Parallel and Mini-Batch Stable Matching for Large-Scale Reciprocal Recommender Systems

Kento Nakada[1,*], Kazuki Kawamura[2] and Ryosuke Furukawa[1]

[1]Sony Network Communications, Inc.
[2]The University of Tokyo

Abstract

Reciprocal recommender systems (RRSs) are crucial in online two-sided matching platforms, such as online job or dating markets, as they need to consider the preferences of both sides of the match. The concentration of recommendations to a subset of users on these platforms undermines their match opportunities and reduces the total number of matches. To maximize the total number of expected matches among market participants, stable matching theory with transferable utility has been applied to RRSs. However, computational complexity and memory efficiency quadratically increase with the number of users, making it difficult to implement stable matching algorithms for several users. In this study, we propose novel methods using parallel and mini-batch computations for reciprocal recommendation models to improve the computational time and space efficiency of the optimization process for stable matching. Experiments on both real and synthetic data confirmed that our stable matching theory-based RRS increased the computation speed and enabled tractable large-scale data processing of up to one million samples with a single graphics processing unit graphics board, without losing the match count.

Keywords

Reciprocal Recommender Systems (RRSs), Recruitment, Stable Matching, Parallel Computation, Sinkhorn's Algorithm

1. Introduction

Two-sided online dating platforms, such as those found in job search and online dating markets, have become increasingly popular. Reciprocal recommender systems (RRSs) are used on these platforms. [1, 2]. In a two-sided matching platform, a recommender system that only considers prediction accuracy may inadvertently cause recommendation inequalities for two main reasons. First, on a two-sided platform, preferences from both sides of the market determine the success of a match. Recommendations based solely on the interests of one side are ineffective, and the recommender system should only recommend when both users have mutual interests. Second, users tend to have difficulties finding compatible partners. For example, on job platforms, employers frequently have a limited number of available interview slots because of time constraints. If the system recommends the same company to many candidates, it may exceed the employer's capacity, leading to missed opportunities for those candidates who applied but were not granted an interview.

The transferable utility (TU) matching model [3, 4] is a framework that considers these matching problems under the assumption that utilities, such as money, can be transferred between matching parties. This allows for resource redistribution among market participants, thereby achieving stable matching states. Choo and Siow [5] applied the TU matching market model to RRSs, which demonstrated higher matching chances in empirical matching applications with TUs. Although stable matching methods have a solid theoretical background, most of them face computational feasibility bottlenecks when executed on real data sizes exceeding 10k. Thus, they are not practically feasible for large-scale user platforms and are only useful for the experimental extraction of a subset of actual service users [6] or matching at the level of the user group clustered by attributes [7, 8].

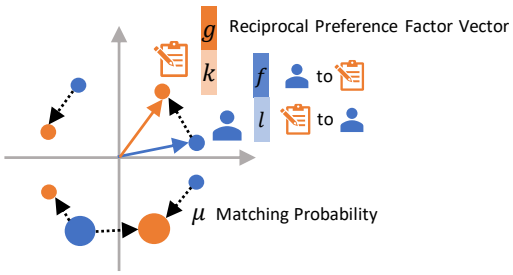

Figure 1: We propose a fast and memory-efficient solution to the TU matching problem, by viewing it as an optimal transport problem with transport costs associated with the inner product of the preference factor vectors.

In this study, we improve the computational efficiency of the iterative proportional fitting procedure (IPFP), a coordinate descent algorithm used to achieve a stable matching state [9], through parallelization and online mini-batch computation. For computational efficiency, we propose a matrix-vector multiplication-based parallel computation method following Sinkhorn's algorithm [10], which considers stable matching as an entropy-regularized optimal transport problem. To improve memory efficiency, we propose a mini-batch update method assuming a model that uses user factor vectors in the unilateral recommendation model, such as matrix factorization [11]. The user set is divided into several partitions (i.e., mini-batches), and a portion of the IPFP coordinate vectors is sequentially updated. This approach enables efficient estimation of update values when the overall preference scores for user combinations do not fit in memory.

In summary, the following are the main contributions of this study.

- We propose an optimization method for RRSs based on the TU matching model to increase computational efficiency using parallel processing.
- Furthermore, we propose a memory-efficient mini-batch update method that does not require approximation and works even for large sizes that do not

RecSys in HR'24: The 4th Workshop on Recommender Systems for Human Resources, in conjunction with the 18th ACM Conference on Recommender Systems, October 14–18, 2024, Bari, Italy.
Kento.Nakada@sony.com (K. Nakada); kwmr@acm.org (K. Kawamura); Ryosuke.Furukawa@sony.com (R. Furukawa)

fit in a single memory.
- Experiments on a synthetic dataset with up to a million users verify the computational efficiency of the proposed method.

To the best of our knowledge, this study is the first to demonstrate how RRSs based on stable matching theory work with large-scale data.

2. Related Work

Reciprocal recommender systems (RRSs) aim to achieve matching by considering the preferences of both parties, which are primarily used in areas where mutual matching is crucial, such as dating sites, talent recruitment platforms, and social networking services [1].

Existing RRSs first predict unilateral preferences that represent the preference of one user toward another. In unidirectional preference calculation, various recommendation methods are employed in recruitment matching problems, including content-based and collaborative filtering approaches [12]. Among these, collaborative filtering is particularly well suited for scenarios with abundant interaction logs. Herein, we focus on collaborative filtering-based methods for reciprocal recommender systems to apply matching platforms with number of users.

These unilateral preferences are then aggregated to calculate reciprocal preference scores, which serve as recommendation probabilities. Aggregation functions that independently calculate for each users pair—such as arithmetic mean [2], harmonic mean [2, 13], and weighted average [14]—can scale to large data size [15]. Although these methods can compute for a considerable number of user data, they lack theoretical background and cannot consider constraints on users' limited capacity of matches.

