# An Approach to the Multi-Function Workers Job Scheduling Problem

by
Kaylee Enevold, Sarah Lanand, Young Park, Zohaib Syed

Minnesota State University, Mankato
Mankato, Minnesota

May, 2017

# Table of Contents

# Chapter 1

# Introduction

Mayo Clinic Health System employs over 17,000 people in more than 70 different communities across four states [2]. Many of the employees travel to offer specialized services to small communities on a regular basis. This creates a large-scale, dynamic scheduling problem in which the true optimal solution becomes unobtainable as size and complexity increases. However, it may be more favorable to find a near optimal solution if the computational load is low. We will attempt to find a balance between accuracy and efficiency.

The goals of our project are [2]

1. develop mathematical algorithms for multi-function workers scheduling problem; and

2. develop strategies to find near optimal solution(s) without going through the usual exhausting search.

To develop an algorithm that can solve the Mayo scheduling problem, we focus our attention on the multi-function worker scheduling problem. We will be able to translate the results back to the context of the Mayo scheduling problem afterwards.

## 1.1 Multi-Function Worker Scheduling Problem

The multi-function worker scheduling problem can be defined through a set of workers, a set of machine cells, and a set of skills shared by the two groups. Let us start with the set of $p$ skills

$$S = \{s_1, s_2, ..., s_p\}.$$

Each machine cell and worker is defined by a subset of $S$. Let us call the set of $n$ workers $W$

$$W = \{w_1, w_2, ..., w_n\}$$

where each $w_i$ represents a worker and $w_i \subseteq S$. Similarly, let us call the set of $m$ machine cells $M$

$$M = \{m_1, m_2, ..., m_m\}$$

where each $m_j$ represents a machine cell and $m_j \subseteq S$.

We have the added requirement that workers are not allowed to move between machine cells. That is, a worker may only work in one machine cell at a time. In order to make sure each skill is covered in each machine cell, we must assign a subset of workers to each machine cell $m_i$ such that $m_i$ is a subset of the union of the workers assigned. We define the optimal assignment as one that uses the least number of workers.

## 1.2 Assumptions of the Problem

In order to develop an algorithm, we need a strict set of assumptions to make the problem well-defined. Our assumptions are as follows:

1. Assignments are considered only for a fixed time.
2. Workers and machine cells are defined by the skills they have or require.
3. There does not exist a hierarchy of the skills.
4. Workers may cover all skills at once in a single machine cell.
5. Movement between machine cells is not allowed.
6. The workers, machine cells, and their respective skill sets are known.
7. Efficiency is defined by minimizing the number of workers assigned.

Because time is fixed, the algorithm will not allow for dynamic scheduling where hours or days vary. We are also assuming that every skill is "equal" in a sense. That is, in the case of the Mayo problem, having a lab technician certification and a degree in pediatric surgery are considered equally valuable. Also, because efficiency is determined by minimizing the number of workers, we are assuming that all workers cost the company the same amount of money. In this case, the true cost to the company in terms of paying salaries is not being considered.

Now, we may turn our attention to developing an algorithm that will find a balance between accuracy and efficiency within the context of the multi-function worker scheduling using machine cells and the assumptions outlined above.

# Chapter 2

# The Algorithm

In this chapter, we explain the development of our algorithm. We are able to avoid searching through each possible solution, and instead finding a solution based on the demand of the skills and the similarity between workers and machine cells.

An optimal solution may always be found using exhaustive search. However, the time required makes it an impossible method for even slightly large problems. By not considering every possible solution, we trade accuracy for efficiency. The question is how to keep accuracy without checking every solution. We attempt to achieve this by ranking skills, workers, and machine cells by priority, and assigning based on similarity between each worker and each machine cell.

Recall the given information of the problem. We have a set of skills $S = \{s_1, s_2, ..., s_p\}$, a set of workers $W = \{w_1, w_2, ..., w_n\}$ and a set of machine cells $M = \{m_1, m_2, ..., m_m\}$ where for each $w_i$ and $m_j$, $w_i \subseteq S$ and $m_j \subseteq S$.

We construct representation matrices $A$ and $B$ where $A$ represents the skills found in the set of workers (rows = workers, columns = skills), and $B$ represents the skills required in the set of machine cells (rows = machine cells, columns = skills).

