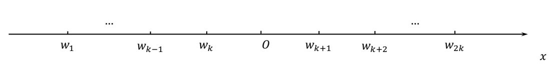
Adaptive Holding for Online Bottleneck Matching with Delays (Supplementary Materials)

Kaixin Wang* Cheng Long* Yongxin Tong† Jie Zhang* Yi Xu†

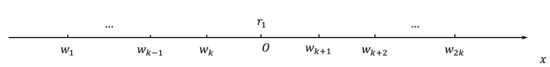
1 Proof of Theorem 2.1

Proof. We prove by constructing an instance with $n = 2k$ workers w_{1-2k} and requests r_{1-2k} located on an axis. We assume that the cost between a worker and a request is equal to the distance between their locations. Specifically, the workers and requests arrive as follows, which are shown in Figure 1.

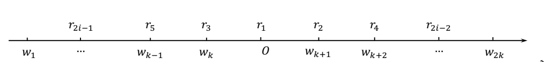
1. At time $t_0 = 0$, workers w_1 to w_{2k} arrive at $-k, -k+1, ..., -1, 1, ..., k-1, k$, respectively.
2. At time $t_1 = 1$, request r_1 arrives at $l_{r_1} = 0$;
3. At time $t_i = i$, r_{2i-2} arrives at $l_{r_{2i-2}} = i-1$ and r_{2i-1} arrives at $l_{r_{2i-1}} = -i+1$ for $i = 2, 3, \cdots, k$. After the first $2k-1$ requests arrive, we wait until $t_{k+1} = 2k$.
4. At time $t_{k+1} = 2k$, request r_{2k} arrives at either k or $-k$ with equal probabilities, which yields a probability distribution X over the input of requests.



(a) At time $t_0 = 0$, all workers arrive.



(b) At time $t_1 = 1$, request r_1 arrives.



(c) At time $t_i = i$, requests r_{2i-2} and r_{2i-1} arrive.

Figure 1: Arriving dynamics of workers and requests

The objective of the offline optimal solution would be equal to 1, which is explained as follows. In the case that request r_{2k} arrives at k, then any request on the positive side can be assigned to a certain worker whose location is the same as the request's while any request on the negative side can be assigned to a certain worker whose location is one unit left to the request. In the case that request r_{2k} arrives at $-k$, then any request on the positive side can be assigned to a certain worker whose location is one unit right to the request while any request on the negative side could be assigned to a certain worker whose location is the same as the request's.

For any online deterministic algorithms, they can be classified into two types according to the way how it assigns the first $2k-1$ requests, denoted by $\mathcal{A}_1$ and $\mathcal{A}_2$.

The first set of algorithms $\mathcal{A}_1$ is to match some of the first $2k-1$ workers after r_{2k}'s arrival. Then, there exists at least one request whose waiting time is greater or equal than k. Then the maximum of all requests' waiting time is no less than k. Thus, for the first type, the competitive ratio is no less than $\frac{k}{1} = k = \frac{n}{2}$.

The other type $\mathcal{A}_2$ is to match all the first $2k-1$ requests before the arrival of r_{2k}. Note that after r_{2k-1} has been matched, there would always be exactly one worker remaining, and we denote its location by $s \in \mathbb{Z}$. Because request r_{2k} would be at k or $-k$ with equal probabilities, the expectation of the cost value under the input distribution X is at least

$$\min_{A \in \mathcal{A}_2} \mathbb{E}\big[obj(A(X))\big] = \frac{1}{2}(s+k) + \frac{1}{2}(k-s) = k$$

Since no deterministic online algorithm can achieve a result better than $\frac{n}{2}$ under the distribution X, by using Yao's principle [2], for any randomized algorithm, we have

