https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH

Submitted to *Operations Research*

Vol. 00, No. 0, Xxxxx 0000, pp. 000–000

ISSN 0030-364X, EISSN 1526-5463

Potential-Based Greedy Matching for Dynamic Delivery Pooling

(Authors' names are not included for peer review)

Abstract. We study the pooling of multiple orders into a single trip, a strategy widely adopted by online delivery platforms. When an order has to be dispatched, the platform must determine which (if any) of the other available orders to pool it with, weighing the immediate efficiency gains against the uncertain, differential benefits of holding each order for future pooling opportunities. In this paper, we demonstrate the effectiveness of using the length of each job as its opportunity cost, via a *potential-based greedy algorithm* (PB). The algorithm is very simple, pooling each departing job with the available job that maximizes the savings in travel distance minus a half of its distance (i.e. the *potential*). On the theoretical front, we show that PB significantly improves upon a naive greedy algorithm in terms of worst-case performance: as the density of the market increases, the *regret per job* vanishes under PB but remains constant under naive greedy. In addition, we show that the potential approximates the marginal cost of dispatching each job in a stochastic setting with sufficient density. Finally, we conduct extensive numerical experiments and show that despite its simplicity, our notion of potential consistently outperforms forecast-aware heuristics that estimate the marginal costs of dispatching different jobs using historical data. Across the greedy dispatch policies we study, PB achieves the strongest performance. Moreover, compared with batching-based heuristics widely used in practice, PB achieves comparable performance with substantially lower computation, and these policies can also benefit from incorporating potential to better capture opportunity costs.

Key words: On-demand delivery, Platform operations, Online algorithms, Dynamic matching

1. Introduction

On-demand delivery platforms have become an integral part of modern life, transforming how consumers search for, purchase, and receive goods from restaurants and retailers. Collectively, these platforms serve more than 1.5 billion users worldwide, contributing to a global market valued at over $250 billion (Statista 2024). Growth has continued at a rapid pace—major companies such as DoorDash in the United States and Meituan in China reported approximately 25% year-over-year growth in transaction volume in 2023 (Curry 2025, Jiang 2024). Managing this ever-expanding stream of orders poses significant operational challenges, particularly as consumers demand increasingly faster deliveries, and fierce competition compels platforms to continuously improve both operational efficiency and cost-effectiveness.

One strategy for improving efficiency and reducing labor costs is to *pool* multiple orders from the same or nearby restaurants into a single trip. This is referred to as "stacked", "grouped", or "batched" orders, and is advertised to drivers as opportunities for increasing earnings and efficiency (DoorDash 2024). The strategy has been generally successful and very widely adopted (UberEats 2025). For example, data from Meituan, made public by the 2024 INFORMS TSL Data-Driven Research Challenge (Zhao et al. 2024), show that more than 85% of orders are pooled, rising to 95% during peak hours and consistently remaining above 50% at all other times (see Figure 1).

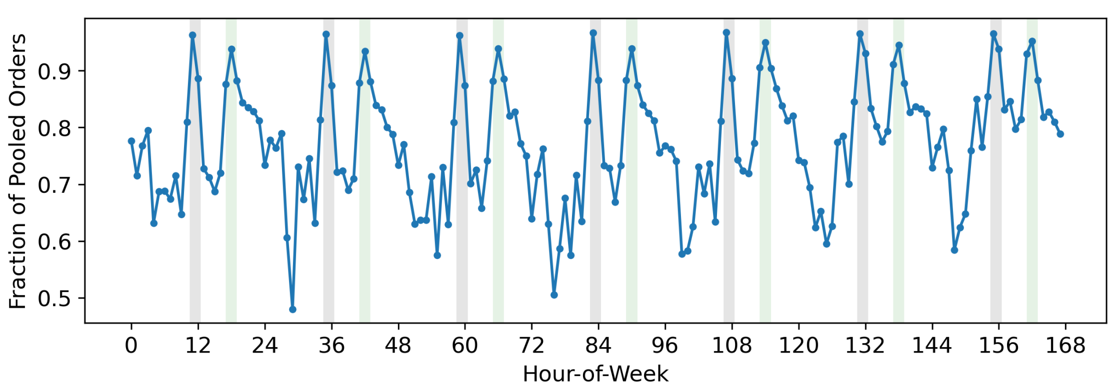


Figure 1 Fraction of pooled orders by hour-of-week in Meituan data. The dataset includes 8 days of order-level data from one city (see Subsection 4.3 for more details). The gray shade indicates peak lunch hours (10:30am-1:30pm), while the green shade indicates peak dinner hours (5pm-8pm).

The delivery pooling approach introduces a highly complex decision-making problem, as orders arrive dynamically to the market and need to be dispatched within a few minutes—orders in the Meituan data are offered to delivery drivers an average of around five minutes after being placed, often before meal preparation is complete (see Figure 11). During this limited window of each order, the platform must determine which (if any) of the available orders to pool with, carefully weighing immediate efficiency gains against the uncertain differential benefits of holding each order for future pooling opportunities. Since a single city may observe over 12 thousand orders/hour during peak times (see Figure 10), a practically-viable algorithm also needs to be efficient and robust. According to Zhao et al. (2024), computational challenges are a primary reason why the Meituan platform currently favors *one-order-to-one-courier* assignments.

Similar problems have been studied in the context of various marketplaces such as ride-sharing platforms and kidney exchange programs, in a large body of work known as dynamic matching. However, we argue that the delivery pooling problem exhibits a fundamentally different *reward structure*, which can be exploited to design simple yet effective algorithms. First, many problems studied in the literature involve connecting two sides of a market, e.g. ridesharing platforms matching drivers to riders. In such bipartite settings, the presence of a large number of agents of identical or similar types (e.g., many riders requesting trips originating from the same area) typically leads to congestion and less efficient outcomes. Meanwhile, the kidney exchange problem is not bipartite, but long queues of "hard-to-match" patient-donor pairs can still form, when many pairs share the same combination of incompatible tissue types and blood types.

In stark contrast, it is unlikely for long queues to build up for delivery pooling, especially in dense markets, in that given enough orders, two of them will have closely situated origins and destinations, resulting in a good match. Illustrating further, it is actually desirable to have many customers request deliveries from similar origins to similar destinations, which we find to be the case in practice, with most requests originating from popular restaurants in busy city centers and ending in residential neighborhoods (see Appendix D.2). Pooling or "stacking" these similar orders together (before assigning them to a courier) drastically reduces total travel distance, effectively increasing the platform's delivery capacity.

In this work, we focus on the pooling problem, and exploit the distinct *reward topology* of delivery pooling where the best-case scenario for a job is to be pooled with a second job of identical origin and destination.

In this best case, the travel-distance-saved for each of the two jobs is half of the job's distance, which we refer to as the *potential* of the job. We demonstrate the effectiveness of using the potential as the opportunity cost of dispatching each job, via our *potential-based greedy algorithm* (PB). Upon the departure of each job, PB makes the pooling decision in order to maximize the travel distance saved minus the potential of the job to be pooled with. On the theoretical front, we prove that PB significantly improves upon a naive greedy approach in terms of worst-case performance, and that the potential approximates the marginal cost of dispatching each job in the stochastic setting, with sufficient density. Finally, we conduct extensive numerical experiments and show that despite its simplicity, our notion of potential consistently outperforms forecast-aware heuristics that estimate the marginal costs of dispatching different jobs using historical data. Across the greedy dispatch policies we study, PB achieves the strongest performance. Moreover, compared with batching-based heuristics widely used in practice, PB achieves comparable performance with substantially lower computation, and these policies can also benefit from incorporating potential to better capture opportunity costs.