RRSs based on fairness-aware aggregation models redistribute recommendation scores to address the concentration of recommendation issues [16, 17, 18]. Several approaches have attempted to prevent concentration in job recommendation by solving the entropy-regularized optimal transport (OT) problem using reciprocal preference scores as a transport cost matrix [19, 20]. These methods can be applied to large-scale data following Sinkhorn's algorithm [21]. However, there is a lack of theoretical background on the alignment between the OT and equilibrium-matching states in terms of modeling user behavior.

The stable matching theory [3] is a behavioral model based on game theory that balances supply and demand in matching markets. Galichon and Salanié [9] demonstrated that the TU matching theory corresponds to the dual problem of a special entropy-regularized OT problem, which provides a theoretical foundation for the application of stable matching in recommender systems.

Although stable matching methods have a solid theoretical background, most of them face computational feasibility bottlenecks when applied to real data sizes exceeding tens of thousands. Galichon and Salanié [9] proposed that their optimization method could be computed in parallel, although their experiment used a dataset of up to a size of 5000. Chen et al. [7] applied stable matching to male–female matching app data of approximately 1000. Tomita et al. [6] proposed a TU matching theory-based memory-efficient inference method for reciprocal scores. To optimize the parameters, all user combinations must be loaded into memory, and a sample size of up to 1000 was used in the experiments. Thousands to hundreds of thousands of users use job or date matching services. Therefore, we extend the applicability of TU matching to the user bases of that scale.

3. Proposed Method

According to [7], this section provides a brief overview of how the TU matching model can be formulated as a convex optimization problem, as described in Section 3.1. Thereafter, we propose a parallel update method using a mini-batch approach for the iterative optimization algorithm in Sections 3.2 and 3.3. This method enables tractable calculations in near-linear memory consumption, even for many market users.

3.1. Matching Model with Transferable Utility

We assume matching between candidate $x \in \mathcal{X}$ and employer $y \in \mathcal{Y}$. The concept of utility explains why an individual chooses where to apply or whom to recruit. Utilities $U_{x,y}, V_{x,y}$ that candidate x or employer y gains by matching are expressed as follows:

$$\begin{aligned} U_{x,y} &= p_{x,y} + \epsilon_{x,y},\ \epsilon_{x,y} \sim \mathcal{P}, \\ V_{x,y} &= q_{y,x} + \eta_{y,x},\ \eta_{y,x} \sim \mathcal{Q}, \end{aligned} \tag{1}$$

where $p_{x,y}$ denotes an observable utility from candidate x to employer y, $q_{y,x}$ denotes an observable utility from y to x, and $\epsilon_{x,y}$ and $\eta_{y,x}$ are unobserved random utilities with probability distributions of $\mathcal{P}$and $\mathcal{Q}$, respectively. We denote the individual who is not matched to any counterpart as 0. Let $\mathcal{X}_0 = \mathcal{X} \cup \{0\}$ and $\mathcal{Y}_0 = \mathcal{Y} \cup \{0\}$ be a set of groups that can be selected as potential partners.

In the TU matching model, we assume that the transferable utility $\tau_{x,y}$ is paid from an employer to a job candidate upon matching (for example, to adjust supply and demand). The matching results are adjusted by optimizing the utility. Specifically, to mitigate the over-concentration of recruitment efforts on highly sought-after candidates, a high cost may be incorporated into the scouting process. Consider observable joint utility $\phi_{x,y} = p_{x,y} + q_{y,x}$ and probability distribution $\mu_{x,y}$ that specifies a match between candidate x and employer y. Assuming that the distribution of random utilities $\mathcal{P}$and $\mathcal{Q}$ are independent and identically distributed (i.i.d.) with a type-I extreme value distribution with scale parameter $\beta > 0$, Galichon and Salanié [9] revealed that optimizing TU $\tau_{x,y}$ to maximize the expected number of matching in the market is defined as the following convex optimization problem, which maximizes social welfare W:

$$\begin{aligned} W(\boldsymbol{\Phi}) = \max_{\boldsymbol{\mu} \in \mathbb{R}^{\mathcal{X} \times \mathcal{Y}}} & \left(\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \mu_{x,y} \phi_{x,y} + \beta E(\boldsymbol{\mu}) \right), \\ \text{s.t.} \sum_{y \in \mathcal{Y}_0} \mu_{x,y} = n_x \quad \forall x \in \mathcal{X}, & \sum_{x \in \mathcal{X}_0} \mu_{x,y} = m_y \quad \forall y \in \mathcal{Y}, \end{aligned} \tag{2}$$

where

$$E(\boldsymbol{\mu}) = -\sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}_0}} \mu_{x,y} \log \frac{\mu_{x,y}}{n_x} - \sum_{\substack{y \in \mathcal{Y} \\ x \in \mathcal{X}_0}} \mu_{x,y} \log \frac{\mu_{x,y}}{m_y} \tag{3}$$

is the standard entropy; n_x and m_y are the normalized capacity constraints of candidate group x and employer group y, respectively. [1] In recruitment matching problems, n_x and m_y can correspond to the number of applications a candidate can submit or the number of positions an employer wants to fill. The parameter β controls the weight of the entropy term in (2). As β increases, it promotes a more uniform match result that is less dependent on individual preferences.

Equation (2) is an OT problem with entropy regularization. We adopt the IPFP proposed in [9], a coordinate decent method to solve (2). The detailed derivation of the TU matching optimization is described in Appendix A.

