PrivateJobMatch: A Privacy-Oriented Deferred Multi-Match Recommender System for Stable Employment

Amar Saini
University of California Merced
asaini4@ucmerced.edu

Florin Rusu
University of California Merced
frusu@ucmerced.edu

Andrew Johnston
University of California Merced
acjohnston@ucmerced.edu

ABSTRACT

Coordination failure reduces match quality among employers and candidates in the job market, resulting in a large number of unfilled positions and/or unstable, short-term employment. Centralized job search engines provide a platform that connects directly employers with job-seekers. However, they require users to disclose a significant amount of personal data, i.e., build a user profile, in order to provide meaningful recommendations. In this paper, we present PrivateJobMatch – a privacy-oriented deferred multi-match recommender system – which generates stable pairings while requiring users to provide only a partial ranking of their preferences. PrivateJobMatch explores a series of adaptations of the game-theoretic Gale-Shapley deferred acceptance algorithm which combine the flexibility of decentralized markets with the intelligence of centralized matching. We identify the shortcomings of the original algorithm when applied to a job market and propose novel solutions that rely on machine learning techniques. Experimental results on real and synthetic data confirm the benefits of the proposed algorithms across several quality measures. Over the past year, we have implemented a PrivateJobMatch prototype and deployed it in an active job market economy. Using the gathered real-user preference data, we find that the match recommendations are superior to a typical decentralized job market—while requiring only a partial ranking of the user preferences.

CCS CONCEPTS

• **Theory of computation → Design and analysis of algorithms**; • **Applied computing → Economics**; • **Security and privacy → Social aspects of security and privacy**.

KEYWORDS

job matching, reciprocal matching, Gale-Shapley algorithm, DAA algorithm, MMDAA algorithm, low-rank matrix factorization

ACM Reference Format:
Amar Saini, Florin Rusu, and Andrew Johnston. 2019. PrivateJobMatch: A Privacy-Oriented Deferred Multi-Match Recommender System for Stable Employment. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3298689.3346983

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3346983

1 INTRODUCTION

Job recommendation is a fundamentally different problem from traditional recommender systems for books, products, or movies [4, 22]. A key difference is that a job posting is meant to hire only one or a few candidates, whereas the same book, product, or movie can be hypothetically recommended to an infinite number of users [28, 29]. Another difference is represented by the physical constraints of the job interview process. Both candidates and employers can dedicate only a limited amount of time to interviews, thus, recommending the most likely candidates/employers is paramount to successful hiring. Given a list of candidates, employers can err either by pursuing candidates who have better offers – which reduces options and risks a failed search – or neglect preferred, attainable candidates in favor of others that seem more likely to accept an offer. Put simply, employers justifiably struggle to identify the best, attainable candidate. On the candidates' side, if too many job-seekers compete for the same job, each one's chances of getting the job are dramatically reduced. Job search failure is costly for employers as well as for job-seekers because many open positions remain unfilled or result in unstable, short-term employment, while job-seekers remain unemployed for longer periods of time. While centralized platforms such as LinkedIn [11] and XING [3] have the potential to address these issues, they use a business-oriented best-fit optimization strategy that favors higher-paying employers and candidates over the well-being of the job market as a whole. However, addressing this coordination failure in a neutral way, may significantly help both job-seekers and employers identify well-suited matches, betting the job market outcome. Centralized matching markets – like those for medical residency in the U.S. [26] and Canada [23] – resolve this coordination failure by eliciting preference rankings from both sides of the market and playing out those preferences to find stable matches between candidates and employers—using what is now known as the Gale-Shapley deferred acceptance algorithm (DAA) [9, 19]. The resulting stable matches perform better than decentralized recommendations [7, 15, 20]. They improve the welfare of both sides of the market—being weakly Pareto optimal [1, 2]. Unfortunately, these benefits are not widely realized in the labor market because they require an ex ante commitment to the outcome of the centralized match. *Our goal is to design a job recommender system that combines the flexibility of decentralized markets and the wise pairing of centralized matching.*

Job recommender systems (Figure 1) operate by matching candidate profiles to job descriptions [4]. The hypothetical interest of a candidate in a job is predicted through a series of explicit and implicit factors aggregated in the candidate profile [18]. Explicit factors are willingly provided by the candidate and include personal details, education, experience, qualifications, desired job types, etc. The premise is that the more data the system knows

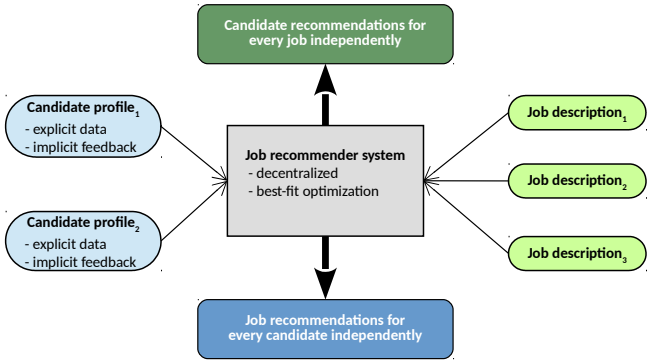

Figure 1: Standard job recommender system.

