to ensure that submission time was not a factor in determining rotation order preference.

#### Linear Sum Optimization

The optimal rotation order was calculated by calculating the linear sum optimization on the padded, square cost matrix in Python (SciPy: 1.9.3, Python 3.9.6).

### Error Testing

To determine the performance of the rotation assignment, we defined a novel error metric, , as the . is a real number in the range , with a lower number signifying less deviation from the optimal result. A of occurs when exactly assign beans to each preference.

## Results

### Definition of Parameters

In practice, we sought to assign students (variable 76-77) to a total of possible rotation orders.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Rotation 1 | Rotation 2 | Rotation 3 | Rotation 4 | Encoding |
| Option 1 | LAB | TBC2 | TBC3 | TBC1 | LAB – TBC2 – TBC3 – TBC1 |
| Option 2 | TBC2 | LAB | TBC1 | TBC3 | TBC2 – LAB – TBC1 – TBC3 |
| Option 3 | TBC3 | TBC1 | LAB | TBC2 | TBC3 – TBC1 –LAB – TBC2 |
| Option 4 | TBC1 | TBC3 | TBC2 | LAB | TBC1 – TBC3 – TBC2 - LAB |

*Table 1. Rotation order definitions.*

* LAB := Longitudinal Ambulatory Block
* TBC1 := Internal Medicine, Surgery
* TBC2 := Pediatrics, Obstetrics and Gynecology, Elective
* TBC3 := Neurology, Psychiatry, Elective

We employed a maximum number of beans per student to allow for integer divisions of beans into preferences.

### The optimal number of beans is highly variable.

An optimal number of beans was selected using hyperparameter optimization. For ease of student use, the minimum number of beans was chosen to be , as it allows for integer divisions between rotation orders. Testing across a wide range of beans revealed that a minimum number of beans minimized error (Figure 1). The cost matrix was sampled randomly and uniformly.

|  |
| --- |
| Figure 1 |
| *Figure 1. Delta error versus number of beans.* |

Analysis of a sample set of real world data showed a skew towards certain rotation orders. Further testing must be done to determine how the number of beans affect the overall error under various sampling distributions. We hypothesize that in real world deployment, increasing the number of beans would decrease the error due to sampling skew and a maximal difference between costs for a given student. In practice, we hypothesizes that the optimal number of total beans for each student is in moduli space due to ease of division for students; is a good choice.

### The error reduces as the number of students increases.

As the number of students increases, the total delta error decreases exponentially (Figure 2). In other words, the error was roughly constant despite increasing the number of students, suggesting better performance as the number of students increases.

|  |
| --- |
| Figure 2 |
| *Figure 2. Delta error versus number of students using a beans-based penalty.* |

### Deployment and Student Satisfaction

* real world this year (number of swaps, satisfaction, etc.)
* compare number of post-assignment swaps from previous years vs this year
* add “what would you like to see in the algo” to satisfaction survey

## Discussion

### Key Findings

#### Optimality and Completeness

In our problem, optimality was defined as a rotation order assignment in which no single swap would benefit all students involved in the swap. Completeness was defined as both an equal number of students assigned to each rotation order as well as all students being assigned to exactly 1 rotation order. In the case that the number of students was not in the moduli space , completeness was defined as a difference of no more than 1 student between the most filled and least filled rotation group. Linear sum optimization provides an optimal solution by definition. Completeness was ensured by matrix tiling.

#### Limitations

Real world behavior in rotation order selection is poorly modeled by a uniform distribution. Sampling of students’ preferences reveals high preferences for certain rotation orders. In practice, we found that rotation order 3 > rotation order 4 > rotation order 2 > rotation order 1 (Table 2).

