Air Force Recruiting Service Job-to-Applicant Matching Optimization

Lance T. Wilhelm
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
lwihelm7@gatech.edu

Abstract

United States Air Force job-to-applicant matching by an optimization algorithm was brought to the enterprise level during the 2020 global pandemic to quickly rematch applicants who had their jobs canceled. Since then, some lower units within the Air Force Recruiting Service (AFRS) have utilized this technique to match jobs and applicants at their respective levels. The most common optimization method was through a Microsoft Excel-based program that matched applicants to jobs recursively. This technique, while simple and potentially quick, relies upon the creativity of the heuristic designer to achieve the greatest number of matches. This report explored multiple recursive heuristics and a deterministic optimization (DO) heuristic to achieve the greatest number of matches while attempting to match applicants with the highest job from their preference list. It is found that no recursive heuristic tested could outperform the DO heuristic in either job-matching performance or top-3 job-matching performance. More work should be done to investigate the performance of optimization algorithms against human-based job-matching optimization, such as the job draft or fair-share distribution. Further studies into the enterprise goaling paradigm should also be conducted as it closely relates to booking strategy.

1. Introduction

The United States military recruiting environment changed dramatically in the fiscal year 2022, prompting many AFRS personnel to rethink how they do business. Depleted applicant queues strain the traditional method of matching jobs to applicants. This method, accepted by most, if not all, the enterprise, is that of a job draft whereby flight leaders select jobs from the squadron's distribution of jobs. The new method of matching jobs considered in this report utilizes various optimization heuristics and automated algorithms to achieve optimal matching performance. This new method is known to many as "the optimizer." The objective function of the optimizer can be tailored to satisfy different objectives depending on the need of the mission.

2. Background

Finding creative solutions to matching problems is a very important and widely studied problem in the field of network optimization [4]. Currently, medical residents within the United States are subject to a matching algorithm that pairs them with hospitals to give both the resident and the hospital the highest possible choice from their rank order list [3]. This algorithm, the basis for a 2012 Nobel Prize in Economic Sciences, relies upon simple recursion to create a transparent and effective matching system [2]. The United States Air Force adopted this type of system in 2018 to match its officers with open jobs during job assignment cycles, which occur every 2-4 years for individual officers [5]. The move to adopt this system was an effort to increase retention through greater transparency in the job-matching process and maximizing both officer and billet owner satisfaction. The same concept of bridging "art and science" [5] through a matching algorithm had existed within AFRS before it took the front stage during the 2020 pandemic. During the pandemic, AFRS utilized the optimizer to navigate job cancellations due to safety precautions enacted at Basic Military Training (BMT) to prevent the spread of COVID-19. At the time of implementation, this enterprise-level optimization was implemented to streamline the job-matching process and quickly match many applicants with jobs.

A 2022 report from the RAND corporation investigated the United States Air Force Enlisted classification and reclassification and found potential benefits for utilizing machine learning techniques in predicting early separation and reenlistment [8]. Of the multiple predictors used to predict success at four stages of an Airman's career, they found that job preference (i.e. the rank of the job that the member was matched with) was of medium to high importance in predicting success in Initial Skills Training (IST) and early separation, and little importance in predicting reenlistment and completion of first-term with subsequent promotion. They further found that while machine learning may not yield

much additional benefit in increasing IST graduation rates, it may be effective in predicting early separation and reenlistment. Expanding the number of predictors used from 21 to 47, which would require process improvements to record and retain more data, decreases prediction errors by 5%. Finally, they theorize that through an optimized job assignment (matching) algorithm, the United States Air Force could achieve 3-5% greater success with Airmen completing their first term, reenlisting, and promoting to the grade of E-5.

Matching applicants to jobs by the optimizer is not a new concept within AFRS, but it has not been the primary matching method for most squadrons and groups. In the traditional matching process, known as the "job draft," flight leaders or their representatives select jobs to match applicants in a turn-based structure. Each flight chief chooses one job after another until all jobs on the distribution have been accounted for. At that point, each flight is responsible for matching the jobs they drafted with the applicants in their Qualified and Waiting (QW) list. They can arrange swaps with other flights if a flight cannot match a job with an applicant. The objective of each flight chief in the job draft is to achieve the maximum number of job bookings for their flight, not necessarily to match an applicant with the highest job from their preference list. This report investigated whether the optimizer can achieve the best of both objectives by achieving near-optimal job match rates while simultaneously matching applicants with jobs near the top of their preference list.