3.2. Parallel Computation of TU Matching Optimization

We propose a parallel update method to alleviate the computational limitation of an IPFP update on a large user base. At the optimum, an equation on a derivative of (2) derives the following relation between $\phi_{x,y}$ and $\mu_{x,y}$:

$$\mu_{x,y} = \exp\left(\frac{\phi_{x,y}}{2\beta}\right)\sqrt{\mu_{x,0}\mu_{0,y}}. \tag{4}$$

By substituting this expression into the constraints in (2), we obtain the following equation:

$$\begin{aligned} \mu_{x,0} + \left(\sum_{y\in\mathcal{Y}} \exp\left(\frac{\phi_{x,y}}{2\beta}\right)\sqrt{\mu_{0,y}}\right)\sqrt{\mu_{x,0}} = n_x, \\ \mu_{0,y} + \left(\sum_{x\in\mathcal{X}} \exp\left(\frac{\phi_{x,y}}{2\beta}\right)\sqrt{\mu_{x,0}}\right)\sqrt{\mu_{0,y}} = m_y. \end{aligned} \tag{5}$$

The IPFP algorithm solves (5) given scaling vector definitions of $u_x = \sqrt{\mu_{x,0}}$ and $v_y = \sqrt{\mu_{0,y}}$, using i as the number of iteration steps to repeatedly run through the following updates:

$$\begin{cases} u_x^{(i+1)} = \sqrt{n_x + s_x^2} - s_x \text{ where } s_x = \frac{1}{2}\sum_{y\in\mathcal{Y}} \exp\left(\frac{\phi_{x,y}}{2\beta}\right) v_y^{(i)} \\ v_y^{(i+1)} = \sqrt{m_y + s_y^2} - s_y \text{ where } s_y = \frac{1}{2}\sum_{x\in\mathcal{X}} \exp\left(\frac{\phi_{x,y}}{2\beta}\right) u_x^{(i+1)} \end{cases}, \tag{6}$$

Once the algorithm has converged after several iterations, the stable match patterns $\boldsymbol{\mu}$ are calculated according to Equation (4).

Equation (6) can be further expressed in matrix-vector arithmetic as follows:

$$\begin{cases} \boldsymbol{u}^{(i+1)} = \sqrt{\boldsymbol{n} + \boldsymbol{s}^2} - \boldsymbol{s} \text{ where } \boldsymbol{s} = \frac{1}{2}\boldsymbol{A}\boldsymbol{v}^{(i)} \\ \boldsymbol{v}^{(i+1)} = \sqrt{\boldsymbol{m} + \boldsymbol{s}^2} - \boldsymbol{s} \text{ where } \boldsymbol{s} = \frac{1}{2}\boldsymbol{A}^T\boldsymbol{u}^{(i+1)} \end{cases}, \tag{7}$$

where $\boldsymbol{A} = \exp\left(\frac{\boldsymbol{\Phi}}{2\beta}\right)$. We denote the update method by Equation (7) as *batch* IPFP. Because Equation (7) only involves the matrix-vector product, it can be efficiently computed through parallel computation. However, because the size of the matrices scales in $O(|\mathcal{X}||\mathcal{Y}|)$, the equation is computationally intractable within the available memory.

3.3. Mini-batch Computation of TU Matching Optimization

To alleviate the memory space limitation, we further propose a memory-efficient update method, *mini-batch* IPFP.

[1] Decker et al. [22] investigated the existence and uniqueness of the equilibrium-matching solution.

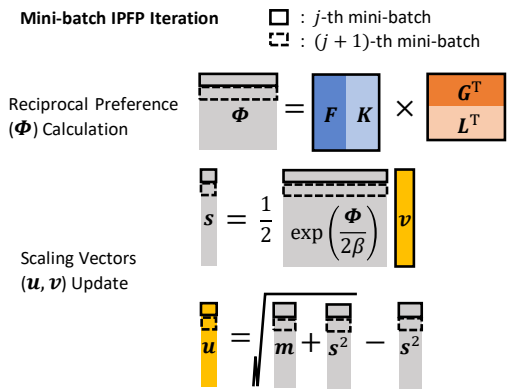

Figure 2: Our Mini-batch IPFP update keeps only a part of the matrices in memory during the update step, achieving parallel computation and memory efficiency.

The inner product of the D dimensional factor vectors $\boldsymbol{f}_x, \boldsymbol{g}_y, \boldsymbol{k}_x, \boldsymbol{l}_y \in \mathbb{R}^D$ through matrix factorization is assumed to compute unilateral preferences $p_{x,y}$ and $q_{y,x}$.

$$p_{x,y} = \langle \boldsymbol{f}_x, \boldsymbol{g}_y \rangle, \; q_{y,x} = \langle \boldsymbol{k}_x, \boldsymbol{l}_y \rangle \tag{8}$$

In such a case, user sets can be substituted for the IPFP algorithm to be executed online. Let $\mathcal{X}_j (j = 1, ..., J_x, 1 < J_x \leq |\mathcal{X}|)$ and $\mathcal{Y}_j (j = 1, ... J_y, 1 < J_y \leq |\mathcal{Y}|)$ be a j-th mini-batch of the candidate and employer sets, respectively. Now, Equation (7) can be calculated for each j-th mini-batch.

$$\begin{cases} \boldsymbol{u}_j^{(i+1)} = \sqrt{\boldsymbol{n}_j + \boldsymbol{s}_j^2} - \boldsymbol{s}_j \text{ where } \boldsymbol{s}_j = \frac{1}{2}\boldsymbol{A}_j\boldsymbol{v}^{(i)} \\ \boldsymbol{v}_j^{(i+1)} = \sqrt{\boldsymbol{m}_j + \boldsymbol{s}_j^2} - \boldsymbol{s}_j \text{ where } \boldsymbol{s}_j = \frac{1}{2}(\boldsymbol{A}^T)_j\boldsymbol{u}^{(i+1)} \end{cases}, \tag{9}$$

where subscript $*_j$ denotes the indices of users in the j-th mini-batch and

$$\boldsymbol{A}_j = \exp\left(\frac{\boldsymbol{F}_j\boldsymbol{G}^T + \boldsymbol{K}_j\boldsymbol{L}^T}{2\beta}\right), \; (\boldsymbol{A}^T)_j = \exp\left(\frac{\boldsymbol{F}\boldsymbol{G}_j^T + \boldsymbol{K}\boldsymbol{L}_j^T}{2\beta}\right). \tag{10}$$