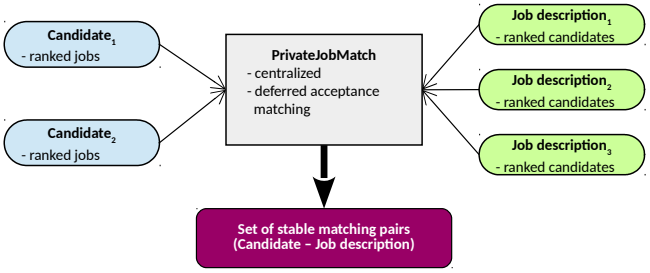

Figure 2: PrivateJobMatch recommender system.

about the candidate, the better the predictions are. However, this raises important privacy concerns and can lead a candidate to not provide the data—or even provide inaccurate or false data. Implicit factors are tracked – typically, without the user consent or even knowledge – by the recommender system from the actions the candidate performs on the platform. Some examples include visited web pages, how long they stayed on a specific page, and bookmarking an item for revisiting at a later time. One of the main drawbacks of recommender systems that require implicit factors is that they inherit the "cold start" problem, where the system has trouble generating meaningful recommendations for new candidates/jobs due to lack of sufficient information. Implicit feedback is even more problematic from a privacy perspective because it can be used to inadvertently target certain candidates—when the interest between a candidate and an employer should be reciprocal [10, 12, 14]. The relative importance of explicit and implicit feedback on the quality of recommendations has been studied empirically in [18]. However, the conclusion that implicit feedback is more important ignores privacy concerns entirely. *Our goal is to design a privacy-oriented job recommender system that uses only a minimum amount of explicitly provided relevant data in order to provide relevant job matchings.*

PrivateJobMatch. In this paper, we introduce PrivateJobMatch (Figure 2) – a privacy-oriented deferred multi-match recommender system – which generates stable pairings (candidate, job description), rather than recommendations that are best-fit for individual candidates/employers. PrivateJobMatch combines the flexibility of decentralized markets with the wise pairing of centralized matching by adapting DAA into a recommender system that generates a ranked list of candidates for employers – and vice-versa – in the order of expected match quality. This is realized by asking candidates/employers to provide only a lightweight partial ranking of their preferences—without the burden of creating extensive and privacy-sensitive profiles. Being driven by DAA, PrivateJobMatch optimizes the stability of the job market as a whole, by ensuring that candidates and employers are matched with each other in stable pairs, i.e., multiple best matches are found for every candidate/employer based on the ranked preferences, and, thus, mitigates the coordination failure between job-seekers and employers.

Contributions. The integration of DAA in PrivateJobMatch poses several technical challenges, chief among them being limited matches for individual candidates/employers and sparse rankings. Therefore, we design three multi-match DAA algorithms – MMDAA, LMF-MMDAA, and Mixed-MMDAA – that produce a ranked list of matches—rather than a single match. MMDAA is a direct extension of DAA to ranked recommendation lists which also addresses unequal number of candidates and jobs. LMF-MMDAA employs low-rank matrix factorization (LMF) collaborative filtering in order to expand the sparse rankings provided by certain candidates/employers. It enriches sparse rankings with recommendations extracted from other similar overlapping rankings—not from candidate profiles or job descriptions. Mixed-MMDAA is our most advanced algorithm that combines the output of MMDAA and LMF-MMDAA to further improve each one's recommendations. We perform an extensive set of experiments over real and synthetic data in which we measure the performance of the proposed MMDAAs based on three metrics—displacement, withholdings, and vacancy. The results show the overall good performance of all three algorithms and confirm the superiority of Mixed-MMDAA over the other two. Over the past year, we have implemented a PrivateJobMatch prototype and deployed it in the academic positions job market [24]. Using the gathered real-user preferences, we find that the PrivateJobMatch recommendations are superior to a typical decentralized job market, confirming the viability of our approach.

Outline. In the remaining of the paper, we first put our work in perspective to other literature on the subject (Section 2). Then, we present our main technical contribution – the multi-match DAA algorithms – in Section 3. The experimental evaluation of the algorithms is included in Section 4, while a job market simulation is in Section 5. The conclusions and plans for the future are in Section 6.

2 RELATED WORK

In this section, we survey relevant work on job recommender systems and deferred acceptance algorithms—the topics of this paper.

Job recommender systems. There has been an increased interest in systems for job recommendation over the past years, as exemplified by the two RecSys Challenges from 2016 [28] and 2017 [29], respectively. In 2016, the challenge was to predict which of the displayed jobs a candidate would interact positively with, based on the profile and previous interactions. The 2017 challenge was the inverse problem—given a new "cold" job posting, identify the appropriate candidates for the job. In both cases, the goal is to optimize the outcome for an individual candidate/job, not for the whole job

market. This is also the case in [17] and [6] where job transitions are used as a feature in machine learning-based recommendations. PrivateJobMatch is different because it takes a global view of the market and generates matches that consider the overall interactions between candidates and jobs. The iHR platform [10] introduces the concepts of reciprocity and availability to job recommendations in order to avoid local minima for a certain candidate/job. The LiJAR system [5] from LinkedIn targets the same goal by controlling the number of candidates a job is displayed to based on a desired application yield. However, none of these systems performs global matching between candidates and jobs based on ranked preferences.

Deferred acceptance algorithms. The original DAA algorithm was introduced by Gale and Shapley [9] in 1962. This algorithm requires that the sizes of the candidate and employer sets are equal, ensuring that everyone finds a pair. The subsequent Roth-Peranson algorithm [20] generalizes Gale-Shapley to one-to-many matchings, unequal candidate/employer sets, and sparse rankings. Our multi-match algorithms are extensions of the original DAA to job recommender systems. Although the Roth-Peranson algorithm was applied to many domains, including medical residency programs [23, 26], student and teacher school assignment [1, 7], sorority selection [15], and supply-chain management [16], we are not aware of its adoption in any modern recommender system. The Roth-Peranson algorithm differs from our multi-match algorithms in terms of exclusivity. According to National Matching Services, Inc., a corporation that delivers matching programs for competitive recruitment through the use of the Roth-Peranson algorithm [25], this algorithm is used to achieve one-to-many matches that can be used as assignments, rather than recommendations. Our multi-match algorithms offer multiple one-to-one matches, i.e., many-to-many matches, in the form of recommendations. The main difference between our MMDAAs and the Roth-Peranson algorithm is that our algorithms do not enforce exclusivity in the matches. In other words, multiple candidates can appear in multiple employers' recommendations, rather than having each candidate only be assigned to a single employer. This property is what allows our MMDAAs to be considered a recommender system, rather than an assignment system. The MMDAAs we propose are different from the secure DAA algorithms [8] since the preference rankings are not encrypted—PrivateJobMatch addresses privacy in the recommender system, not in the DAA.

