Towards Capacity-Aware Broker Matching: From Recommendation to Assignment

Shuyue Wei[1], Yongxin Tong[1], Zimu Zhou[2], Qiaoyang Liu[1], Lulu Zhang[3], Yuxiang Zeng[4], Jieping Ye[3]
[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
[2] City University of Hong Kong, Hong Kong, China [3] Ke Holdings Inc., Beijing, China
[4] The Hong Kong University of Science and Technology, Hong Kong SAR, China
[1] {weishuyue, yxtong, qiaoyangliu}@buaa.edu.cn, [2] zimuzhou@cityu.edu.hk
[3] {zhanglulu026, yejieping}@ke.com, [4] yzengal@cse.ust.hk

***Abstract*— Online real estate platforms are gaining increasing popularity, where a central issue is to match brokers with clients for potential housing transactions. Mainstream platforms match brokers via top-*k* recommendation. Yet we observe through extensive data analysis that such top-*k* recommendation tends to overload the top brokers, which notably degrades their service quality. In this paper, we propose to avoid such overloading in broker matching via the paradigm shift from recommendation to assignment. To this end, we design learned assignment with contextual bandits (LACB), a data-driven capacity-aware assignment scheme for broker matching which estimates broker-specific workload capacity in an online fashion and assigns brokers to clients from a global perspective to maximize the overall service quality. Extensive evaluations on synthetic and real-world datasets from an industrial online real estate platform validate the efficiency and effectiveness of our solution.**

***Index Terms*—Capacity-Aware, Broker Matching, Real Estate Platform**

I. Introduction

Online real estate platforms, such as Compass[1], Zillow[2] and Ke Holdings Inc. (a.k.a Beike)[3] are increasingly exploiting data-driven approaches to improve their business and service quality. A central function of these platforms is to match clients who are interested in house purchases to appropriate brokers[4] for followup services [1]. The status quo of such broker matching is top-*k* recommendation [2], [3]. Take Beike, the largest Chinese online real estate platform, as an example. When clients click for detailed information about a house on the platform app, three brokers associated with that house will be recommended by the App (see Fig. 1).

We observe, through extensive data analysis on online real-estate platforms, that *(i)* brokers have limited workload capacity and *(ii)* the top-*k* recommendation mechanism leads to the *overloaded of top brokers*, also known as the *overloaded phenomenon*, which impairs both the service quality and the long-term development of the platform. Specifically, our study shows that due to the top-*k* mechanism, the brokers' sign-up rate can drop from 14.3%∼27.5% to 2.5%∼17.8%, if they respond to over 40 client requests per day (see Sec. II-B). Here

[1]https://www.compass.com/
[2]https://www.zillow.com
[3]https://www.ke.com
[4]We do not differ an agent from a broker, and collectively refer them as brokers.

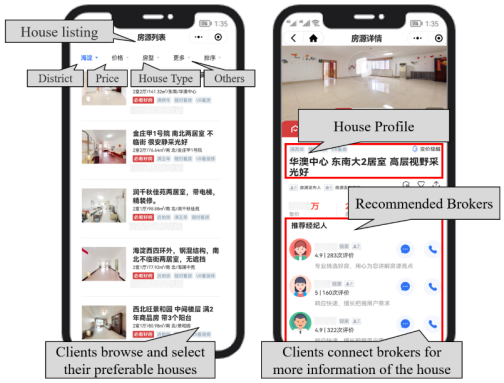


Fig. 1: An illustration of top-*k* broker recommendation on Beike, a major online real estate platform. When house buyers click for more information about their favorite houses, the platform App recommends three brokers by default.

the sign-up rate of a broker is a common indicator of service quality, which is the ratio between the number of clients who sign up with him/her and the total clients he/she served. We also observe the Matthew effect [4] when adopting the top-*k* recommendation mechanism. That is, many requests are occupied by top brokers, leaving others few opportunities to improve their skills. It may discourage those neglected brokers and harm the platform in the long run.

We argue that the overloaded phenomenon is caused by ignorance of brokers' workload capacity, which motivates us to take an *assignment* [5]–[7] perspective for capacity-aware broker matching. That is, rather than blindly recommend the few top brokers to all clients, we propose to first estimate the workload capacity of individual brokers, and then assign them to clients in a global view without overwhelming the brokers. However, it faces two practical challenges to implement capacity-aware assignment for broker matching.

- *Challenge 1: how to estimate broker-specific workload capacity in an online fashion?* We observe that the workload capacity differs across brokers (see Sec. II), making personalized estimation necessary. However, it is impractical to collect data on a broker's service quality under all possible workloads in advance, which makes online estimation of workload capacity preferable. Prior workload capacity estimation schemes [8], [9] fail to

support such online learning of personalized estimation.

- *Challenge 2: how to assign brokers under capacity limit to maximize the overall utility over time?* It is common that the amount of real estate transactions at present affects that in the near future. Consequently, broker assignments among batches tend to be correlated, making it difficult to assign brokers holistically. Most assignment schemes [10], [11] consider clients and brokers in each batch independently, making them sub-optimal in terms of the collective utility of multiple batches.

To address these challenges, we propose Learned Assignment with Contextual Bandits (LACB), a data-driven capacity-aware assignment scheme for real estate broker matching. It tackles *Challenge 1* via contextual bandits for data-efficient and online personalized capacity estimation. LACB overcomes *Challenge 2* with a capacity-aware value function, which accounts for both the short-term and the long-term utility of brokers for matching. Our main contributions and results are summarized as follows.

- To the best of our knowledge, we are the first to identify the overload of top brokers problem for online real estate platforms. Extensive data analysis shows that brokers have limited workload capacity and their service quality tends to drop when overloaded, which motivates the shift from recommendation to assignment for broker matching.
- We design LACB, a data-driven capacity-aware assignment scheme for broker matching. It estimates broker-specific capacity in an online manner and assigns brokers to clients from a global view. We further propose LACB-Opt, which accelerates assignments via broker selection.
- We conduct extensive experiments on synthetic and real-world datasets from Beike, the largest Chinese online real estate platform. Experimental results validate the efficiency and effectiveness of our solutions.

In the rest of this paper, we first identify the overloaded phenomenon in Sec. II and formulate the problem in Sec. III. Then we present an overview of our solution in Sec. IV, and introduce each module in Sec. V and Sec. VI, respectively. We present the evaluations in Sec. VII, review related work in Sec. VIII and finally conclude in Sec. IX.

II. Motivations

We motivate our study through measurements from Beike, an online real estate platform in China. We observe a phenomenon called *the overload of top brokers*, where a few brokers are tasked to serve amounts of requests that exceed their capacities, which eventually leads to drop in the brokers' service quality and the overall utility of the platform.

A. Limited Broker Capacity

Our first motivation is that brokers have limited capacity. As with other service industries, we assume a real estate broker has limited capacity, *i.e.*, the number of services he/she can provide in unit time with high quality. Since the low quality of service in housing transactions easily leads to the churn of clients, we hypothesize that the service quality of brokers

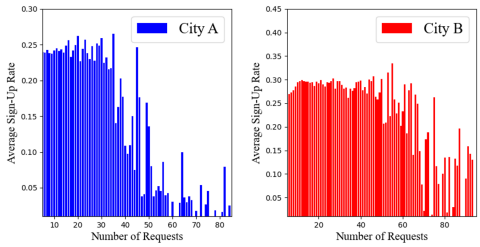


Fig. 2: The average sign-up rate of brokers in two cities.

tends to drop with the increase of served requests. We test this hypothesis via the measurements below.

Measurements. We analyzed data from an online real estate platform from June 1st to August 31st, 2021 in two major cities in China to explore the relationship between service quality and capacity of brokers. We use the broker's sign-up rate, *i.e.*, the ratio between the number of clients who sign up with the broker and that served in total, as the proxy for the service quality. We measure the sign-up rates as the increase of workload, *i.e.*, the number of requests served daily, at both the city and individual levels.

Observations. We observe that the sign-up rates tend to drop with the increase of workload, and the decreasing patterns seem complex and broker-specific.

- Fig. 2 shows the average sign-up rate of brokers in the two cities with the increase of requests served per day. Take City A (in blue) as the example. When the number of requests served is below 40 per day, the average sign-up rate is 14.3$\sim$27.5%. Once brokers have to tackle more than 40 requests a day, their average sign-up rate drops to 2.5$\sim$17.8%. By employing Welch's t-test, we find that the sign-up rate is statistically significantly correlated to the number of requests served daily (p-value < 0.0001). Thus excessive workload of brokers lowers the service quality, and even leads to client churn. Similar decreasing trends are observed for City B (in red) as well.
- We further study the top 50 brokers who serve most requests in City A, where 21 of them serve more than 40 requests occasionally. Fig. 3 plots the sign-up rates of these 21 brokers with higher workloads in City A. Among all the 21 brokers, their sign-up rates exhibit a decreasing trend as the number of requests served daily increases.
- Despite the decreasing trend, the relationship between the sign-up rate and the number of requests served tends to be complex, non-linear and broker-specific patterns, as observed from both Fig. 2 and Fig. 3.

B. Overloaded Top Brokers

Our second motivation is that the top brokers tend to be overloaded due to the top-k recommendation mechanism in current online real estate platforms. This is because the platform lists the top-k brokers without accounting for their capacities, while clients incline to select from the top brokers listed by the platform. We test this claim as follows.