According to Tomita et al. [6], once optimal $\boldsymbol{u}$ and $\boldsymbol{v}$ are obtained, stable match patterns $\log \boldsymbol{\mu}$ can be calculated as the dot product of the following two vectors with dimensions $(2D + 2)$.

$$\begin{aligned} \log\left(\mu_{x,y}\right) &= \frac{1}{2\beta}\left\langle \boldsymbol{\psi}_x, \boldsymbol{\xi}_y \right\rangle, \\ \boldsymbol{\psi}_x &= \text{Concat}\left(\boldsymbol{f}_x, \boldsymbol{k}_x, \beta\log\left(\boldsymbol{u}\right), 1\right), \\ \boldsymbol{\xi}_y &= \text{Concat}\left(\boldsymbol{g}_y, \boldsymbol{l}_y, 1, \beta\log\left(\boldsymbol{v}\right)\right), \end{aligned} \tag{11}$$

where Concat(*) denotes the concatenation of vectors. In the case of large data sizes (i.e., $D \ll |X|$), the space complexity is reduced to $O(|X|)$ or $O(|Y|)$. In practice, this reduction allows the execution of the IPFP by adjusting the batch size to fit within the memory limit.

The pseudocodes of the *batch* and *mini-batch* IPFP algorithm is shown in Appendix B.

4. Experiments

Here, we first validate the IPFP algorithm by comparing it with existing methods in terms of the expected number of matches, using both real and synthetic data.

Thereafter, as our contribution, we validate the computational efficiency of the proposed batch and mini-batch updates in the CPU/GPU settings. Furthermore, we investigate the memory efficiency and calculation performance of the mini-batch IPFP algorithm using various batch sizes. We aim to improve the efficiency of the IPFP algorithm because the expected number of matches is the same as that of IPFP.

4.1. Experiments on Expected Number of Matches

4.1.1. Datasets.

We compared the IPFP algorithm with existing methods using data from the online dating platform Libimseti [13]. This dataset includes user reciprocal ratings. We chose 500 male and 500 female users who submitted the highest number of ratings. Any missing ratings were filled in using probabilistic matrix factorization with the alternating least squares (ALS) method [23].

In experiments with real data, it is impossible to know the true preferences of each user in the market. Consequently, the expected number of matches can only be calculated using estimated preferences. To address this limitation, we also conducted experiments using synthetic data, which allowed us to control the true preferences in a structured manner.

We evaluated the methods using synthetic data generated based on the process described in [18]. The preference matrices $p_{x,y}$ and $q_{y,x}$ are generated by interpolating random values sampled from an independent uniform distribution and values proportional to the index of samples, indicating crowding of preferences in the market. The degree of preference crowding can be adjusted using the parameter $\lambda \in [0, 1]$. The experiment involved setting the number of employers at 500, candidates at 1000, and the crowding parameter was varied over $0.0, 0.25, 0.5, 0.75$. The observational data are simulated by sampling binary values $\{0, 1\}$ from Bernoulli distributions based on the probabilities of the generated preference matrices. The factor vectors are obtained by the implicit alternating least squares (iALS) method [24] from observational data.

For both real and synthetic dataset, we repeated this evaluation 10 times to obtain the average and standard error.

4.1.2. Algorithms and Metrics.

We compared the TU method with three baselines: naive, reciprocal and cross-ratio (CR) methods. The naive method uses the unidirectional preference from the candidate to employer $p_{x,y}$ to create the presentation list. The reciprocal method uses the product of preferences from both sides $p_{x,y} * q_{y,x}$ to create the presentation list. The CR method proposed in [25] uses a cross ratio uninorm of preferences from both sides. We attempted to apply the method proposed by Su et al.[18] to both real and synthetic data, however the optimization process did not complete in tractable time.

We further compared the optimization methods of the TU method: batch and mini-batch IPFP. For the TU method, we used $\beta = 1.0$. In both the Libimseti and synthetic data experiment, the baseline and batch IPFP methods utilized the imputed preference matrix, which is the product of these factor vectors. The mini-batch IPFP directly employed the factor vectors.

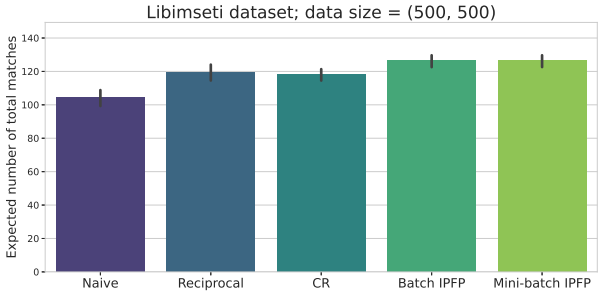

Figure 3: Results of Libimseti data experiments. The market size is 500 men and 500 women, and the examination function is $v(k) = 1/\exp(k-1)$.

We evaluate our method and the baselines in terms of the expected total number of matches, which is calculated by social welfare in a two-sided market, as defined in [18]. To simulate user groups with highly congested populations, we used the exponentially decaying examination function in the position-based model.

$$v(k) = 1/\exp(k-1), \tag{12}$$

, where k denotes the index in the ranking list presented to the candidate or employer. For synthetic datasets we calculate social welfare using the generated preference matrix. For the Libimseti dataset, we calculate social welfare using the imputed preference matrix.

4.1.3. Results.

Figure 3 presents the results of the Libimseti dataset. The batch and mini-batch IPFP methods demonstrated the highest number of matches, indicating its effectiveness in optimizing match outcomes. Figure 4 presents the results of the synthetic data experiments conducted with various crowding parameters. The results demonstrate that the IPFP-based recommendation policy maintains superior performance as crowding parameters increase. Specifically, the expected number of matches not only remains consistently higher than the baseline methods but also exhibits remarkable resilience to performance degradation under increased crowding conditions. This suggests that IPFP is particularly effective in highly crowded markets. Considering that factor vector-based preference matrix calculations, such as matrix factorization, are commonly used for large-scale real-world recommender systems, we believe that mini-batch IPFP remains a viable approach for efficiently handling large-scale matching problems.