3 MULTI-MATCH DEFERRED ACCEPTANCE

In this section, we first present the Gale-Shapley DAA algorithm [9, 20] applied to candidate-job matching. Then, we introduce our novel MMDAA algorithms for job recommendation.

3.1 DAA Algorithm

The DAA algorithm finds a stable matching between two sets with the same size. It requires as input a full ranking of the opposite set for each element. DAA was initially introduced for the stable marriage problem [9]: given n men and n women, where each person has ranked all the members of the opposite sex in order of preference, determine the marriages that are stable, i.e., there are no two persons of opposite sex who would rather be with each other than with their assigned partners. The extension to candidate-job matching is immediate. In this case, DAA requires the candidates' ranking/preference of each job/employer and – vice-versa – the employers' ranking of each candidate. These rankings represent a measure of how stable a match between a candidate and an employer is—the optimal matching is returned by DAA.

Algorithm 1: DAA Algorithm

Input : Set of candidates $C = \{c_1, \ldots, c_n\}$
Set of jobs/employers $E = \{e_1, \ldots, e_n\}$
Job ranking for candidate i: $RE_i = \left\{e_i^1, \ldots, e_i^n\right\}$
Candidate ranking for job j: $RC_j = \left\{c_j^1, \ldots, c_j^n\right\}$
Output : Set of matching pairs $M = \left\{\left(c_i, e_j\right)\right\}$, $c_i \in C$, $e_j \in E$
Init: Empty match set $M = \emptyset$

```
while ∃c_i with RE_i ≠ ∅ do
    e_j ← RE_i.head
    if e_j has empty pair then
        Add pair (c_i, e_j) to M
    end
    else if ∃ (c_k, e_j) ∈ M and (RC_j[i] < RC_j[k]) then
        Replace pair (c_k, e_j) with (c_i, e_j) in M
    end
end
return M
```

Formally, the input to DAA consists of a fully-connected directed bipartite graph having as vertexes the two sets of size n. There are two directed edges – one in each direction – between any two opposing vertexes—the total number of edges is $2{\times}n^2$. The rankings can be encoded as weights on edges, e.g., the weight for the first choice is 1, for the second is 2, and so on. The output is an undirected bipartite graph of much smaller size—there are only n edges and each vertex has degree 1. Figure 3 illustrates a brief example for job recommendation that is used throughout the paper. There are 3 candidates and 3 jobs. The input graph is split into two adjacency matrix representations—one for each set. Each matrix – depicted at the top of Figure 3 – encodes the rankings of each candidate and employer/job, respectively—smaller ranks correspond to higher preferences. Out of the 18 edges, DAA selects only 3 that form the most stable – or reciprocal – matches. In our simplified example, both the candidates and the employers get their first choice.

DAA – depicted in Algorithm 1 – is a greedy algorithm that builds matches starting from one side of the bipartite graph, e.g., the candidates. A candidate c_i is matched with jobs e_j in increasing order of its rankings. At a given rank j, two situations are possible. First, job e_j is not part of a match yet. In this case, a match $\left(c_i, e_j\right)$ is created. Second, when job e_j is already part of a match with another candidate c_k, the ranking RC_j of e_j has to be considered. If e_j prefers c_k over c_i, candidate c_i's next rank is considered since the current match $\left(c_k, e_j\right)$ is more stable. Otherwise, it is replaced with the more stable match $\left(c_i, e_j\right)$ and candidate c_k is considered again for subsequent matches. This process repeats until all the candidates find matches. The full ranking for all candidates and jobs guarantee that DAA finds a solution. In the worst case, a quadratic number of edges is considered—for a complexity of $O\left(n^2\right)$.

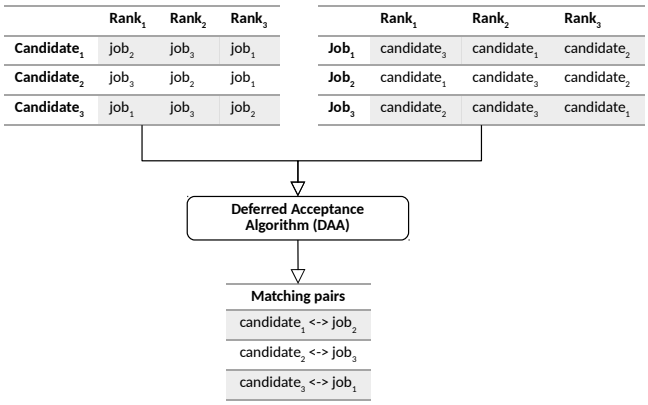

Figure 3: Deferred acceptance (DAA).

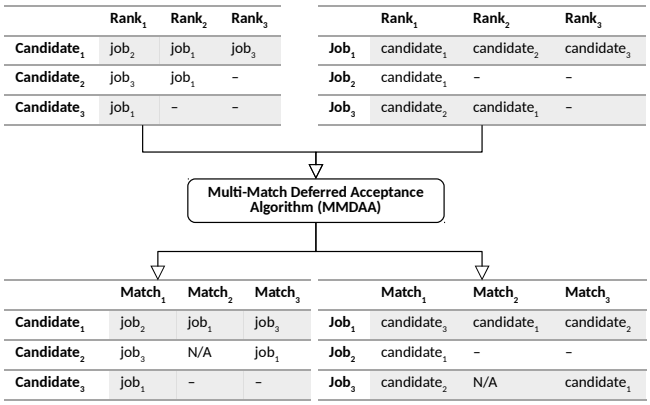

Figure 4: Multi-match deferred acceptance (MMDAA).

The DAA algorithm has two drawbacks that limit its application to recommender systems. First, the number of candidates and jobs has to be identical. This is hardly the case in the real world. Second, complete rankings are required from all the participants. When the size of the two subsets increases, fully-ranking everyone is not scalable anymore. Moreover, the DAA output is too constrained—there is a single match for every candidate/job. In recommender systems, the goal is to provide a series of alternatives rather than a single match. We address all these limitations by proposing a new family of Multi-Match DAA (MMDAA) algorithms. Additionally, unequal sets and convergence analysis are discussed further in the extended report [21].

