Detailing Marketplace: Pilot for US Navy

Ian Shaw, LTJG, USN Zac Dannelly, LTJG, USN Brian Yarborough, 1LT, USAF

February 6, 2019

Abstract

This paper seeks to outline the mathematics necessary to enable a detailing marketplace for the Department of Defense (DoD). A Detailing Marketplace is a system by which military members due to rotate (due to the mandate to change jobs every 1-3 years) can transparently rank their job preferences and the people who own those jobs can supply their preference of incoming personnel. This problem is rather unique to the military due to the captive market of many members obligated to remain in service due to contract, desiring to stay in for a pension, wishing to stay in through a sense of service, and an inability for lateral entry (almost all members needing to start from the bottom).

This process is shared in some regards by the medical school graduates applying to the residency stage of their training. These graduates are applying to the pool of U.S. residency programs. The difference to college admissions is that the pool is also rather narrow, mostly coming from U.S. based medical schools, and the specificity of skill-set required for success is better understood. This similarity is why our initial matching algorithm (Gale-Shapely Deferred Acceptance Algorithm) is the same one used by this process, the National Residency Match Program. The Gale-Shapely algorithm's application to the residency matching problem earned the Nobel Prize in 2012.

This paper also proposes an optimization based solution to the matching process. The optimization is not found in the National Residency Match Program as medical students have the ability to reject their assigned position, a choice not always given to military members. Thus an optimization can often find a more optimal solution for the system (Department of Defense) at the expense of a few forced members.

Due to the importance of co-locating dual-military households (where two family members are in the military), we focus our matching algorithm and optimization proposals on solutions that would guarantee 95% or greater co-location rate.

Acknowledging the difficulty of wrangling disparate and dated personnel data, this paper also explores helpful metrics that can be gleaned simply from submitted, ordered preferences by job seekers and job owners. These are competitiveness, similarity, generalism, and specialization. Interestingly, due to the fact that preferences are expressed on job seekers and the jobs themselves, these metrics can be developed about the jobs or the job seekers.

Further the paper ends with suggested metrics that would be gleaned if personnel data beyond preferences was accessible, clean, and structured. These include a similarity measure based on quality encodings and a suggested ordering of possible jobs or applicants. The latter is proposed to be enabled by deep learning (the underlying technology of Artificial Intelligence (AI)). The suggested ordering would not make decisions on placement, but rather provide job seekers and job owners with metrics distilling the vast amount of information about that which they intend the rank. The metric would aid them in the process of making their rankings.

The code to demonstrate the matching algorithms, optimization, and preference-based metrics can be found in Ian Shaw's Github Repository `Dynamic_Manning`. [1]

[1] `https://github.com/ieshaw/Dynamic_Manning`

Contents

1 Overview

In this report we outline the mathematical basis for a detailing marketplace. We recognize that previous efforts have been attempted in this arena, many failing due to the inaccessibility of Navy personnel data. For this reason, most of this report focuses on what can be accomplished without personnel data, just the submitted preferences of job seekers and job owners. Sections 2 and 3 focus on efforts that would be enabled by only preferences, not until Section 4 do we allude to efforts that could be pursued if further data were available.

1.1 Notation

Throughout the paper, we will reference the notation listed in this section.

$$m = \text{number of different jobs available} \tag{1}$$
$$n = \text{number of persons} \tag{2}$$
$$O_i = \text{set of seekers hired by owner of job } i \tag{3}$$
$$S_i = \text{job of seeker } i \tag{4}$$
$$C = \text{Co-Location Matrix, upper triangular} \tag{5}$$
$$C_{ij} = \begin{cases} 1 & j > i, \text{ and Seeker } i \text{ requests co-location with Seeker } j \\ 0 & j \leq i, \text{ or Seeker } e_{i,1} \text{ does not request co-location} \end{cases} \tag{6}$$
$$E = \text{Seeker Entity Matrix}, \in \mathbb{Z}^{+,n_e \times 2} \tag{7}$$
$$e_{i,2} = \begin{cases} r & \text{Seeker } e_{i,1} \text{ requests co-location with Seeker } r \\ 0 & \text{Seeker } e_{i,1} \text{ does not request co-location} \end{cases} \tag{8}$$
$$n_c = \text{number of couples requesting co-location} \tag{9}$$
$$= \sum_{i=1}^{n} \mathbb{1}(e_{i,2} == 0) \tag{10}$$
$$n_e = n - n_c \tag{11}$$
$$\vec{P_i^S} = \text{Preference vector of job seeker } i, \ \in \mathbb{Z}^{+,m \times 1} \tag{12}$$
$$P^S = [\vec{P_i^S} | \dots | \vec{P_n^S}] \in \mathbb{Z}^{+,m \times n}, \tag{13}$$
$$\text{Preference Matrix of Seekers} \tag{14}$$
$$\vec{P_i^O} = \text{Preference vector of job owner } j, \ \in \mathbb{Z}^{+,n \times 1} \tag{15}$$
$$P^O = [\vec{P_i^O} | \dots | \vec{P_n^O}] \in \mathbb{Z}^{+,n \times m}, \tag{16}$$
$$\text{Preference Matrix of Job Owners} \tag{17}$$
$$\vec{A} = \text{Position Available vector } \in \mathbb{Z}^{+,m \times 1} \tag{18}$$
$$a_j = \text{Amount of positions for job } j, \ \in \mathbb{Z}^{+} \tag{19}$$
$$X = \text{Placement Matrix} \in \{0,1\}^{n \times m} \tag{20}$$
$$x_{i,j} = \begin{cases} 1 & \text{if } S_i \text{ is slated for job } j \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