1.1. Model Description

Different models of arrivals and departures have been proposed in the dynamic matching literature (see Subsection 1.3.1), and we believe our insight about potential is relevant across all of them. However, in this paper we focus on a single model well-suited for the delivery pooling application.

To elaborate, we consider a dynamic non-bipartite matching problem, with a total of n jobs arriving sequentially to be matched. Jobs are characterized by (potentially infinite) types $\theta \in \Theta$ representing features (e.g. locations of origin and destination) that determine the reward for the platform. Given the motivating application to pooled deliveries, we always model jobs' types as belonging to some metric space. Following Ashlagi et al. (2019), we assume that an unmatched job must be dispatched after d new arrivals, which we interpret as a known *internal* deadline imposed by the platform to incentivize timely service. The platform is allowed to make a last-moment matching decision before the job leaves, termed as the job becoming *critical*. At this point, the platform must decide either to match the critical job to another available one, collecting reward $r(\theta, \theta')$ where θ, θ' are the types of the matched jobs and r is a known reward function, or to dispatch the critical job on its own for zero reward. The objective is to maximize the total reward collected from matching the n jobs. We believe this to be an appropriate model for delivery platforms (cf. Subsection 1.3.1) because deadlines are known upon arrival and sudden departures (order cancellations) are rare.

Dynamic matching models can also be studied under different forms of information about the future arrivals. Some papers (e.g. Kerimov et al. 2024, Aouad and Saritaç 2020, Eom and Toriello 2023, Wei et al. 2023) develop sophisticated algorithms to leverage stochastic information, which is often necessary to derive theoretical guarantees under general matching rewards. In contrast, our heuristic does not require any knowledge of future arrivals, and our theoretical results hold in an "adversarial" setting where no stochastic assumptions are made. In our experiments, we consider arrivals generated both from stochastic distributions and real-world data, and find that our heuristic can outperform even algorithms that are given the correct stochastic distributions, under our specific reward topology.

1.2. Main Contributions

Notion of potential. As mentioned above, we design a simple greedy-like algorithm that is based purely on topology and reward structure, which we term potential-based greedy (PB). More precisely, PB defines the *potential* of a job type θ to be $p(\theta) = \sup_{\theta' \in \Theta} r(\theta, \theta')/2$, measuring the highest-possible reward obtainable from matching type θ. The potential acts as an opportunity cost, and the relevance of this notion arises in settings where the potential is heterogeneous across jobs. To illustrate, let $\Theta = [0, 1]$ and $r(\theta, \theta') = \min\{\theta, \theta'\}$, a reward function used for delivery pooling as we will justify in Section 2. Under this reward function, a job type (which is a real number) can never be matched for reward greater than its real value, which means that jobs with higher real value have greater potential. Our PB algorithm matches a critical job (with type θ) to the available job (with type θ') that maximizes

$$r(\theta, \theta') - p(\theta') = \min\{\theta, \theta'\} - \frac{1}{2}\theta', \tag{1}$$

being dissuaded to use up job types θ' with high real values. Notably, this definition does not require any forecast or partial information of future arrivals, but rather assumes full knowledge of the universe of possible job types.

Theoretical results. Our theoretical results assume $\Theta = [0, 1]$. We compare PB to the naive greedy algorithm GRE, which selects θ' to maximize $r(\theta, \theta')$, instead of $r(\theta, \theta') - p(\theta')$ as in (1). We first consider an offline setting ($d = \infty$), showing that under reward function $r(\theta, \theta') = \min\{\theta, \theta'\}$, our algorithm PB achieves regret $O(\log n)$, whereas GRE suffers regret $\Omega(n)$ compared to the optimal matching. Our analysis of PB is tight, i.e. it has $\Theta(\log n)$ regret. Building upon the offline analysis, we next consider the online setting ($d < n$), showing that PB has regret $\Theta(\frac{n}{d} \log d)$, which improves as d increases. This can be interpreted as there being $\frac{n}{d}$ "batches" in the online setting, and our algorithm achieving a regret of $O(\log d)$ per batch. Alternatively, it can be interpreted as the *regret per job* of our algorithm being $O(\frac{\log d}{d})$. By contrast, we show that the naive greedy algorithm GRE suffers a total regret of $\Omega(n)$, regardless of d.

For comparison, we also analyze two other reward topologies, still assuming $\Theta = [0, 1]$.

1. We consider the classical min-cost matching setting (Reingold and Tarjan 1981) where the goal is to minimize total match distance between points on a line, represented in our model by the reward function $r(\theta, \theta') = 1 - |\theta - \theta'|$. For this reward function, both PB and GRE have regret $\Theta(\log n)$; in fact, they are the same algorithm because all job types have the same potential. Like before, this translates into both algorithms having regret $\Theta(\frac{n}{d} \log d)$ in the online setting.

2. We consider reward function $r(\theta, \theta') = |\theta - \theta'|$, representing an opposite setting in which it is worst to match two jobs of the same type. For this reward function, we show that any index-based matching policy must suffer regret $\Omega(n)$ in the offline setting, and that both PB and GRE suffer regret $\Omega(n)$ (irrespective of d) in the online setting.

Our theoretical results are summarized in Table 1. As our model is a special case of Ashlagi et al. (2019), their 1/4-competitive randomized online edge-weighted matching algorithm can be applied, which translates in our setting to an $O(n)$ upper bound for regret (under any definition of reward). However, their regret does not improve with d, while we show that for specific reward topologies, the regret is $O(n\frac{\log d}{d})$ and hence does improve with d, using a completely different algorithm and analysis. We now outline how to prove our two main technical results, Theorems 1 and 2.

<table>
<tr><th>$\theta, \theta' \in [0,1]$</th><th>$r(\theta,\theta') = \min\{\theta,\theta'\}$</th><th>$r(\theta,\theta') = 1 - |\theta-\theta'|$</th><th>$r(\theta,\theta') = |\theta-\theta'|$</th></tr>
<tr><td>Regret of PB</td><td>Offline: $\Theta(\log n)$
(Theorem 1, Proposition 2)
Online: $\Theta(\frac{n}{d}\log d)$
(Theorem 2, Proposition 4)</td><td rowspan="2">Offline: $\Theta(\log n)$
Online: $\Theta(\frac{n}{d}\log d)$
(Appendix B.1)</td><td rowspan="2">Offline: $\Omega(n)$
Online: $\Omega(n)$
(Appendix B.2)</td></tr>
<tr><td>Regret of GRE</td><td>Offline: $\Omega(n)$
(Proposition 1)
Online: $\Omega(n)$
(Proposition 3)</td></tr>
</table>

Table 1 Summary of theoretical results under different reward functions $r: [0,1]^2 \to \mathbb{R}$. The total number of jobs is denoted by n, and the batch size in the online setting is approximately d.

