Optimizing Matching Markets with Graph Neural Networks and Reinforcement Learning

Satoshi Waki, Toyotaro Suzumura, Hiroki Kanezashi

The University of Tokyo
waki-satoshi508@g.ecc.u-tokyo.ac.jp, {suzumura, hkanezashi}@ds.itc.u-tokyo.ac.jp

Abstract

This study introduces a novel recommendation system designed for matching markets, such as job placement and online dating, which goes beyond the traditional focus on individual user preferences. Traditional Reciprocal Recommendation Systems in these markets often fail to consider the overall market dynamics, leading to a narrow focus on specific popular choices and neglecting the diversity of user needs. To address this, our approach conceptualizes the market as a network, utilizing Graph Neural Networks to analyze the intricate connections within this network. We also incorporate Reinforcement Learning to optimize outcomes for the entire market, not just individual users. Furthermore, to address the issue of sparse user-item interactions in matching markets, our approach incorporates a novel graph data augmentation technique. This method enriches the network by adding labeled edges, enhancing the market's representation. This augmentation facilitates more effective and varied recommendations, leading to a noticeable increase in successful matches in various market scenarios, as evidenced by our offline experiments with both synthetic and real-world data.

Introduction

In the contemporary digital landscape, Recommendation Systems (RSs) are integral, enhancing personalized experiences in areas such as online retail, entertainment, and social networking(Schafer, Konstan, and Riedl 1999; Sivapalan et al. 2014; Ben-Shimon et al. 2015; Barragáns-Martínez et al. 2010; Gomez-Uribe and Hunt 2016; Alvarado et al. 2020). A specialized category within these, known as Reciprocal Recommendation Systems (RRSs), is gaining prominence in matching markets(Hu et al. 2023; Tomita et al. 2023; Su, Bayoumi, and Joachims 2022; Borisyuk, Zhang, and Kenthapadi 2017). These markets, such as job placement and online dating, are based on the concept of pairing individuals or entities with matching requirements and preferences. RRSs play a crucial role here, aligning the preferences and requirements of both parties to facilitate optimal, mutually beneficial matches. For example, in job markets, RRSs not only help applicants find fitting job opportunities but also ensure that these opportunities align with the qualifications and expectations of employers.

Current RRSs are typically designed to match a single user with a multitude of jobs or items, focusing on finding the best fit for that particular user. This approach, while effective for individual needs, often neglects the broader dynamics of the market and the diverse needs of other users.In the realm of job placement, as an example, this can lead to a disproportionate emphasis on certain popular positions, thereby limiting the diversity of hiring opportunities. An approach that considers the market as an integrated entity can not only connect users to suitable jobs but also enhance the overall balance and diversity of the market, thereby improving its health and efficiency.

To improve RRSs in matching markets, this paper introduces an innovative method that transcends traditional single-user-focused approaches. We conceptualize the market as a network, treating users and items as nodes within a graph. This network is analyzed using Graph Neural Networks (GNNs), which are adept at capturing the intricate interconnections among these nodes. Additionally, we incorporate Reinforcement Learning (RL) to optimize market-wide outcomes over time. Moreover, our study addresses the common issue of sparse interactions between users and items in matching markets, particularly in job placements. To overcome this challenge, we have developed a novel graph data augmentation (GDA) technique that, unlike traditional GDA methods that focus on adding only one type of edge (Ling et al. 2023; Zhao et al. 2022a,b; Mu et al. 2022), predicts various types of potential interactions between users and items and adds labeled edges to the graph. The resulting augmented graph offers a more detailed representation of the market, enabling our system to provide more accurate recommendations. This approach ensures optimal individual matches in a sparse market while maintaining overall market equilibrium.

In a nutshell, our contributions are as follows:

- We present an optimized market-wide recommendation strategy, leveraging GNNs and RL, to improve matching outcomes throughout the market. This strategy effectively counters popularity bias and fosters market equilibrium by increasing the long-term success rate of matches.
- To combat the prevalent issue of sparse interactions in matching markets, we introduce an innovative GDA technique. This method enhances the market's graph repre-

sentation by predicting potential interactions and adding labeled edges, thereby enriching the recommendation process.

- Our empirical evaluations, utilizing both synthetic and real-world data, demonstrate that our approach significantly enhances the total number of successful matches across various market settings.