Measurements. We analyze data of the same online real estate platform in June, 2021 in City A and plot the workload dis-

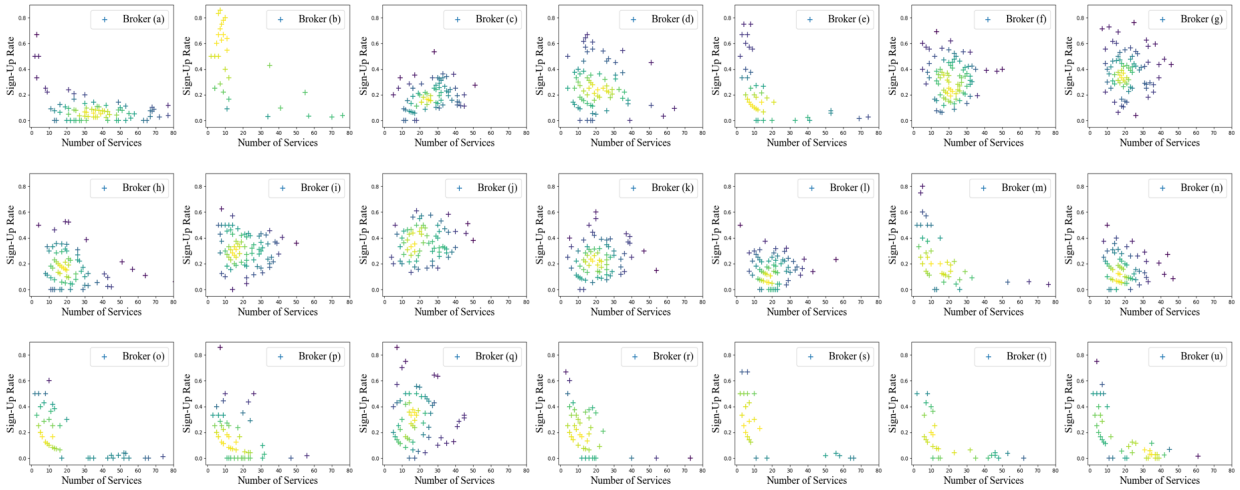


Fig. 3: The average sign-up rate of top brokers in City A. We apply the Gaussian kernel density estimation to fit the empirical distribution of broker's performance with workload. The center of the performance distribution is in the lighter color, which represents their accustomed workload area. Most top brokers perform better in light area compared with large workload area.

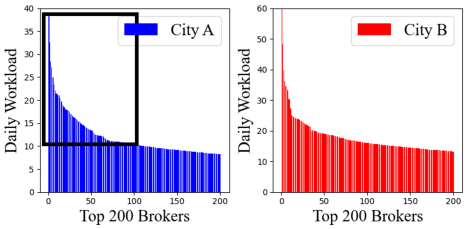


Fig. 4: The workload distribution of the top brokers in City A and City B. People tend to follow the recommended brokers and most requests are served by these top brokers.

tribution breakdown of brokers recommended by the platform and those not listed by the platform. By default, the platform recommends the top-3 brokers (see Fig. 1).

Observations. We observe that the workload distribution is highly unbalanced towards the top-3 brokers recommended by the platform. In City A, the top-1 broker serves 38.26 requests daily on average, while a broker serves 3.18 requests per day on average, *i.e.*, the top-1 broker's workload is 12.03× larger than the average workload. Fig. 4 plots the workload distribution of top-200 brokers in both City A and City B. As is shown, their workloads are all notably higher than the city average workload. Note that from Fig. 3, a workload of 10 to 20 results in higher sign-up rates. Hence, roughly a hundred brokers in the black box risk exceeding their limited capacity. Furthermore, the *Matthew effect* may occur if top brokers are continually tasked requests. That is, most requests are occupied by top brokers, while others are overlooked. As a result, the neglected brokers have few opportunities to improve their home-finding skills, which has a negative impact on the development of the platform. Overloaded top brokers are also common in City B.

C. Key Observations and Insights

In summary, we observe that the top-k recommendation mechanism used by prior online real estate platforms tends to overload the top brokers, which we call the *overload of top brokers* problem. Overloaded brokers exhibit drop in service quality, eventually leading to a decrease in sign-up rates. The overloaded problem occurs because the top-k recommendation ignores the broker's capacity.

The overloaded phenomenon motivates us to rethink broker matching from an *assignment* perspective. Rather than blindly recommend a small group of top brokers to all clients, we propose to assign brokers to clients from a global view, while accounting for the capacity of top brokers. As next, we formulate our perspectives into a capacity-aware assignment problem and propose practical solutions for efficient broker matching in case of unknown workload capacity.

III. Problem Statement

In this section, we formally define the capacity-aware assignment (CAA) problem to avoid overloading top brokers during broker matching.

Definition 1 (Broker)**.** *A broker $b \in B$ is a triple $(\boldsymbol{x}_b, w_b, s_b)$, where $\boldsymbol{x}_b$, w_b, and s_b represent broker b's attributes, the number of requests served daily, and the daily sign-up rate.*

The vector $\mathbf{x}_b$ includes attributes such as working years, job title, average dialogue rounds, etc. (details in Table II), which reflects the current working status of broker b. Aligned with our measurements in Sec. II, we use the number of requests served daily w_b to quantify a broker's workload, and the daily sign-up rate s_b as the proxy of his/her service quality. For ease of presentation, we use $\mathcal{T}_b$ to represent the trial triples of broker b, *i.e.*, $\mathcal{T}_b = \{(\mathbf{x}_1, w_1, s_1), ..., (\mathbf{x}_d, w_d, s_d)\}$.

In broker matching, brokers are tasked to serve clients' requests. Following the conventions in the real estate industry, we perform broker matching on a batch basis and adopt the sign-up rate as the utility of assignment. We take the fixed-time window batched assignment, *i.e.*, the platform presets the time interval and assigns brokers to all the requests appeared. Denote the requests appearing in time interval $i \in I$ as $R^{(i)}$.

From the assignment perspective, broker matching can be formulated as the following problem.

Definition 2 (Capacity-Aware Assignment (CAA) Problem)**.** *Given a set of brokers B, requests in each interval $R^{(i)}$, and the matching utility between them $u_{r,b}$ (utility of assigning broker b to request r), we aim to find assignments $\mathcal{M}^{(i)}$ for each time interval i that maximizes the total utility under the unknown capacity constraint.*

- *Maximizing Total Utility.*

$$\max \sum_{i \in I} \sum_{r,b} u_{r,b} \mathcal{M}_{r,b}^{(i)} \tag{1}$$

- *Capacity Constraint.*

$$\forall b, \quad \sum_{i \in I} \sum_{r} \mathcal{M}_{r,b}^{(i)} \leq c_b \tag{2}$$

where $\mathcal{M}_{r,b}^{(i)} = 1$ if broker b is assigned to request r and $\mathcal{M}_{r,b}^{(i)} = 0$ otherwise. $u_{r,b}$ is the utility obtained if broker b is assigned to request r, which is the input and can be learned from historical assignments using models such as XGboost [12]. c_b is the unknown capacity of broker b to be estimated. Table I summarizes the major notions throughout this paper.

Discussions. We make two notes on the CAA problem.

- Although the batched assignment modeling has been widely adopted in crowdsourcing applications *e.g.*, ride-hailing [10], [13], it is the first time it is catered for broker matching for online real estate platforms.
- A unique challenge of the CAA problem against the general batched assignment lies in the broker capacity c_b, which is not given in advance. Thus, prior to assignment, effective capacity estimation is necessary. Consequently, we aim to devise both capacity estimation and assignment algorithms tailored for online real estate applications.
- Other well-known applications include online healthcare (*e.g.*, ZhongAn Insurance[5]) and legal consultation (*e.g.*, Lvshiguan[6]), where a similar capacity-aware matching problem exists due to the limited workload capacity of workers to serve their assigned requests.

TABLE I: Summary of major symbols.

Notation	Description
$\mathbf{x}_b, w_b, s_b, c_b$	attributes, workload, sign-up rate and capacity of broker b
$\mathcal{T}_b$	trial triples of broker b
I, i	time intervals horizons and time interval i
B, B_+	set of all brokers and set of brokers with residue capacity
$R, R^{(i)}$	set of all requests and set of requests in interval i
$u_{r,b}$	utility of assigning broker b to request r
$\mathcal{M}^{(i)}, \mathcal{M}_{r,b}^{(i)}$	assignment in interval i and assignment indicator of (r, b)
$UCB_{\mathbf{x},c}$	upper confidence bound of capacity c under context $\mathbf{x}$
$\mathcal{B}_{\theta,\mathbf{D}}$	contextual bandit with learned parameters $\theta, \mathbf{D}$
$\mathcal{S}_\theta, g_\theta$	reward mapping function and its gradient
$\mathcal{L}(\theta)$	loss function of neural networks in contextual bandit
$\mathcal{B}_b$	exclusive contextual bandit of broker b
$\mathcal{V}(cr)$	capacity-aware value function under residue capacity cr
β, γ, δ	learning rate, discount factor and update threshold

[5]https://www.zhongan.com/

[6]http://www.lvshiguan.com

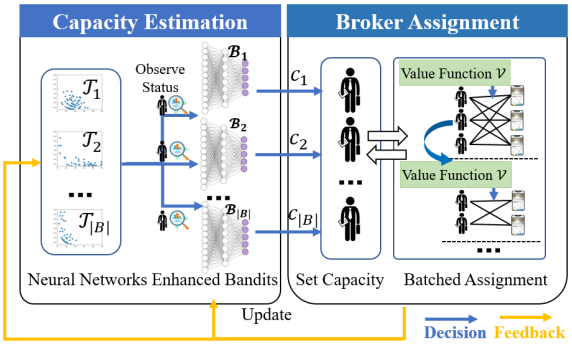


Fig. 5: Workflow of LACB.

IV. Overview of Our Solution

To solve the CAA problem, we propose Learned Assignment with Contextual Bandits (LACB), which learns the unknown broker capacity via contextual bandit and assigns brokers from the global perspective to maximize the total utility without overloading the top brokers. We first present an overview of LACB and explain each functional module.