Proof techniques. We establish an upper bound on the regret of PB by comparing its performance to the sum of the potential of all jobs. Under reward function $r(\theta,\theta') = \min\{\theta,\theta'\}$, the key driver of regret is the sum of the distances between jobs matched by PB. In Theorem 1, we study this quantity by analyzing the intervals induced by matched jobs in the offline setting ($d = \infty$). We show that the intervals formed by the matching output of PB constitute a *laminar set family*; that is, every two intervals are either disjoint or one fully contains the other. We then prove that more deeply nested intervals must be exponentially smaller in size. Finally, we use an LP to show that the sum of interval lengths remains bounded by a logarithmic function of the number of intervals.

In Theorem 2, we partition the set of jobs in the online setting into roughly $\frac{n}{d}$ "batches" and analyze the sum of the distances between matched jobs within a single batch. Our main result is to show that we can use Theorem 1 to derive an upper bound for an arbitrary batch. However, a direct application of the theorem on the offline instance defined by the batch would not yield a valid upper bound, because the resulting matching of PB in the offline setting could be inconsistent with its online matching decisions. To overcome this challenge, we carefully construct a modified offline instance that allows for the online and offline decisions to be coupled, while ensuring that the total matching distance did not go down. This allows us to upper-bound the regret per batch by $O(\log d)$, for a total regret of $O(\frac{n}{d}\log d)$.

Simulations on synthetic data. We test PB on random instances in the setting of our theoretical results, except that job types are drawn uniformly at random from $[0, 1]$ (instead of adversarial). We benchmark its performance against GRE, as well as more sophisticated algorithms, including (i) batching-based heuristics that are highly relevant both in theory and practice, and (ii) forecast-aware heuristics that use historical data to compute shadow prices. Our extensive simulation results show that PB consistently outperforms all benchmarks starting from relatively low market densities, achieving over 95% of the hindsight optimal solution that has full knowledge of arrivals.

Simulations on real data. Finally, we test the practical applicability of PB via extensive numerical experiments using order-level data from the Meituan platform, made available from the 2024 INFORMS TSL Data-Driven Research Challenge (Zhao et al. 2024). The key information we extract from this dataset is the exact timestamps for the creation of each request, and the geographic coordinates of pick-up and drop-off locations. These aspects differ from our theoretical setting in that (i) requests may not be available to be pooled for a fixed number of new arrivals before being dispatched, and (ii) locations are two-dimensional with delivery orders having heterogeneous origins. To address (i), we assume that the platform sets a fixed time window after which an order becomes critical, corresponding to each job being able to wait for at most a fixed sojourn time before being dispatched. For (ii), we extend our definition of reward (that captures the travel distance saved) to two-dimensional heterogeneous origins. Under this new reward definition, the main insight from our theoretical model remains true: longer deliveries have higher potential reward from being pooled with other jobs in the future, and thus PB aims to keep them in the system.

In contrast to our simulations with synthetic data, the real-life delivery locations may now be correlated and exhibit time-varying effects. Regardless, we find that PB improves significantly upon the naive greedy and forecast-aware greedy policies, achieving over 80% of the maximum possible travel distance saved from pooling Although batching-based policies can achieve better performance (over 90%), this comes at a high computational cost, which is at least one order of magnitude larger than PB. Moreover, we show that these policies can benefit from including shadow prices, increasing at least 1% in the particular case of PB.

Explanation for the surprising unbeatability of PB. Our potential-based greedy heuristic consistently performs at or near the best, across a wide range of experimental setups including both synthetic and real data. This result is surprising to us, given that PB is a simple index-based rule that ignores forecast information. To provide some explanation for this finding, we analyze a stylized setting in Appendix C, where we show that the "correct" shadow prices converge to our notion of potential as the market thickness increases.

That being said, we also end with two caveats. First, our findings about the effectiveness of PB are specific to our definition of matching reward for delivery pooling, based on travel distance saved, and may not extend to settings with different reward structures. Second, while we carefully tuned the batching-based and dual-based heuristics that we compare against, it remains possible that more sophisticated algorithms, particularly those using (approximate) dynamic programming in stochastic settings, could outperform PB.

1.3. Further Related Work

1.3.1. Dynamic matching. Dynamic matching problems have received growing attention from different communities in economics, computer science, and operations research. We discuss different ways of modeling the trade-off between matching now vs. waiting for better matches, depending on the application that motivates the study.

One possible model (Kerimov et al. 2024, 2025, Wei et al. 2023) considers a setting with jobs that arrive stochastically in discrete time and remain in the market indefinitely, but use a notion of *all-time regret* that evaluates a matching policy, at every time period, against the best possible decisions until that moment, to disincentivize algorithms from trivially delaying until the end to make all matches.

A second possible model, motivated by the risk of cancellation in ride-sharing platforms, is to consider sudden departures modeled by jobs having heterogeneous *sojourn times* representing the maximum time that they stay in the market (Aouad and Saritaç 2020). These sojourn times are unknown to the platform, and delaying too long risks many jobs being lost without a chance of being matched. Aouad and Saritaç (2020) formulates an MDP with jobs that arrive stochastically in continuous time, and leave the system after an exponentially distributed sojourn time. They propose a policy that achieves a multiplicative factor of the hindsight optimal solution.

A third possible model (Huang et al. 2018) also considers jobs that can leave the system at any period, but allows the platform to make a last-moment matching decision right before a job leaves, termed as the job becoming *critical*. In this model, one can without loss assume that all matching decisions are made at times that jobs become critical.

Finally, the model we study also makes all decisions at times that jobs become critical, with the difference being that the sojourn times are known upon the arrival of a job, as studied in Ashlagi et al. (2019), Eom and Toriello (2023). Under these assumptions, Eom and Toriello (2023) assume stationary stochastic arrivals, while Ashlagi et al. (2019) allow for arbitrary arrivals. Our research focuses on deterministic greedy-like algorithms that can achieve good performance as the market thickness increases. Moreover, our work differs from these papers in the description of the matching value. While they assume arbitrary matching rewards, we consider specific reward functions known to the platform in advance, and use this information to derive a simple greedy-like algorithm with good performance. This aligns with a stream of literature on online matching, that captures more specific features into the model to get stronger guarantees (see Chen et al. 2023, Balkanski et al. 2023).

We should note that our paper also relates to a recent stream of work studying the effects of batching and delayed decisions in online matching (e.g. Feng and Niazadeh 2024), as well as works studying the relationship between market thickness and quality of online matches (e.g. Ashlagi and Roth 2021).

1.3.2. Delivery operations. On-demand delivery routing has been widely studied using approximate dynamic programming techniques (e.g. Reyes et al. 2018, Ulmer et al. 2021) to tackle complex mathematical programs, often lacking theoretical performance guarantees and offering limited interpretability. Closer to our research goal, Chen and Hu (2024) analyze the optimal dispatching policy on a stylized queueing model representing a disk service area centered at one restaurant, while Cachon and Besbes (2023) study the interplay between the number of couriers and platform efficiency, assuming a one-dimensional geography with one single origin, which is also the primary model in our theoretical results.

2. Preliminaries

In this section, we introduce a dynamic non-bipartite matching model for the delivery pooling problem. A total of n jobs arrive to the platform sequentially. Each job $j \in [n] = \{1, \ldots, n\}$, indexed in the order of arrival, has a *type* $\theta_j \in \Theta$. We adopt the criticality assumption from Ashlagi et al. (2019), in the sense that each job remains available to be matched for $d \geq 1$ new arrivals, after which it becomes *critical*. The parameter d can also be interpreted as the market density. When a job becomes critical, the platform decides whether to (irrevocably) match it with another available job, in which case they are dispatched together (i.e. *pooled*) and the platform collects a *reward* given by a known function $r : \Theta^2 \to \mathbb{R}$. If a critical job is not pooled with another job, it has to be dispatched by itself for zero reward.