Preliminary

Understanding Job Recommendation Systems

Job recommendation systems, a specialized segment within the broader domain of recommendation systems, play a pivotal role in the labor market by bridging the gap between job seekers (users) and employment opportunities (items). These systems are designed to recommend jobs to job seekers based on various criteria such as skills, experience, and preferences. The process begins with the system recommending jobs to job seekers. When a job aligns with a job seeker's interests and qualifications, they may apply for it. This action is analogous to making a purchase in e-commerce or selecting a movie in streaming services. However, a distinctive feature in the job market is the subsequent phase where the employer assesses the candidate's application. They evaluate the applicant's fit for the role based on their aptitude and skills, determining whether to proceed with the hiring process.

The primary objective of a job recommendation system is to maximize successful employment matches (hires), paralleling the goal in e-commerce systems, which is to maximize purchases and sales. This goal underlines the need for a system that accurately understands and aligns the preferences and qualifications of job seekers with the requirements and cultures of employers, thereby facilitating optimal job placements.

In the following, unless misunderstood, job seekers are simply referred to as users, employment opportunities as jobs, and being hired for a job as a match.

Two-Tower Models for Recommendation Systems

Two-tower models, widely used in RSs (Tomita et al. 2023; Rendle et al. 2020), process users, denoted as u, and items, denoted as i, independently through their respective encoders, f_{θ_u} and f_{θ_i}. The user and item embeddings are then computed as $\boldsymbol{e}_u = f_{\theta_u}(u)$ and $\boldsymbol{e}_i = f_{\theta_i}(i)$. These embeddings are placed in a common vector space. The learning objective of these models is to align the embeddings such that their inner product or cosine similarity corresponds to the likelihood of user interaction with the item, like clicking or purchasing. By achieving this, the systems can effectively recommend items that align with the users' interests.

Advanced Embedding Representations with Graph Neural Networks

Two-tower models encode user and item embeddings independently, so they cannot capture complex relationships between users and items. Therefore, GNNs can be applied to bipartite graphs of users and items, or knowledge graphs with item attributes and external knowledge, to obtain richer embedding representations. In particular, Relational Graph Convolutional Networks (R-GCNs) (Schlichtkrull et al. 2018) can calculate the embeddings of vertices considering multiple edge types. Specifically, for a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ defined by vertices $v_i \in \mathcal{V}$, edge types $r \in \mathcal{R}$, and edges $(v_i, r, v_j) \in \mathcal{E}$, the hidden state of the node v_i in the $l+1$th layer of the neural network is calculated as follows:

$$\boldsymbol{h}_i^{(l+1)} = g\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\boldsymbol{W}_r^{(l)}\boldsymbol{h}_j^{(l)} + \boldsymbol{W}_0^{(l)}\boldsymbol{h}_i^{(l)}\right) \quad (1)$$

where g is an activation function, $\mathcal{N}_i^r$ is the set of neighbor indices of node v_i under relation $r \in \mathcal{R}$, $c_{i,r}$ is the normalization constant, $\boldsymbol{W}_r^{(l)}$ and $\boldsymbol{W}_0^{(l)}$ are trainable variables. Finally, the embeddings of the vertex v_i are obtained by concatenating the hidden states up to layer L:

$$\boldsymbol{e}_i = \boldsymbol{h}_i^{(0)} \parallel \boldsymbol{h}_i^{(1)} \parallel \cdots \parallel \boldsymbol{h}_i^{(L)} . \quad (2)$$

Reinforcement Learning for Recommendation Systems

To achieve effective RSs, it is important to consider not only short-term metrics, but also long-term metrics. For example, when a new item is added to the market, there may be little data about the item initially, so it may be more effective to recommend other well-known items from a short-term perspective. However, in the long term, as more data about the new item is accumulated, recommending it may be a more valuable choice for users. Thus, it is important to strike a balance between short-term results and long-term strategies in building high-quality RSs.

Reinforcement learning is particularly useful for focusing on these long-term metrics. In RL, a 'learning agent' – in our case, the RS – interacts with an environment made up of users and employers. The RS learns from the feedback it receives after making recommendations. The main goal here is to learn strategies that ensure long-term success, not just immediate rewards.

