The Mathematics of a Detailing Marketplace

Ian Shaw LTJG, USN

January 13, 2019

1 Matching Algorithm

The intent of this algorithm is to provide stable pairings between job owners and job seekers based on their ranked preferences. The algorithm's initial conception and definition of stability can be found in Gale and Shapely's 1962 publication in the January *The American Mathematical Monthly* [1]. Interestingly, this research was funded by the Office of Naval Research. In short, a stable system is one where every job owner and job seeker are paired with the best possible preference for each; in other words no two job seekers or job owners can switch their assignments to both their benefits. Depending on how to construct this algorithm, the result could be focused on giving preference to the job seeker or to the job owner; we choose here to provide optimality for the job seeker. A given situation could be optimal to both, but by nature of needing an initiating agent we give that to the seeker. Also important to note, since a job owner can have multiple positions (e.g. four ensigns allotted for a ship's CO), that job owner can be matched with multiple seekers, but no seeker can have multiple owners.

This is the same algorithm that won the Nobel Prize in 2012 for its application in the National Residency Match Program. In their case, the algorithm is constructed to provide optimality for the applicant rather than the hospital program. They surmised that providing the best position for the seeker improved organizational performance.

[1] http://www.u.arizona.edu/~mwalker/501BReadings/Gale&Shapley_AMM1962.pdf

1.1 Gale-Shapely Algorithm

Algorithm 1: Deferred Acceptance

Result: There are no pairs $(O_i, S_i), (O_j, S_j)$ such that the pairings $(O_i, S_j), (O_j, S_i)$ would be preferred by all parties.

Consider a set of preferences $\{P^x_{y,z} \in \mathbb{Z}^+ : x \in \{O, S\}, y \in \{1, \ldots, n\}, z \in \{1, \ldots, m\}\}$. This indicates the positive integer preference ranking of either the job owner or seeker (O or S) for the n available jobs and m seekers. Also know that l_n is the number of available positions in each job.;

for $c \in \{1, \ldots, n\}$ **do**
    For each job, initialize the job owner's hiring slate to be empty ;
    $O_c = \{\}$;
end
for $c \in \{1, \ldots, m\}$ **do**
    For each job seeker initialize their indicator to say *un-slated* ;
    $I_c = 0$;
    Also initialize to look at the first preference of each job seeker ;
    $g_c = 1$;
end
Check if there are more jobs than seekers, or more seekers than jobs ;
$a = \max\left(0, m - \sum_{c=1}^{n} l_c\right)$;
Seek jobs until either all seekers are hired or all jobs are spoken for ;
while $\left(\sum_{c=1}^{m} I_c \leq a \right)$ **do**
    Iterate through all the job seekers ;
    for $c \in \{1, \ldots, m\}$ **do**
        Find a job for a seeker c only if they are not slated for a job ;
        if $I_c == 0$ **then**
            Look for the job that is seeker c's g^{th} preference ;
            $j = r$ such that $P^S_{r,c} = g_c$;
            If job j has open positions ;
            if $|O_j| < l_j$ **then**
                Add the seeker c to the slate of owner of job j ;
                $O_j += S_c$;
                Indicate that seeker c tentatively has a job ;
                $I_c = 1$;
            Or if seeker c preferred by job owner j than their least preferred person currently on their slate ;
            else if $P^O_{j,c} < P^O_{j,w}$ *such that* $P^O_{j,w} = \max\{P^O_{j,v} | S_v \in O_j\}$ **then**
                Remove the seeker w (least preferred) from the slate of owner of job j ;
                $O_j -= S_w$;
                Indicate that seeker w tentatively does not have a job ;
                $I_w = 0$;
                Add the seeker c to the slate of owner of job j ;
                $O_j += S_c$;
                Indicate that seeker c tentatively has a job ;
                $I_c = 1$;
            end
            Indicate that the $g^{th} + 1$ preference of seeker c has been considered ;
            g_c++;
        end
    end
end

1.2 Co-Location Algorithm

Suppose the goal of a system owner is to allow couples to co-locate more often, as was another intention of the National Residency Math Program. This can be achieved by allowing couples to submit preferences as pairs. In this scenario, a position is only given if the position improves the combined preference of the couple.

Algorithm 2: Deferred Acceptance with Co-Location

Result: There are no pairs $(O_i, S_i), (O_j, S_j)$ such that the pairings $(O_i, S_j), (O_j, S_i)$ would be preferred by all parties.