LACB consists of two functional modules, *capacity estimation* and *capacity-based assignment*.

- The *capacity estimation* module decides the daily workload capacity according to the broker's current status by neural network enhanced contextual bandits.
- The *capacity-based assignment* module selects a set of brokers satisfying the capacity constraint and assigns them to requests via the capacity-aware value function.

Fig. 5 shows the workflow of LACB. It operates in two phases: estimation and assignment. First, we observe the broker's working status and set a daily workload capacity for him/her by the neural network enhanced bandit. In the assignment phase, we take the broker's estimated capacity and adopt a capacity value function to guide the assignment, capturing the long-term utility of brokers with different workloads. Finally, we store the results of batched assignment as feedback to improve future decisions.

V. Capacity Estimation

This section introduces our capacity estimation method. We formulate the workload capacity estimator as a contextual bandit and propose a neural network enhanced policy to decide the daily workload capacities for each broker.

A. Basic Idea

Our method is motivated by the following three challenges when estimating the broker capacities.

- *Online training*. It is impractical to collect data of a broker's sign-up rate under all possible workloads prior to model deployment. The workload capacity estimator is expected to routinely adapt itself to the observations of workloads and sign-up rates after deployment. Our solution is to apply the contextual bandit algorithm [14], a learning method to explore the unknown environment and make decisions under various contextual information.

TABLE II: The broker's attributes.

Attribute Type	Attribute	Description
Basic Info.	Age	Broker's age.
	Working Year	The working years as a broker.
	Education	Education background (*e.g.*,undergraduate, master).
	Title	Job title (*e.g.*,assistant, clerk, manager).
Work Profile	Response Rate	The rate of the broker's response to a request in one minute.
	Dialogue rounds	The average dialogue rounds via the App in recent 7/14/30/90 days.
	Number of Housing Presentation	The number of broker's presenting houses offline in recent 7/14/30/90 days.
	Number of Presentation via VR	The number of broker's presenting houses via VR in recent 7/14/30/90 days.
	Time of Presentation via VR	The time of broker's presenting houses via VR in recent 7/14/30/90 days.
	Number of Consultation via Phone	The number of broker answering clients via phone in recent 7/14/30/90 days.
	Time of Consultation via Phone	The time of broker answering clients via phone in recent 7/14/30/90 days.
	Number of Consultation via App	The number of broker answering clients via App in recent 7/14/30/90 days.
	Time of Consultation via App	The time of broker answering clients via App in recent 7/14/30/90 days.
	Number of Maintained Houses	The number of houses currently maintained by the broker.
	Number of Served Clients	The number of clients who are served by the broker in recent 7/14/40/90 days.
	Number of Housing Transactions	The number of housing transactions through the broker in recent 7/14/40/90 days.
Preference	Districts Information	Broker's preferable communities and area around POIs.
	Housing Information	Broker's preferable price, area and type of houses.

- *Complex relations between a broker's performance and workload*. The broker's performance is closely related to the current working status. For instance, a broker may be more exhausted in the sales seasons, and thus less resilient to heavy workloads. As observed in Sec. II-A, the relationship between a broker's performance and his/her workload is non-linear, and we model such non-linear complexity via neural networks.
- *Personalized workload capacity*. As shown in Fig. 3, the relationship between sign-up rate and workload is not only complex but also broker-specific. However, it is challenging to directly learn a personalized capacity estimator due to the sparsity of broker-specific data. We first learn a generic capacity estimation model and fine-tune it for individual brokers.

As next, we elaborate on the designs in sequel.

B. Workload Capacity Estimator as Contextual Bandit

As mentioned above, we utilize contextual bandits to learn a generic broker capacity estimator in an online fashion by interacting with the real estate platform. The reinforcement learning [15] (*e.g.*, Q-learning) mainly models the effect of the decision on the state. However, in our scenario, the broker's intrinsic working status is not affected by our decisions, so approaches like Q-learning are infeasible to capacity estimation.

Review on Contextual Bandit. We start with a quick review of the contextual bandit. A bandit with k-*arms* is widely used for online decision-making in an unknown environment over n batches, where each arm represents a decision. In each batch, the bandit chooses one arm (decision) and receives a *reward* from the environment. It then updates its decision-making strategy based on the reward and tries to maximize the total rewards over the n batches. A contextual bandit further allows the bandit to make decisions with additional information (*i.e.*, the *context*) at the beginning of each batch.

Our Formulation. We now explain how to formulate the workload capacity estimator as a contextual bandit. We consider the broker's candidate workload capacities as arms of the bandit (represented as $\mathcal{C}$). The broker's working status $\mathbf{x}_b$ is viewed as the context, based on which the bandit chooses a capacity $c_b \in \mathcal{C}$. The daily sign-up rate s_b under workload w_b is used as the reward. The workload capacity estimator interacts with the real estate platform, which is viewed as the unknown environment. In each batch, the real estate platform executes assignment algorithms and reveals the reward s_b. We use $(\mathbf{x}_b, w_b, s_b)$ as a trial triple to update the reward function of the bandit (workload capacity estimator) since a broker's workload w_b is usually lower than her/his capacity c_b.

C. Choosing Capacity with Neural Network Enhanced UCB

With the workload capacity estimator formulated as a contextual bandit, the next question is to determine the *policy* to choose the workload capacity that maximizes the daily sign-up rates for the given broker working status.

Standard UCB. A common decision-making policy for a contextual bandit is the upper confidence bound (UCB) algorithm [14]. It acts as if the environment is as nice as plausibly possible and uses a context vector to calculate the upper confidence bound of the expected reward. Then it chooses the arm with the maximum upper confidence bound as the decision. For our workload capacity estimation, we can use the broker's working status $\mathbf{x}$ as the context, and calculate the upper confidence bound $UCB_{\mathbf{x},c}$ of each workload capacity c using the equation below [14].

$$UCB_{\mathbf{x},c} = f_\theta(\mathbf{x}, c) + \alpha\sqrt{f'_\theta(\mathbf{x}, c)^T D^{-1} f'_\theta(\mathbf{x}, c)} \tag{3}$$

where $f_\theta(\mathbf{x}, c)$ is a linear model which maps the context $\mathbf{x}$ to the expected reward of a workload capacity and $f'_\theta(\mathbf{x}, c)$ is its derivative. α is a preset coefficient parameter. θ is the parameters of the linear model and D is the covariance matrix. θ and D are parameters to be trained.

NN-enhanced UCB. A limitation of the standard UCB algorithm is the assumption on the linear relation between the expected reward and the context, *i.e.*, $f_\theta(\mathbf{x}, c)$ in Eq. (3). Hence, the standard UCB fails to depict the non-linear relation between the broker's sign-up rate (expect reward) and the working status (context) in our scenario (see Sec. II-A). As a remedy, we replace the linear model by a neural network.

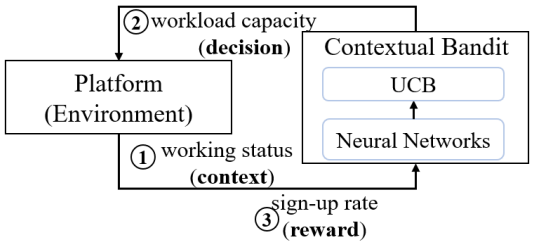


Fig. 6: Workflow of the NN-enhanced UCB.

We name the corresponding capacity choosing policy as NN-enhanced UCB.

Fig. 6 shows the workflow of our NN-enhanced UCB. We replace the linear model $f_\theta(\mathbf{x}, c)$ in standard UCB by a learned reward mapping function $\mathcal{S}_\theta(\mathbf{x}, c)$. For simplicity, we adopt a fully connected MLP network with L layers:

$$\mathcal{S}_\theta(\mathbf{x}, c) = W_L \cdot \sigma_{L-1}(\cdots \sigma_1(W_1[\mathbf{x}; c])) \tag{4}$$

where W_i $(1 \le i \le L)$ are the learned parameters of each layer and $W_1 \in \mathbb{R}^{m\times d}$, $W_i \in \mathbb{R}^{m\times m}$ $(2 \le i \le L-1)$, $W_L \in \mathbb{R}^{m\times 1}$. σ_i is the ReLU activation.

Let $\theta \in \mathbb{R}^d$ be all the learnable parameters of the neural network and its gradient is denoted by $g_\theta(\mathbf{x}, c) = \nabla_\theta \mathcal{S}_\theta \in \mathbb{R}^d$. Then we can easily extend the upper confidence bound of the expected reward in Eq. (3) as follows.

$$UCB_{\mathbf{x},c} = \mathcal{S}_\theta(\mathbf{x}, c) + \alpha\sqrt{g_\theta(\mathbf{x}, c)^T \mathbf{D}^{-1} g_\theta(\mathbf{x}, c)} \tag{5}$$

That is, we replace the linear model $f_\theta(\mathbf{x}, c)$ and its derivative $f'_\theta(\mathbf{x}, c)$ by a neural network $\mathcal{S}_\theta(\mathbf{x}, c)$ and its gradients $g_\theta(\mathbf{x}, c)$, with θ and $\mathbf{D}$ to be trained, as in the standard UCB.