To apply RL, the environment must be formulated as a Markov Decision Process (MDP) consisting of four elements $(\mathcal{S}, \mathcal{A}, T, R)$ (Bellman 1957). Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the state transition function, and R is the reward function. In RL given an MDP, the agent learns a policy $\pi(a_t \mid s_t)$ that maximizes the discounted reward sum defined as $\sum_{t=0}^{\infty}\gamma^t r_t$, where r_t is the reward at time t, and $\gamma \in [0, 1]$ is the discount rate.

When the policy is represented as a function of parameters $\boldsymbol{\theta}$, the optimal policy can be learned by updating the parameters based on the gradient of the expected discounted reward sum with respect to $\boldsymbol{\theta}$, as shown in Equation (3). In addition, a method called actor-critic (Konda and Tsitsiklis 1999) approximates the gradient efficiently by approximat-

ing it as shown in Equation (4):

$$J(\boldsymbol{\theta}) = \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi_{\theta_i}\right] \tag{3}$$

$$\simeq \sum_{t=0}^{\infty} \nabla_\theta \log \pi_{\theta_i}(a_t \mid s_t)\left(r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t)\right) \tag{4}$$

where $V_\omega(s_t)$ is the state-value function, and $V_\omega(s_t)$ is learned to approach $r_t + \gamma V_\omega(s_{t+1})$.

Proposed Method

Reinforcement Learning-Based Recommendation System Utilizing Graph Neural Networks

In this study, we introduce a recommendation system enhanced by RL and GNNs. This system models its policy using GNNs, allowing it to account for the interactions between users and items. We build a bipartite graph, with users and items as nodes, and use GNNs to generate their embeddings. The system then ranks recommendations based on the inner product of user and job embeddings, denoted as $\langle \boldsymbol{e}_u, \boldsymbol{e}_i \rangle$. The recommendation system receives various feedback from the environment, such as job applications and acceptances, as well as disparities in job popularity. By designing rewards based on these feedbacks, we can realize the desired recommendation system. We design rewards considering both short-term and long-term gains; short-term rewards are given for job applications, and long-term rewards are assigned upon successful job matches. Our strategy aims to balance personalization with market needs, recommending jobs that not only attract immediate interest but also contribute to maximizing successful matches in the market over time.

Furthermore, in our Actor-Critic model, we employ a Multilayer Perceptron (MLP) for the state-value function. This MLP computes the state value by taking the average embedding vectors of both user and item nodes, represented as $\overline{\boldsymbol{e}}_u$ and $\overline{\boldsymbol{e}}_i$, respectively. The state value calculation for the entire graph is as follows:

$$V_\omega(s_t) = \mathrm{MLP}_\omega\left(\overline{\boldsymbol{e}}_u \parallel \overline{\boldsymbol{e}}_i\right). \tag{5}$$

Here, $\overline{\boldsymbol{e}}_u \parallel \overline{\boldsymbol{e}}_i$ represents the concatenation of the average embedding vectors of user and item nodes.

Graph Data Augmentation Technique for labeled edges

In RSs, the scarcity of interactions between users and items often poses a problem. For instance, in a job recommendation scenario, applications and hirings might be less frequent compared to views or clicks in an online retail setting, exacerbating the challenge of data sparsity. In GNNs, the embedding of a vertex is calculated based on the hidden states of connected vertices; thus, pronounced sparsity can degrade the quality of embeddings. To mitigate this, we propose a GDA technique that adds edges between users and jobs. By adding these edges, we enhance the embeddings for both users and jobs. This GDA technique allows us to alleviate sparsity issues. Specifically, for a user vertex u and an item vertex i, we concatenate the embeddings $\boldsymbol{e}_u$ and $\boldsymbol{e}_i$ from the learned Two-Tower models and a one-hot vector $\boldsymbol{e}_r$ representing the edge type between vertices u and i, and pass them through an MLP and a sigmoid function σ to calculate the probability p_{ui} of adding an edge

$$\boldsymbol{p}_{ui} = \sigma\left(\mathrm{MLP}_A\left(\boldsymbol{e}_u \parallel \boldsymbol{e}_i \parallel \boldsymbol{e}_r\right)\right). \tag{6}$$

The decision to include or exclude an edge is determined by sampling from a Bernoulli distribution, based on the probability $\boldsymbol{p}_{ui}$. By increasing the output dimension of the MLP, it is possible to add multiple types of edges. That is, the u, i components of the adjacency matrix for edge type r are as follows:

$$A_{ui}^{(r)} \sim \text{Bernoulli}\left(\boldsymbol{p}_{ui}^{(r)}\right). \tag{7}$$

Here, we utilize the Gumbel-Softmax reparameterization trick (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 2017; Ling et al. 2023; Maddison, Tarlow, and Minka 2014), a method for sampling from categorical distributions, to ensure differentiability in our model.