2 Matching Algorithms

2.1 Gale-Shapely 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 is 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 benefit. 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 commanding officer), 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.

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.

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

2.2 Co-Location Algorithm: Adopting Gale-Shapely

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. A couple is considered co-located if they both receive a position in a submitted pair. 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.

for $i \in \{1, \ldots, m\}$ **do**
  For each job, initialize the job owner's hiring slate to be empty ;
  $O_i = \{\}$;
end
for $i \in \{1, \ldots, n\}$ **do**
  For each job seeker initialize their job to be *un-slated* ;
  $J_i = 0$;
end
for $i \in \{1, \ldots, n_e\}$ **do**
  For each job seeking entity initialize their status to be *un-slated* ;
  $I_i = 0$;
end
Check if there are more jobs than seekers, or more seekers than jobs ;
$q = \max\left(0, n - \sum_{i=1}^{m} a_i\right)$;
Seek jobs until either all seekers are hired or all jobs are spoken for ;
while $\left(\sum_{i=1}^{n} \mathbb{1}(J_i \neq 0) \leq a \right)$ **do**
  Iterate through all the job seeking entities ;
  for $e \in \{1, \ldots, n_e\}$ **do**
    Choose the appropriate seeking process of a single or a couple;
    if $e_2 == 0$ **then** SeekSingle(*seeker priorities, owner priorities, owner slates, seeker indicators*) ;
    else SeekCouple(*seeker priorities, owner priorities, owner slates, seeker indicators*) ;
  end
end