Bandit Training. Alg. 1 illustrates how to train a contextual bandit $\mathcal{B}_{\theta,\mathbf{D}}$ over time, *i.e.*, learn the parameters θ and $\mathbf{D}$ defined in Eq. (5) of our NN-enhanced UCB. We first observe the working status $\mathbf{x}_b$ and then preferentially explore the capacity with maximum upper confidence bound of the expected reward based on Eq. (5) (lines 6 to 10). Next, we update the covariance matrix $\mathbf{D}$ using the same extension as Eq. (5), *i.e.*, replacing the derivative $f'_\theta(\mathbf{x}, c)$ of the linear model by gradients $g_\theta(\mathbf{x}, c)$ of neural networks (line 12). Afterwards we observe the actual workload w_b as well as the reward s_b (line 13). We make such explorations and observations for a batch size of $batchSize$ (preset as 16), where we store the triples $(\mathbf{x}, w, s)$ in buffer ob. After collecting observations for $batchSize$ batches, we update the parameters θ by minimizing the following loss.

$$\mathcal{L}(\theta) = \sum_{o\in ob} \|\mathcal{S}_\theta(\mathbf{x}_o, w_o) - s_o\|^2 + \lambda\|\theta\|_2^2 \tag{6}$$

where (x_o, w_o, s_o) is an observation and λ is the regularization parameter to avoid over-fitting.

Estimating Capacity with Bandit. After training the bandit $\mathcal{B}_{\theta,\mathbf{D}}$ as Alg. 1, it can be used to estimate the workload capacity for subsequent batches. That is, given a working status $\mathbf{x}$, we choose the workload capacity c with the maximum upper confidence bound calculated by Eq. (5). For ease of presentation, we use $\mathcal{B}.estimate(\mathbf{x})$ to represent the workload capacity estimation, which chooses a suitable workload capacity given working status $\mathbf{x}$.

Algorithm 1: NN-enhanced UCB

Input: Regularization parameter λ, upper confidence bound coefficient α, and batch size $batchSize$
Output: Contextual bandit $\mathcal{B}_{\theta,\mathbf{D}}$

1 Initialize deviation matrix $\mathbf{D} \leftarrow \lambda\mathbf{I}$;
2 Initialize observation buffer $ob \leftarrow []$;
3 Initialize parameters θ with Gauss Distribution;
4 **for** *each trial* $t \in \mathcal{T}$ **do**
5 // Explore the workload capacity;
6 Observe the broker's working status $\mathbf{x}$;
7 **for** *each candidate capacity* $c \in \mathcal{C}$ **do**
8 $U_{\mathbf{x},c} \leftarrow \mathcal{S}_\theta(\mathbf{x}, c) + \alpha\sqrt{g_\theta(\mathbf{x}, c)^T \mathbf{D}^{-1} g_\theta(\mathbf{x}, c)}$;
9 Choose b's capacity $c^* \leftarrow \arg\max_c U_{\mathbf{x},c}$;
10 **end**
11 // Update upper confidence and reward function;
12 $\mathbf{D} \leftarrow \mathbf{D} + g_\theta(\mathbf{x}, c) \cdot g_\theta(\mathbf{x}, c)^T$;
13 Observe broker's workload w and the reward s;
14 Store observed triple $(\mathbf{x}, w, s)$ in buffer ob;
15 **if** $ob.size = batchSize$ **then**
16 Define loss function $\mathcal{L}(\theta)$ as $\sum_{o\in ob} \|\mathcal{S}_\theta(\mathbf{x}_o, c_o) - s_o\|^2 + \lambda\|\theta\|_2^2$;
17 $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}(\theta)$;
18 Clear observation buffer $ob \leftarrow []$;
19 **end**
20 **end**
21 **return** contextual bandit $\mathcal{B}_{\theta,\mathbf{D}}$;

D. Personalized Workload Capacity Estimator

As previously mentioned, the contextual bandit only learns a generic capacity estimator for all brokers, yet the workload capacity estimation may be broker-specific. We enable personalized workload capacity estimation by fine-tuning the neural network $\mathcal{S}_\theta(\mathbf{x}, c)$ in Eq. (5) on broker-specific data.

Concretely, we first train a base reward mapping function θ^{base}, *i.e.*, the neural network defined in Eq. (4), on the observations $\cup_{b\in B}\mathcal{T}_b$ of all brokers. Then we copy the first $L-1$ layers of θ^{base} to the broker-specific reward mapping function θ_b of broker b. Afterwards, we freeze the first $L-1$ layers of θ_b, and fine-tune the last full-connected layer based on the broker's observations $\mathcal{T}_b$ following Alg. 1. This way, we obtain the personalized reward mapping function.

E. Effectiveness of NN-Enhanced UCB

We now theoretically analyze the effectiveness of our NN-enhanced UCB policy for contextual bandit based workload capacity estimation. We assess the effectiveness via the *regret*, a standard performance metric for bandits [14].

The regret is defined as the difference in total rewards between the optimal policy and a learned decision-making policy. For our workload capacity estimation, the regret can be defined as the difference between the sum of sign-up rates

with ideal capacities and the sum of sign-up rates with the estimated capacities, which is formalized as follows,

$$\mathcal{R}egret = \sum_{t=1}^{|\mathcal{T}|} \max_{c \in \mathcal{C}} s(\mathbf{x}_b, c) - \sum_{t=1}^{|\mathcal{T}|} s_t \tag{7}$$

where $\mathcal{T}$ is the trial triples, $\max_{c \in \mathcal{C}} s(\mathbf{x}_b, c)$ is the ideal rewards under working status $\mathbf{x}_b$, and s_t is rewards produced by our NN-enhanced UCB algorithm.

We have the following claim on the regret of our NN-enhanced UCB algorithm.

Theorem 1. *For a contextual bandit adopting NN-enhanced UCB algorithm with an L-layer MLP network, if there are $|\mathcal{C}|$ candidate capacities, the regret of n batches is no more than $\frac{n|\mathcal{C}|\xi^L}{\pi^{L-1}}$, where $\pi \approx 3.14$ is the circular constant and ξ is the maximum single value of parameters in the MLP model.*

Proof. Let $c_t^* = \arg\max_c S_{\theta^*}(\mathbf{x}_t, c)$ denote the optimal capacity under the context $\mathbf{x}_t$ and $\hat{c}_t = \arg\max_c U_{\mathbf{x}_t, c}$ in batch t. Firstly, we analyze the instantaneous regret r_t in batch t, which is defined as:

$$r_t = S_{\theta^*}(\mathbf{x}_t, c^*) - S_{\theta^*}(\mathbf{x}_t, \hat{c}) \tag{8}$$

Using the *Lipschitz continuous* [16] to bound instantaneous the regret of the bandit, we have,

$$\|S_{\theta^*}(\mathbf{x}_t, c^*) - S_{\theta^*}(\mathbf{x}_t, \hat{c})\|_2 \leq Lip(S_{\theta^*}) \|c^* - \hat{c}\|_2 \tag{9}$$

where $\|\cdot\|_2$ denotes the L_2-norm and $Lip(S_{\theta^*})$ is called the *Lipschitz constant* of S_{θ^*}.

Then, we analyze the upper bound of Lipschitz constant $Lip(S_{\theta^*})$ of the reward function S_{θ^*}. According to Lemma 2 in [16], we can construct its upper bound,

$$Lip(S_{\theta^*}) \leq \frac{\|W_L \text{diag}(\sigma_{L-1}) W_{L-1} \cdots \text{diag}(\sigma_1) W_1\|_2}{\pi^{L-1}} \tag{10}$$

where $\pi \approx 3.14$ is the circular constant. If we take the *ReLU* as the activation function σ, we have $\|\sigma\|_2 \leq \|\mathbf{I}\|_2$, where I is the identity matrix. Thus, the upper bound of Lipschitz constant $Lip(S_{\theta^*})$ can be rewritten as,

$$Lip(S_{\theta^*}) \leq \frac{\Pi_{i=1}^{L} \|W_i\|_{op} \cdot \Pi_{i=1}^{L-1} \|\mathbf{I}\|_2}{\pi^{L-1}} \leq \frac{\xi^L}{\pi^{L-1}} \tag{11}$$

where $\|W_i\|_{op}$ is the operator norm of W_i, *i.e.*,, the largest single value of W_i ($\|W_i\|_{op} \leq \xi$). Since both c^* and $\hat{c}$ belong to $\mathcal{C}$, we have $\|c^* - \hat{c}\|_2 \leq |\mathcal{C}|$. Thus, for instantaneous regret r_t, we can give its upper bound,

$$r_t = S_{\theta^*}(\mathbf{x}_t, c^*) - S_{\theta^*}(\mathbf{x}_t, \hat{c}) \leq Lip(S_{\theta^*}) \|c^* - \hat{c}\|_2 \leq \frac{|\mathcal{C}|\xi^L}{\pi^{L-1}} \tag{12}$$

Finally, we prove that the regret bound for our NN-enhanced UCB policy over n batches is,

$$\mathcal{R}egret = \sum_{t=1}^{n} r_t \leq \frac{n|\mathcal{C}|\xi^L}{\pi^{L-1}} \tag{13}$$

□

Discussions. We draw two practical notes from Theorem 1.

- Setting a suitable number of candidate capacities is beneficial to select an optimal decision. We empirically determine the range of candidate sets based on historical observations in Sec. II and do not explore the workload capacity with a prominent low sign-up rate.
- Although a deeper network may model more complex relationships between a broker's performance and working status, it may also prevent the bandit from choosing the optimal workload capacity. We empirically adopt a 3-layer MLP network in our NN-enhanced UCB algorithm to balance the complexity of the neural network and the effectiveness of workload estimation.

VI. Capacity-Based Assignment

Now we present the assignment module of LACB, which takes the estimated capacity as input and makes assignments by accounting for both the capacity constraint and the dependency of assignments across batches.

A. Batched Assignment as Markov Decision Process

Unlike previous studies [10], [17] that *independently* make assignments for each batch, we propose to match brokers in a more holistic view by modeling the assignments over time as a *Markov Decision Process (MDP)* [14]. Such modeling accounts for the dependencies of assignments across batches (*i.e.*, residual capacities of brokers over time) and potentially results in a higher total utility.