See Figure 1 for an overview of our proposed method.

Experiments

This section introduces real-world datasets, synthetic data generation procedures, and baselines we used for comparison.

Experimental Settings

Description of Datasets Our offline experiments utilized two datasets.

The first dataset, referred to as the SMS dataset, was provided by SMS Co., Ltd., a company operating a nursing care recruiting site in Japan. This dataset, which is a proprietary dataset not publicly available, includes 1,250,581 users and 96,465 jobs, along with both application and match histories. Using Two-Tower models, we trained MLPs to estimate application and match probabilities, considering probabilities above 0.5 as positive indications.

The second, the CareerBuilder2012 dataset[1], was sourced from the CareerBuilder 2012 Job Recommendation Challenge (Ben Hamner and Krupa 2012) hosted on Kaggle. It comprises 389,708 users, 1,091,923 jobs, and 1,603,111 application histories. However, this dataset lacks match (employment) history data. To address this, we trained a MLP using the Two-Tower models, estimating application probabilities from the application history. Applications were considered made if the probability exceeded 0.5, and employments were assumed if the probability exceeded 0.8. This threshold value of 0.8 was set so that the Maximum #Matches value of the first private dataset was close. Maximum #Matches is the maximum number of matches that can actually be observed.

[1] https://kaggle.com/competitions/job-recommendation

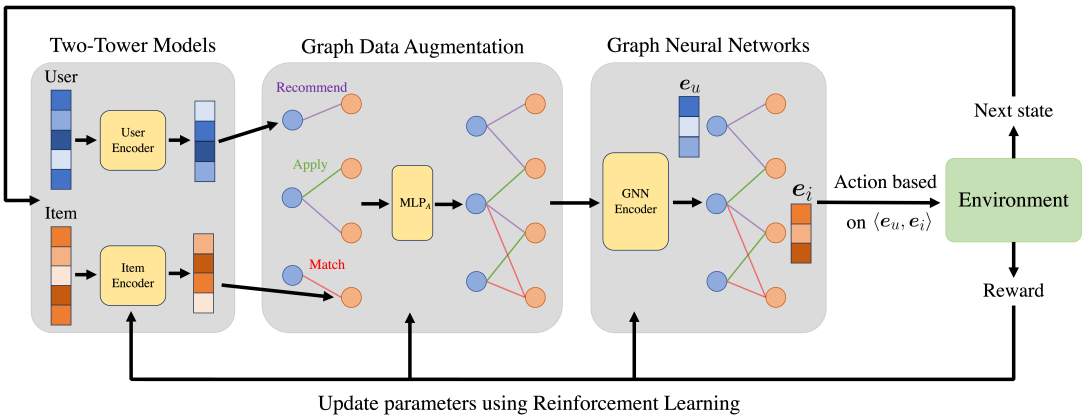

Figure 1: An overview of our framework.

	SMS	CareerBuilder2012	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Synthetic 5
ApplyPredictor	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
MatchPredictor	0.00 ± 0.00	0.00 ± 0.00	0.33 ± 0.47	0.33 ± 0.47	0.00 ± 0.00	0.00 ± 0.00	5.00 ± 0.00
LiJAR	1.33 ± 0.47	0.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	8.00 ± 0.00	16.67 ± 0.47	20.33 ± 0.94
StableMatching	18.00 ± 2.94	4.33 ± 1.25	3.33 ± 1.25	9.00 ± 0.00	11.33 ± 4.11	16.33 ± 1.70	20.67 ± 3.86
Ours w/o GNNs	1.00 ± 0.00	5.33 ± 1.89	4.33 ± 1.89	15.67 ± 0.47	26.00 ± 0.00	28.67 ± 0.47	55.33 ± 1.25
Ours w/o GDA	19.33 ± 1.70	6.00 ± 0.00	13.00 ± 1.41	18.33 ± 3.68	31.33 ± 3.40	37.67 ± 0.47	64.00 ± 1.41
Ours	$\mathbf{23.00} \pm 3.56$	$\mathbf{10.00} \pm 2.94$	$\mathbf{23.33} \pm 2.87$	$\mathbf{31.67} \pm 4.99$	$\mathbf{50.67} \pm 1.89$	$\mathbf{62.33} \pm 3.09$	$\mathbf{71.67} \pm 1.70$