2.2.1 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 e = (e_1, 0) only if they are not slated for a job;
Iterate down the list of seeker e_1's preferences until they are places
 in a job;
p = 1;
while I_{e_1} == 0 do
    j = r such that P^S_{r,e_1} = p;
    If job j has open positions;
    if |O_j| < a_j then
        Add the seeker e_1 to the slate of owner of job j;
        O_j += S_{e_1};
        Indicate that seeker e_1 tentatively has job j;
        J_{e_1} = j;
        I_e = 1;
    Or if seeker e_1 is preferred by job owner j over their least
     preferred person currently on their slate;
    else if P^O_{j,e_1} < P^O_{j,w} such that P^O_{j,w} = max{P^O_{j,v} | S_v ∈ 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 as an
         individual;
        J_w = 0;
        Indicate that their entity does not have a job, thus if submitted
         as a couple even if the other member has a job they must move
         onto their next preference;
        I_{w_e} = 0 such that w_{e_1} ∈ w_e;
        Add the seeker e_1 to the slate of owner of job j;
        O_j += S_{e_1};
        Indicate that seeker e_1 tentatively has job j;
        J_{e_1} = j;
        I_e = 1;
    end
end
```

2.2.2 Couple Seeker Function

Algorithm 4: Seeking Function for Couples

Function `CheckPriority(`*seeker, job*`)`
- `If job` j `has open positions or if seeker` c `preferred by job owner` j `than their least preferred person currently on their slate ;`
- **if** $|O_j| < a_j$ *or* $P^O_{j,c} < P^O_{j,w}$ *such that* $P^O_{j,w} = \max\{P^O_{j,v} | S_v \in O_j\}$ **then**
  - **return** *1*;
- **return** *0*;

Function `SeekCouple(`*seeker priorities, owner priorities, owner slates, seeker indicators*`)`
- `Find a job for the seeking couple` $e = (e_1, e_2)$ `if either of them are not slated for a job ;`
- `While both members of the seeking entity do not have a job, iterate down the list of preferences of seeker entity` e `;`
- Initialize $p = 1$;
- **while** $I_e == 0$ **do**
  - $j_1 = r$ such that $P^S_{r,e_1} = p$;
  - $j_2 = r$ such that $P^S_{r,e_2} = p$;
  - `Check if this set of jobs could go to the couple ;`
  - **if** `CheckPriority(`e_1, j_1`)` *and* `CheckPriority(`e_2, j_2`)` **then**
    - `Flag that they are tentatively assigned positions ;`
    - $I_e = 1$;
    - `Assign the positions;`
    - **for** $(c, j) \in ((e_1, j_1), (e_2, j_2))$ **do**
      - **if** $|O_j| < a_j$ **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 `is preferred by job owner` j `over 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 as an individual;`
        - $J_w = 0$;
        - `Indicate that their entity does not have a job, thus if submitted as a couple even if the other member has a job they must move onto their next preference ;`
        - $I_{w_e} = 0$ such that $w \in w_e$;
        - `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**
  - `If you have not found a position, look at the next preference ;`
  - $p++$;
- **end**

2.3 Optimization

Alternate to deterministic algorithms is a linear programming (optimization) approach. This allows system owners to input strategic objectives in the seeker-owner job matching process. The importance of job seeker preference can be weighted to be more important than owner, or vice versa. Requirements can also be added, as you will see in the formulation below.

The optimization function takes the form:

$$\min \quad \sum_{i=1}^{i=n} \sum_{j=1}^{j=m} f(x_{i,j}) \tag{22}$$

$$\text{such that} \quad \sum_{j=1}^{m} x_{i,j} \leq 1 \quad \forall i \in \{1, \dots n\} \quad \texttt{only one job per person} \tag{23}$$

$$\sum_{i=1}^{i=n} \sum_{j=1}^{j=m} x_{i,j} == \min(n, m) \quad \forall j \in \{1, \dots m\} \tag{24}$$

$$\texttt{either all the jobs are filled or everyone has a job} \tag{25}$$

$$\sum_{i=1}^{n} x_{i,j} \leq a_j \quad \forall j \in \{1, \dots m\} \quad \texttt{all jobs are at or below capacity} \tag{26}$$

$$\frac{1}{n_c} \sum_{i=1}^{n_e} C(e_1, e_2) \geq 0.95 \quad \texttt{at least 95\% of couples are co-located} \tag{27}$$

The Goodness Function f is the strategic objective function of the assignment process. For the sake of this paper, we set it to value the preference of the seeker twice as much as the preference of th job owner.

$$f(x_{i,j}) = 2P^S_{i,j} + P^O_{j,i}$$

In matrix form this can be re-written (with $tr()$ indicating the trace):

$$\min \quad tr(2X^T P^O) + tr(XP^S) \tag{28}$$

$$\text{such that} \quad \sum_{j=1}^{m} x_{i,j} \leq 1 \quad \forall i \in \{1, \dots n\} \quad \texttt{only one job per person} \tag{29}$$

$$X^T \bullet 1 \leq A \quad \texttt{all jobs are at or below capacity} \tag{30}$$

$$\frac{1}{n_c} \sum_{i=1}^{n_e} X^T C X D \geq 0.95 \quad \texttt{at least 95\% of couples are co-located} \tag{31}$$

Co-location Function C returns 1 if the couple is considered co-located, 0 if not or if single. Here we choose 50 miles between job locations to be consider co-located because that is the threshold for receiving dislocation allowance (DLA) for a permanent change of station (PCS) according to the Joint Travel Regulations (JTR). The location function $L(S_i)$ returns the lat/long location of the stationing for Seeker i.