## 2.1 Step 1: Compute frequencies and check existence

First, we sum all of the columns of $A$ and $B$ to get the counts of skills found in the workers and the counts of skills required by the machine cells. For each skill $s_i$, the count of $s_i$ in the set of workers is

$$W_{s_i} = |\{w \in W | s_i \in w\}| = \text{colsum}\left(A_i\right)$$

and the count of $s_i$ in the set of machine cells is

$$M_{s_i} = |\{m \in M | s_i \in m\}| = \text{colsum}\left(B_i\right).$$

For each $s_i$, we can compute the difference

$$D_{s_i} = W_{s_i} - M_{s_i}$$

to check existence. If $D_{s_i} < 0$ for any $i \in \{1, 2, ..., p\}$, no solution exists. In this case, we stop the algorithm. This small check may save a lot of time and computation. However, this is not a guaranteed condition for existence so we may have problems that pass this check which do not actually have a solution.

## 2.2 Step 2: Sort skills

In order to avoid exhaustive search, we rank the skills by priority which is defined by the demand of the skills with respect to the machine cells and workers. We use the difference between the count of a skill in the set of workers and the count of the skill in the set of machine cells to define the demand of the skill. For each $s_i$, the demand is defined as the difference

$$D_{s_i} = W_{s_i} - M_{s_i}.$$

Then when comparing differences, a lower value indicates that the demand for $s_i$ is higher. Thus, we sort the skills using these differences where a lower value means a skill is listed higher. This results in an arrangement of $S$ which we denote as $L$,

$$L = [l_1, l_2, ..., l_p],$$

where $D_{l_1} \leq D_{l_2} \leq \cdots \leq D_{l_p}$.

## 2.3 Step 3: Sort workers and machine cells

Next, we use $L$ to sort the rows of $A$ and $B$. A worker or machine cell with a skill or skills higher on the list $L$ will be listed higher within the matrices $A$ and $B$. The columns of the matrices are also rearranged according to $L$. This results in new matrices $A'$ and $B'$ where the top row has more 1's in the left columns, and the bottom row has 1's in the right columns. An example is

$$A' = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

## 2.4 Step 4: Compute similarity

Next, we attempt to define the similarity between a worker and a machine cell. We do this by counting the number of skills the two have in common and subtracting the number of skills that the worker has but the machine cell does not require. The number of skills each worker and each machine cell has in common can be found by multiplying $A'$ and $(B')^T$. This new matrix has $n$ rows (number of workers) and $m$ columns (number of machine cells). Each cell of the matrix represents the number of skills in common between a worker and a machine cell. For example,

$$\begin{aligned} AB_{ij} &= A_i \cdot (B_j)^T \\ &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \\ &= 1 + 0 + 0 + 0 + 1 = 2 \end{aligned}$$

which gives us the number of skills that the $i$th worker in the matrix $A'$ and the $j$th machine cell in the matrix $B'$ have in common.

To further define the similarity between each worker and machine cell, we remove the number of unused skills the worker has. This number of unused skills is the total number of skills a worker has minus the number of skills in common with a machine cell. For example, the number of unused skills by assigning the $i$th worker in $A'$ to the $j$th machine cell in $B'$ is

$$\text{rowsum}\,(A'_i) - AB_{ij}.$$

We then subtract this number from the cell $AB_{ij}$ to define the similarity between the $i$th worker and the $j$th machine cell as

$$\begin{aligned} &AB_{ij} - (\text{rowsum}\,(A'_i) - AB_{ij}) \\ &= 2 \cdot AB_{ij} - \text{rowsum}\,(A'_i)\,. \end{aligned}$$

This results in a newly updated version of $AB$ which we call $AB'$.

## 2.5 Step 5: Assign

Now, we can assign each machine cell a worker using the information in $AB'$. Each column of $AB'$ represents a machine cell, and each row represents a worker. We start with the first column. Choose the highest value in that column such that the corresponding cell in $AB$ is not zero. We assign this corresponding worker to the given machine cell (represented by this column), and then remove it as a future option by making the corresponding row in $AB$ all zeros. We continue through each column making similar assignments.

At the end of this process, we check whether there are any skills left uncovered in any machine cell. If all skills are covered, we stop. If not, we restart from the beginning.

In the next chapter, we showcase the results of our algorithm on two small examples.

# Chapter 3

# Examples

This chapter focuses on two multi-function worker scheduling examples where the number of workers, machine cells, and skills are small. We discuss the results by comparing the algorithm's solutions to the true optimal solutions.