4.2. Experiments on Computational Efficiency

4.2.1. Datasets.

We generated synthetic market data with the same number of candidates and employers. For batch IPFP experiments, we set the sample size parameters $|\mathcal{X}| = |\mathcal{Y}|$ in $\{10^2, 10^3, 10^4\}$. For mini-batch IPFP experiments, we further expanded the size parameters in $\{10^2, ..., 10^6\}$.

To calculate the unidirectional preference score, we sampled the factor vectors of candidates and employers $\boldsymbol{f}_x, \boldsymbol{k}_x, \boldsymbol{g}_y, \boldsymbol{l}_y$ from a uniform distribution $U\left[0, \frac{1}{\sqrt{D}}\right]$, where the

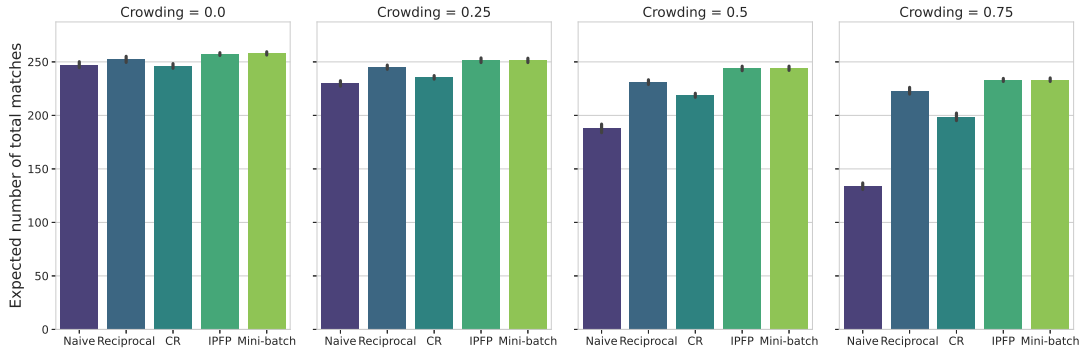

Figure 4: Synthetic data experiment results at various crowding parameter levels. The market size is 500 jobs and 1000 candidates, and the examination function is $v(k) = 1/\exp(k-1)$. When following the IPFP-based recommendation policy, the expected number of matches remains higher than the baseline methods as crowding parameters increase. For mini-batch IPFP, the number of matches slightly decreases because the preference matrix is approximated using the product of factor vectors.

dimension of the factor vector is $D = 50$ during the experiments. We assumed that all users have the same capacity value – $\forall x \in \mathcal{X}$, $n_x = C/|\mathcal{X}|$ and $\forall y \in \mathcal{Y}$, $m_y = C/|\mathcal{Y}|$, where C is a constant value.

4.2.2. Algorithms and Metrics.

We compared the following four IPFP variants in terms of computational cost and memory efficiency: (a) Batch IPFP on CPU (vanilla IPFP, as baseline), (b) Batch IPFP on GPU, (c) Mini-batch IPFP on CPU, and (d) mini-batch IPFP on GPU. We set $\beta = 1.0$ for all methods. For mini-batch IPFP methods, we experimented with various batch sizes in $B \in \{1, 10, 100\}$ to fit within our computer memory. We executed iteration $I = 100$ loops and evaluated the average calculation time per loop and the overall CPU or GPU memory consumption.

We also measured the computation time and memory consumption for a population of $|\mathcal{X}| = |\mathcal{Y}| = 10^4$ while varying the dimension of factor vectors in the mini-batch IPFP. [2]

The source codes were written by inheriting from OTT-JAX [26], a solver kit for solving optimal transport problems using the vector computation library JAX [27]. [3]

4.2.3. Results.

Figure 5 presents the average computation time per iteration for each method. The GPU implementation of IPFP is faster than the CPU implementation for both batch and mini-batch. Batch IPFP requires more memory than mini-batch IPFP because it requires loading the preference matrices into memory in advance. In our experimental environment, an out-of-memory error prevented execution when the data size exceeded 10^5. The mini-batch IPFP handled memory consumption via its online factor vector product calculation and yielded a calculation time comparable to batch IPFP.

Figure 6 presents the computation time and memory usage of mini-batch IPFP for large datasets with different batch sizes. The execution time increases by a constant factor with the batch size. However, because of the effective memory processing of JAX, memory usage remains linear scaling regardless of batch size. In our calculation environment, setting $B = 100$ allows IPFP to be performed in tractable time even with a data size of 10^6. In practical applications to

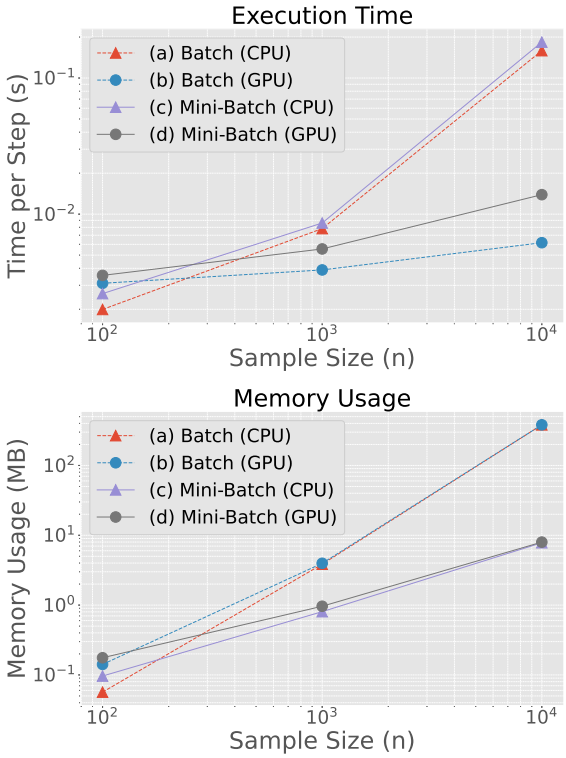

Figure 5: Calculation time (left) and memory usage (right) of batch and mini-batch IPFP with varying data size values. The average calculation times span over 100 iterations. Memory usage is measured on CPU memory for (a) and (c) and GPU memory for (b) and (d).

real-world scenarios, it is conceivable that the overall computation time could be reduced by implementing an early stopping criterion. This could be achieved by terminating the process when the ranking for candidates and employers remains stable for a predetermined number of epochs.