3.2 MMDAA Algorithm

The MMDAA algorithms accept input rankings that are non-square and sparse, i.e., the bipartite sets have different size and rankings can be partial. These relaxations impact the stability of the DAA algorithm—MMDAA generates partially stable matches. However, the output consists of a ranked set of multiple one-to-one matches or multi-matches—not only a single match. This increases the probability that each participant – candidate and employer – receives at least one match—for this type of input, DAA does not provide any match. We argue that multi-matches are sufficient in a recommender system. Thus, MMDAAs are recommendation – not matching – algorithms.

Figure 4 depicts an example that clearly contrasts MMDAA with DAA. Two differences are clear. The input ranking matrices contain empty cells. The output is two matrices with a series of ranked matches for each candidate/employer. Each column in these matrices is the output of a DAA execution round. Due to incomplete rankings, there are unmatched participants – the N/A cells – in certain rounds. For example, candidate c_2 does not have a match in round 2. The reason for this occurrence is the higher preference of employer e_1 for c_1. However, the match (c_2, e_1) appears in a later round, meaning that such a recommendation – while possible – is less relevant because it is less likely to materialize. In DAA, this match is never possible. While the MMDAA output has to be interpreted from the perspective of each individual candidate/employer, its recommendations consider the overall input state.

MMDAA – depicted in Algorithm 2 – invokes DAA as a subroutine to generate the desired number of recommendations m. The matches generated in each round are the best possible for the given rankings. These matches are appended to the result set and removed from the rankings in subsequent rounds to ensure that different recommendations are considered. This process – invoke DAA then alter the input rankings – is repeated until all the stated preferences are exhausted or the desired number of recommendations is reached—whichever comes first. Since m is finite, the MMDAA algorithm is guaranteed to finish in at most $O\left(m \cdot n^2\right)$ time, which corresponds to m executions of DAA.

Algorithm 2: MMDAA Algorithm

Input : Set of candidates $C = \{c_1, \ldots, c_n\}$
Set of jobs/employers $E = \{e_1, \ldots, e_n\}$
Job ranking for candidate i: $RE_i = \left\{e_i^1, \ldots, e_i^n\right\}$
Candidate ranking for job j: $RC_j = \left\{c_j^1, \ldots, c_j^n\right\}$
Number of recommendations m

Output : Ranked job matches ME_i for each candidate i
Ranked candidate matches MC_j for each job j

```
for k ← 1 to m do
    M ← DAA(C, E, RE, RC)
    Append matches M to ME_i and MC_j
    Remove matches M from RE_i and RC_j
end
return (ME_i, MC_j)
```

3.3 LMF-MMDAA Algorithm

MMDAA transforms the DAA matching algorithm into a recommender algorithm by generating a list of ranked recommendations for every candidate/employer. However, this list contains only explicitly stated preferences. For example, in Figure 4, candidate c_3 is matched only with job e_1 because this is the single ranked job it provides. If, for whatever reason, this match does not materialize, c_3 remains unmatched, even though there are other available jobs. It is very unlikely that this is the desired outcome. More likely, c_3 has not provided rankings for other jobs because of other reasons.

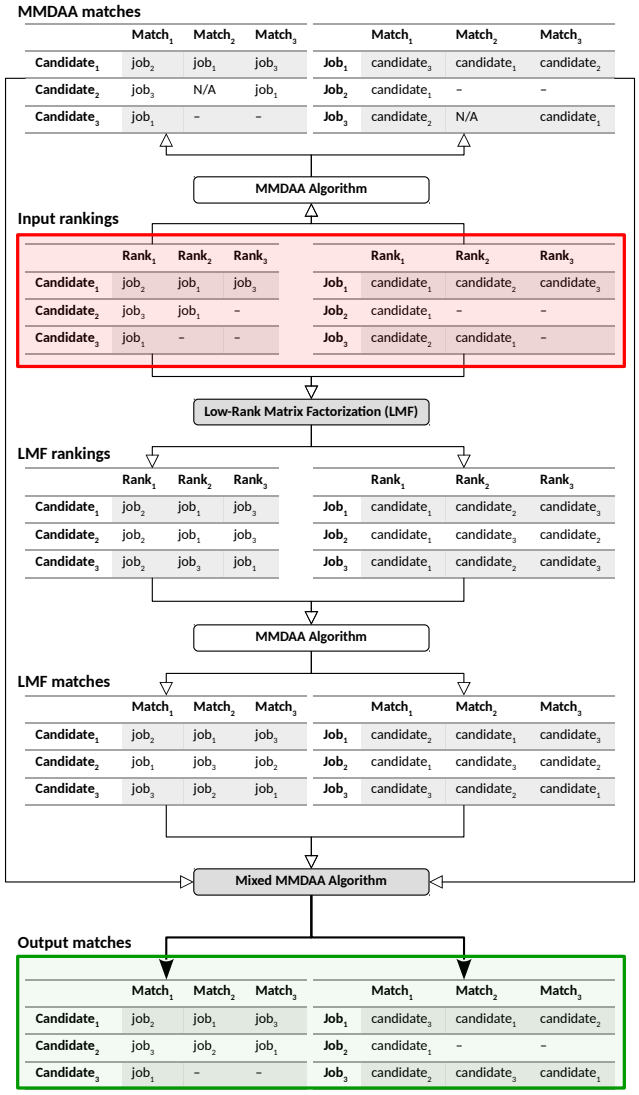

Figure 5: Mixed-MMDAA algorithm.

LMF-MMDAA addresses this limitation by enriching the list of ranked preferences provided by a candidate/employer with other relevant preferences derived through collaborative filtering [4, 22]. In the extreme case, a ranking is inferred for every (candidate, job) and (employer, candidate) pair, respectively. Essentially, the sparse preference matrices are filled to become dense. This is the key feature of LMF-MMDAA. It allows us to satisfy the strict requirements of DAA without having to ask each candidate/employer to provide a rank for every single employer/candidate. We argue that this truly makes the DAA algorithm applicable to real-world recommender systems. While the DAA algorithm can be applied on the dense matrices, the one-to-one matches are too restrictive due to the uncertain nature of the derived rankings. For example, if a candidate is matched exclusively with a job it has not applied for – derived as relevant through collaborative filtering – there is a high risk the match is not stable. This is the reason we perform the recommendation MMDAA algorithm which outputs a list of ranked matches. The information contained in this output is more valuable because it gives an explanation why the desired match is not realized.