## 3.1 Example 1

In the first example, we have skills $S = \{1, 2, 3, 4, 5, 6\}$, and the following workers and machine cells:

- workers

  $$w_1 = \{1, 3, 6\}, w_2 = \{1, 2, 3, 4\}, w_3 = \{2, 5, 6\}, w_4 = \{2, 3, 5, 6\}, w_5 = \{1, 6\}$$

- machines

  $$m_1 = \{1, 2, 5, 6\}, m_2 = \{1, 3, 4\}, m_3 = \{2, 3, 6\}$$

We will give the main steps of the algorithm for this example. The starting matrices $A$ and $B$ are

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

**Step 1**: Compute the frequencies and check existence

| | | | | | |
|---|---|---|---|---|---|
| $W_1 = 3$ | $W_2 = 3$ | $W_3 = 3$ | $W_4 = 1$ | $W_5 = 2$ | $W_6 = 4$ |
| $M_1 = 2$ | $M_2 = 2$ | $M_3 = 2$ | $M_4 = 1$ | $M_5 = 1$ | $M_6 = 2$ |
| $D_1 = 1$ | $D_2 = 1$ | $D_3 = 1$ | $D_4 = 0$ | $D_5 = 1$ | $D_6 = 2$ |

Since $D_i \geq 0$ for each $i \in \{1,2,3,4,5,6\}$, the problem passes the existence check.

**Step 2**: Sort skills

Since $D_4 \leq D_1 \leq D_2 \leq D_3 \leq D_5 \leq D_6$, the sorted skill list is

$$L = [4,1,2,3,5,6].$$

**Step 3**: Sort workers and machine cells

Using $L$, the workers and machine cells are sorted in the following orders

$$W_L = [w_2, w_1, w_5, w_4, w_3]$$

$$M_L = [m_2, m_1, m_3]$$

and thus, $A'$ and $B'$ are

$$A' = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$B' = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

**Step 4**: Compute similarity

$$AB = A' \cdot (B')^T = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 3 & 3 \\ 0 & 3 & 2 \end{bmatrix}$$

The matrix we subtract from $AB$ to update the similarity is

$$\begin{bmatrix}4&4&4\\3&3&3\\2&2&2\\4&4&4\\3&3&3\end{bmatrix} - \begin{bmatrix}3&2&2\\2&2&2\\1&2&1\\1&3&3\\0&3&2\end{bmatrix} = \begin{bmatrix}1&2&2\\1&1&1\\1&0&1\\3&1&1\\3&0&1\end{bmatrix}$$

Then the updated $AB$ is

$$AB' = \begin{bmatrix}3&2&2\\2&2&2\\1&2&1\\1&3&3\\0&3&2\end{bmatrix} - \begin{bmatrix}1&2&2\\1&1&1\\1&0&1\\3&1&1\\3&0&1\end{bmatrix} = \begin{bmatrix}2&0&0\\1&1&1\\0&2&0\\-2&2&2\\-3&3&1\end{bmatrix}$$

**Step 5**: Assign

Now, we can use $AB'$ to assign a worker to each machine cell. Since 2 is the maximum element in column 1, we assign $w_2$ to $m_2$ (row 1 represents worker 2 and column 1 represents machine cell 2). Immediately after this assignment, the 1st row of $B'$ is updated so that the skills supplied by the assignment are now 0. Also, the 1st row of $A'$ is changed to zeros to prevent worker 2 from being assigned again.

Using the same process, $w_3$ is assigned to $m_1$, and $w_4$ is assigned to $m_3$. This gives the assignment

- $w_3 \rightarrow m_1$
- $w_2 \rightarrow m_2$
- $w_4 \rightarrow m_3$

and $A'$ is now

$$A' = \begin{bmatrix}0&0&0&0&0&0\\0&1&0&1&0&1\\0&1&0&0&0&1\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

Now, we start the whole process over from Step 1 and let $A = A'$ and $B = B'$. Running the process one more time produces the final assignment:

- $w_3, w_5 \to m_1$
- $w_2 \to m_2$
- $w_4 \to m_3$

We use 4 out of 5 workers which is the true optimal value. Thus, in this example, the algorithm gives a true optimal solution.