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Order | Beans Assigned | Number All In |
| 1 | LAB - TBC2 - TBC3 - TBC1 | 27 | 1 |
| 2 | TBC2 - LAB - TBC1 - TBC3 | 475 | 13 |
| 3 | TBC3 - TBC1 - LAB - TBC2 | 859 | 26 |
| 4 | TBC1 - TBC3 - TBC2 - LAB | 490 | 15 |

*Table 2. Summary statistics from real-world deployment.*

This selection led to an increased number of students receiving a deeply unfavorable rotation order based on their choice of bean assignment.

#### Optimal Student Strategy

Due to a student-determined cost penalty and the unequal popularity of certain rotations, students could employ game theory to optimize their odds of receiving a certain rotation. For simplicity, consider a scenario with 75 students, where every student wanted rotation order 1 and no students wanted rotation order 4. In this case, only a maximum of 19 students could receive the top choice rotation. Thus, students had the option of assigning all their beans to rotation order 1 to maximize their chance of getting this rotation. However, if more than 19 students employed this strategy, several would be randomly assigned to a different rotation. Due to the relative unpopularity of rotation order 4, it is likely that most of these students would be assigned to this rotation. Hence, it may benefit students to “take the L” and assign all their beans to their second choice rotation.

Our algorithm allows for easy modification and eliminated of this aspect by having students rank their preferences followed by deterministic assigning a cost penalty to an unfavorable rotation order without the students consultation.

|  |
| --- |
| Figure 3 |
| *Figure 3. Delta error versus number of students using a linear penalty.* |
| Figure 4 |
| *Figure 4. Delta error versus number of beans using a linear penalty.* |

Testing of this method showed that there was decreased error with increase beans (Figure 4), indicating that error is roughly constant under a linear penalty regime. In practice, this was not used in order to allow finer control in students’ choices.

### Future Directions

#### Skewed Costs

Applying a weight to the cost matrix can skew the results to avoid assigning students to their last-choice preference. For example, adding an exponential penalty would more significantly penalize rotation orders with fewer beans, skewing the optimal result away from those set of solutions. Our algorithm allows for easy modification of a cost matrix. Hyperparameter optimization should be used to determine the best cost penality function for a given application.

#### Adding Distance Penalties

Within each rotation, students can be placed at several sites. Suburban hospital campuses pose additional environmental and travel cost for students. Future iterations of a non-random rotation matching algorithm can modify the cost function based on the distance a student has to travel to a given rotation. An example implementation could be to recursively run the algorithm within each rotation assignment using the distance traveled in miles as the of a rotation.

#### Adding “Couples Matching”

Often, students live with another medical student. To encourage carpooling, the cost function can be further modified to increase the odds that such students are placed in the same rotation.

#### Unequal Rotation Sizes

While our usage mandated an equal number of students in each rotation, updating the algorithm for unequal distributions is as simple as modifying the tiling function to include more repetitions of options that can accommodate a higher number of students.

## References

1. Munkres J. Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics 1957;5(1):32–8.
2. Crouse DF. On implementing 2D rectangular assignment algorithms. IEEE Transactions on Aerospace and Electronic Systems 2016;52(4):1679–96.
3. MacLean MT, Lysikowski JR, Rege RV, Sendelbach DM, Mihalic AP. Optimizing Medical Student Clerkship Schedules Using a Novel Application of the Hungarian Algorithm. Academic Medicine 2021;96(6):864–8.
4. Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 1955;2(1–2):83–97.
5. Kuhn HW. Variants of the hungarian method for assignment problems. Naval Research Logistics Quarterly 1956;3(4):253–8.

## License

Shield: [](http://creativecommons.org/licenses/by-sa/4.0/)

This work is licensed under a [Creative Commons Attribution-ShareAlike 4. International License](http://creativecommons.org/licenses/by-sa/4.0/).

[CC BY-SA 4.0](http://creativecommons.org/licenses/by-sa/4.0/)

### Authors

1. a[Roshan Lodha](https://roshanlodha.github.io)
2. aTori Rogness, aAlan Shen, aNikhil Pramod, aNitesh Mohan
3. aNeil Mehta, aCraig Nielsen

aCleveland Clinic Lerner College of Medicine