$$C(e_1, e_2) = \begin{cases} 0 & \text{if } e_2 == 0 \\ \mathbb{1}\big(||L(S_i) - L(S_j)|| \leq 50\big) & \text{otherwise} \end{cases}$$

The inspiration and initial formulation of this optimization was done by a young Air Force officer who has since moved onto the private sector.

The extension of the formulation to include the Co-Location Function C and have more than one positions for each job a_i are the contributions of this team.

2.3.1 Explanation of Co-location constraint math

$$\frac{1}{n_c}\sum\sum X^T C X \geq 0.95$$

This works because

$$(CX)_{ij} = \begin{cases} 1 & \text{Seeker } i\text{'s mate is assigned to job } j \\ 0 & \text{otherwise} \end{cases}$$

$$(X^T)_{ij} = \begin{cases} 1 & \text{Seeker } i \text{ is assigned to job } j \\ 0 & \text{otherwise} \end{cases}$$

$$(X^T C X)_{ij} = \begin{cases} n & \text{Number of couples assigned to the } (i,j) \text{ job pairing} \\ 0 & \text{otherwise} \end{cases}$$

$$D_{ij} = \begin{cases} 1 & \text{The job pairing } (i,j) \text{ is considered co-location} \\ 0 & \text{otherwise} \end{cases}$$

$$(X^T C X)\dot{D} = \text{ The Hadamard (element-wise) multiplication of } (X^T C X) \text{ and } D$$

$$\left((X^T C X)\dot{D}\right)_{ij} = \begin{cases} n & \text{Number of co-located couples assigned to the } (i,j) \text{ job pairing} \\ 0 & \text{otherwise} \end{cases}$$

$$\sum\sum (X^T C X)_{ij} = \text{Number of couples co-located}$$

$$\frac{1}{n_c}\sum\sum (X^T C X)_{ij} = \text{Ratio of couples co-located}$$

3 Metrics with Preference Data

3.1 Competitiveness

3.1.1 Average Ranking

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

$$\texttt{Competitiveness}_j = \frac{1}{n}\sum_{i=1}^{m} P^S_{i,j}$$

3.1.2 Adapted Sciorintino Ratio

The issue with the *Average Ranking* approach is that, though conveying the desirability of a position, loses much of the information contained in the distribution of preferences. Consider two jobs out of a pool of ten jobs with five applicants.

$$O_1 = [1,1,1,1,5] \tag{32}$$
$$C_1 = 1.8 \tag{33}$$
$$O_2 = [2,2,2,2,1] \tag{34}$$
$$C_2 = 1.8 \tag{35}$$

We see that though job 1 is more competitive, job 2 has the same score, thus considered metrically, equally competitive.

An average of preference rankings, . For example, a position ranked first by ten individuals and hundredth by a twenty individuals has the same average ranking as a job ranked seventh by all thirty individuals; yet the prior is much more competitive.

The Sciorintino Ratio is a metric from the finance industry used to measure the performance of a investment vehicle based on the distribution of their returns. An investment vehicle with consistent, small positive returns is much different than one with consistent negative returns and one big win, even though the average return may be the same. We see the same problem in finance as the detailing marketplace, conveying the nature of a distribution in a single metric. The difference in this ratio, as compared to the more popular Sharpe Ratio, is that the Sciorintino ratio does not let positive volatility negatively effect the score, only volatility on the negative side is punishing. Here we want to adapt the the concept that the volatility of preference is reflected only with higher ranked preferences. For example, if a job has an average ranking of seven, the fact that many people ranked it first is much more important for competitiveness than the fact that many other people ranked it hundredth.

The formulation for our *Adapted Sciorintino Ratio* for the competitiveness of job j takes the form

$$\mu_j = \frac{1}{n}\sum_{i=1}^{m} P_{i,j}^S \tag{36}$$

$$\sigma_j = \frac{1}{m^2}\sum_{i=1}^{m} \mathbb{1}\big(P_{i,j}^S < \mu\big)\big(P_{i,j}^S - \mu\big)^2 \tag{37}$$

$$\texttt{Competitiveness}_j = \frac{\mu_j}{\sigma_j} \tag{38}$$

3.1.3 Weighted Scaling

The issue with the *Adapted Sciorintino Ratio* approach is that it has trouble adjusting for consistent ranking around the mean. Consider two jobs out of a pool of ten jobs with five applicants.

$$O_3 = [3, 3, 3, 2, 1] \tag{39}$$

$$C_3 = 56 \tag{40}$$

$$O_4 = [4, 3, 4, 2, 1] \tag{41}$$

$$C_4 = 24 \tag{42}$$

We see that though job 1 is more competitive, job 2 has a lower score, thus considered metrically more competitive.

So now we turn to a different ranking that scales based on the number of people, the number of jobs available, and the number of positions available in each job. This job scales from 1 being most competitive, and 0 being not competitive at all.

$$\texttt{Competitiveness}_j = 1 - \frac{1}{mn\sqrt{a_j}}\sum_{i=1}^{n}\sqrt{P_{i,j}^S}$$

Applying this to the previous example (assuming each has only one position available) we see the scores are in the proper order (job 2 is less competitive than job 1).

$$O_3 = [3, 3, 3, 2, 1] \tag{43}$$

$$C_3 = 0.912 \tag{44}$$

$$O_4 = [4, 3, 4, 2, 1] \tag{45}$$

$$C_4 = 0.906 \tag{46}$$