The current optimizer employed by most organizations is based on Microsoft Excel and utilizes custom macros written in Visual Basic for Applications (VBA). This platform is convenient for running on government systems as it is available on every government computer but is limited in computational power and flexibility compared to modern programming languages available for free (i.e. Python and R). Furthermore, each job-matching algorithm performs its matches recursively. This allows for flexibility in heuristic design, ease of understanding, and portability to Microsoft Excel if necessary. Further models using DO have been explored by the AFRS Analysis Branch [7] and were reinvigorated in this report. Such models can guarantee optimal matching, whereas recursive models may never be able to unless cleverly designed.

3. Methodology

This study analyzed applicant and job distribution data as well as developed and ran matching optimization using Jupyter notebook written in Python and R. It is interested in improving the job-to-applicant matching performance of the current systems to show the potential upside of using an automated system. Currently, Python cannot be installed on bare metal on government systems. Therefore a secure Air Force instance of the clustered computing platform Databricks was utilized to host the data and run a Python notebook containing the code used to perform job-to-applicant matching and analysis. Various heuristics were written for the matching algorithm within the Python notebook, including recursive and DO models. Multiple trials were run using different date ranges and job/applicant listings to compare the performance of each method.

4. Data

This analysis critically requires two independent datasets. Combining these two datasets allows the algorithm to match applicants with available jobs based on dates and preferences optimally.

1. **Applicant List**: This list includes information such as when the applicant is available to leave for BMT and their rank order job preference list. This list also includes many other pieces of data that could be useful in additional analysis or quality control.

2. **Job Distribution List**: This list contains every job available for new applicants. The two pieces of critical information in this data set are the Air Force Specialty Code (AFSC) and the EAD date or "ship date," which is when the applicant will ship to BMT.

For this report, 4 different applicant lists and job distribution lists were considered, which had jobs with entry dates spanning from May 1, 2022, to December 30, 2022. The applicant lists correspond to the current list of available applicants in the QW when those job lists were acquired. In total, there were 2431 total applicants matched against 3924 jobs.

5. Implementation

For this analysis, multiple different heuristics were investigated for matching jobs:

A) Prioritize jobs with a larger interest count; applicants with later DEP entrance dates

B) Prioritize jobs based on a greater difference between interest and availability; applicants with later DEP entrance dates

C) Prioritize jobs with larger interest counts; applicants with fewer jobs listed on their preference list

D) Prioritize jobs based on a greater difference between interest and availability; applicants with fewer jobs listed on their preference list

E) DO with custom heuristic weights (equation 1)

$$
\begin{aligned}
max \quad & \frac{w_1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} + \frac{w_2}{365m}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}d_i \\
& + \frac{w_3}{20m}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}p_{ij} \\
s.t. \quad & \sum_{i=1}^{m} x_{ij} \leq 1 \quad \forall j \\
& \sum_{j=1}^{n} x_{ij} \leq 1 \quad \forall i \\
& x_{ij} \leq e_{ij} \quad \forall i, \forall j \\
& x_{ij} \in \{0,1\} \quad \forall i, \forall j \\
& e_{ij} \in \{0,1\} \quad \forall i, \forall j \\
& 0 \leq w_i \leq 1 \quad \forall i \\
& \sum_{i}^{3} w_i = 1
\end{aligned} \tag{1}
$$

In the mixed integer linear program 1, the objective function maximizes the weighted combination of the total number of matches, days in the DEP, and preference score. The matrix x represents the match matrix where $x_{ij} = 1$ represents a match of applicant i with job j. The vector d_i is the number of days in the DEP for applicant i, p_{ij} is the preference matrix that holds the preference scores for each applicant. Each component of the objective function is normalized before combining to allow the weights to have an effect.

The first constraint ensures that only 1 applicant can be matched to a job, and the second constraint ensures that an applicant can only be matched to one job. The third constraint ensures that an applicant can only be matched to a job they are eligible for where e_{ij} represents the eligibility matrix and $e_{ij} = 1$ represents that applicant i is eligible to match with job j. The last constraint about w_i ensures that each weight is between 0 and 1, and the sum of the weights must total 1.