Figure 7 presents the results on the computation time and memory efficiency of the algorithm on synthetic dataset with various dimensions of factor vectors. The increase in computation time and memory consumption of minibatch-IPFP has an almost linear relationship with the increase in the dimension of factor vectors.

[2]We conducted experiments on an Intel Core i9-12900K CPU and single NVIDIA GeForce RTX 3080 (10GB memory) GPU computer.
[3]https://github.com/74hcnklULDuids89/minibatch-ipfp.git

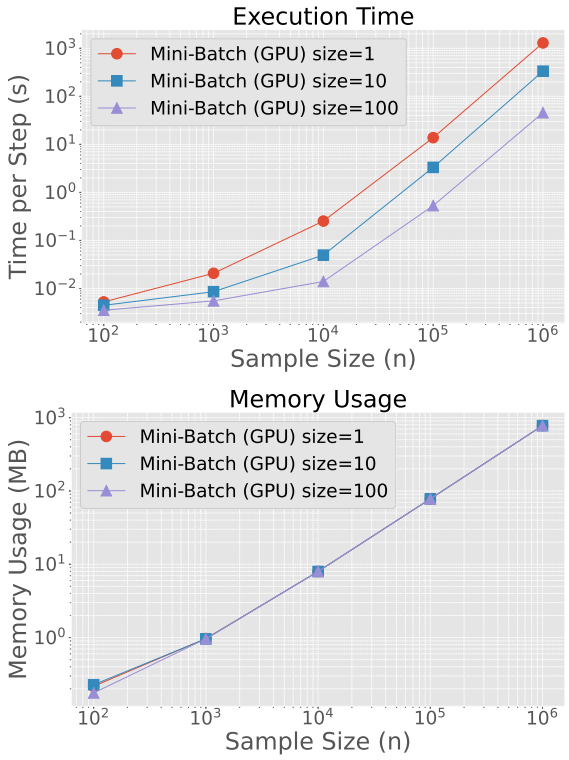

Figure 6: Calculation time and memory usage of mini-batch IPFP for large sample sizes with various batch sizes. The average calculation times span over 100 iterations.

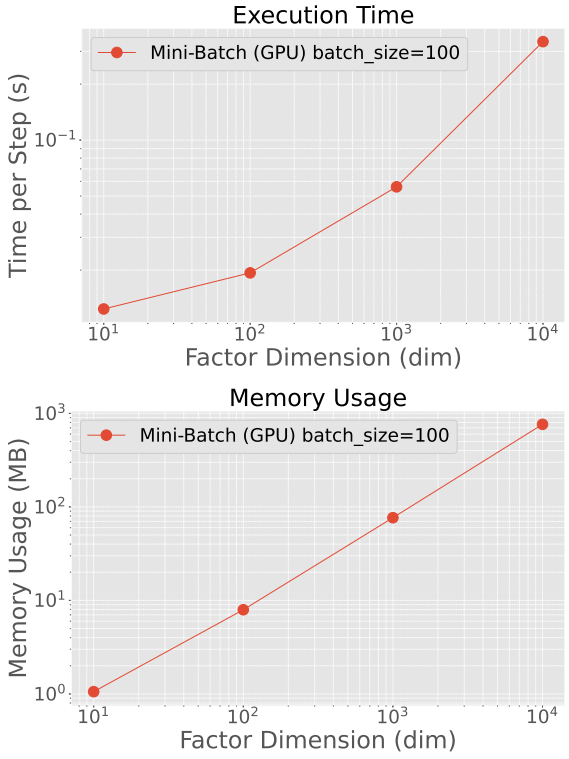

Figure 7: Calculation time and memory usage of mini-batch IPFP for various dimensions of factor vectors. The average calculation times span over 100 iterations.

5. Conclusion and Future Work

In this study, we propose a novel method that significantly improves the computational efficiency and memory usage of the IPFP algorithm to use the TU matching theory in RSSs with large data. Our method achieved the same matching probabilities as the conventional IPFP algorithm, while operating efficiently even for large-scale data using parallel and mini-batch computation techniques. Our experiments on synthetic and real datasets have demonstrated that our method can process computations for up to a million users, even in typical CPUs/GPUs.

In future work, we plan to apply the acceleration techniques developed in the field of OT problems. Initially, we approximated the transportation cost matrix using the low-rank Sinkhorn factorization algorithm[28], reducing the spatial complexity. Second, we calculated the derivative values for the preference matrix and backpropagated them to a unilateral recommendation model [20].

References

[1] I. Palomares, C. Porcel, L. Pizzato, I. Guy, E. Herrera-Viedma, Reciprocal recommender systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation, 2021. doi:10.1016/j.inffus.2020.12.001.

[2] L. Pizzato, T. Rej, T. Chung, I. Koprinska, J. Kay, Recon: A reciprocal recommender for online dating, in: RecSys, New York, NY, USA, 2010, p. 207–214. doi:10.1145/1864708.1864747.

[3] L. S. Shapley, M. Shubik, The Assignment Game I: The Core, volume 1, Physica-Verlag GmbH, DEU, 1971. doi:10.1007/BF01753437.

[4] G. S. Becker, A theory of marriage: Part i, Journal of Political Economy 81 (1973) 813–846. URL: http://www.jstor.org/stable/1831130.