## 3.2 Example 2

In the second example, we have skills $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, and

- workers

$$w_1 = \{1, 2, 3, 4, 9, 10\}, w_2 = \{1, 3, 4, 5, 6\}, w_3 = \{3, 7, 8, 9\},$$

$$w_4 = \{1, 4, 5, 6, 7, 9, 10\}, w_5 = \{2, 5, 7, 8, 9, 10\},$$
$$w_6 = \{5, 9\}, w_7 = \{1, 4, 8\}, w_8 = \{1, 2, 4, 6, 8\},$$
$$w_9 = \{4, 7, 8\}, w_{10} = \{2, 4, 5, 7, 10\},$$
$$w_{11} = \{1, 2, 4\}, w_{12} = \{1, 3, 4, 10\}, w_{13} = \{1, 2, 3, 4, 5, 8\},$$
$$w_{14} = \{3, 4, 6, 7, 8\}, w_{15} = \{2, 3, 10\}$$

- machine cells

$$m_1 = \{1, 3, 4, 6, 7, 9\}, m_2 = \{1, 3, 5, 6, 8, 10\}, m_3 = \{1, 2, 4, 5, 6, 8\},$$

$$m_4 = \{1, 2, 4, 5, 7, 9, 10\}, m_5 = \{2, 3, 7, 8, 9, 10\}$$

The algorithm gives the following assignment

- $w_4, w_{15} \to m_1$
- $w_2, w_9, w_{12} \to m_2$
- $w_6, w_8 \to m_3$
- $w_1, w_{10}, w_{11} \to m_4$
- $w_3, w_5 \to m_5$

We used 12 out of 15 workers in this case. However, it is known that the optimal value is 10 workers so the algorithm does not produce an optimal solution. The solution of 12 workers is still an improvement from assigning all 15, and two extra workers is not a large increase in cost.

# Chapter 4

# Conclusion

Here, we summarize the notable aspects of our algorithm, and discuss future directions of study.

## 4.1 Discussion

Our algorithm gives optimal or near optimal results for the two examples presented in the last chapter. We accomplished this using a small number of operations and avoiding an exhaustive search of the possible solutions. It should be noted that in Example 2, we did not achieve the optimal solution which is 10 workers, but our solution is near optimal with 12 workers. It is unclear whether the algorithm will produce a near optimal solution for larger problems. In order to translate the results of our study to the Mayo Clinic problem, we need to know how our algorithm performs in problems that contain thousands of workers and hundreds of machine cells with a large number of skills.

If our algorithm does work relatively efficiently with large-scale problems, we can easily translate the result to the Mayo Clinic Health System scheduling problem for a fixed time frame. Each machine cell represents a department, clinic, or hospital, and each worker is defined by their certifications or degrees. The departments require certain types of workers defined by their certifications, and thus, the optimal assignment uses the minimal number of workers that cover all of the required certifications. However, we would need to tweak the assumptions and algorithm to account for more complex situations.

## 4.2 Future Work

First and foremost, we need to test the algorithm with larger problems in order to get a better picture of its efficiency and when it may or may not result in near optimal solutions. In the future, we would like to expand upon the work done in this project by extending our list of assumptions to accommodate more general scheduling problems. First and foremost, we need to test the algorithm with larger problems in order to get a better picture of its efficiency and when it may or may not result in near optimal solutions. There are three aspects of our assumptions that we would like to change in order to make the problem more realistic.

The first is the fixed time assumption. In the larger Mayo scheduling problem, employees work in different locations on different days which means the scheduling is dynamic (it changes with respect to time). This would require additional constraints and an additional layer within the algorithm to perform assignment at different times. The constraints may be defined by bounding the number of hours worked per week for each worker, scheduling blocks of time so that workers' hours follow normal scheduling (i.e. 9:00 to 5:00), and allowing only workers with certain skills to be assigned to different machine cells on different days (orthopedic surgeon may visit rural clinics two days a week, but clinic nurse works in only one clinic).

Another assumption we would like to change is the lack of a hierarchy within the set of skills. Realistically, not all skills are treated equal in terms of hiring employees. For example, surgical specialties are ranked higher than nursing certifications. There are going to be far more registered nurses than there are cardiac surgeons. Having this hierarchy predefined could allow us to improve the sorting of skills, workers, and machine cells so that assignment is more efficient.

Lastly, the assumption that the optimal assignment only depends on number of workers should be replaced. The true condition for optimality is with respect to cost. If there are hierarchies to the skills, we can define appropriate salaries for the workers. Now, the goal is to minimize cost while adhering to all the previous constraints. A doctor would not be filling the function of a nursing assistant because they would cost more than hiring a nurse who does not have the certifications of a doctor. This would help to further ensure that assignment makes sense in terms of "wasting" workers with a vast array of skills on just covering a small function within a machine cell.

# Bibliography

[1] Rowell, Eric. "Big-O Complexity Chart." *Algorithmic Complexity Cheat Sheet*, bigocheatsheet.com. Accessed 29 Mar. 2017.

[2] Lee, Namyong. "Industrial Math." *Industrial Mathematics Problem*, Minnesota State University, Mankato, 2017, mnsu.ims.mnscu.edu/d2l/.