5.1. Python Notebook

The Python Notebook starts by setting some important variables and parameters used later in the algorithm. The file path locations of the two necessary data sets and the start and end date control what jobs are matched. Next, the notebook imports the two datasets into Pandas dataframes, cleaned up to fix dates, remove null entries, group jobs based on extended active duty (EAD) date, and AFSC, and establish new blank columns with resulting data later.

The notebook has several functions to allow for matching and other analyses against different datasets. One important function, `generate_job_deltas`, takes both the list of available applicants and jobs for the specified date range and returns a table that displays the difference between the interest in each job and the number of jobs available. This table serves multiple purposes and helps the recursive heuristic algorithms function, but its most simple and powerful purpose is in sales assistance. This table allows the recruiter to see which jobs are available and how many are available. This can then be relayed to the applicant to find the best match given their circumstances.

However, the key function to the DO heuristic portion of the matching algorithm is `match_MIP_weighted`. This function takes the list of available applicants, jobs, and weights as arguments and returns the list of applicants with jobs they have been matched with. First, the function builds and populates a matrix that is (num_apps $\times$ num_jobs) in shape as dictated by the data passed to the function. The algorithm utilizes the Python package `cvxpy` to express the optimization problem and then utilizes the additional package `cvxopt` to install the solver that would allow it to be solved as a mixed integer linear program. This is essential as the availability of both the jobs and applicants are represented as either a 1 or a 0 and return either a 1 if an applicant is matched to a particular job or a 0 if not. After expressing the problem and constraints detailed in equation 1, The `is_dcp` function within `cvxpy` is used to ensure that the problem is Disciplined Convex Programming (DCP). This function verifies that the problem follows DCP rules, indicating that it is convex and thus can be solved [1]. `cvxpy` is told to solve the problem, and the result is stored in the variable `match_values`. Finally, the function parses the resulting list `match_values` and stores the results of which jobs were matched to which applicants in the `resulting_qw` and `resulting_jobs` dataframes which are passed back to the caller of the function.

The recursive algorithms which drive heuristics A, B, C, and D all are written in similar functions. However, each of these is solved using nested for loops instead of a convex optimization problem solver. The only difference between the 4 algorithms is how the jobs and applicants are sorted. This dictates which jobs and applicants are considered first during matching. The first loop considers each job in its respective order by filtering out the applicant list to only those who have expressed interest in that job via their preference list. Then the algorithm determines if the applicant is eligible by comparing the EAD value for the job with the EAD From value for the applicant. If the applicant has expressed that they would like to enter the Air Force at an earlier date than when the job indicates the applicant will enter, the applicant is eligible. Based on sorting the interested applicants, the first applicant that meets the EAD eligibility criteria and has not been matched to an earlier job is matched with the job in question. The algorithm then

proceeds to the next job on the list and repeats the applicant matching algorithm. The algorithm is complete when all jobs are considered from the available list. The function returns `resulting_qw` and `resulting_jobs` in the same fashion as the DO heuristic so that the same analysis functions can be used.

5.2. Trials

To increase the number of trials run, the 4 lists of jobs were broken down into different date ranges to provide results from having more or fewer jobs available to match against. This resulted in 12 unique trials where all 5 heuristics were run, including 2 additional trials of the DO heuristic to determine the optimal weights, totaling 62 trials. The initial trials to determine an optimal weighting scheme included the following weight schemes for (number of matches, time in DEP, job preference rank):

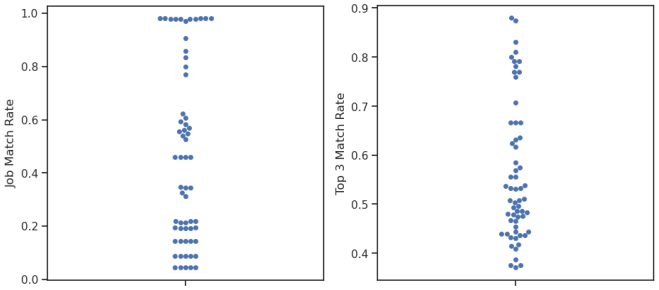
- (1.0, 0, 0)
- (0, 0, 1.0)
- (0, 0.5, 0.5)