2.1. Reward Topology

Our modeling approach directly imposes structure on the type space, as well as the reward function that captures the benefits from pooling delivery orders together. Similar to previous numerical work on pooled trips in ride-sharing platforms (Eom and Toriello 2023, Aouad and Saritaç 2020), we assume that the platform's goal is to reduce the total distance that needs to be traveled to complete all deliveries. The reward of pooling two orders together is therefore the travel distance saved when they are delivered by the same driver in comparison to delivered separately. Formally, we consider a *linear city model* where job types $\theta \in \Theta = [0, 1]$ represent destinations of the delivery orders, and assume that all orders need to be served from the origin 0 (similar to that analyzed in Cachon and Besbes (2023)). If a job of type θ is dispatched by itself, the total travel distance from the origin 0 is exactly θ. If it is matched with another job of type θ', they are pooled together on a single trip to the farthest destination $\max\{\theta, \theta'\}$. Thus, the distance saved by pooling is

$$r(\theta, \theta') = \theta + \theta' - \max\{\theta, \theta'\} = \min\{\theta, \theta'\}. \tag{2}$$

Although our main theoretical results leverage the structure of this reward topology, our proposed algorithm can be applied to other reward topologies. In particular, we derive theoretical results for two other reward structures for $\Theta = [0, 1]$. First, we consider

$$r(\theta, \theta') = 1 - |\theta - \theta'| \tag{3}$$

to capture the commonly-studied spatial matching setting (e.g. Kanoria 2021, Balkanski et al. 2023), where the reward is larger if the distance $|\theta - \theta'|$ is smaller. We also consider

$$r(\theta, \theta') = |\theta - \theta'| \tag{4}$$

with the goal of matching types that are far away from each other, contrasting the other two reward functions. In addition, we perform numerical experiments for delivery pooling in two-dimensional (2D) space, where the reward function corresponds to the travel distance saved in the 2D setting.

2.2. Benchmark Algorithms

Given market density d and reward function r, if the platform had full information of the sequence of arrivals $\boldsymbol{\theta} \in \Theta^n$, the hindsight optimal $\mathsf{OPT}(\boldsymbol{\theta}, d)$ can be computed by the integer program (IP) defined in (5). We denote the hindsight optimal matching solution as $\mathcal{M}_{\mathsf{OPT}} = \{(j,k) : x^*_{j,k} = 1\}$, where x^* is an optimal solution of (5).

$$
\begin{aligned}
\mathsf{OPT}(\boldsymbol{\theta}, d) = \max_{x} \quad & \sum_{j,k: j\neq k, |j-k|\leq d} x_{jk} r(\theta_j, \theta_k) \\
\text{s.t.} \quad & \sum_{k: j\neq k} x_{jk} \leq 1, \qquad j \in [n], \\
& x_{jk} \in \{0,1\}, j,k \in [n], j \neq k.
\end{aligned}
\tag{5}
$$

An online matching algorithm operates over an instance $\boldsymbol{\theta} \in \Theta^n$ sequentially: when job $j \in [n]$ becomes critical, the types of future arrivals $k > j + d$ are unknown, and any matching decision has to be made based on the currently available information. Given an algorithm ALG, we denote by $\mathsf{ALG}(\boldsymbol{\theta}, d)$ the total reward collected by an algorithm on such instance. We analyze the performance of algorithms via *regret*, as follows:

$$\mathsf{Reg}_{\mathsf{ALG}}(\boldsymbol{\theta}, d) = \mathsf{OPT}(\boldsymbol{\theta}, d) - \mathsf{ALG}(\boldsymbol{\theta}, d).$$

We study a class of online matching algorithms that we call *index-based greedy matching algorithms*. Each index-based greedy matching algorithm is specified by an *index function* $q : \Theta^2 \to \mathbb{R}$, and only makes matching decisions when some job becomes critical (in our model, it is without loss of optimality to wait to match). A general pseudocode is provided in Algorithm 1. When a job $j \in [n]$ becomes critical, the algorithm observes the set of available jobs in the system $A(j) \subseteq [n]$ that have arrived but are not yet matched (this is the set $A \setminus \{j\}$ in Algorithm 1), and chooses a match $m(j) \in A(j)$ that maximizes the index function (even if negative), collecting a reward $r(\theta_j, \theta_{m(j)})$. If there are multiple jobs that achieve the maximum, the algorithm breaks the tie arbitrarily. The algorithm makes a set of matches $\mathcal{M} = \{(j, m(j)) : j \in C\}$, where C denotes the set of jobs that are matched when they become critical, and its total reward is

$$\mathsf{ALG}(\boldsymbol{\theta}, d) = \sum_{j \in C} r(\theta_j, \theta_{m(j)}).$$

Note that $m(j)$ is undefined if j is either unmatched, or was not critical at the time it was matched.

Algorithm 1 Index-based Greedy Matching Algorithm

Input: Instance $\boldsymbol{\theta}$, density d, index function q, reward function r

Output: $C, m : C \to [n]$

Initialize set of available jobs $A = \emptyset$, and jobs that are critical when matched $C = \emptyset$
for job $t = 1, \ldots, n + d + 1$ **do**
  if $t \leq n$ **then**
    $A = A \cup \{t\}$ ▹ job t arrives
  end if
  if $t - d \in A$ **then**
    $j = t - d$ ▹ job $j = t - d$ becomes critical
    if $A \setminus \{j\} \neq \emptyset$ **then**
      Choose $m(j) \in \arg\max_{k \in A \setminus \{j\}} q(\theta_j, \theta_k)$ ▹ Ties are broken arbitrarily
      $C \leftarrow C \cup \{j\}$
      $A \leftarrow A \setminus \{j, m(j)\}$ ▹ jobs $j, m(j)$ are dispatched together
    else
      $A \leftarrow A \setminus \{j\}$ ▹ job j is dispatched by itself
    end if
  end if
end for
return $C, \{m(j)\}_{j \in C}$

As an example, the *naive greedy* algorithm (GRE) uses index function $q_{\mathsf{GRE}}(\theta,\theta') = r(\theta,\theta')$. When a job becomes critical, the algorithm matches it with an available job to maximize the (instant) reward. To illustrate the behavior of this algorithm under $r(\theta,\theta') = \min\{\theta,\theta'\}$, we first make the following observation, that the algorithm always chooses to match each critical job with a higher type job when possible.

Remark 1. When a job $j \in [n]$ becomes critical, if the set $A_+(j) = \{k \in A(j) : \theta_k \geq \theta_j\}$ is nonempty, then under the naive greedy algorithm GRE, we have $m(j) \in A_+(j)$ and $r(\theta_j, \theta_{m(j)}) = \theta_j$.

3. Potential-Based Greedy Algorithm

We introduce in this section the potential-based greedy algorithm and prove that it substantially outperforms the naive greedy approach in terms of worst case regret.