A depiction of the LMF-MMDAA algorithm is included in Figure 5 as part of the Mixed-MMDAA algorithm which is presented later. We observe that the sparse input matrices are first filled with missing rankings, before MMDAA is performed to generate the matches. This allows matches to be generated at each round for every candidate. For example, candidate c_2 is unmatched in round 2 of MMDAA – the upper part of Figure 5 – even though it has another stated preference. In LMF-MMDAA, c_2 gets matched with job e_3. Although every candidate/employer in our example is matched in every round – there are no more N/A entries – this is not always the case. In general, LMF-MMDAA reduces the rate of non-matches because it also considers non-stated rankings.

The most challenging part of the LMF-MMDAA algorithm is filling the missing rankings with accurate values, i.e., the jobs ranked high for a candidate are indeed relevant to the candidate. Our approach is to apply the Low-rank Matrix Factorization (LMF) technique [13] to the preference input matrices, where an entry (i, j) corresponds to the rank of candidate c_i for job e_j. LMF produces two factor matrices whose product results in a dense matrix with decimal values at each entry (i, j). The factor matrices are "learned" by minimizing the difference between the explicit rankings and the value obtained at that entry by their multiplication. This preserves the given rankings, while deriving optimal values for the missing ones based on the rankings of all the candidates/employers. In order to convert the decimal matrix entries to discrete ranks, we iterate over each generated ranking list and scale-up the decimals to integers such that the relative order between ranks is preserved. However, it is not guaranteed that the resulting order follows the originally stated rankings—the "learning" is not exact. Several approaches are possible. We can use the LMF output as is. In this case, the stated rankings are taken into account only to the extent they are preserved by LMF. We can preserve the relative ordering between the given ranks. This stops LMF to change the stated order, however, the missing rankings can interleave with the given ones. The last alternative is to fully keep the given rankings and only use the LMF output for the ordering of the non-stated entries. We experimented with each of these solutions and found empirically that preserving only the relative order of the given ranks generates the best results. Thus, we use this approach in LMF-MMDAA. The computation of the factor matrices, i.e., the LMF training, is a time-consuming process. However, this is a pre-processing step that is done once and offline in LMF-MMDAA. Since the online step consists of performing MMDAA on the dense output produced by LMF, the algorithmic complexity of MMDAA is preserved.

3.4 Mixed-MMDAA Algorithm

LMF-MMDAA solves the problem of having a high number of unmatched candidates and employers by incorporating "learned" rankings in recommendation. This is extremely important from a practical point of view because it eliminates the strict requirement to work with complete rankings. However, due to the inclusion of both explicit and derived rankings, the recommendation accuracy

may decline since their relationship is not vetted by the user. A possible solution is to run the LMF rankings by the user in order to reconcile them with the original input and then perform MMDAA on the unified rankings. While this approach eliminates the uncertainty in recommendations, it requires another round of input from the user, possibly delaying the entire process.

We propose a different alternative – the Mixed-MMDAA algorithm – that considers only the original incomplete rankings, performs MMDAA twice – once on the original rankings and another time on the LMF rankings – and combines the two resulting matches in a set of recommendations that are more accurate and also minimize the number of unmatched candidates and employers. Thus, MMDAA and LMF-MMDAA have to be executed before running Mixed-MMDAA on their output. Mixed-MMDAA is not a matching algorithm in the sense of DAA and MMDAA. Its main novelty is that it combines matches rather than rankings. Mixed-MMDAA prioritizes the explicit rankings provided by the user and includes LMF recommendations only when candidates/employers remain unmatched. This guarantees that stable matches are produced whenever the explicit rankings warrant it and useful recommendations are provided instead of blank results.

Algorithm 3: Mixed-MMDAA Algorithm

1 $RE_L \leftarrow$ `LMF(`RE`)` ▷ LMF to fill missing rankings
2 $RC_L \leftarrow$ `LMF(`RC`)`
3 $\left(ME_i^M, MC_j^M\right) \leftarrow$ `MMDAA(`C, E, RE, RC, m`)` ▷ MMDAA
4 $\left(ME_i^L, MC_j^L\right) \leftarrow$ `MMDAA(`C, E, RE_L, RC_L, m`)` ▷ LMF
5 Fill empty matches in ME_i^M with best matches from ME_i^L
6 Fill empty matches in MC_j^M with best matches from MC_j^L
7 **return** $\left(ME_i^M, MC_i^M\right)$

As depicted in Figure 5, Mixed-MMDAA takes as input four sets of matches: the MMDAA candidate and employer match results from the explicit rankings (the upper-most matrices in Figure 5); and the candidate and employer match results from the LMF-MMDAA rankings (the next-to-bottom matrices in Figure 5). The main task of the algorithm is to combine these matches into a better set of recommendations for each candidate/employer (line 5-6 in Algorithm 3). Specifically, we have to find all of the matching rounds that result in a no match, i.e., N/A, and fill them with a substitute match. To achieve this, we iterate over the explicit MMDAA matches and check if there is a match for each round. If there is a round that has no match, then we fill it with the next best match—found in the LMF matches. The chosen LMF match has to satisfy two requirements. First, the new match must not have been matched to this particular candidate/employer in any other round. Otherwise, this results in duplicate matches. Second, the new match must not have been matched to any other candidate/employer in the current round. Otherwise, two entities share the same match in the same round, which violates the stability and the one-to-one match requirements of the algorithm. The worst-case runtime complexity of this process is $O(k \cdot m \cdot n)$, where n is the largest of the number of candidates and employers, m is the number of matching rounds in MMDAA, and k is the number of matching rounds in LMF-MMDAA. However, notice that m and k are typically constants much smaller than n, thus, the complexity of merging in Mixed-MMDAA is closer to $O(n)$—the MMDAA calls remain the most time-consuming part of the entire algorithm.