After background research, this study intended to maximize the number of matches while also maximizing the rank of the job matched to an applicant to increase training success and job satisfaction and prevent early separation [8]. The initial trials showed that a weighting scheme of (0, 0.5, 0.5) yielded the same number of matches as the (1.0, 0, 0) weighting scheme yet increased the top-3 match rate (i.e. the percentage of applicants matched with one of their top-3 job choices) from 47.5% to 70.71%. A weighting scheme of (0, 0, 1) focusing only on matching applicants with a job as high on their preference list as possible yielded slightly lower overall matches but improved the top-3 match rate to 72.76%. Considering that the recruiting enterprise is in a time of consistently less qualified and available applicants than jobs, the weighting scheme ensures the highest number of matches possible while also considering job rank-order preference was chosen.

An interesting observation is that the problem can maximize the number of matches when there is 0 weight placed on the number of matches. This is likely because the objective function parts regarding days in DEP and preference contain the variable x_{ij}, which is the total number of matches. Thus, even though x_{ij} is being discounted by various factors within those parts of the objective function, a higher score will still come from having more matches altogether because x_{ij} is greater.

6. Results and Analysis

The results from each trial were stored in an excel sheet containing the following fields: Heuristic, QW, Job_Distro, Start Date, End Date, Date Range Length, Total QW, QW Available, Jobs, Jobs/QW Ratio, Matches, Time Elapsed, Job Match Rate, Applicant Match Rate, Top 3 Match Rate, and Weights. These fields ensured a run could be duplicated by recording which dataset was used with what heuristic and what date range. All categorical and numerical fields were considered in the following analysis.



(a) Match rate for each trial (b) Top-3 job match rate for each trial

Figure 1: Initial results from all 60 trials

The study's objective is to find which algorithm performs best in matching performance (i.e. Job Match Rate) and matching applicants with jobs high on their preference list (i.e. Top 3 Match Rate). Figure 1 shows the overall match rate and the top-3 job match rate results for all trials.

The overall performance across all trials is not consistent, which is expected given that the trials are run across multiple periods with 4 different datasets and 5 different matching heuristics. More insight should be gleaned from the result attributes and heuristic performance.

6.1. Result Attribute Analysis

The results of the different statistics should be compared to observe any correlations that may be useful to understanding the performance of each trial. This may be parlayed into regression analysis if reasonable. Figure 2 shows a heatmap correlation matrix of the initial trial results.

There is a clear positive correlation between jobs and date range length, which follows the logic: as the date range increases, more jobs become available. Next, there is a positive relationship between the total number of applicants on the QW list and who is available. More significantly, there is a positive correlation between the job match rate and the number of applicants on the QW and available QW. This indicates that the more applicants are in our QW and available, the better our match rate will be. Figure 3 shows this correlation and plots a linear regression. However, the best-fit line should not be utilized to make useful projections other than rough job match rate performance estimates. Ultimately the recruiting goal is to have a job match rate of 100%, so our positive correlation indicates that to increase