The *potential-based greedy* algorithm (PB) is an index-based greedy matching algorithm (as defined in Algorithm 1) whose index function is specified as

$$q_{\mathsf{PB}}(\theta,\theta') = r(\theta,\theta') - p(\theta'), \tag{6}$$

where $p(\theta)$ is the *potential* of a job of type θ, formally defined as follows

$$p(\theta) = \frac{1}{2} \sup_{\theta' \in \Theta} r(\theta,\theta'). \tag{7}$$

This new notion of potential can be interpreted as an optimistic measure of the marginal value of holding on to a job that could be later matched with a job that maximizes instant reward. Intuitively, this "ideal" matching outcome appears as the optimal matching solution when the density of the market is arbitrarily large. We provide a formal statement of this intuition in Appendix C. Note that this definition of potential is quite simple in the sense that it relies on only in the knowledge of the reward topology: reward function and type space. In particular, for our topology of interest, $r(\theta,\theta') = \min\{\theta,\theta'\}$ on $\Theta = [0,1]$, the ideal scenario is achieved when two identical jobs are matched and the potential $p(\theta) = \theta/2$ is simply proportional to the length of a solo trip, capturing that longer deliveries have higher potential reward from being pooled with other jobs in the future.

We begin by giving a straightforward interpretation for this algorithm under the linear city model and our reward of interest, as we did for the naive greedy algorithm GRE in Remark 1.

REMARK 2. Under the 1-dimensional type space $\Theta = [0,1]$ and the reward function $r(\theta,\theta') = \min\{\theta,\theta'\}$, the PB algorithm always matches each critical job to an available job that's the closest in space, since

$$m(j) \in \arg\max_{k\in A(j)} q_{\mathsf{PB}}(\theta_j,\theta_k) = \arg\min_{k\in A(j)} |\theta_j - \theta_k|.$$

This follows from the fact that $|\theta_j - \theta_k| = \theta_j + \theta_k - 2\min\{\theta_j,\theta_k\} = \theta_j - 2q_{\mathsf{PB}}(\theta_j,\theta_k)$.

In the rest of this section, we first assume $d = \infty$ and study the *offline* performance of various index-based greedy matching algorithms under the reward function $r(\theta,\theta') = \min\{\theta,\theta'\}$. The performance of greedy procedures for offline matching is a fundamental question in its own right, but this analysis will also later help us analyze the algorithms' *online* performances when $d < \infty$.

Results for the two alternative reward structures defined in (3) and (4) are reported in Appendix B.

3.1. Offline Performance under Reward Function $r(\theta,\theta') = \min\{\theta,\theta'\}$

Suppose $d = \infty$, meaning that all jobs are available to be matched when the first job becomes critical. We study the *offline* performance of both the naive greedy algorithm GRE and potential-based greedy algorithm. We show that the regret of GRE grows linearly with the number of jobs, while the regret of PB is logarithmic.

PROPOSITION 1 (**proof in Appendix A.1**). *Under reward function $r(\theta,\theta') = \min\{\theta,\theta'\}$, when the number of jobs n is divisible by 4, there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ for which $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},\infty) \geq n/4$.*

In particular, the *regret per job* of GRE, i.e. dividing the regret by n, is constant in n. In contrast, we prove in the following theorem that PB performs substantially better. In fact, its regret per job gets better for larger market sizes n.

THEOREM 1. *Under reward function $r(\theta,\theta') = \min\{\theta,\theta'\}$, we have $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},\infty) \leq 1 + \log_2(n/2+1)/2$, for any n, and any instance $\boldsymbol{\theta} \in [0,1]^n$.*

Proof. Let $(C, m(\cdot))$ be the output of PB on $\boldsymbol{\theta}$ (generated as in Algorithm 1). In particular, $\mathsf{PB}(\boldsymbol{\theta},\infty) = \sum_{j\in C} r(\theta,\theta')$. On the other hand, since $r(\theta,\theta') \leq p(\theta) + p(\theta')$ for all $\theta,\theta' \in [0,1]$ and $p(\theta) = \theta/2$, we have

$\mathsf{OPT}(\boldsymbol{\theta},\infty) \le \sum_{j=1}^{n} p(\theta_j) \le \frac{1}{2} + \sum_{j\in C} \frac{\theta_j}{2} + \frac{\theta_{m(j)}}{2}$, where the last inequality comes from the fact that PB, and in fact any index-based matching algorithm, leaves at most one job unmatched (when n is odd). Hence,

$$\begin{aligned}\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},\infty) &= \mathsf{OPT}(\boldsymbol{\theta},\infty) - \mathsf{PB}(\boldsymbol{\theta},\infty)\\ &\le \frac{1}{2} + \sum_{j\in C} \frac{\theta_j}{2} + \frac{\theta_{m(j)}}{2} - \min\{\theta_j,\theta_{m(j)}\}\\ &= \frac{1}{2} + \sum_{j\in C} \frac{|\theta_j - \theta_{m(j)}|}{2}. \end{aligned} \tag{8}$$

To bound the right-hand side, we first prove the following. For each $j \in C$, consider the interval $I_j = (\min\{\theta_j,\theta_{m(j)}\},\max\{\theta_j,\theta_{m(j)}\}) \subseteq [0,1]$ and $\ell(j) = |\{k : I_j \subseteq I_k\}|$. We argue that

$$|\theta_j - \theta_{m(j)}| \le 2^{1-\ell(j)}. \tag{9}$$

First, note that $\{I_j\}_{j\in C}$ is a *laminar set family*: for every $j,k \in C$, the intersection of I_j and I_k is either empty, or equals I_j, or equals I_k. Indeed, without loss of generality, assume $j < k$. Then, when job j becomes critical we have $m(j),k,m(k) \in A(j)$. Therefore, from Remark 2, $m(j)$ is the closest to j and thus $k,m(k) \notin I_j$, i.e. either $I_j \cap I_k = \emptyset$, or $I_j \cap I_k = I_j$. Moreover, if $I_j \subsetneq I_k$ (i.e. $I_j \subsetneq I_k$ and $I_j \ne I_k$), then necessarily $j < k$, since otherwise k becomes critical first and having $j,m(j) \in A(k)$ closer in space, PB would not have chosen $m(k)$. Now, we can prove (9) by induction. If $\ell(j) = 1 = |\{j\}|$, clearly $|\theta_j - \theta_{m(j)}| \le 1$. Suppose that the statement is true for $\ell(j) = \ell \ge 1$. If $\ell(j) = \ell + 1$ then there exists k such that $I_j \subsetneq I_k$ and $\ell(k) = \ell$. By the previous property, $j < k$ and since when job j becomes critical we had $k,m(k) \in A(j)$, it must be the case that $|\theta_j - \theta_{m(j)}| < \max\{\min\{\theta_j,\theta_{m(j)}\} - \min\{\theta_k,\theta_{m(k)}\}, \max\{\theta_k,\theta_{m(k)}\} - \max\{\theta_j,\theta_{m(j)}\}\} \le \min\{\theta_j,\theta_{m(j)}\} - \min\{\theta_k,\theta_{m(k)}\} + \max\{\theta_k,\theta_{m(k)}\} - \max\{\theta_j,\theta_{m(j)}\} = |\theta_k - \theta_{m(k)}| - |\theta_j - \theta_{m(j)}|$. Thus, $|\theta_j - \theta_{m(j)}| \le |\theta_k - \theta_{m(k)}|/2 \le 2^{1-(\ell+1)}$, completing the induction.