Table 1: Comparisons between our method and baselines on the final total number of matches obtained in the test environment simulations. The best results are shown in bold.

Synthetic Data To complement our real-world data and test our model in various scenarios, we generated synthetic data. User features were sampled from a uniform distribution $\mathcal{U}^d(0,1)$, and item features from $\mathcal{U}^d(0, b(i))$. We introduced popularity bias in items by varying the upper limit $b(i)$ for each item. The details of the function b that determines the popularity bias and other experimental settings are written in Appendix. Five different b were used to generate synthetic data with various popular characteristics. Summaries for each environment are also in the Appendix.

Baseline Methods To evaluate the performance of the proposed method, several methods were chosen as baselines. In existing research, various innovations have been made to improve accuracy, such as the use of Bayesian estimation (Borisyuk, Zhang, and Kenthapadi 2017) and multitask learning(Hu et al. 2023). However, to measure the effect of using RL and GNNs, the models in the comparative methods were designed to be relatively simple, using MLPs and trained through supervised learning.

- ApplyPredictor: Predicts application probability among user-item pairs, akin to general RSs in e-commerce.
- MatchPredictor: Forecasts the matching probability between user-item pairs, commonly used in RRSs.
- LiJAR: Estimates application probability, implementing penalties or bonuses to ensure equitable application distribution (Borisyuk, Zhang, and Kenthapadi 2017).
- StableMatching: Predicts both application and matching probabilities, applying the Gale-Shapley algorithm (Gale and Shapley 1962) to find stable matching(Bills and Ng 2021).

Experimental Results

Table 1 presents the final total number of matches obtained in the test environment simulations. The results demonstrate our method's superiority over baselines in all datasets and synthetic environments. In our experiments, we compare different configurations of our model. 'Ours w/o GNNs' and 'Ours w/o GDA' represent our model without GNNs and GDA, respectively. A comparison between 'Ours w/o GNNs' and MatchPredictor reveals a significant improvement in the total number of matches, averaging 24.06 times higher, which highlights the importance of considering long-term rewards. Additionally, comparing 'Ours w/o GNNs' with 'Ours w/o GDA', there's an average improvement of 1.75 times in the total number of matches, while a comparison between Ours and 'Ours w/o GDA' shows an average improvement of 1.14 times. These results underscore the importance of using GNNs and GDA to consider the overall market information. Moreover, comparing the proposed method with LiJAR and StableMatching, the models applying only RL performed poorly in several environments, while the introduction of GNNs outperformed them in all environments.

Conclusion

In this study, we proposed a recommendation system for matching markets that considers information from the entire market. Our approach involved a model combining graph neural networks and graph data augmentation methods to add labeled edges, which was trained using reinforcement learning to optimize long-term market-wide evaluation metrics. Experiments conducted in various environments using real-world datasets and synthetic data achieved state-of-the-art performance in all settings. This research demonstrates that considering the entire market's information using graph neural networks and reinforcement learning can significantly improve performance, especially in tasks like matching markets that require long-term and holistic optimization. Additionally, our work addresses the issue of data sparsity commonly faced in recommendation tasks, showing that our Graph Data Augmentation method can mitigate this sparsity. The combination of more sophisticated reinforcement learning methods and graph neural networks models holds the potential for further improvements in recommendation systems, presenting a valuable future direction for the research community.

Acknowledgement

This work is supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN)" and "High Performance Computing Infrastructure (HPCI)" in Japan (Project ID: jh221002).

In addition, this work was supported by JSPS KAKENHI Grant Numbers JP21K17749, JP23H03408.