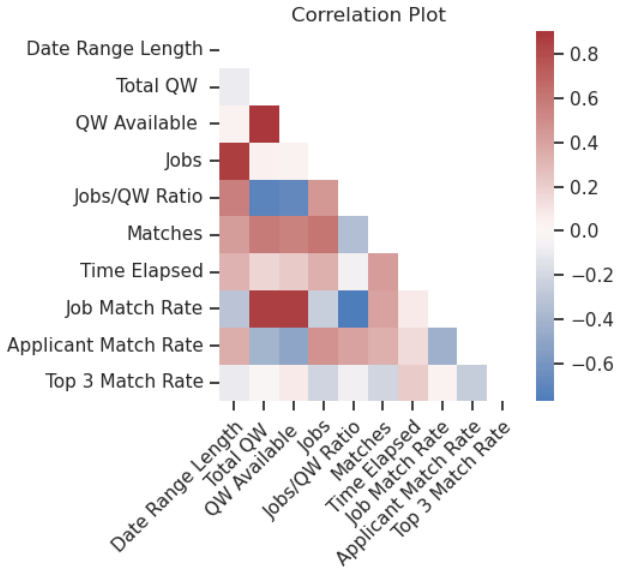


Figure 2: Correlations between numerical variables in the results table

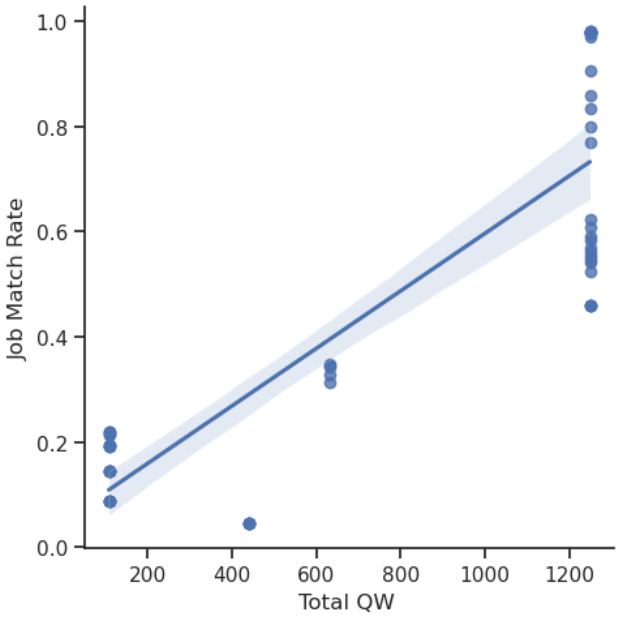


Figure 3: Plot of total QW count versus job match rate

the number of matches, the number of applicants on the QW and available should increase. This has long been known within the recruiting enterprise and has been the focus of remedial recruiting efforts in the past year.

Lastly, there appears to be a negative correlation between the job match rate and the jobs to QW ratio. This means that the job match rate decreases as there are more jobs than applicants on the QW. This is a logical relationship as only as many jobs as there are applicants can be filled. A more interesting data subsection would be to observe the matching performance when the job-to-QW ratio is less than or equal to 1.0. This means there are as many or more applicants available as there are jobs. Unfortunately, there are only 20 results when the job to QW ratio was ≤ 1.0, which is likely because of the fact that there are only 4 unique datasets. Also, many of those 20 runs come from the same datasets. This matters because the QW and job distributions will be mostly the same for those periods.

After correlation analysis and consideration of the result attributes, it does not make sense to consider a linear regression for the numerical results given above. Only a few independent predictors could be considered for predicting matches: number of jobs, total QW, available QW, and possibly date range length. All other numerical results in the dataframe are derived from these variables in some fashion and, therefore, should not be considered in a regression analysis as they depend on one another. This logic follows from the above correlation analysis as there may appear to be correlations between variables that depend on one another for calculation.

6.2. Heuristic Analysis

No recursive heuristic outperformed the DO heuristic in job-matching performance. A new variable labeled `Matches Difference to E` was made to describe the difference between each heuristic and the respective DO heuristic for their trial. This allows a finer analysis of the difference between each heuristic. Figure 4 gives box plots representing the differences for each trial between the different heuristics and heuristic E. As there is no data greater than 0, there is no heuristic that outperforms E.

However, there are trials where their performance matches, albeit only for trials when the total number of matches is low. Figure 5 shows a clear relationship between the number of matches and the spread of performance in the heuristics. As the number of matches increases, the performance of the recursive heuristics decreases, and the spread in performance increases, as indicated by 6.

Run Time The recursive heuristic runs have lower mean values and consistent standard deviations, and the DO heuristic is much larger. This likely has a lot to do with solving a convex optimization problem. Table 1 shows the differences between the run time for each heuristic. Figures 7 and 8 show the relationship between run time and the independent variables that drive run time.

As the number of matches increases, the time it takes for the recursive algorithms to complete is linearly proportional to (QW Available x Jobs). The time it takes for the DO problem to complete grows exponentially with (QW Avail-