A standard MDP model consists of four elements: the *state*, *action*, *state transition*, and *reward*. We explain these elements in the context of broker assignment below.

- **State**. As our assignment decision is capacity-aware, we define the state of each broker as $cr_b \in [0, c_b]$, where cr_b and c_b denotes the broker's residue capacity and the workload capacity, respectively.
- **Action**. In batch i, the action is an assignment policy $\mathcal{M}_{r,b}^{(i)}$ for each request r and each broker b. If $\mathcal{M}_{r,b}^{(i)} = 1$, broker b is assigned to request r, and $\mathcal{M}_{r,b}^{(i)} = 0$ otherwise. In this work, we adopt a value function guided assignment policy (see Sec. VI-B).
- **State Transition**. The state of a broker changes as the result of action $\mathcal{M}_{r,b}^{(i)}$. If $\sum \mathcal{M}_{r,b}^{(i)} = 0$, the state of broker b remains the same. Otherwise, state cr_b will transit to $cr_b - \sum \mathcal{M}_{r,b}^{(i)}$. That is, the residue capacity of the broker is reduced by the number of requests assigned to him/her.
- **Reward**. The rewards of an action is defines as the utility of all brokers in batch i (a.k.a batch utility), *i.e.*, $r(\mathcal{M}^{(i)}) = \sum_{r,b} u_{r,b} M_{r,b}^{(i)}$. Note that the reward of the MDP model differs from the reward of a bandit (in Sec. V), and the latter is the accumulative utility of a single broker, *i.e.*, $s_b = \sum_i \sum_r u_{r,b} \mathcal{M}_{r,b}^{(i)}$.

Note that we formulate the batched assignment as an MDP model (and adopt reinforcement learning based solution) rather than a bandit algorithm because the latter is unfit for long-term planning with state transitions [14].

B. Capacity-Aware Assignment

Given the MDP model above, we utilize a capacity-aware value function to guide the assignments.

Capacity-Aware Value Function. It is common to make decisions in an MDP by learning the value function of a state [10], [13]. In this work, we define a capacity-aware value function $\mathcal{V}(i, cr)$, which represents the expected utility of the broker after batch i, where cr is the broker's residue capacity. Such a capacity-aware value function captures the long-term utility of brokers with different residue capacities. We then use Q-learning [10], a classical method with relatively low time overhead, to train the capacity-aware value function, which is based on the Temporal-Difference (TD) equation below.

$$\mathcal{V}(cr) \leftarrow \mathcal{V}(cr) + \beta[u + \gamma\mathcal{V}(cr') - \mathcal{V}(cr)] \tag{14}$$

where cr/cr' are current/transited state after taking an action, u is the reward of an action, β and γ are the learning rate and discount factor of the TD, respectively.

Algorithm 2: Value Function Guided Assignment

Input: Brokers $B = \{b_1, b_2, ..., b_{|B|}\}$, requests $R = \{R^{(1)}, R^{(2)}, ..., R^{(|I|)}\}$ over I intervals, brokers' exclusive bandits $\{\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_{|B|}\}$

Output: Matching results $\mathcal{M}$

1 **for** *each broker* b *in* B **do**
2 $c_b \leftarrow \mathcal{B}_b.estimate(\mathrm{x}_b)$;
3 **for** *each interval* $i \in I$ **do**
4 Get available brokers $B_+ = \{b | w_b < c_b \wedge b \in B\}$;
5 **for** *each candidate pair* $(r, b) \in R^{(i)} \times B_+$ **do**
6 $$u'_{r,b} \leftarrow \begin{cases} u_{r,b} + 0, & \text{if } f_b \le \delta \\ u_{r,b} + \gamma\mathcal{V}(cr') - \mathcal{V}(cr), & \text{if } f_b > \delta \end{cases}$$
7 Execute the Kuhn-Munkres algorithm based on the refined utility $\mathcal{M}^{(i)} = KM(u', R^{(i)}, B_+)$;
8 **for** *each broker* b *in* B_+ **do**
9 $w_b \leftarrow w_b + \sum_r \mathcal{M}^{(i)}_{r,b}$;
10 Update capacity-aware value function based on the Temporal-Difference equation Eq. (14);
11 **for** *each broker* b *in* B **do**
12 $s_b \leftarrow \sum_i \sum_r u_{r,b}\mathcal{M}^{(i)}_{r,b}$;
13 $\mathcal{B}_b.update(\mathrm{x}_b, w_b, s_b)$;
14 **return** $\mathcal{M} = \cup_{i \in I}\mathcal{M}^{(i)}$

Value Function Guided Assignment (VFGA). We can now leverage the above capacity-aware value function to assign brokers from a global view to maximize the total utility. Alg. 2 shows the overall assignment algorithm. First, we determine the personalized capacity of brokers from the contextual bandit $\mathcal{B}_b$ (see Sec. V). In lines 4-14, we make assignment in each batch. Specifically, in line 5, we first select a set of available brokers B_+, whose workload w_b is lower than his/her capacity c_b. Then we update the utility of each candidate matching pair as follows.

$$u'_{r,b} = \begin{cases} u_{r,b} + 0, & \text{if } f_b \le \delta \\ u_{r,b} + \gamma\mathcal{V}(cr') - \mathcal{V}(cr), & \text{if } f_b > \delta \end{cases} \tag{15}$$

where $u_{r,b}/u'_{r,b}$ is the original/refined utility respectively, $cr = c_b - w_b$ and $cr' = cr - 1$. Note that only top brokers may reach their workload capacity, and we only use the value function for top brokers whose frequency f_b of reaching capacity is more than δ, where δ is a positive number close to 1. In line 9, we run the classical Kuhn–Munkre (KM) [18] algorithm on a bipartite graph with the refined utility $u'_{r,b}$ and return the assignment policy $\mathcal{M}^{(i)}$. In lines 10-12, the workloads of the assigned brokers and the capacity-aware value function are updated according to Eq. (14). In lines 15-17, once we have assigned requests in all batches, we collect the brokers' workloads and performance as feedback to update the bandit of each broker. As aforementioned, we update the parameters of bandits whenever the observation buffer is full. Otherwise, we only add new observations to the buffer. Finally, VFGA returns the assignment results $\mathcal{M}$.

Discussions. We make two notes on the assignment module.

- Since the number of requests $|R|$ is usually smaller than that of brokers $|B|$, a common practice is to add some dummy vertices to the smaller part to construct a balanced bipartite graph [19]–[21]. By adding $|B| - |R|$ dummy vertices, we obtain a balanced one with $|B|$ vertices on both sides and can execute the classical KM algorithm.
- Once a client is unsatisfied with the assigned broker, she/he can appeal to the platform for another broker. The platform sets the utility between the client and the assigned broker to 0, restores the broker's workload, and chooses another broker in the next time interval.

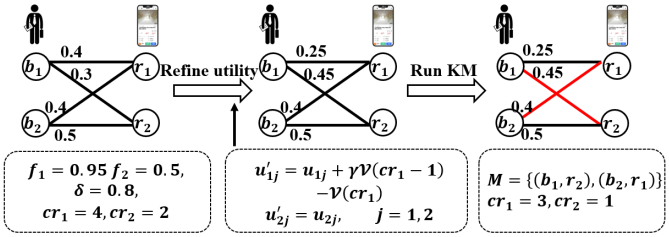


Fig. 7: Example of the VFGA algorithm.

Fig. 7 shows an example of the core steps in the VFGA algorithm. There are two brokers b_1, b_2 and two requests r_1, r_2. The utility of (b_1, r_1), (b_1, r_2), (b_2, r_1) and (b_2, r_2) are $0.4, 0.3, 0.4$ and 0.5 respectively. Based on Eq. (15), b_1 reaches the threshold $\delta = 0.8$ and we refine b_1's utility as $u_{11} = 0.25$ and $u_{12} = 0.45$, while b_2's utility remains the same. Then we run the KM algorithm to get the maximum assignment over refined weights, *i.e.*, $\{(b_1, r_2), (b_2, r_1)\}$. Finally, residue capacities of b_1 and b_2 are updated to 3 and 1, respectively.

Complexity Analysis. For each batch, the time complexity of the VFGA algorithm is determined by the KM algorithm [18]. Since the KM algorithm is run on a balanced bipartite graph with $|B|$ vertices on both sides, the time complexity of the VFGA algorithm is $\mathcal{O}(|B|^3)$.

C. Accelerating Assignment via Broker Selection

The real estate platform needs to assign brokers to clients as efficiently as possible, which is an essential factor in

Algorithm 3: Candidate Broker Selection (CBS)

Input: The candidate size k, request r and brokers B
Output: The candidate brokers Top_k^r

```
if |B| ≤ k then
    return B;
end
Choose an random value p from u_{r,b} (b ∈ B);
LC ← {b ∈ B | u_{r,b} ≥ p};
RC ← {b ∈ B | u_{r,b} < p};
if |LC| ≥ k then
    Top_k^r ← CBS(k, r, LC)
end
else
    Top_k^r ← LC ∪ CBS(k − |LC|, r, RC);
end
return Top_k^r
```

user experience. It usually requires responding in 2 seconds for reasonable user experience [22]. There is an opportunity to accelerate the VFGA algorithm because the numbers of brokers and requests are highly imbalanced in online real estate platforms. In each batch, the platform typically assigns thousands of brokers to only tens of requests. If we prune the brokers that are unlikely to be matched, we can notably reduce the number of dummy vertices added to the request side, and thus lower the time complexity.

Theoretical Evidence. We claim that only a small set of brokers are necessary for effective assignments, as shown by Theorem 2 and Corollary 1.