We would also like to express our sincere gratitude to SMS Co., Ltd. (SMS) for providing the data essential for our study.

References

Alvarado, O.; Heuer, H.; Vanden Abeele, V.; Breiter, A.; and Verbert, K. 2020. Middle-Aged Video Consumers' Beliefs About Algorithmic Recommendations on YouTube. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW2).

Barragáns-Martínez, A. B.; Costa-Montenegro, E.; Burguillo, J. C.; Rey-López, M.; Mikic-Fonte, F. A.; and Peleteiro-Ramallo, A. 2010. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Inf. Sci.*, 180: 4290–4311.

Bellman, R. 1957. A Markovian Decision Process. *Indiana University Mathematics Journal*, 6: 679–684.

Ben Hamner, R. W.; and Krupa, W. 2012. Job Recommendation Challenge. https://kaggle.com/competitions/job-recommendation. Accessed: 2023-12-01.

Ben-Shimon, D.; Tsikinovsky, A.; Friedmann, M.; Shapira, B.; Rokach, L.; and Hoerle, J. 2015. RecSys Challenge 2015 and the YOOCHOOSE Dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, 357–358. New York, NY, USA: Association for Computing Machinery. ISBN 9781450336925.

Bills, J.; and Ng, Y.-k. D. 2021. Looking for Jobs? Matching Adults with Autism with Potential Employers for Job Opportunities. In *Proceedings of the 25th International Database Engineering; Applications Symposium*, IDEAS '21, 212–221. New York, NY, USA: Association for Computing Machinery. ISBN 9781450389914.

Borisyuk, F.; Zhang, L.; and Kenthapadi, K. 2017. LiJAR: A System for Job Application Redistribution towards Efficient Career Marketplace. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, 1397–1406. New York, NY, USA: Association for Computing Machinery. ISBN 9781450348874.

Gale, D.; and Shapley, L. S. 1962. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1): 9–15.

Gomez-Uribe, C. A.; and Hunt, N. 2016. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4).

Hu, X.; Cheng, Y.; Zheng, Z.; Wang, Y.; Chi, X.; and Zhu, H. 2023. BOSS: A Bilateral Occupational-Suitability-Aware Recommender System for Online Recruitment. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, 4146–4155. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701030.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Konda, V.; and Tsitsiklis, J. 1999. Actor-Critic Algorithms. In Solla, S.; Leen, T.; and Müller, K., eds., *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Ling, H.; Jiang, Z.; Luo, Y.; Ji, S.; and Zou, N. 2023. Learning Fair Graph Representations via Automated Data Augmentations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Maddison, C. J.; Tarlow, D.; and Minka, T. 2014. A$*$ Sampling. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Mu, S.; Li, Y.; Zhao, W. X.; Wang, J.; Ding, B.; and Wen, J.-R. 2022. Alleviating Spurious Correlations in Knowledge-Aware Recommendations through Counterfactual Generator. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, 1401–1411. New York, NY, USA: Association for Computing Machinery. ISBN 9781450387323.

Rendle, S.; Krichene, W.; Zhang, L.; and Anderson, J. R. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In Santos, R. L. T.; Marinho, L. B.; Daly, E. M.; Chen, L.; Falk, K.; Koenigstein, N.; and de Moura, E. S., eds., *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, 240–248. ACM.

Schafer, J. B.; Konstan, J.; and Riedl, J. 1999. Recommender Systems in E-Commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, 158–166. New York, NY, USA: Association for Computing Machinery. ISBN 1581131763.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, 593–607. Springer.

Sivapalan, S.; Sadeghian, A.; Rahnama, H.; and Madni, A. M. 2014. Recommender systems in e-commerce. In *2014 World Automation Congress (WAC)*, 179–184.

Su, Y.; Bayoumi, M.; and Joachims, T. 2022. Optimizing Rankings for Recommendation in Matching Markets. In *Proceedings of the ACM Web Conference 2022*, WWW '22, 328–338. New York, NY, USA: Association for Computing Machinery. ISBN 9781450390965.

Tomita, Y.; Togashi, R.; Hashizume, Y.; and Ohsaka, N. 2023. Fast and Examination-Agnostic Reciprocal Recommendation in Matching Markets. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, 12–23. New York, NY, USA: Association for Computing Machinery. ISBN 9798400702419.