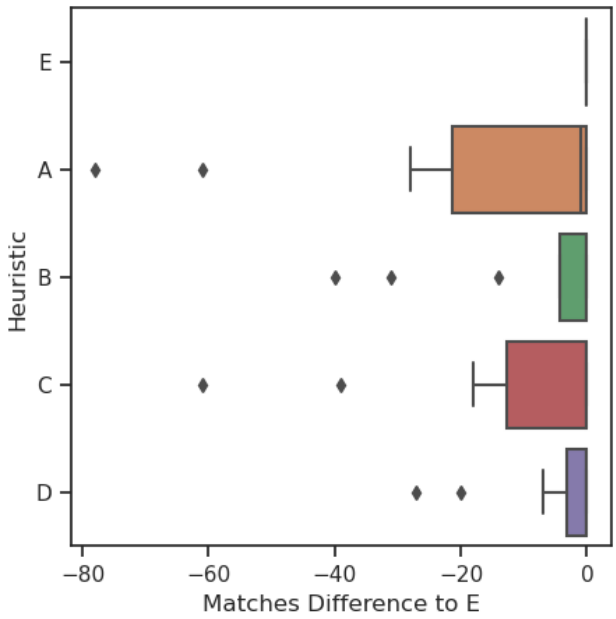


Figure 4: Differences in matches for each heuristic compared to heuristic E

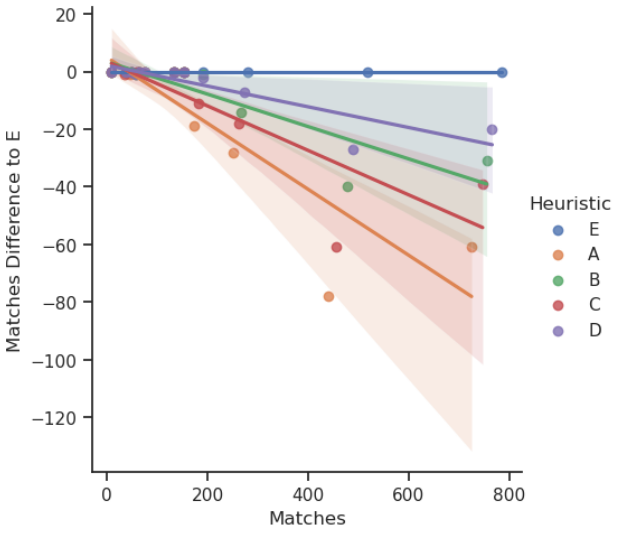


Figure 5: Matches versus matches difference to E

able + Jobs). This did not pose a particular problem for the trials, as the longest run was 839.34 seconds or about 14 minutes. The run that resulted in that large time elapse had 1039 on the available QW list and 1380 jobs to match. Any run using this technique that utilizes job and applicant totals larger than this must consider the time it will take.

Top-3 Job Match Rate Table 2 clearly shows that the DO program is the best performer in giving applicants a

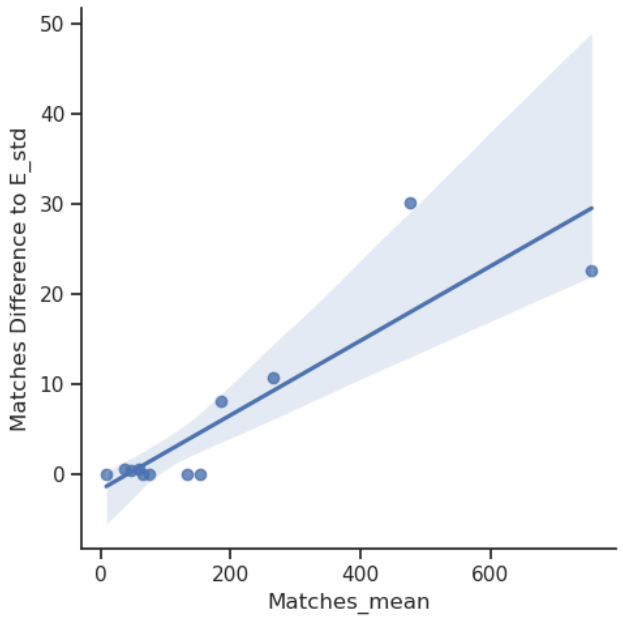


Figure 6: Matches versus standard deviation of matches difference to E

Heuristic	mean	max	min	std
A	8.515000	30.89	1.50	7.946796
B	8.178333	32.10	1.62	8.196935
C	8.249167	31.72	2.09	8.063265
D	8.256667	31.98	1.86	8.201959
E	97.330000	839.34	0.49	235.422577

Table 1: Run time statistics by heuristic ($n = 12$)

job from high on their preference list. The minimum from its results is higher than the mean of any of the recursive heuristics. It also has one of the lowest standard deviations, meaning that it is consistent in its performance of matching at a top-3 job. The next best performer could be considered a toss-up between heuristics C and D. Logically, heuristics C and D perform better in matching a top-3 job because both techniques match applicants who listed fewer jobs on their preference lists. If an applicant only lists 3 jobs on their list, then there is a 100% chance they will match to a top-3 job if they do match. This applicant would be favored over an applicant who lists 10 jobs and may be more flexible.