Theorem 2. *Given a bipartite graph $G =< U, V, E >$ ($|U| \leq |V|$), and let $opt_{\mathcal{M}}(u)$ be the matched vertex of u in the optimal assignment $\mathcal{M}$. There exists an optimal assignment $\mathcal{M}$, such that for any vertex $u \in U$, we have $opt_{\mathcal{M}}(u) \in Top^u_{|U|}$, where $Top^u_{|U|}$ is a set of vertices with $|U|$ largest edge weight among all vertices connected u.*

Proof. The proof is by contradiction. We start by assuming the opposite. Let u be any vertex of U and the vertex matched to u be v^*. Assume there is no optimal assignment $\mathcal{M}$, such that $v^* = opt_{\mathcal{M}}(u) \in \text{Top}^u_{|U|}$. Consider a corner case, where any other vertex of U is matched to a vertex of $\text{Top}^u_{|U|}$ in the optimal assignment $\mathcal{M}$. There is still at least one unmatched vertex $v' \in \text{Top}^u_{|U|}$. By the definition of $\text{Top}^u_{|U|}$, we have $w(u, v') \geq w(u, v^*)$. If we replace (u, v') with (u, v^*) and keep the other matching of $\mathcal{M}$ the same, we can construct another optimal assignment $\mathcal{M}'$, which contradicts the assumption that "*there is no optimal assignment*." □

Corollary 1. *Given an imbalanced bipartite graph $G =< U, V, E >$, we need at most $|U|$ candidate vertices for each vertex $u \in U$ to find an optimal assignment,* i.e.*, taking the $Top^u_{|U|}$ as the candidate set for any $u \in U$.*

Candidate Broker Selection Algorithm. To efficiently select the candidate set $\text{Top}^r_{|R|}$ for each request r, we devise the Candidate Broker Selection (CBS) algorithm (see Alg. 3), which is inspired by solutions to the classical *selection problem* [18]. We mainly introduce how to integrate this optimization into our VFGA. In each batch, we execute the CBS algorithm to select the necessary brokers $\cup_{r \in R^{(i)}} \text{Top}^r_{|R^{(i)}|}$ after getting the available brokers B_+ (line 5 in Alg. 2). Then we can perform assignment on a much smaller bipartite graph.

Complexity Analysis. The expected time complexity of candidate broker selection is $\mathcal{O}(|R||B|)$. Based on the above optimization, we can assign brokers on the bipartite graph with $|R|$ vertices and $|R|^2$ edges, so the time complexity of executing the KM algorithm is reduced from $\mathcal{O}(|B|^3)$ to $\mathcal{O}(|R|^3)$, where $|R|$ are usually much smaller than $|B|$. As a result, the overall time complexity of the VFGA algorithm with CBS is $\mathcal{O}(|R|^3 + |R||B|)$.

TABLE III: Synthetic datasets.

Factor	Setting
The number of brokers $\|B\|$	500, 1000, **2000**, 5000, 10000
The number of requests $\|R\|$	10K, 20K, **50K**, 100K, 200K
The number of covering days Day	7, 10, **14**, 17, 21
The degree of imbalance σ	0.005, 0.01, **0.015**, 0.02, 0.05

TABLE IV: Real-world datasets.

City	Dates	Brokers	Requests
City A	Aug. 1 ∼ Aug. 21, 2021	5515	103106
City B	Jul. 1 ∼ Jul. 21, 2021	8155	387339
City C	Jun. 8 ∼ Jun. 28, 2021	3689	74831

VII. Experimental Study

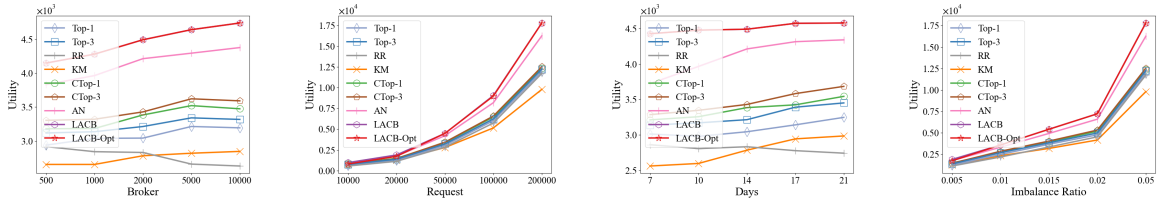
This section presents the evaluations of LACB.

A. Experiment Setup

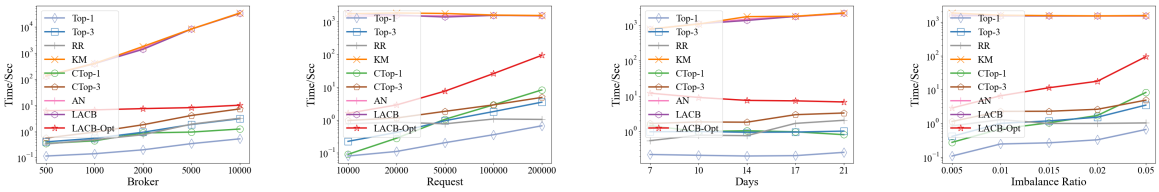
Datasets. We test our capacity-aware assignment algorithm over both synthetic and real datasets.

- *Synthetic Datasets*. We generate 2000 brokers and 50000 requests in total. Then we vary the number of brokers, requests, covering days, and the degree of imbalance. The degree of imbalance $\sigma = |R|/|B|$ is the ratio between requests and brokers in each batch. To vary σ, we keep the number of brokers the same and change the number of requests. Table III summarizes the configuration of synthetic datasets. We mark the default settings in bold.
- *Real-world Datasets*. We collect data in three cities covering 21 days from Beike, the largest Chinese online real estate platform. Table IV summarizes the statistics of real datasets.

Implementation. Our experiments are conducted on a simulator of Beike, which takes the same utility function deployed and outputs the utility between requests and brokers, so that we can take such utility as the input and assess algorithms over real world matching instances. In the NN-enhanced UCB, we adopted a 3-layer MLP network and set the size of input layer, hidden layer and output layers as 128, 64 and 16, respectively. We take the *ReLU* as the activation function. In Alg. 1, we set α to 0.001, the batch size $batchSize$ to 16 and the regularization parameter λ to 0.001. In Alg. 2, we set



(a) Utility of varying $|B|$ (b) Utility of varying $|R|$ (c) Utility of varying Day (d) Utility of varying σ



(e) Time of varying $|B|$ (f) Time of varying $|R|$ (g) Time of varying Day (h) Time of varying σ

Fig. 8: Results on synthetic datasets.

the learning rate β to 0.25, discount factor γ to 0.9 and the threshold δ to 0.8. All the experiments were conducted on Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz with 32GB main memory. The algorithms are implemented in Python 3.

Compared Algorithms. We compare our LACB *i.e.*, Sec. VI-B and LACB with CBS *i.e.*, Sec. VI-C (denoted as LACB-Opt) with two categories of baselines. The first category (Top-K, RR and KM) does not set explicit capacity for brokers, while the second category (CTop-K and AN) first chooses brokers' capacities and then assigns them to requests.

- Top-K Recommendation (Top-K) [2]: It ranks and returns K brokers with the highest utility. We evaluate both Top-1 and Top-3 recommendation.
- Randomized Recommendation (RR): It takes the broker's service quality as the sampling weight and recommends a random broker. RR extends prior *fair matching* algorithms [23] and views service quality as the fairness index. It can avoid the overloaded phenomenon by apportioning online requests to all brokers.
- Kuhn–Munkre (KM) algorithm [18]: It runs the KM algorithm to assign brokers to requests in each batch.
- Constrained Top-K (CTop-K) [24]: It is an extension of Top-K where CTop-K observes the relations of workload and the sign-up rate at city levels (Fig. 2) and empirically chooses a capacity for all brokers. The empirical workload capacity is set as 45, 55 and 40 for brokers in City A, City B and City C, respectively. We test CTop-K with both K=1 and K=3.
- Assignment with NeuralUCB (AN): It explores the workload capacity by NeuralUCB [9] and assigns brokers to requests by the KM algorithm [18].

B. Performance on Synthetic Datasets

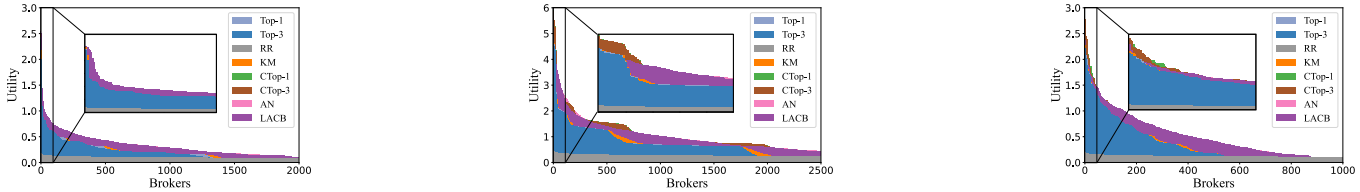
In this set of experiments, we test the impact of different parameters on the performance of different algorithms.

Impact of # brokers. The first column of Fig. 8 shows the results of varying $|B|$. For the total utility, our LACB and LACB-Opt dominate other baselines, including Top-K, CTop-K, KM, RR and AN. We also observe that the utility of Top-K Recommendation decreases as $|B|$ increases, indicating that providing more brokers will not improve the total utility due to the overloaded phenomenon. Finally, LACB-Opt achieves the same utility as LACB, which is consistent with our theoretical analysis in Corollary 1, *i.e.*, the candidate broker selection does not sacrifice the total utility. In terms of the running time, as $|B|$ increases, KM, AN, and LACB become inefficient due to their cubic time complexity, yet the running time of Top-k, RR and CTop-k only increases marginally. The running time of LABT-Opt remains stable since its time complexity is mainly decided by the number of requests and is faster than other KM-based algorithms (KM, AN, LACB).