[5] E. Choo, A. Siow, Who marries whom and why, Journal of Political Economy 114 (2006) 175–201. URL: http://www.jstor.org/stable/10.1086/498585.

[6] Y. Tomita, R. Togashi, Y. Hashizume, N. Ohsaka, Fast and examination-agnostic reciprocal recommendation in matching markets, in: RecSys, New York, NY, USA, 2023, p. 12–23. doi:10.1145/3604915.3608774.

[7] K.-M. Chen, Y.-W. Hsieh, M.-J. Lin, Reducing recommendation inequality via two-sided matching: a field experiment of online dating, International Economic Review 64 (2023) 1201–1221. doi:10.1111/iere.12631.

[8] A. Saini, F. Rusu, A. Johnston, Privatejobmatch: A privacy-oriented deferred multi-match recommender system for stable employment, In: RecSys, New York, NY, USA, 2019, p. 87–95. doi:10.1145/3298689.3346983.

[9] A. Galichon, B. Salanié, Cupid's invisible hand: Social surplus and identification in matching models, The Review of Economic Studies 89 (2021) 2600–2629. doi:10.1093/restud/rdab090.

[10] P. Knopp, R. Sinkhorn, Concerning nonnegative matrices and doubly stochastic matrices., Pacific Journal of Mathematics 21 (1967) 343–348.

[11] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37. doi:10.1109/MC.2009.263.

[12] G. Özcan, S. G. Ögüdücü, Applying different classification techniques in reciprocal job recommender system for considering job candidate preferences, in: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 235–240. doi:10.1109/ICITST.2016.7856703.

[13] P. Xia, B. Liu, Y. Sun, C. Chen, Reciprocal recommendation system for online dating, in: ASONAM, New York, NY, USA, 2015, p. 234–241. doi:10.1145/2808797.2809282.

[14] A. Kleinerman, A. Rosenfeld, F. Ricci, S. Kraus, Optimally balancing receiver and recommended users' importance in reciprocal recommender systems, in: RecSys, ACM, New York, NY, USA, 2018, p. 131–139. doi:10.1145/3240323.3240349.

[15] R. Ramanathan, N. K. Shinada, M. Shimatani, Y. Yamaguchi, J. Tanaka, Y. Iizuka, S. K. Palaniappan, A reciprocal embedding framework for modelling mutual preferences, in: AAAI, volume 35, 2021, pp. 15385–15392. doi:10.1609/aaai.v35i17.17807.

[16] F. Borisyuk, L. Zhang, K. Kenthapadi, Lijar: A system for job application redistribution towards efficient career marketplace, in: KDD, New York, NY, USA, 2017, p. 1397–1406. doi:10.1145/3097983.3098028.

[17] V. Do, N. Usunier, Optimizing generalized gini indices for fairness in rankings, 2022. doi:10.48550/arXiv.2204.06521.

[18] Y. Su, M. Bayoumi, T. Joachims, Optimizing rankings for recommendation in matching markets, in: WWW, 2022. doi:10.1145/3485447.3511961.

[19] G. Bied, E. Perennes, V. A. Naya, P. Caillou, B. Crépon, C. Gaillac, M. Sebag, Congestion-avoiding job recommendation with optimal transport, in: FEAST workshop ECML-PKDD 2021, Bilbao, Spain, 2021.

[20] Y. Mashayekhi, B. Kang, J. Lijffijt, T. De Bie, Recon: Reducing congestion in job recommendation using optimal transport, in: RecSys, ACM, New York, NY, USA, 2023, p. 696–701. doi:10.1145/3604915.3608817.

[21] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, in: NIPS, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 2292–2300.

[22] C. Decker, E. H. Lieb, R. J. McCann, B. K. Stephens, Unique equilibria and substitution effects in a stochastic model of the marriage market, Journal of Economic Theory 148 (2013) 778–792. doi:10.1016/j.jet.2012.12.005.

[23] A. Mnih, R. R. Salakhutdinov, Probabilistic matrix factorization, in: J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems, volume 20, Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf.

[24] A. Paterek, Improving regularized singular value decomposition for collaborative filtering, Proceedings of KDD Cup and Workshop (2007).

[25] J. Neve, I. Palomares, Latent factor models and aggregation operators for collaborative filtering in reciprocal recommender systems, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 219–227. URL: https://doi.org/10.1145/3298689.3347026. doi:10.1145/3298689.3347026.

[26] M. Cuturi, L. Meng-Papaxanthos, Y. Tian, C. Bunne, G. Davis, O. Teboul, Optimal transport tools (ott): A jax toolbox for all things wasserstein (2022). doi:10.48550/arXiv.2201.12324.

[27] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs, 2018. URL: http://github.com/google/jax.

[28] M. Scetbon, M. Cuturi, G. Peyré, Low-rank sinkhorn factorization, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of *PMLR*, PMLR, 2021, pp. 9344–9354.

A. Derivation of Transferable Utility Matching

Let $\mu_{x,y}$ be a probability distribution that specifies a match between candidate x and employer y. The set of feasible matching $\mathscr{M}$ is defined under the following constraint:

$$\mathscr{M}(m,n) = \Bigg\{ \mu_{x,y} \geq 0 \mid \sum_{y \in \mathscr{Y}_0} \mu_{x,y} = m_x \, \forall x \in \mathscr{X}, \sum_{x \in \mathscr{X}_0} \mu_{x,y} = n_y \, \forall y \in \mathscr{Y} \Bigg\}, \tag{13}$$

where m_x and n_y are the normalized masses of the candidate group x and employer group y, respectively. We set the total mass of each group to a constant C.