Overall If the results from the top-3 job match rate performance and the overall match rate performance are combined, as shown in table 3, heuristic D is the best recursive technique, having, on average, 6 more matches per run. This increase in performance stems from the fact that it prioritizes jobs with more available jobs that applicants are in-

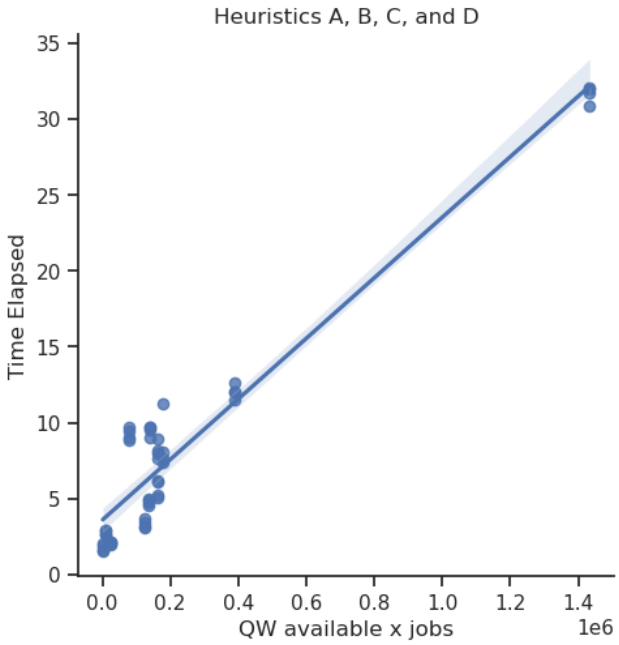


Figure 7: Run time relationship for the recursive heuristics

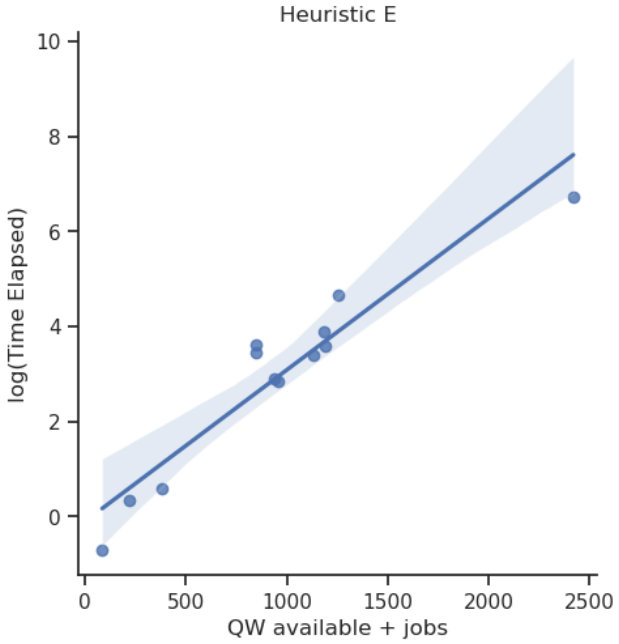


Figure 8: Run time relationship for the DO heuristic

terested in. It also stems from its applicant sorting method, which matches less flexible applicants first (i.e. less listed jobs), leaving those applicants who are more flexible (i.e. more jobs listed) to the end. This heuristic exemplifies how clever algorithms can achieve near-optimal performance but require ingenuity.

	mean	max	min	std
Heuristic				
A	0.4927	0.5846	0.4310	0.0520
B	0.4665	0.6667	0.3711	0.0832
C	0.5453	0.7692	0.4444	0.0908
D	0.5270	0.8000	0.3750	0.1294
E	0.7699	0.8800	0.5745	0.0866

Table 2: Top-3 job match statistics by heuristic ($n = 12$)

	mean	max	min	std
Heuristic				
A	180.500000	725	9	208.680051
B	189.000000	755	9	220.435932
C	185.416667	747	9	215.999772
D	191.500000	766	9	224.780903
E	196.250000	786	9	232.900730

Table 3: Match statistics by heuristic ($n = 12$)

Altogether, it is clear to see the performance increase by utilizing the DO approach. This algorithm allows flexibility in all aspects currently available in the recursive techniques while ensuring an optimal matching result. The only downside, which can be circumstantial, is the time cost of the approach. For shorter periods with fewer jobs and applicants available, the time cost is on the same order as the recursive techniques. However, the user must consider the time cost once the number of applicants and jobs consider eclipses 1000.