Zhao, T.; Liu, G.; Wang, D.; Yu, W.; and Jiang, M. 2022a. Learning from Counterfactual Links for Link Prediction. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 26911–26926. PMLR.

Zhao, T.; Tang, X.; Zhang, D.; Jiang, H.; Rao, N.; Song, Y.; Agrawal, P.; Subbian, K.; Yin, B.; and Jiang, M. 2022b. AutoGDA: Automated Graph Data Augmentation for Node Classification. In Rieck, B.; and Pascanu, R., eds., *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, 32:1–32:17. PMLR.

Detailed Experimental Settings

Popularity Bias Function Used in Synthetic Data Generation

The $b(i_k)$ for item i_k is set using two parameters, $imbW$ and $imbLen$, as follows:

$$b(i_k) = \exp\left(imbW \cdot \frac{\max\{0, imbLen - k\}}{imbLen}\right). \quad (8)$$

These parameters control the popularity gap and the length of the distribution tail. Applications and employments were determined based on the sigmoid function of the inner product of user and item features.

For user features u and item features i generated from each uniform distribution, an application was considered made if the sigmoid function of their inner product $\sigma(\langle u, i\rangle)$ exceeded 0.8, and an employment was assumed if it exceeded 0.9.

MDP Experimental Settings

For MDP experiments, users and items were divided into disjoint sets of 300 each for training, evaluation, and testing. Users to recommend at each timestep were randomly selected, with the simulation ending at 100 timesteps.

Other Experimental Settings

The policy adopted an epsilon-greedy approach, where a random item is recommended with a probability that decays with each episode. In the training environment, simulations were run for 100 timesteps, and based on this data, the parameters for the policy and the state-value function were updated. Subsequently, in the evaluation environment, simulations were conducted for 100 timesteps three times. The model with the highest average reward sum across the three simulations was then tested in the test environment for 100 timesteps, three times. The average total number of matches obtained in these simulations was considered the final evaluation value.

Experiment Summary

In Table 2, the values of $imbW$ and $imbLen$ used for generating synthetic data, as well as statistical values for all test environments including real data, are presented. For example, the test environment created from the CareerBuilder2012 dataset has 300 users and 300 jobs each. Furthermore, out of the total 90,000 pairs between users and jobs, 25,873 pairs are in an application relationship, and 3,776 are in a matching relationship. However, since the number of recruitments per job is set to one, the maximum number of matches that can actually be observed is 101. In the experiment, recommendations are made to one user at each time step, and the simulation is conducted for only 100 time steps, so the maximum number of matches that can be obtained as an experimental indicator is 100 in the best-case scenario. Additionally, when calculating the Gini coefficient for the disparity in popularity among jobs, the Gini coefficient at the time of application is 0.27, whereas the Gini coefficient for matches is 0.74, indicating that considering matches, the environment has a large disparity in popularity.

Evaluation on Other Metrics

While our study's primary objective is to maximize the final total number of matches, indicating a successful matchmaking process, we also examine secondary metrics. It's crucial to understand that these metrics, while informative, involve trade-offs and may not directly contribute to our primary goal.

We used the total number of applications and the entropy of the number of applications between jobs to evaluate the

	SMS	CareerBuilder2012	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Synthetic 5
$imbW$	-	-	0.1	1.0	1.0	1.0	2.0
$imbLen$	-	-	100	50	100	150	100
#Users	300	300	300	300	300	300	300
#Jobs	300	300	300	300	300	300	300
#Applications	36,746	25,873	34,697	39,176	44,894	51,017	49,694
#Matchings	14,784	3,776	4,359	8,682	13,623	19,004	22,119
Maximum #Matchings	126	101	115	134	126	184	173
Apply Gini Coefficient	0.23	0.27	0.41	0.40	0.36	0.31	0.35
Match Gini Coefficient	0.70	0.74	0.79	0.80	0.75	0.68	0.71

Table 2: Summary of statistics for environments.

results. The performance is better when both the total number of applications and the entropy are larger; a larger total number of applications means that the user is able to recommend jobs of interest, which indicates the performance of individual optimization. Also, the larger the entropy, the better the system is able to recommend a variety of jobs, which is the performance of market equilibrium and fairness of the recommendation system.