Yet we see that this is just the mean ranking squared scaled by the number of jobs available and the number of positions for the job. So, this scoring method still suffers from the issue of the average ranking,

heavy tailed distribution of preferences may have the same competitiveness score as a normal distribution, even though the former is more competitive. To attempt to compensate for this, we adjust the average by a power of one half is to lessen the impact of preferences closer to m on the score, essentially weighting favorable preference more in our score consideration.

This can be seen by returning to our first example.

$$O_1 = [1, 1, 1, 1, 5] \tag{47}$$
$$C_1 = 0.892 \tag{48}$$
$$O_2 = [2, 2, 2, 2, 1] \tag{49}$$
$$C_2 = 0.885 \tag{50}$$

3.2 Similarity

To understand the multi-dimensional similarity of two vectors, the natural choice is the squared Euclidean distance between the two vectors.

$$\sum_{k=1}^{m} (P_{i,k}^S - P_{j,k}^S)^2$$

Yet, a job seeker's first preference is much more important to them than their hundredth preference, indicating that the similarity of top preferences of two job seekers is indicative of their overall similarity than the comparison of their much lower ranked preference. Thus we want a weight on the distance according to the average importance of the metric to the two seekers.

$$w_k = \frac{2m - (P_{i,k}^S + P_{j,k}^S)}{2m}$$

Thus we propose a similarity measure of two job seekers using preference data should be a weighted squared Euclidean distance between their two preference vectors. To ensure the maximum value is 1 for complete similarity (identical preferences), we scale the metric by a factor of $(m-1)^2$ (because the maximum difference between two rankings is $m-1$)and subtract it from 1.

Thus the similarity function is

$$\texttt{Similarity}(P_i^S, P_j^S) = 1 - \frac{1}{(m-1)^2} \sum_{k=1}^{m} \frac{2m - (P_{i,k}^S + P_{j,k}^S)}{2m} (P_{i,k}^S - P_{j,k}^S)^2$$

3.3 Generalism

The generalism metric can be though of as a score for the amount by which a job seeker has transferable skills. A job seeker's generalist quotient to be the variance of their ranking by all job owners. If a seeker is a generalist, they will have a low variance compared to specialists whose skills will make them highly desirable to some job owners but not so much to others.

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} P_{j,i}^O \tag{51}$$
$$\texttt{Generalism}_i = \frac{1}{m^2} \left(P_{j,i}^O - \mu\right)^2 \tag{52}$$

3.4 Specialization

This metric was developed by the aforementioned former, junior Air Force officer.

A job seeker is a specialist to the extent that their skill set will serve a specific function well. Thus, a job seeker's specialization can be measured as the best preference ranking they are given by any job owner.

$$\texttt{Specialization}_i = \min\{P^O_{j,i} : \forall j \in \{1, \ldots, m\}\}$$

4 Metrics Beyond Preference Data

4.1 Similarity Using Quality Encodings

Consider the following definitions.

$$\vec{w} = \text{ weight vector, } \in \mathbb{R}^{q\times 1} \tag{53}$$
$$1 = \sum_{k=0}^{q} \vec{w}_k \tag{54}$$
$$Q = \text{ Quality Matrix, } \in \mathbb{R}^{q\times n} \tag{55}$$
$$0 \leq Q \leq 1 \tag{56}$$
$$S = \text{ Similarity Matrix, } \in \mathbb{R}^{n\times n} \tag{57}$$
$$0 \leq S \leq 1 \tag{58}$$
$$q + 1 = \text{number of qualities encoded} \tag{59}$$
$$m = \text{number of persons} \tag{60}$$
$$\tag{61}$$

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

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

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.*

4.1.1 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

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

4.1.2 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.

4.1.3 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.

4.1.4 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.

4.2 Suggested Ordering

If data is managed and stored properly it could be used to train a machine learning algorithm to predict the optimal ordering for a job seeker and for a job owner. These metrics would not be used to set the matching, but rather suggest to the seeker and owner what the best preference ranking would be based on historic tendencies. This could be useful especially if job seeker is asked to rank on the order of a hundred or more jobs, or if the billet owner has a hundred or more applicant seekers. This suggested ordering would provide a more holistic default than either no ranking, an arbitrary ranking such as alphabetically, or a single metric ranking such as PT score.

The previous data from preferences, performance evaluations, and other qualities in the personnel data (if managed accurately and structured properly) could all be taken into account to provide a holistic assessment beneficial to both sides. Essentially the computer could take in all the information and distill it to a single metric in a more consistent and timely manner than the humans who are already overwhelmed by the other demands of the job seeking/hiring process.

References

[1] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69:9–15, January 1962.