The matching rate results of these trial runs cannot be used to justify one recursive heuristic over another. The implications of using one technique over another must be considered by observing the distributions of matched and unmatched jobs and applicants. For example, one heuristic may match more applicants overall, but it leaves more hard-to-fill jobs as a result. Similarly, a heuristic may match more jobs but often gives applicants jobs that are lower on their preference list.

7. Future Work

A greater comparison must occur between the optimizer and the widely accepted and utilized job draft matching strategy. This would require an effective simulation of the job draft technique or results from real job drafts combined with the QW and job distributions at the draft time. Even then, it may not be enough to replicate the job draft accurately. More matching actions occur after the draft when flight leaders swap jobs and new applicants become available. The job draft would be hard to simulate as other factors, such as flight chief skill, must also be accounted for. More skilled or experienced flight leaders may be able to

secure jobs that are better for their flight.

Instead, a better simulation solution might be a fair-share distribution of jobs to the flights based on their quota share. This would attempt to give every flight the same number of jobs within each major assignment category. This distribution method does not perfectly mimic the job draft, but it could effectively simulate the distribution of jobs to the flight level.

Job booking strategies are closely interwoven into the goaling and incentive structure of recruiting. Recruiting is a sales enterprise that utilizes quotas and awards to incentivize performance. flight leaders often adopt a job matching strategy such as the job draft because it allows them the most control and visibility of available jobs. Understandably, having an algorithm book jobs at the low, mid, or enterprise level removes some of the recruiter's control over meeting their expectation. AFRS recently adjusted its goaling calculations to increase fairness, but goals remain at the recruiter level. Larger studies into the overarching goaling paradigm, such as RAND's from 2022 [6], should be conducted considering improvements in applicant-to-job booking strategies to increase job matching performance targeting improved talent management described in Robson et al. [8].

8. Conclusion

A DO algorithm for matching available United States Air Force jobs to applicants proves to be the most effective solution in achieving the most matches while allowing more applicants to match with jobs higher on their preference list. Recursive techniques can achieve similar results in asset-constrained circumstances and be faster to finish when input data is larger. However, their performance in achieving the greatest number of overall matches considering applicant preferences should never beat that of the DO heuristic. More research or studies should be considered to compare the performance of job-to-applicant matching optimization with human-based techniques such as the job draft.

References

[1] Disciplined convex programming. https://www.cvxpy.org/tutorial/dcp/index.html. 3

[2] How it works. https://www.nrmp.org/intro-to-the-match/how-matching-algorithm-works/. 1

[3] National resident matching program. https://www.nrmp.org/, Nov 2022. 1

[4] A Agnetis. Introduction to matching problems. https://www3.diism.unisi.it/~agnetis/matchingENGnew.pdf. 1

[5] Kat Bailey. Afpc adopting innovative officer assignment system it platform, Apr 2019. 1

[6] Tiffany Berglund, David Schulker, Tracy C. Krueger, and Nelson Lim. *Improving the Goal-Setting Process for U.S. Department of the Air Force Recruiters*. RAND Corporation, Santa Monica, CA, 2021. 8

[7] Rebecca Butler. Job distribution optimizer, May 2020. 2

[8] Sean Robson, Maria C. Lytell, Matthew Walsh, Kimberly Curry Hall, Kirsten M. Keller, Vikram Kilambi, Joshua Snoke, Jonathan W. Welburn, Patrick S. Roberts, Owen Hall, and Louis T. Mariano. *U.S. Air Force Enlisted Classification and Reclassification: Potential Improvements Using Machine Learning and Optimization Models*. RAND Corporation, Santa Monica, CA, 2022. 1, 4, 8