Table 3 shows the results for the total number of applications and Table 4 shows the results for entropy. Comparing the results in terms of the total number of applications, the proposed method recorded the highest values in three environments, LiJAR in one environment, and ApplyPredictor in four environments. Therefore, from the standpoint of individual optimization, ApplyPredictor has the best performance, followed by the proposed method. In terms of entropy, the proposed method and Stable matching recorded the highest entropy values in three and four environments, respectively. Therefore, in terms of market equilibrium, Stable matching has the best performance, followed by the proposed method.

These results indicate that the proposed method is able to maximize the final total number of matches by successfully balancing the trade-off relationship between individual optimization and market equilibrium.

	SMS	CareerBuilder2012	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Synthetic 5
ApplyPredictor	25.33 ± 3.09	12.33 ± 2.49	68.00 ± 2.16	$\mathbf{99.00} \pm 0.82$	$\mathbf{99.00} \pm 0.82$	$\mathbf{99.00} \pm 0.82$	$\mathbf{100.00} \pm 0.00$
MatchPredictor	33.33 ± 5.19	7.67 ± 1.70	12.00 ± 2.94	35.67 ± 6.55	5.00 ± 0.82	24.67 ± 4.50	97.00 ± 0.82
LiJAR	23.00 ± 4.08	17.33 ± 6.55	44.67 ± 0.47	96.67 ± 1.25	97.33 ± 0.47	98.00 ± 0.82	$\mathbf{100.00} \pm 0.00$
StableMatching	40.33 ± 1.70	25.33 ± 4.19	30.67 ± 2.87	36.67 ± 3.68	38.00 ± 2.94	48.00 ± 2.83	47.33 ± 3.68
Ours w/o GNNs	34.33 ± 3.40	$\mathbf{26.67} \pm 1.25$	47.67 ± 4.99	54.00 ± 0.82	49.33 ± 8.01	64.00 ± 5.89	82.67 ± 0.47
Ours w/o GDA	30.00 ± 4.32	20.00 ± 2.94	65.67 ± 3.30	66.33 ± 4.50	71.67 ± 3.86	78.00 ± 3.74	88.00 ± 1.63
Ours	$\mathbf{43.67} \pm 4.99$	14.00 ± 2.45	$\mathbf{78.67} \pm 2.36$	84.67 ± 3.40	87.67 ± 0.47	90.33 ± 2.05	93.33 ± 0.47

Table 3: Comparisons between our method and baselines on the final total number of applications obtained in the test environment simulations. Results that indicate the highest values are shown in bold.

	SMS	CareerBuilder2012	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Synthetic 5
ApplyPredictor	2.11 ± 0.10	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
MatchPredictor	1.17 ± 0.12	0.00 ± 0.00	0.69 ± 0.24	0.00 ± 0.00	0.22 ± 0.31	0.18 ± 0.13	1.18 ± 0.16
LiJAR	3.68 ± 0.28	3.28 ± 0.73	3.47 ± 0.05	2.06 ± 0.01	2.43 ± 0.02	3.01 ± 0.03	3.38 ± 0.02
StableMatching	$\mathbf{5.19} \pm 0.08$	$\mathbf{4.57} \pm 0.22$	$\mathbf{4.76} \pm 0.15$	$\mathbf{5.07} \pm 0.20$	5.10 ± 0.16	5.42 ± 0.04	5.44 ± 0.11
Ours w/o GNNs	1.22 ± 0.15	2.70 ± 0.44	2.07 ± 0.38	3.00 ± 0.09	4.38 ± 0.31	4.33 ± 0.04	5.41 ± 0.02
Ours w/o GDA	3.87 ± 0.21	2.66 ± 0.18	3.33 ± 0.15	3.53 ± 0.58	4.52 ± 0.24	4.64 ± 0.18	$\mathbf{6.18} \pm 0.05$
Ours	3.50 ± 0.37	3.20 ± 0.41	4.36 ± 0.19	4.41 ± 0.36	$\mathbf{5.35} \pm 0.07$	$\mathbf{5.71} \pm 0.12$	6.01 ± 0.05

Table 4: Comparisons between our method and baselines on the entropy of the number of applications between jobs obtained in the test environment simulations. Results that indicate the highest values are shown in bold.