Impact of # requests. The second column of Fig. 8 shows the results of varying $|R|$. The total utility generally increases as $|R|$ increases. Similar to the previous experiment, LACB and LACB-Opt achieve the same utility and perform better than other algorithms. As for the running time, the KM-based algorithms are slower than others. LACB-Opt is up to $16.4\times \sim 1091.9\times$ faster than KM, AL and LACB. LABT-Opt is also competitively fast compared with Top-K and CTop-K, especially with a small number of requests.

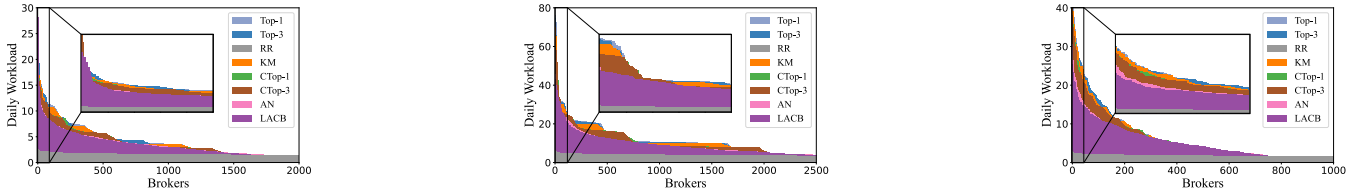
Impact of # covered days. The third column in Fig. 8 presents the results of varying the covered days Day. Our methods still outperform other baselines in terms of total utility. Particularly, AN yields less utility in covering seven days, indicating that it may face a cold start, while LACB consistently performs well when varying covering days. Again, LACB and LACB-Opt perform the same in terms of utility. As for the running time, a similar trend is observed as varying $|B|$. Our LACB-Opt is $65.5\times \sim 329.4\times$ faster than other KM-based algorithms.

Impact of the degree of imbalance. The fourth column in Fig. 8 plots the different imbalance ratios σ. Since we fix $|B|$ and change $|R|$ to test different imbalance ratios, all algorithms have similar trends in utility as σ increases. The acceleration of LACB-Opt over LACB is closely related to the imbalance ratio σ. For example, LACB-Opt is $641.7\times$ and $16.4\times$ faster than LACB when $\sigma = 0.005$ and $\sigma = 0.05$, respectively.



(a) City A (b) City B (c) City C

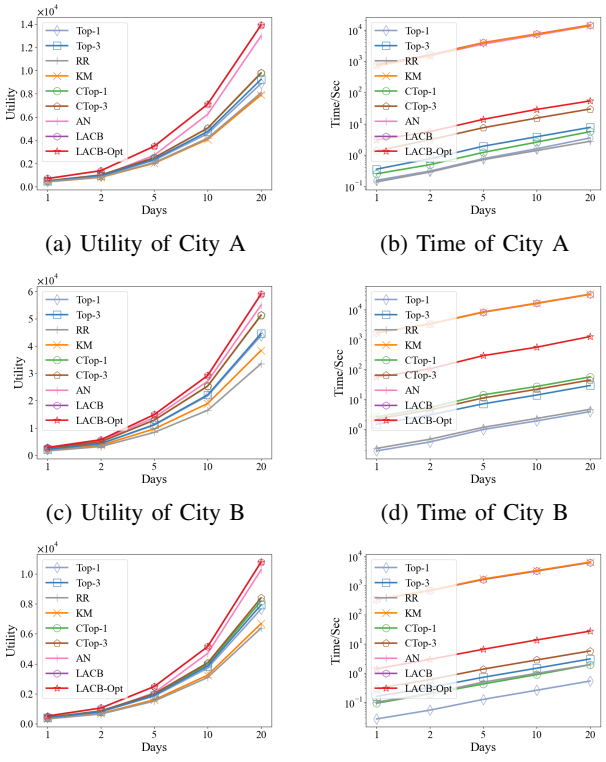
Fig. 9: The utility distribution of all compared algorithms. We have a further look on top brokers' utility.



(a) City A (b) City B (c) City C

Fig. 10: The workload distribution of all compared algorithms, where we can take Top-K as overloaded results.

C. Performance on Real Datasets



(a) Utility of City A (b) Time of City A

(c) Utility of City B (d) Time of City B

(e) Utility of City C (f) Time of City C

Fig. 11: Results on real-world datasets.

Overall Performance. Fig. 11 presents the results on three real-world datasets. Take City A as an example and we can first make the following observations in total utility. As expected, Top-K performs poorly on all three datasets. Top-3 slightly outperforms Top-1 because Top-1 more easily overloads the recommended brokers. CTop-K improves the total utility over Top-K, indicating the necessity of capacity awareness. AN outperforms most baselines due to contextual bandits while our LACB and LACB-Opt outperform AN. The running time increases linearly over days. Similar to the results on synthetic datasets, KM, AN, and LACB are the slowest due to their cubic time complexity. LACB-Opt is competitive in efficiency, only $1.7 \sim 24.2$ seconds slower than Top-K and CTop-K and $233.4\times \sim 284.9\times$ faster than other KM-based algorithms. We have similar observations on data from all three cities.

In-depth Analysis of Brokers. To understand the gain of our algorithms over baselines, we take a closer look at the distributions of both utility and workload of baselines. As aforementioned, LACB-Opt only differs from LACB in efficiency, so there is no need to include LACB-Opt in the following analysis. As we focus on the utility/workload of top brokers, we only show brokers with higher utility or workload. The utility/workload of other brokers exhibits a similar long-tail distribution as Fig. 9 and Fig. 10, and is omitted here.

- *Utility Distribution*. Fig. 9 plots the utility distribution of all algorithms. Take City A as an example. Capacity-based assignment algorithms (CTop-K, AN, and LACB) achieve higher utility than Top-K for most brokers, *i.e.*, avoiding the overloaded phenomenon is also beneficial at individual levels. Particularly, 80.8% brokers in LACB have an improvement in utility compared with Top-K. RR results in closer utilities for most brokers, as it randomly apportions requests to all brokers. However, RR decreases the utility of 25.7% brokers compared with Top-K. Similar observations are found in City B and City C. To summarize, LACB can improve the utility of both top brokers and the remaining brokers.
- *Workload Distribution*. Fig. 10 plots the workload distribution of all algorithms. As expected, Top-K leads to the highest workload of top brokers. RR randomly apportions requests and results in the lowest workload of top brokers. Yet it also prevents top brokers from serving more even if they have spare capacity, thus limiting the potential of top brokers. Except for RR, the workloads of top brokers in LACB are the lowest, which means that top brokers in LACB are at low risk of overload.

To summarize, LACB outperforms baselines thanks to its suitable workload capacity for top brokers.

D. Summary of Experimental Results

We summarize our major experimental findings as follows.

- CTop-K outperforms Top-K on all datasets, indicating the necessity of setting a workload capacity.
- Our solutions, LACB and LACB-Opt, generally outperform other baselines in total utility. They also improve 72.0%$\sim$82.2% brokers' utility compared with Top-K.
- Without loss of utility, LACB-Opt is up to $284.9\times$ faster than other KM-based algorithms on real-world datasets, which is consistent with our theoretical analysis.

VIII. Related Work

Our work is mainly related to three lines of research: *online task assignment*, *data science for real estate*, and *contextual bandits*. We review the representative work as follows.

Online Task Assignment. Task assignment is the core operation of crowdsourcing [25]–[29]. We focus on the online task assignment and summarize previous work into two categories: the vertex-based mode and the batch-based mode. *(i) vertex-based mode*. In [30], Kazemi *et al.* study the bipartite matching in spatial crowdsourcing and adopt greedy-based assignment algorithms to optimize total utility. Tong *et al.* [31] conduct the experimental study on online bipartite matching and show that the greedy algorithm is competitive in many practical settings. Later studies explore task assignment with different objectives or constraints, such as fairness [23], [32], [33], privacy [34]–[36] and incentive [37], [38]. Particularly, the fair-aware task assignment algorithm in [23] can be extended as a baseline to avoid the overloaded phenomenon in our scenario. *(ii) batch-based mode*. This mode prevails in ride-hailing [10], [11], [13]. Wang *et al.* [10] adopt reinforcement learning to adaptively adjust the time window of each batch in the assignment. Zhang *et al.* [11] solve the batched task assignment by a heuristic hill-climbing method. Authors in [13] and [39] take the value function to optimize the sequential decision-making over a long horizon, which inspires one of the core ideas of our method. Existing solutions mainly focus on the assignment strategies and assume a known capacity, however, our scenario is more challenging as we need to estimate the personalized capacity before assignments.

Data Science For Real Estate. The availability of big data has stimulated the interest in adopting data-driven approaches in the real estate industry [3]. Prior work mainly focus on two topics, housing appraisal [40] and housing finding [41]. In [40], Fu *et al.* enhance real estate appraisal by modeling dependencies of nearby estates and affiliated business areas. Zhang *et al.* [42] propose the graph neural networks based method for the asynchronously spatio-temporal dependencies. Grbovic *et al.* [41] design embedding techniques for real-time personalized housing searching. In [43], Weng *et al.* improve the home-finding service by analyzing the reachability between the home locations and concerned POIs. In this work, we investigate broker matching, an essential issue for online real estate platforms yet has rarely been optimized from a data-driven perspective.