$$\sum_{x \in \mathscr{X}} n_x = \sum_{y \in \mathscr{Y}} m_y = C. \tag{14}$$

We assume that the two random utility distributions $\mathscr{P}$ and $\mathscr{Q}$ are independent and identically distributed (i.i.d.) with a type-I extreme value distribution with a scale parameter $\beta > 0$. Equation (13) is transformed into the following equation:

$$\begin{aligned} \mu_{x,y} &= m_x \frac{\exp\left(p_{x,y} + \tau_{x,y}\right)}{\sum_{y' \in \mathscr{Y}_0} \exp\left(p_{x,y'} + \tau_{x,y'}\right)} \\ &= n_y \frac{\exp\left(q_{y,x} - \tau_{x,y}\right)}{\sum_{x' \in \mathscr{X}_0} \exp\left(q_{y,x'} - \tau_{x',y}\right)}. \end{aligned} \tag{15}$$

Let $\phi_{x,y} = p_{x,y} + q_{y,x}$ be an observable joint utility. According to [9], the optimization problem in $\tau_{x,y}$ in (15) is a dual expression of the convex (primary) optimization problem for social welfare W.

B. Algorithm of Batch and Mini-Batch IPFP

Algorithm 1 displays the algorithm used in the batch IPFP computation. Algorithm 2 displays the algorithm used in the computation of the mini-batch IPFP.

Algorithm 1 Solving Equilibrium Matching by *Batch* IPFP

Require: preference score matrices $\boldsymbol{P}, \boldsymbol{Q}$ in size $(|\mathscr{X}|, |\mathscr{Y}|)$, normalized mass vectors $\boldsymbol{m}$ of size $|\mathscr{X}|$, $\boldsymbol{n}$ of size $|\mathscr{Y}|$, scale parameter β
Require: a maximum number of iterations I
Ensure: a matrix $\boldsymbol{\mu}$ of size $(|\mathscr{X}|, |\mathscr{Y}|)$ denotes an equilibrium matching pattern.

$\boldsymbol{u} \leftarrow \boldsymbol{1}$ {size $(|\mathscr{X}|)$}
$\boldsymbol{v} \leftarrow \boldsymbol{1}$ {size $(|\mathscr{Y}|)$}
$\boldsymbol{A} \leftarrow \exp(\frac{\boldsymbol{P}+\boldsymbol{Q}}{2\beta})$
for $i = 1, ..., I$ **do**
  $\boldsymbol{s} \leftarrow \boldsymbol{A}\boldsymbol{v}/2$
  $\boldsymbol{u} \leftarrow \sqrt{\boldsymbol{s}^2 + \boldsymbol{m}} - \boldsymbol{s}$
  $\boldsymbol{s} \leftarrow \boldsymbol{A}^T\boldsymbol{u}/2$
  $\boldsymbol{v} \leftarrow \sqrt{\boldsymbol{s}^2 + \boldsymbol{n}} - \boldsymbol{s}$
  $i \leftarrow i + 1$
end for
$\boldsymbol{\mu} \leftarrow \boldsymbol{A} \odot (\boldsymbol{u} \otimes \boldsymbol{v})$
return $\boldsymbol{\mu}$

Algorithm 2 Solving Equilibrium Matching by *Mini-batch* IPFP

Require: preference factor matrix $\boldsymbol{F}, \boldsymbol{K}$, in size $(|\mathscr{X}|, D)$, $\boldsymbol{G}, \boldsymbol{L}$ in size $(|\mathscr{Y}|, D)$, normalized mass vectors $\boldsymbol{m}$ of size $|\mathscr{X}|$, $\boldsymbol{n}$ of size $|\mathscr{Y}|$, scale parameter β, number of mini-batches J_x, J_y
Require: a maximum number of iterations I
Ensure: stable factor matrices $\boldsymbol{\Psi}$ of size $(|\mathscr{X}|, 2D + 2)$ and $\boldsymbol{\Xi}$ in size $(|\mathscr{Y}|, 2D + 2)$.

$\boldsymbol{u} \leftarrow \boldsymbol{1}$ {size $(|\mathscr{X}|)$}
$\boldsymbol{v} \leftarrow \boldsymbol{1}$ {size $(|\mathscr{Y}|)$}
for $i = 1, ..., I$ **do**
  for $j = 1, ..., J_x$ **do**
    $\boldsymbol{A}_j \leftarrow \exp\left(\frac{\boldsymbol{F}_j\boldsymbol{G}^T + \boldsymbol{K}_j\boldsymbol{L}^T}{2\beta}\right)$ {size $(B, |\mathscr{Y}|)$}
    $\boldsymbol{s}_j \leftarrow \boldsymbol{A}_j\boldsymbol{v}/2$
    $\boldsymbol{u}_j \leftarrow \sqrt{\boldsymbol{m}_j + \boldsymbol{s}_j^2} - \boldsymbol{s}_j$
    $j \leftarrow j + 1$
  end for
  for $j = 1, ..., J_y$ **do**
    $(\boldsymbol{A}^T)_j \leftarrow \exp\left(\frac{\boldsymbol{F}\boldsymbol{G}_j^T + \boldsymbol{K}\boldsymbol{L}_j^T}{2\beta}\right)$ {size $(B, |\mathscr{X}|)$}
    $\boldsymbol{s}_j \leftarrow (\boldsymbol{A}^T)_j\boldsymbol{u}/2$
    $\boldsymbol{v}_j \leftarrow \sqrt{\boldsymbol{n}_j + \boldsymbol{s}_j^2} - \boldsymbol{s}_j$
    $j \leftarrow j + 1$
  end for
  $i \leftarrow i + 1$
end for
$\boldsymbol{\Psi} \leftarrow \text{Concat}\left(\boldsymbol{F}, \boldsymbol{K}, \beta \log\left(\boldsymbol{u}\right), 1\right)$
$\boldsymbol{\Xi} \leftarrow \text{Concat}\left(\boldsymbol{G}, \boldsymbol{L}, 1, \beta \log\left(\boldsymbol{v}\right)\right)$
return $\boldsymbol{\Psi}, \boldsymbol{\Xi}$