Consider a set of preferences $\{P^x_{y,z} \in \mathbb{Z}^+ : x \in \{O, S\}, y \in \{1, \ldots, n\}, z \in \{1, \ldots, m\}\}$ and some of the pairs are submitted together $\{(P^x_{y_1,z}, P^x_{y_2,z}) \in \mathbb{Z}^+ : x \in \{O, S\}, y_1, y_2 \in \{1, \ldots, n\}, z \in \{1, \ldots, m\}\}$, and indicate pairing by $p_y = 0$ if submitted as a single or $p_{y_1} = y_2, p_{y_2} = p_{y_1}$ if as a couple. This indicates the positive integer preference ranking of either the job owner or seeker (O or S) for the n available jobs and m seekers. Also know that l_n is the number of available positions in each job.;

for $c \in \{1, \ldots, n\}$ **do**
  For each job, initialize the job owner's hiring slate to be empty ;
  $O_c = \{\}$;
end
for $c \in \{1, \ldots, m\}$ **do**
  For each job seeker initialize their job to be *un-slated* ;
  $J_c = 0$;
end
Find the number of job seekers who submitted preferences as singles ;
$m_s = \sum_{c=1}^{m} \mathbb{1}(p_c == 0)$;
Find the number of job seeking entities, in this case a couple counts as one entity. ;
$m_e = m_s + \frac{m - m_s}{2}$;
Check if there are more jobs than seekers, or more seekers than jobs ;
$a = \max\left(0, m - \sum_{c=1}^{n} l_c\right)$;
Seek jobs until either all seekers are hired or all jobs are spoken for ;
while $\left(\sum_{c=1}^{m} \mathbb{1}(J_c \neq 0) \leq a \right)$ **do**
  Iterate through all the job seeking entities ;
  for $c \in \{1, \ldots, m_e\}$ **do**
    Choose the appropriate seeking process of a single or a couple;
    if *single* **then** `SeekSingle(`*seeker priorities, owner priorities, owner slates, seeker indicators*`)` ;
    else `SeekCouple(`*seeker priorities, owner priorities, owner slates, seeker indicators*`)` ;
  end
end

1.3 Single Seeker Function

Algorithm 3: Seeking Function for Singles

Function `SeekSingle(`*seeker priorities, owner priorities, owner slates, seeker indicators*`)`

Find a job for a seeker c only if they are not slated for a job ;
if $J_c == 0$ **then**
  Iterate down the list of seeker c's preferences ;
  for $p \in \{1, \ldots, n\}$ **do**
    $j = r$ such that $P^S_{r,c} = p$;
    If job j has open positions ;
    if $|O_j| < l_j$ **then**
      Add the seeker c to the slate of owner of job j ;
      $O_j += S_c$;
      Indicate that seeker c tentatively has job j ;
      $J_c = j$;
    Or if seeker c preferred by job owner j than their least preferred person currently on their slate ;
    else if $P^O_{j,c} < P^O_{j,w}$ *such that* $P^O_{j,w} = \max\{P^O_{j,v} | S_v \in O_j\}$ **then**
      Remove the seeker w (least preferred) from the slate of owner of job j ;
      $O_j -= S_w$;
      Indicate that seeker w tentatively does not have a job ;
      $J_w = 0$;
      Add the seeker c to the slate of owner of job j ;
      $O_j += S_c$;
      Indicate that seeker c tentatively has job j ;
      $J_c = j$
    end
  end
end

1.4 Couple Seeker Function

Algorithm 4: Seeking Function for Singles

Function `SeekCouple(`*seeker priorities, owner priorities, owner slates, seeker indicators*`)`

Find a job for the seeking couple c if either of them are not slated for a job ;
if $J_c == 0$ **then**
  Iterate down the list of seeker c's preferences ;
  for $p \in \{1, \ldots, n\}$ **do**
    $j = r$ such that $P^S_{r,c} = p$;
    If job j has open positions ;
    if $|O_j| < l_j$ **then**
      Add the seeker c to the slate of owner of job j ;
      $O_j += S_c$;
      Indicate that seeker c tentatively has job j ;
      $J_c = j$;
    Or if seeker c preferred by job owner j than their least preferred person currently on their slate ;
    else if $P^O_{j,c} < P^O_{j,w}$ *such that* $P^O_{j,w} = \max\{P^O_{j,v} | S_v \in O_j\}$ **then**
      Remove the seeker w (least preferred) from the slate of owner of job j ;
      $O_j -= S_w$;
      Indicate that seeker w tentatively does not have a job ;
      $J_w = 0$;
      Add the seeker c to the slate of owner of job j ;
      $O_j += S_c$;
      Indicate that seeker c tentatively has job j ;
      $J_c = j$
    end
  end
end