Contextual Bandits. The contextual bandit utilizes additional information to make decisions, which has been adopted in recommender system [8], query optimization [44] and dynamic pricing [37]. Early efforts mainly focus on the theoretical analysis of regret bound [45], [46]. Li *et al.* [8] propose the LinUCB algorithm, which combines a linear predictor with upper bound confidence algorithms in 2010 and follow-up researchers extend the basic framework of LinUCB for efficient learning [47] and latent relations representation [48]. Recently, some work utilize neural networks to optimize the contextual bandit [9], [44], [49]. Zhou *et al.* [9] design a neural contextual bandit algorithm without assumption about the reward function, which can be extended as a baseline in our scenario. Marcus *et al.* [44] propose contextual bandits with tree convolutional neural networks to learn query execution strategies in databases. Ban *et al.* [49] propose multi-facet contextual bandits, where each facet is designed to characterize one of users' needs. However, these algorithms are not directly applicable in case of personalized estimation, as is in our capacity-aware broker matching scenario. Our LACB is built upon this thread of research but improves prior studies by adopting layer transfer for more data-efficient bandit learning.

IX. Conclusion

In this paper, we investigate the overloaded phenomenon in broker matching for online real estate platforms. Specifically, we find that the top-k recommendation mechanism adopted by mainstream platforms for broker matching tends to overwhelm top brokers, which decreases both sign-up rates and total utility. The root cause lies in the ignorance of the broker's workload capacity. In response, we propose LACB, which effectively estimates the personalized capacity by transferable neural contextual bandits and assigns brokers to clients from a global view. We further propose LACB-Opt, which largely improves the efficiency of LACB on an imbalanced bipartite graph. Extensive evaluations on three cities based on a real-world online real estate platform demonstrate the effectiveness and efficiency of our approach. We envision our work as an insightful reference for a wide spectrum of service industries where workers have limited workload capacity.

X. Acknowledgements

We are grateful to anonymous reviewers for their constructive comments. This work is partially supported by the National Key Research and Development Program of China under Grant No. 2018AAA0101100, the National Science Foundation of China (NSFC) under Grant No. U21A20516, U1811463 and 62076017, WeBank Scholars Program, and the Basic Research Funding in Beihang University No. YWF-22-L-531. Yongxin Tong is the corresponding author in this paper.

REFERENCES

[1] H. Zuo, "The rise of platform-based networks: Outlook for the real estate brokerage industry," in *Understanding China's Real Estate Markets*. Springer, 2021, pp. 299–305.

[2] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *RecSys*, 2010, pp. 39–46.

[3] R. Bekkerman, V. Josifovski, and F. J. Provost, "Data science for the real estate industry," in *KDD*. ACM, 2020, pp. 3559–3560.

[4] D. Rigney, *The Matthew effect: How advantage begets further advantage*. Columbia University Press, 2010.

[5] Y. Zhao, K. Zheng, Y. Cui, H. Su *et al.*, "Predictive task assignment in spatial crowdsourcing: A data-driven approach," in *ICDE*. IEEE, 2020, pp. 13–24.

[6] Y. Zheng, J. Wang, G. Li, R. Cheng *et al.*, "QASCA: A quality-aware task assignment system for crowdsourcing applications," in *SIGMOD*. ACM, 2015, pp. 1031–1046.

[7] Y. Tong, J. She, B. Ding, L. Wang *et al.*, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE*. IEEE, 2016.

[8] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *WWW*. ACM, 2010, pp. 661–670.

[9] D. Zhou, L. Li, and Q. Gu, "Neural contextual bandits with ucb-based exploration," in *ICML*, vol. 119. PMLR, 2020, pp. 11 492–11 502.

[10] Y. Wang, Y. Tong, C. Long, P. Xu, *et al.*, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *ICDE*. IEEE, 2019, pp. 1478–1489.

[11] L. Zhang, T. Hu, Y. Min, G. Wu *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *KDD*. ACM, 2017, pp. 2151–2159.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. KDD*. ACM, 2016, pp. 785–794.

[13] Z. Xu, Z. Li, Q. Guan, D. Zhang *et al.*, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *KDD*. ACM, 2018, pp. 905–913.

[14] T. Lattimore and C. Szepesvári, *Bandit algorithms*. CU Press, 2020.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[16] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," in *NeurIPS*, 2018, pp. 3839–3848.

[17] Z. Chen, P. Cheng, Y. Zeng, and L. Chen, "Minimizing maximum delay of task assignment in spatial crowdsourcing," in *ICDE*. IEEE, 2019, pp. 1454–1465.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT press, 2009.

[19] T. Abeywickrama, V. Liang, and K. Tan, "Optimizing bipartite matching in real-world applications by incremental cost computation," *VLDB*, vol. 14, no. 7, pp. 1150–1158, 2021.

[20] P. Shi, M. Zhao, W. Wang, Y. Zhou *et al.*, "Best of both worlds: Mitigating imbalance of crowd worker strategic choices without a budget," *Knowl. Based Syst.*, vol. 163, pp. 1020–1031, 2019.

[21] L. Ramshaw and R. Tarjan, "On minimum-cost assignments in unbalanced bipartite graphs," 2012.

[22] F. F. Nah, "A study on tolerable waiting time: how long are web users willing to wait?" *Behav. Inf. Technol.*, vol. 23, no. 3, pp. 153–163, 2004.

[23] F. Basik, B. Gedik, H. Ferhatosmanoglu, and K.-L. Wu, "Fair task allocation in crowdsourced delivery," *IEEE Trans. Serv. Comput.*, vol. 14, no. 4, pp. 1040–1053, 2021.

[24] K. Christakopoulou, J. Kawale, and A. Banerjee, "Recommendation with capacity constraints," in *CIKM*. ACM, 2017, pp. 1439–1448.

[25] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal*, vol. 29, pp. 217–250, 2020.

[26] X. Yu, G. Li, Y. Zheng, Y. Huang *et al.*, "Crowdota: An online task assignment system in crowdsourcing," in *ICDE*. IEEE, 2018, pp. 1629–1632.

[27] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," in *ICDE*. IEEE, 2017, pp. 39–40.

[28] S. Amer-Yahia and S. B. Roy, "Human factors in crowdsourcing," *VLDB*, vol. 9, no. 13, pp. 1615–1618, 2016.

[29] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan *et al.*, "Task assignment optimization in knowledge-intensive crowdsourcing," *The VLDB Journal*, vol. 24, no. 4, pp. 467–491, 2015.

[30] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *GIS*. ACM, 2012, pp. 189–198.

[31] Y. Tong, J. She, B. Ding, L. Chen *et al.*, "Online minimum matching in real-time spatial data: Experiments and analysis," *VLDB*, vol. 9, no. 12, pp. 1053–1064, 2016.

[32] D. Shi, Y. Tong, Z. Zhou *et al.*, "Learning to assign: Towards fair task assignment in large-scale ride hailing," in *KDD*. ACM, 2021, pp. 3549–3557.

[33] Y. Zhao, K. Zheng, J. Guo, B. Yang *et al.*, "Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches," in *ICDE*. IEEE, 2021, pp. 265–276.

[34] Q. Tao, Y. Tong, Z. Zhou, Y. Shi *et al.*, "Differentially private online task assignment in spatial crowdsourcing: A tree-based approach," in *ICDE*. IEEE, 2020, pp. 517–528.

[35] H. To, C. Shahabi, and L. Xiong, "Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server," in *ICDE*. IEEE, 2018, pp. 833–844.

[36] Y. Xu, S. Wei, and Y. Wang, "Privacy preserving online matching on ridesharing platforms," *Neurocomputing*, vol. 406, pp. 371–377, 2020.

[37] Y. Tong, L. Wang, Z. Zhou, L. Chen *et al.*, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *SIGMOD*. ACM, 2018, pp. 773–788.

[38] H. Cheng, S. Wei, L. Zhang, Z. Zhou *et al.*, "Engaging drivers in ride hailing via competition: A case study with arena," in *MDM*. IEEE, 2021, pp. 19–28.

[39] D. Shi, Y. Tong, Z. Zhou *et al.*, "Adaptive task planning for large-scale robotized warehouses," in *ICDE 2022*. IEEE, 2022, pp. 3327–3339.

[40] Y. Fu, H. Xiong, Y. Ge, Z. Yao *et al.*, "Exploiting geographic dependencies for real estate appraisal: a mutual perspective of ranking and clustering," in *KDD*. ACM, 2014, pp. 1047–1056.

[41] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *KDD*. ACM, 2018, pp. 311–320.

[42] W. Zhang, H. Liu, L. Zha, H. Zhu *et al.*, "Mugrep: A multi-task hierarchical graph representation learning framework for real estate appraisal," in *KDD*. ACM, 2021, pp. 3937–3947.

[43] D. Weng, H. Zhu, J. Bao, Y. Zheng *et al.*, "Homefinder revisited: Finding ideal homes with reachability-centric multi-criteria decision making," in *CHI*. ACM, 2018, p. 247.

[44] R. Marcus, P. Negi, H. Mao, and N. T. and, "Bao: Making learned query optimization practical," in *SIGMOD*. ACM, 2021, pp. 1275–1288.

[45] D. Bouneffouf, I. Rish, and C. C. Aggarwal, "Survey on applications of multi-armed and contextual bandits," in *CEC*. IEEE, 2020, pp. 1–8.

[46] J. Langford and T. Zhang, "The epoch-greedy algorithm for multi-armed bandits with side information," in *NeurIPS*, 2007, pp. 817–824.

[47] D. N. Hill, H. Nassif *et al.*, "An efficient bandit algorithm for realtime multivariate optimization," in *KDD*. ACM, 2017, pp. 1813–1821.

[48] H. Wang, Q. Wu, and H. Wang, "Learning hidden features for contextual bandits," in *CIKM*. ACM, 2016, pp. 1633–1642.

[49] Y. Ban, J. He, and C. B. Cook, "Multi-facet contextual bandits: A neural network perspective," in *KDD*. ACM, 2021, pp. 35–45.