Having established (9), we use it to derive an upper bound for $\sum_{j\in C} |\theta_j - \theta_{m(j)}|$. Note that since $\ell(j) \in \{1,\ldots,\lfloor n/2\rfloor\}$, we have

$$\sum_{j\in C} |\theta_j - \theta_{m(j)}| = \sum_{\ell=1}^{\lfloor n/2\rfloor} \sum_{j\in C:\ell(j)=\ell} |\theta_j - \theta_{m(j)}|. \tag{10}$$

For every ℓ, we have $\sum_{j\in C:\ell(j)=\ell}|\theta_j-\theta_{m(j)}|\le\min\{1,2^{1-\ell}|\{j:\ell(j)=\ell\}|\}$ by the laminar property and (9). Let z_ℓ denote $2^{1-\ell}|\{j:\ell(j)=\ell\}|$, where we note that $\sum_{\ell=1}^{\lfloor n/2\rfloor}2^{\ell-1}z_\ell=\sum_{\ell=1}^{\lfloor n/2\rfloor}|\{j:\ell(j)=\ell\}|=|C|\le n/2$. We can then use the following LP to upper-bound the value of (10) under an adversarial choice $\{z_\ell:\ell=1,\dots,\lfloor n/2\rfloor\}$:

$$\begin{aligned}\max_{z}\quad & \sum_{\ell=1}^{\lfloor n/2\rfloor} z_\ell\\ \text{s.t.}\quad & \sum_{\ell=1}^{\lfloor n/2\rfloor}2^{\ell-1}z_\ell\le n/2,\\ & 0\le z_\ell\le 1,\quad \ell=1,\dots,\lfloor n/2\rfloor\,.\end{aligned}\tag{11}$$

This is a fractional knapsack problem, whose optimal value is at most $\log_2(n/2+1)$, and thus

$$\sum_{j\in C}|\theta_j-\theta_{m(j)}|\le\log_2(n/2+1).\tag{12}$$

Substituting back into (8) yields $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},\infty)\le 1/2+\log_2(n/2+1)/2$, completing the proof. □

The following result shows that this analysis of PB is tight, up to constants.

PROPOSITION 2 (**proof in Appendix A.2**). *Under reward function $r(\theta,\theta')=\min\{\theta,\theta'\}$, when the number of jobs is $n=2^{k+3}-4$ for some integer $k\ge0$, there exists an instance $\boldsymbol{\theta}\in[0,1]^n$ for which* $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},\infty)\ge(\log_2(n+4)-3)/4$.

3.2. Online Performance under Reward Function $r(\theta,\theta')=\min\{\theta,\theta'\}$

We now study the performance of algorithms in the online setting, i.e. when $d<n$, and derive informative performance guarantees conditional on d, the number of new arrivals before a job becomes critical. In fact, the regret per job of GRE remains constant, whereas the one of PB decreases with d. To build upon our results proved in Subsection 3.1 for the offline setting, we partition the set of jobs into $b=\lceil n/(d+1)\rceil$ *batches*. To be precise, we define the t-th batch as $B_t=\{(t-1)(d+1)+1,\dots,t(d+1)\}\cap[n]$, for each $t=1,\dots,b$. Scaling d while keeping the number of batches b fixed can be interpreted as increasing the density of the market. We first show that the regret under GRE remains linear in the number of jobs n, independent of d, which can be understood as suffering a regret of order $\Omega(d)$ per batch.

Proposition 3 **(proof in Appendix A.3).** *Under reward topology* $r(\theta,\theta') = \min\{\theta,\theta'\}$*, if* $(d+1)$ *is divisible by 4, then for any number of jobs* n *divisible by* $(d+1)$*, there exists an instance* $\boldsymbol{\theta} \in [0,1]^n$ *for which* $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta}, d) \geq n/4$.

In contrast, the following theorem shows that the performance of PB improves as market density increases.

Theorem 2. *Under reward topology* $r(\theta,\theta') = \min\{\theta,\theta'\}$*, we have* $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}, d) \leq 1/2 + (\frac{n}{d+1} + 1)(1 + \log(d+2))/2$ *for any* n*, and any instance* $\boldsymbol{\theta} \in [0,1]^n$.

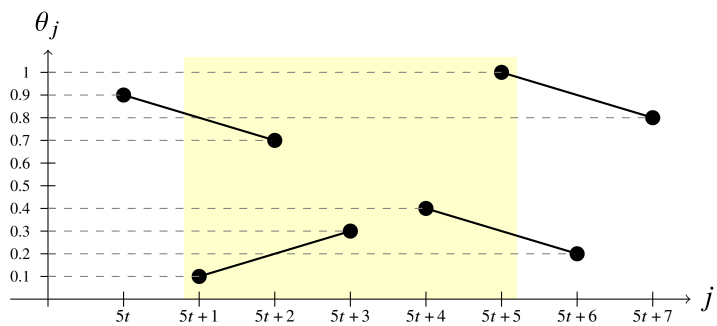
Proof. Let $d \geq 1$ and $(C, m(\cdot))$ be the output of PB on $\boldsymbol{\theta}$. Similar to the proof of Theorem 1, we have

$$\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}, d) \leq \frac{1}{2} + \frac{1}{2}\sum_{t=1}^{b}\left(\sum_{j\in C\cap B_t} |\theta_j - \theta_{m(j)}|\right), \tag{13}$$

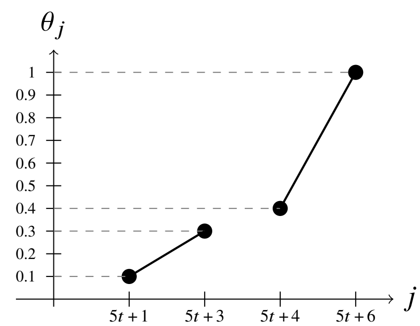
where we split the sum into the $b = \lceil n/(d+1)\rceil$ batches, defined as $B_t = \{(t-1)(d+1)+1, \ldots, t(d+1)\} \cap [n]$ for $t = 1, \ldots, b$. We analyze the term in large parentheses for an arbitrary t. Intuitively, we would like to consider an offline instance consisting of jobs $(\theta_j)_{j\in C\cap B_t}$ and their matches $(\theta_{m(j)})_{j\in C\cap B_t}$, and apply Theorem 1 on the offline instance which has size $2|C \cap B_t| \leq 2(d+1)$. However, since the offline setting allows jobs to observe the full instance, as opposed to only the next d arrivals, the resulting matching of PB on the offline instance could be inconsistent with its output on the online instance. In particular, executing PB on the offline instance could match jobs with indices more than d apart, which is not possible in the online setting. Thus, to derive a proper upper bound on regret, we need to construct a modified offline instance for which each resulting match of PB coincides with the matching output in the online counterpart, and in which matching distances were not decreased.