2 Competitiveness Score

The competitiveness score for any given job could be defined as the average preference ranking of the job across all seekers

Consider a set of preferences $\{P^x_{y,z} \in \mathbb{Z}^+ : x \in \{O, S\}, y \in \{1, \ldots, n\}, z \in \{1, \ldots, m\}\}$. This indicates the positive integer preference ranking of either the job owner or seeker (O or S) for the n available jobs and m seekers.

Thus the competitiveness score C for a given job j could be defined as

$$C_j = \frac{1}{m} \sum_{y=1}^{m} P^x_{y,j}$$

3 Similarity Measure

3.1 Math

Consider the following definitions.

$$\vec{w} = \text{ weight vector, } \in \mathbb{R}^{q\times 1} \tag{1}$$
$$1 = \sum_{k=0}^{q} \vec{w}_k \tag{2}$$
$$Q = \text{ Quality Matrix, } \in \mathbb{R}^{q\times n} \tag{3}$$
$$0 \leq Q \leq 1 \tag{4}$$
$$S = \text{ Similarity Matrix, } \in \mathbb{R}^{n\times n} \tag{5}$$
$$0 \leq S \leq 1 \tag{6}$$
$$q + 1 = \text{number of qualities encoded} \tag{7}$$
$$m = \text{number of persons} \tag{8}$$
$$\tag{9}$$

The element-wise the similarity score between Person i and Person j is

$$S_{ij} = 1 - (Q_i - Q_j)^2 \bullet \vec{w} = 1 - \frac{1}{q}\sum_{k=0}^{q}(Q_{k,i} - Q_{k,j})^2\vec{w}_k$$

In words:

> *The similarity of two people is the square distance of their encoded quality vector, weighted by the importance of similarity for each quality.*

3.2 Possible Alteration

Suppose a quality q is considered to be more valuable as it increases (for example, competency in some skill), without any negative consequences. In order to not negatively impact a person for being different but better in a quantitative way (but still different), the definition of similarity would need to be altered to be non-equal ($S_{ij} \neq S_{ji}$), but could take the following formulation.

$$S_{ij} = 1 - \max(Q_i - Q_j, 0) \bullet \vec{w} = 1 - \frac{1}{q}\sum_{k=0}^{q}\max(Q_k, i - Q_k, j, 0)\vec{w}_k$$

In the case that Person i is better than or equal to Person j in some metric, yet absolutely greater in at least one quality

$$S_{ij} > S_{ji}$$

3.3 Explanation

For a given pair of people, Person i and Person j, their similarity score S_{ij}, is a number between 0 and 1. With 0 being completely dissimilar, 1 being completely identical in terms of the encoded qualities.

The Matrix Q is the quality matrix. The entry Q_{ki} is a number between 0 and 1 indicating the strength of quality k for Person i.

The vector $\vec{w}$ is the weight vector. The entry $\vec{w}_k$ is a number between 0 and 1 indicating the importance of the quality k to determine the value of a person for the position in question.

3.3.1 Implementation

1. Hiring official determines the qualities important for their position (past jobs, competence in certain skills, etc.). These are the qualities 1 through q.
2. Hiring official determines the importance of each quality to the position on a scale of 1 (absolutely necessary) to 0 (irrelevant). These values form the vector $\vec{w}$.
3. Position seekers (the n individuals) or some authority generate the quantitative measure for each quality q, these populate the matrix Q. These could be 1/0 for yes/no (ex: have or have not attended a certain training), or some continuous scale (ex: 0.7 for 70% qualified in a particular skill). This can be done by asking the position seekers, referencing their official records, or administering some sort of evaluation.
4. Evaluate Similarity scores across the set of people.
5. Leverage similarity score information.

3.3.2 Use Cases of Similarity Score

1. If hiring official wants a new employee most similar to a previous one, choose a position seeker with the highest similarity score.
2. If a hiring official wants a diverse group, can create an optimization function with the minimal similarity across the team.

3.4 Future State

After enough records and performance metrics developed, can instead use records as data and performance metrics as labels to develop a machine learning algorithm to predict people's performance in a specific job. This could be used in concert with, or replace the need for a similarity score.