Figure 5 illustrates how match merging is performed in Mixed-MMDAA. Starting from the explicit input rankings, two paths are followed. On the first path – going up – the MMDAA matches are generated by executing the MMDAA algorithm. We observe that there are no matches both for candidates and employers in round 2. On the second path – going down – the explicit rankings are enriched with LMF collaborative filtering. Then, the MMDAA algorithm is executed over these LMF rankings to generate the complete set of LMF matches. Notice that there are no empty cells in the LMF matches. Mixed-MMDAA merges the MMDAA and LMF matches in the final output matches. We observe that the final matches are identical to the MMDAA matches, while eliminating the N/A entries. The replacement of N/A is selected from the LMF matches. The final output does not contain matches where there is no ranking specified, e.g., for candidate c_3 and job e_2. Alternatively, these entries can be populated from the LMF matches.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate experimentally the three MMDAA algorithms proposed in the paper. Our main focus is the quality of the recommendations. The runtime performance is secondary because matching algorithms are performed offline, after the ranking preferences are gathered—executing the algorithm typically takes a fraction of the time to obtain the input data. Nonetheless, all the algorithms finish execution on the tested data in less than a minute.

Datasets. We perform experiments on six datasets—five synthetic and one real. The synthetic datasets are generated based on the simulation of a traditional job market with different number of candidates and employers. Each candidate/employer is randomly assigned two attribute values drawn from a normal distribution with mean 0 and standard deviation 1. They are also randomly assigned one of two types with even odds. Type-1 candidates and employers find the first attribute twice as important as the second. Type-2 candidates and employers find the second attribute twice as important as the first. Candidates rank 10 random jobs by a utility function defined over the two attributes, while the employers rank the candidates in their pool by their own utility function defined over the same two attributes. Having a utility function makes these preference datasets more realistic, rather than being purely random. We apply this data generation process to five combinations of candidates and jobs – $(10, 100)$, $(50, 100)$, $(100, 100)$, $(110, 100)$, and $(150, 100)$ – by keeping the number of jobs constant and varying the number of candidates such that their ratio covers a large range.

We obtain the real dataset from a user study we conducted on the 2019 academic job market that involves candidates for faculty positions and higher education institutions [24]. A total of 24 employers and 75 candidates provided their rankings on our website that allows users to create candidate or employer profiles. Candidates can search for jobs posted on a popular website for academic jobs and rank them in the order of their preference. Employers can manage jobs they are responsible for and rank candidates who

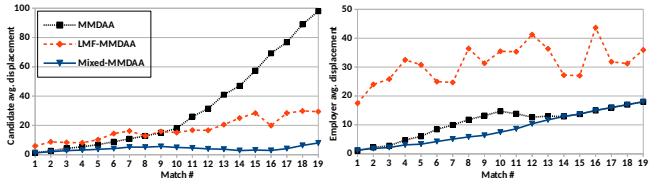

Figure 6: Average displacement on synthetic data.

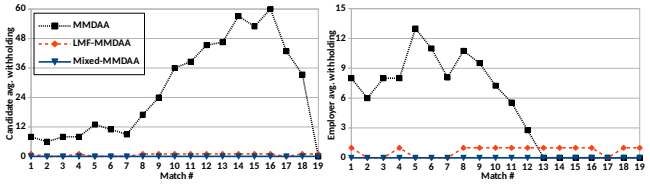

Figure 7: Average withholding on synthetic data.

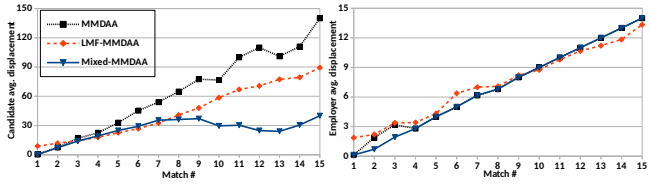

Figure 8: Average displacement on real data.

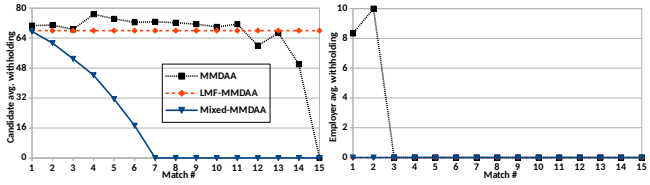

Figure 9: Average withholding on real data.

applied for those jobs. This website allows us to gather preference data from candidates and employers and provide recommendations using the proposed MMDAA algorithms. More specifications on these datasets can be found in [21].

Metrics. The qualitative metrics used to evaluate the MMDAA algorithms are displacement and withholding. *Displacement* measures the match accuracy of an MMDAA algorithm with respect to the ranked preferences. It tracks where each match recommendation lies in the ordered input preferences and returns the absolute difference between the desired rank and the assigned recommendation. For accurate results, an MMDAA algorithm has to produce low average displacement across all the candidates/employers, as this indicates that most of the recommendations are matching their rank. To handle situations where a candidate/job does not get a recommendation in a matching round of MMDAA, we apply a displacement penalty. Essentially, displacement is a measure of precision across ordered sets. *Withholding* measures the number of candidates/employers that did not get a recommendation in a given matching round. This metric helps insure that candidates and employers are getting as many recommendations as possible and do not sit out in many rounds. The combination of these two metrics determines which MMDAA algorithm is more accurate in terms of meaningful recommendations and more exhaustive in providing a sufficient number of recommendations. Displacement and withholding are evaluated on each dataset, for each MMDAA algorithm. For LMF-MMDAA and Mixed-MMDAA, we set the maximum size of the output to the number of recommendations required by MMDAA to converge—the numbers on the x-axis in all the figures, e.g., MMDAA generates all the matches in 19 rounds on the synthetic dataset and 15 rounds on the real, respectively. This is because MMDAA terminates after considerably fewer matching rounds and we compare the algorithms on a round-by-round basis. Running LMF-MMDAA to acquire every possible recommendation is not practical since the most important recommendations are found within the first few matching rounds.