To construct this modified offline instance, first note that if $m(j) \in B_t$ for all $j \in C \cap B_t$, then no modification is needed, since all matched jobs have indices less than d apart. However, if $m(j) \in B_{t+1}$ for some $j \in C \cap B_t$, then $m(j)$ can be available in the offline instance for some job $k < j$ with $|m(j) - k| > d$ and be matched to k in the offline instance even though this would not be possible in the online instance (see Figure 2). Thus, in the modified offline instance, we "move" job $m(j)$ ensure job k would not choose

$m(j)$ for its match. To do so, we distinguish three cases. For each $j \in C \cap B_t$, let $A(j)$ be the set of jobs that the online algorithm could have chosen from to match with j, i.e. A_j is the set $A \setminus \{j\}$ in Algorithm 1 right after job j becomes critical. If $m(j) \in A(j) \cap B_t$, then both jobs $j, m(j)$ are included in the offline instance we construct. In the second case, if $m(j) \in A(j) \cap B_{t+1}$ and $A(j) \cap B_t \neq \emptyset$, then we modify $\theta_{m(j)}$ to be equal to the best available matching candidate within batch i.e. we "move" job $m(j)$ to location $\arg\min\{|\theta_j - \theta| : \theta = \theta_k,\ k \in A(j) \cap B_t\}$. In the third case, if $m(j) \in A(j) \cap B_{t+1}$ and $A(j) \cap B_t = \emptyset$, then we don't consider job j or its match (if any) in the offline instance we construct. This can happen at most once per batch, since any other job $k < j$ would have $j \in A(k)$.



(a) Batch B_t of the online instance $\boldsymbol{\theta}$.



(b) Modified offline instance $\hat{\boldsymbol{\theta}}$.

Figure 2 Illustration of the construction of the modified offline instance, with $d = 4$. In Figure 2a, job $5t+1$ chooses job $5t+3$ in the online execution of PB, because only jobs up to $5t+5$ would have arrived when job $5t+1$ becomes critical. However, in the offline execution of PB, job $5t+1$ would choose job $5t+6$, because all jobs are available. Our modified instance in Figure 2b corrects the inconsistent decision.

To formally define the construction, fix an arbitrary batch t. Let $\hat{C} = \{j \in C \cap B_t : A(j) \cap B_t \neq \emptyset\}$ and $\hat{n} = 2|\hat{C}| \leq 2(d+1)$. $\hat{C}$ is the set of critical jobs in cases one or two above, and we include jobs $j \in \hat{C}$ and their matches $m(j)$ in the offline instance. There are $\hat{n}$ jobs in the offline instance and the ordering of indices is consistent with the original instance. Define modified locations in the offline instance as follows: $\hat{\theta}_j = \theta_j$ for $j \in \hat{C}$, $\hat{\theta}_{m(j)} = \theta_{m(j)}$ for $j \in \hat{C}$, if $m(j) \in A(j) \cap B_t$ (case one), and $\hat{\theta}_{m(j)} = \arg\min\{|\theta_j - \theta| : \theta = \theta_k,\ k \in A(j) \cap B_t\}$ for $j \in \hat{C}$, if $m(j) \in A(j) \cap B_{t+1}$ (case two).

Now, we show that the output of (offline) PB on $\hat{\boldsymbol{\theta}}$ provides an upper bound for the expression $\sum_{j \in C \cap B_t} |\theta_j - \theta_{m(j)}|$ in (13). In particular, we show that the output of PB is exactly $(\hat{C}, m(\cdot))$ when executed on the modified offline instance. First, note that the construction potentially introduces ties if there is a job $k \in C \cap B_t$ with $m(k) \in B_{t+1}$. However, since Theorem 1 holds under arbitrary tie-breaking, we assume PB breaks ties on $\hat{\boldsymbol{\theta}}$ to maintain consistent decisions with PB on the original instance $\boldsymbol{\theta}$. Thus, from Remark 2, it suffices to show that for every $j \in \hat{C}$, $m(j)$ minimizes the distance among available jobs.

Indeed, note that every job in the offline instance has a type that is identical to a job in B_t. Then, proceeding in the same order as in the online instance, the job types of the set of available jobs in the offline instance is a subset of the original available jobs. For a match in case one, we know that it minimizes the distance among the original available jobs, and hence it minimizes the distance among the offline available jobs. Moreover, we have $|\hat{\theta}_j - \hat{\theta}_{m(j)}| = |\theta_j - \theta_{m(j)}|$. In the second case, the match minimizes distance among offline available jobs by construction, and moreover $|\hat{\theta}_j - \hat{\theta}_{m(j)}| \geq |\theta_j - \theta_{m(j)}|$. Consequently, we can upper bound $\sum_{j \in C \cap B_t} |\theta_j - \theta_{m(j)}| \leq 1 + \sum_{j \in \hat{C}} |\hat{\theta}_j - \hat{\theta}_{m(j)}|$. Then, recalling (12) in the proof of Theorem 1, we have $\sum_{j \in \hat{C}} |\hat{\theta}_j - \hat{\theta}_{m(j)}| \leq \log(\hat{n}/2 + 1) \leq \log(d+2)$, and

$$\sum_{j \in C \cap B_t} |\theta_j - \theta_{m(j)}| \leq 1 + \log(d+2). \tag{14}$$

Substituting back into (13), and using the fact that $b \leq \frac{n}{d+1} + 1$, we get $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}, d) \leq 1/2 + (\frac{n}{d+1} + 1)(1 + \log(d+2))/2$, completing the proof. $\square$

Lastly, we show that this online analysis of PB is also tight, up to constants.

Proposition 4 **(proof in Appendix A.4)**. *Under reward topology $r(\theta, \theta') = \min\{\theta, \theta'\}$, if $d + 1 = 2^{k+3} - 4$ for some $k \in \{0, 1, \ldots\}$, then for any number of jobs n divisible by $(d+1)$, there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ for which* $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}, d) \geq \frac{n}{3(d+1)}(\log_2(d+5) - 3)/4$.

4. Numerical Experiments

In this section, we show via numerical simulations that under the reward structure from delivery pooling, our proposed potential-based greedy algorithm (PB) outperforms a number of benchmark algorithms, given sufficient density. We first generate synthetic data under a setting more aligned with our theoretical results, and find that PB outperforms all benchmarks starting from very low market densities ($d \geq 5$). We then apply all algorithms and benchmarks on real data from the Meituan platform. Although the real-life setting differs from our theoretical model in a number of ways—that we address in Subsection 4.3—the results remain qualitatively similar: PB performs the best, starting from the modest density level corresponding to "each job is able to wait for one minute before being dispatched". Additional simulation results, including additional performance metrics and more general synthetic environments (e.g. two-dimensional locations, different spatial distributions, and different reward topologies) are provided in Appendix E.

The benchmark algorithms that we compare with can incorporate forecast-based information into the greedy framework or depart from it entirely by solving explicit matching optimizations at each decision epoch. We now describe each of them.

4.1. Benchmark Algorithms

As we have discussed earlier, the potential-based greedy algorithm PB does not rely on any historical data/demand forecast. The same is true for naive greedy GRE, and we refer to heuristics with this property as *forecast-agnostic*