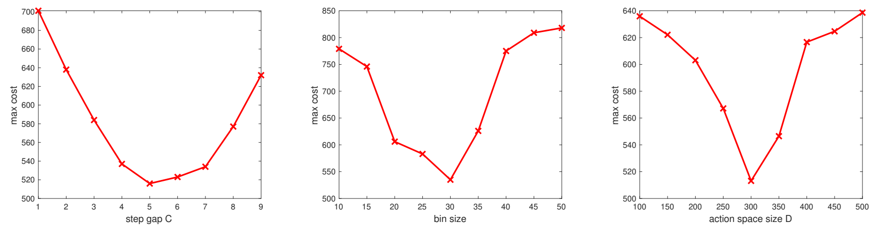
$$\max_{x \in \mathcal{X}} \mathbb{E}\big[obj(A(x))\big] \geqslant \min_{A \in \mathcal{A}_2} \mathbb{E}\big[obj(A(X))\big] = k = \frac{n}{2}$$

Therefore, we conclude that no randomized algorithm for the OBM-D problem with n workers (requests) can achieve a competitive ratio better than $\frac{n}{2}$. □

2 Additional Experiments and Results

Synthetic Datasets. We follow the existing study [1] for generating synthetic datasets. Specifically, we assume that both requests and workers are located in the grid cells of a 2D spatial space, and the cost between a request and a worker is defined as the number of grid cells the worker needs to traverse in order to reach the grid cell in which the request is

*Nanyang Technological University, Singapore, {kaixin.wang, c.long, zhangj}@ntu.edu.sg
†Beihang University, China, {yxtong, xuy}@buaa.edu.cn

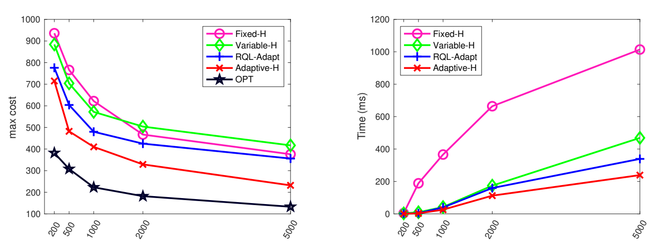


(a) max cost of varying step gap (b) max cost of varying bin size (c) max cost of varying action space size

Figure 2: Results on varying time step gap C, bin size W and action space size D on synthetic dataset

located. We explore two distributions, namely Uniform and Gaussian, with appropriate parameters for generating the locations (or grid cells) of requests and workers, and use the Uniform distribution as the default one. In addition, we assume that the same number n of requests and workers arrive dynamically following some distribution during a predefined time window $(0, 1, 2, ..., t_{max})$. We explore three distributions, namely Uniform, Gaussian, and Zipf, with appropriate distribution parameters for the arriving times, and use the Uniform distribution as the default one. For experiments on synthetic datasets, given a setting, the datasets are generated 20 times and the average results are reported.

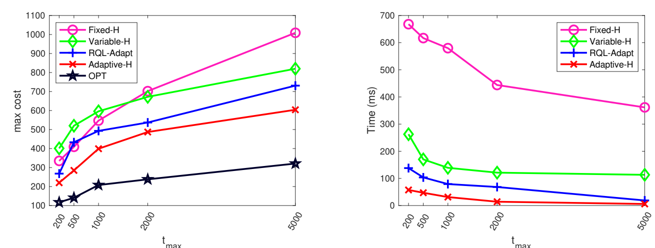
Hyperparameter Selection. In Figure 2, we show the results of the performance of our method when varying one of the hyperparameters while fixing the other two with their default settings under the synthetic dataset. On the synthetic dataset, our method has the best performance under the settings of $C = 5$, $W = 30$, and $D = 300$. On the real datasets Didi and Olist, we also adopt the same process for tuning the hyperparameters, and the results show the same trends. We have included the best hyperparameter setting for each real dataset with our codes at *https://anonymous.4open.science/r/0466e3c1-7381-410e-bc53-400bd9b1ed5f/*.