Results. The performance of the MMDAA algorithms is depicted in Figure 6-9. Due to lack of space, we include only the results for the (100, 100) synthetic dataset—the results for the other combinations are available in an extended report [21], along with their runtimes.

Average displacement. Average displacement (Figure 6 and 8) starts off with a low value and gradually increases as we progress through the matching rounds. This is primarily due to later recommendations having lower match accuracy because the better matches with higher preference rankings are found in earlier rounds. The increased displacement for later recommendations demonstrates the stability of the matches. The algorithms display the same trend for candidate displacement on both datasets—the left side of the figures. As the rounds increase, Mixed-MMDAA clearly outperforms the other two. MMDAA exhibits large displacement because it returns no matches, which incurs a high penalty. While LMF-MMDAA always returns a match, these are not very accurate in the tail of the distribution. Mixed-MMDAA replaces the MMDAA no matches – eliminates penalties – with the best available LMF recommendation—not the complete default order. Average displacement for employers – the right side of Figure 6 and 8 – is different on the two datasets. This is due to the much smaller number of employers in the real dataset providing fewer preference rankings. As a result, matches in later rounds are recommendations that do not improve displacement. LMF-MMDAA has the worst displacement on synthetic data because the original preference rankings are overwritten with identical values for all the employers. This forces out-of-order matches that increase displacement.

Average withholding. As expected, the withholding on the synthetic (100, 100) dataset is (close to) 0 for LMF-MMDAA and Mixed-MMDAA—Figure 7. These algorithms almost always return a match because they consider the complete set of candidates/jobs. While MMDAA also ends up with 0 withholding in the final matching round, it incurs many no matches in the intermediate rounds because of unsolved conflicting preferences. On the real dataset – Figure 9 – there is a clear discrepancy between candidates and employers because of their different number—75 and 24, respectively. The withholding for employers is 0 across all the algorithms after 3 rounds because of the much larger number of candidates. Due to the lack of employers, at most 24 candidates are recommended

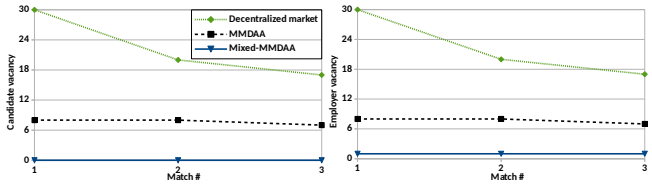

Figure 10: Vacancy on synthetic data.

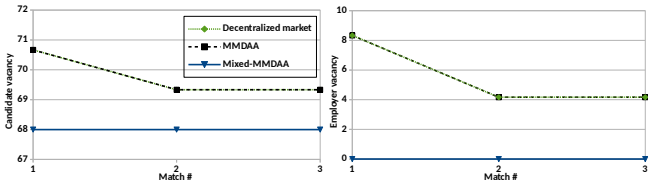

Figure 11: Vacancy on real data.

in a matching round, while the rest must be withheld. Thus, MMDAA results in high average candidate withholding in the first rounds, which decreases to 0 in the end. Mixed-MMDAA achieves 0 withholding earlier because it considers alternative matches. LMF-MMDAA blocks in a state where certain candidates cannot be matched because of identical rankings with other candidates—exactly one candidate can get a job.

5 JOB MARKET SIMULATION

In order to confirm that our MMDAA algorithms are stabilizing a job market, we have created a realistic job market simulator. This simulator shows the results of what happens if: 1) employers offer positions to candidates based off of their explicit preferences, without the use of MMDAA algorithms—this is a typical decentralized job market; 2) employers offer positions to candidates based off of their recommendation results from MMDAA algorithms. Theoretically, MMDAA minimizes the number of jobless candidates and unfilled positions compared to having employers chase candidates independently. The decentralized job market simulator is designed as follows. There are three classes of employers: 1) high-class employers; 2) medium-class employers; and 3) low-class employers. We allocate a third of the employers in a dataset to each of the three classes. Employers offer positions to candidates based off of the employers' class type in three position offering sessions. In other words, each employer has three rounds/chances to hire a candidate. High-class employers offer positions only to the most preferred candidates. Medium-class employers offer positions to candidates that are preferred, but not most preferred. Specifically, medium-class employers offer positions to candidates who are not among the top 33% of their preferred candidates. The reason for this behavior is due to medium-class employers knowing that their highly-qualified, most preferred candidates probably receive better offers from an employer in a higher class. Following the same reasoning, low-class employers offer positions to candidates who are not among the top 66% of their preferred candidates. If employers use the recommendations from the MMDAA algorithms, each employer offers the job to the top three candidate match results. We do not have to consider the class type of an employer because MMDAA includes preferences from both candidates and employers when computing the stable matches. Essentially, the employer's class type appears implicitly in the candidates' preferences—candidates rank high-class employers higher than middle-class or low-class employers. Because of this, the stable MMDAA one-to-one matches can be used directly by employers to offer positions to candidates. In other words, the simulator has employers always offering positions to candidates based off of their best matches—the top 3 matches. On the candidate side, candidates accept the first offer they receive—if they even receive one. Moreover, a candidate cannot accept more than one offer—additional offers are declined.

Vacancy is used to asses the performance of the simulator. For employers, vacancy measures the number of unfilled positions after a job offering round. For candidates, vacancy measures the number of jobless candidates after a round. Vacancy is maximum after the first job offering round and decreases with each additional round because 1) employers receive more chances to fill a position; and 2) candidates receive more chances to accept an offer. Using the same synthetic and real datasets in Section 4, we compare the job market simulator with MMDAA and Mixed-MMDAA by measuring vacancy—LMF-MMDAA is excluded since Mixed-MMDAA is superior. The results for three matching rounds are depicted in Figure 10 and 11. The decentralized job market simulator always has higher vacancy than Mixed-MMDAA both for jobs and candidates. MMDAA is inferior to Mixed-MMDAA. Although it stabilizes the job market on synthetic data, MMDAA still lets some jobs vacant and some candidates jobless because of the ranking conflicts. Nonetheless, its vacancy rate is much lower than that of the simulator. Mixed-MMDAA uses the LMF recommendations to substitute the rounds that result in no matches. Because of this, it yields a perfect match in which there are no vacant jobs and no unemployed candidates. This is also the case for employers in the real dataset—they all find a candidate to hire. However, candidates remain unemployed because the number of jobs is significantly smaller. Being a direct extension of the DAA algorithm, MMDAA does not perform better than the decentralized simulator on the real data. The main reason is the different number of candidates and employers. Additionally, there are preference conflicts that cannot be reconciled in three matching rounds. Overall, the job market simulation confirms that Mixed-MMDAA minimizes vacancy—up to complete elimination when resources are available.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduce PrivateJobMatch – a privacy-oriented deferred multi-match recommender system – which generates stable matches between candidates and employers. PrivateJobMatch uses only lightweight partial rankings of user preferences – instead of extensive and privacy-sensitive profiles – in order to optimize the stability of the job market as a whole. In the near future, we plan to perform a more extensive study applied to other markets beyond academic jobs. In the long term, we plan to investigate further how to integrate deferred acceptance algorithms into recommender systems. Additionally, we are interested in how traditional user profiles can be used to generate the required partial rankings of a user's preferences.

REFERENCES

[1] Atila Abdulkadiroglu, Parag A. Pathak, and Alvin E. Roth. The New York City High School Match. *American Economic Review*, 2005.

[2] Atila Abdulkadiroglu and Tayfun Sonmez. Matching Markets: Theory and Practice. *Advances in Economics and Econometrics*, 1:3–47, 2013.

[3] Fabian Abel. We Know Where You Should Work Next Summer: Job Recommendations. In *ACM Conference on Recommender Systems (RecSys)*, 2015.

[4] Shaha T. Al-Otaibi and Mourad Ykhlef. A Survey of Job Recommender Systems. *Int. J. Phys. Sci.*, 7(29):5127–5142, 2012.

[5] Fedor Borisyuk, Liang Zhang, and Krishnaram Kenthapadi. LiJAR: A System for Job Application Redistribution towards Efficient Career Marketplace. In *ACM Conference on Knowledge and Data Discovery (KDD)*, 2017.

[6] Vachik S. Dave, Baichuan Zhang, Mohammad Al Hasan, Khalifeh AlJadda, and Mohammed Korayem. A Combined Representation Learning Approach for Better Job and Skill Recommendation. In *ACM International Conference on Information and Knowledge Management (CIKM)*, 2018.

[7] Jonathan M.V. Davis. The Short and Long Run Impacts of Centralized Clearinghouses: Evidence from Matching Teach for America Teachers to Schools. *tinyurl.com/Davis-JMP*, 2017.

[8] Jack Doerner, Dave Evans, and Abhi Shelat. Secure Stable Matching at Scale. In *Conference on Computer and Communications Security (CCS)*, 2016.

[9] David Gale and Lloyd Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1), 1962.

[10] Wenxing Hong, Lei Li, Tao Li, and Wenfu Pan. iHR: An Online Recruiting System for Xiamen Talent Service Center. In *ACM Conference on Knowledge and Data Discovery (KDD)*, 2013.

[11] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. In *ACM Conference on Recommender Systems (RecSys)*, 2017.

[12] Akiva Kleinerman, Ariel Rosenfeld, Francesco Ricci, and Sarit Kraus. Optimally Balancing Receiver and Recommended Users' Importance in Reciprocal Recommender Systems. In *ACM Conference on Recommender Systems (RecSys)*, 2018.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

[14] Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. Matching People and Jobs: A Bilateral Recommendation Approach. In *Hawaii International Conference on Systems Science (HICSS)*, 2006.

[15] Susan Mongell and Alvin E. Roth. Sorority Rush as a Two-Sided Matching Mechanism. *American Economic Review*, 81(3), 1991.

[16] Michael Ostrovsky. Stability in Supply Chain Networks. *American Economic Review*, 98(3):897–923, 2008.

[17] Ioannis K. Paparrizos, Berkant B. Cambazoglu, and Aristides Gionis. Machine Learned Job Recommendation. In *ACM Conference on Recommender Systems (RecSys)*, 2011.

[18] Michael Reusens, Wilfried Lemahieu, Bart Baesens, and Luc Sels. A Note on Explicit versus Implicit Information for Job Recommendation. *Decision Support Systems*, 98:26–35, 2017.

[19] Alvin E. Roth. The Evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory. *Journal of Political Economy*, 92:991–1016, 1984.

[20] Alvin E. Roth and Elliott Peranson. The Redesign of the Matching Market for American Physicians: Some Engineering Aspects of Economic Design. *American Economic Review*, 1999.

[21] Amar Saini. PrivateJobMatch: A Privacy-Oriented Deferred Multi-Match Recommender System for Stable Employment. *arXiv:1905.04564*, 2019.

[22] Zheng Siting, Hong Wenxing, Zhang Ning, and Yang Fan. Job Recommender Systems: A Survey. In *International Conference on Computer Science & Education (ICCSE)*, 2012.

[23] Canadian Resident Matching Service. The Match Algorithm. http://www.carms.ca/en/residency/match-algorithm/, 2016.

[24] EconMatch. https://github.com/AmarSaini/EconMatch, 2019.

[25] National Matching Services, Inc. https://natmatch.com/, 2019.

[26] National Resident Matching Program. 2016 Main Residency Match. http://www.nrmp.org/wp-content/uploads/2016/04/Main-Match-Results-and-Data-2016.pdf, 2016.

[27] PrivateJobMatch Repository. https://github.com/AmarSaini/PrivateJobMatch, 2019.

[28] *RecSys Challenge '16: Proceedings of the Recommender Systems Challenge*, 2016.

[29] *RecSys Challenge '17: Proceedings of the Recommender Systems Challenge*, 2017.