(a) Max cost (b) Running time

Figure 3: Results of varying cardinality (synthetic datasets)

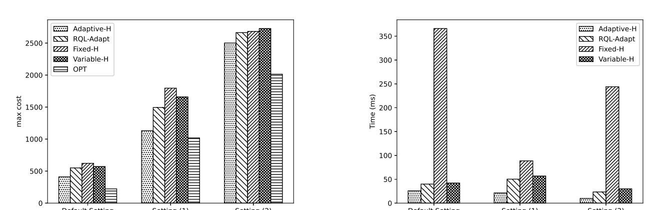
(1) Effects of Cardinality n **(Synthetic Datasets)**. The results of varying the cardinality n, i.e., the number of requests/workers, are presented in Figure 3, where $t_{\max} = 2000$. Consider the maximum cost (Figure 3(a)). We observe that it decreases when n increases. This could be explained by the fact that with more requests and workers, the density would be higher and in general it would take less time for a worker to travel to a request. Besides, our Adaptive-H algorithm outperforms other algorithms over all settings. Consider the running time (Figure 3(b)). All algorithms have their running times increase when n increases. Besides, Adaptive-H runs faster than other algorithms over all settings.



(a) Max cost (b) Running time

Figure 4: Results of varying $t_{\max}$ (synthetic datasets)

(2) Effects of $t_{\max}$ **(Synthetic Datasets)**. The results of varying $t_{\max}$, i.e., the maximum time steps, are presented in Figure 4, where $n = 1000$. All algorithms return smaller maximum costs when $t_{\max}$ increases. This could be explained by the fact that when $t_{\max}$ gets larger, which implies that the requests and workers arrive within a longer period of time, the requests would usually need to wait for a longer time before a favorable worker could be matched to it, resulting a larger cost. Besides, Adaptive-H outperforms other algorithms in terms of maximum cost and running time over all settings.



(a) Max cost (b) Running time

Figure 5: Results of varying distributions

(3) Effects of Distribution Settings (Synthetic Datasets). We denote the distributions Uniform (location), Gaussian

$N(\mu_s = 500, \sigma_s = 50)$ (location), Uniform (arriving time), Gaussian $N(\mu_t = \frac{t_{max}}{2}, \sigma_t = 200)$ (arriving time), and Zipf $f(N = t_{\max}, s = 2)$ (arriving time) by L_1, L_2, T_1, T_2, and T_3, respectively. We follow [1] and test two specific distribution settings (apart from the default one, i.e., (L_1, T_1) for both requests and workers): (1) (L_2, T_2) for requests and (L_1, T_1) for workers; and (2) (L_1, T_3) for requests and (L_1, T_1) for workers. The results are shown in Figure 5. We observe that Adaptive-H enjoys its advantages of effectiveness and efficiency across different distribution settings. In addition, consider the Setting (1). The maximum costs become larger compared with those under the default distribution setting. This could be explained as follows. First, since requests' locations follow the Gaussian distribution, they tend to arrive near the center. For those workers whose locations are far from the center, they would need to travel a long distance. Second, since the requests' arrival times follow the Gaussian distribution, the requests tend to arrive at around $\frac{t_{\max}}{2}$. Some workers would arrive late and the waiting times of requests would be large. Consider the Setting (2). Since the requests' arrival times follow the zipf distribution, up to 80% of requests arrive early, whereas workers' arrival times follow a Uniform distribution. As a result, the maximum costs under this distribution setting become even larger. We also notice that Fixed-H runs exceptionally slow for the default setting and Setting (2). This is possibly because for these settings, the requests and workers arrive in a more scattered way and Fixed-H would perform more matchings.

References

[1] Z. Chen, P. Cheng, Y. Zeng, and L. Chen, *Minimizing maximum delay of task assignment in spatial crowdsourcing*, in 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 1454–1465.

[2] A. C.-C. Yao, *Probabilistic computations: Toward a unified measure of complexity*, in 18th Annual Symposium on Foundations of Computer Science (sfcs 1977), IEEE, 1977, pp. 222–227.