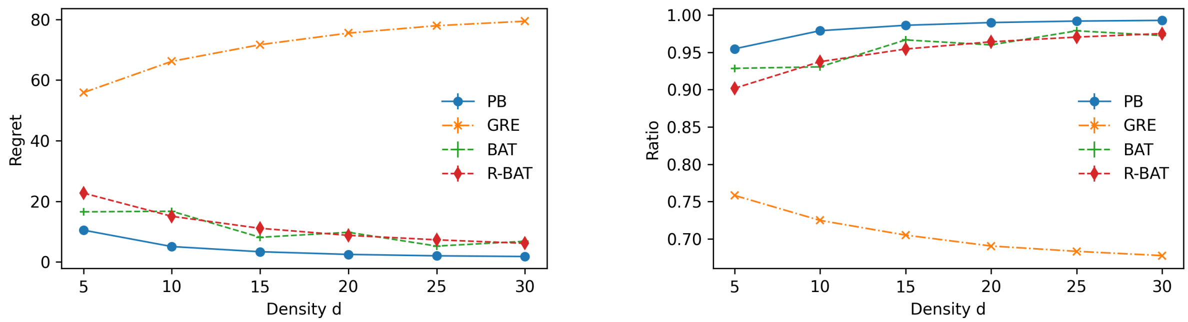
In practice, delivery platforms typically have access to past order data that allows to predict patterns of future arrivals to some extent. In contrast to the aforementioned algorithms, we say that a policy is *forecast-aware* if it uses historical data to make pooling decisions. We consider dual-based benchmark algorithms that use optimal dual solutions (i.e. shadow prices) to approximate the opportunity cost of matching each job. More specifically, given an instance from the historical data (i.e. a sequence of delivery requests with information about origin, destination, and timestamp), we can solve the dual program of a linear relaxation of the IP defined in (5). We provide a detailed description of the primal and dual linear programs in Appendix C.1. The optimal dual variables associated with the capacity constraint of each job will then provide a proxy for the marginal value of the job, which we use to construct two additional benchmark algorithms.

Hindsight Dual. Ideally, if we had access to the dual optimal solution for an instance in advance, we may use them as the opportunity cost of matching each job. In an attempt to measure the value of this hindsight information, we simulate a *hindsight-dual* algorithm (HD) that operates retrospectively on the same instance. Formally, given an instance $\boldsymbol{\theta} \in \Theta^n$ and parameter d, let $\lambda_k = \lambda_k(\boldsymbol{\theta}, d)$ be the optimal dual variable associated with the constraint $\sum_{k:j\neq k} x_{jk} \leq 1$ in the linear relaxation. The HD algorithm operates as an index-based greedy matching algorithm (as defined in Algorithm 1) where the index function is given by $q_{\mathsf{HD}}(\theta_j, \theta_k) = r(\theta_j, \theta_k) - \lambda_k$, for all $j, k \in [n]$, such that $j \neq k, |j-k| \leq d$. Note that this index function is computed based on the entire realized instance $\boldsymbol{\theta}$, thus this algorithm is only for benchmarking purposes and is not practically feasible.

Average Dual. A more realistic approach for estimating the opportunity costs is to average the hindsight duals of the jobs from historical data. However, this estimation is not free of challenges. In particular, computing the dual solutions yields a shadow price for every observed type, which might not cover the entire type space and thus many future arrivals may not have an associated shadow price. To address this issue, we discretize the type space and estimate average duals for each discrete type as the empirical average of the shadow prices for jobs associated to the discrete type. To be precise, let H be the set of all instances in the historical data, and consider a discretization $\{\Theta_1, \Theta_2, \dots, \Theta_p\}$ that partitions the space into p cells (i.e. these cells are mutually exclusive and jointly exhaustive). Each cell $i = 1, 2, \dots, p$ is associated with a shadow price $\bar{\lambda}_i$ that is the average shadow price of historical jobs lying in the same cell, i.e. $\bar{\lambda}_i$ is the average of $\{\lambda(\boldsymbol{\theta}, d) : \boldsymbol{\theta} \in H, \theta_j \in \Theta_i\}$. Then, the *average-dual* algorithm (AD) is an index-based greedy matching algorithm that uses the shadow prices for the cells, i.e. uses index function $q_{\mathsf{AD}}(\theta, \theta') = r(\theta, \theta') - \sum_{i=1}^{p} \bar{\lambda}_i \mathbb{1}\{\theta' \in \Theta_i\}$.

We also consider non-greedy forecast-agnostic dispatch rules, which are highly relevant both in theory and in practice.

Full Batching. Batching heuristics periodically compute optimal matching solutions given the available jobs. Since it is without loss in our model to wait to match, we consider batching algorithms that re-compute optimal matchings whenever any available job becomes critical. The *full batching* algorithm (BAT) dispatches all available jobs according to this optimal solution. In particular, under the theoretical model introduced in



(a) Average regret. (b) Average ratio.

Figure 3 Average regret and reward ratio for forecast-agnostic heuristics in random 1D instances.

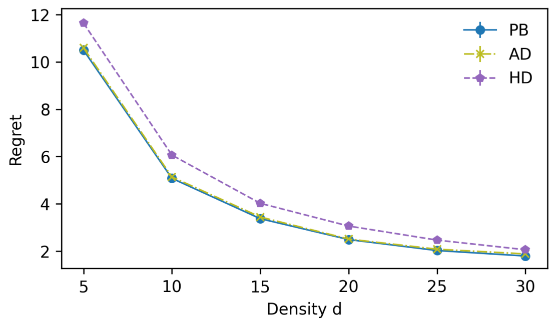
Section 2, BAT clears the market every $d+1$ arrivals, which is equivalent to the one analyzed in Ashlagi et al. (2019). However, an obvious suboptimality of this algorithm is that there are pairs of jobs that are matched and dispatched before any of them becomes critical.

Rolling Batching. To address the previously mentioned issue of dispatching non-critical pairs of jobs, we also consider a modified batching heuristic, termed *rolling batching* ($\mathsf{R-BAT}$). $\mathsf{R-BAT}$ also computes an optimal matching solution every time any available job becomes critical. Instead of dispatching all jobs, however, $\mathsf{R-BAT}$ only dispatches the critical job and its match (if any) according to the optimal solution.

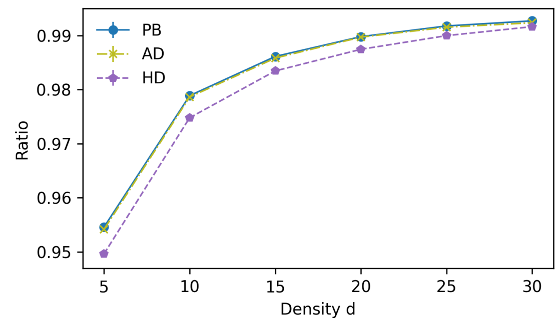
4.2. Synthetic Data

We first consider the setting analyzed in our theoretical results, where job destinations are uniformly distributed on a one-dimensional space. We fix the total number of jobs at $n = 1000$, and compare algorithms at density levels $d \in \{5, 10, 15, 20, 25, 30\}$. For each (n, d) pair, we generate 100 instances $\boldsymbol{\theta} \in [0,1]^n$ uniformly at random, and compute the average regret and reward *ratio* ($\mathsf{ALG}(\boldsymbol{\theta}, d)/\mathsf{OPT}(\boldsymbol{\theta}, d)$) achieved by each algorithm and benchmark. Additional performance metrics, including the fraction of jobs that are pooled and the fraction of the total distance that is reduced by pooling, are presented in Appendix E.1. Similar results for the 1D, non-uniform setting are presented in Appendix E.2.

4.2.1. Comparison with forecast-agnostic heuristics. Figure 3 compares potential-based greedy PB with naive greedy GRE and the two batching algorithms. As expected from our analysis, GRE performs poorly even at high densities. To our surprise, PB outperforms both batching algorithms for all considered



(a) Average regret.



(b) Average ratio.

Figure 4 Average regret and reward ratio for forecast-aware heuristics in random 1D instances.

density values. To assess the performance of the algorithms relative to the hindsight optimal, we also compute the *ratio* for each instance, and plot its empirical average in Figure 3b. We can see that even at relatively low density levels, PB achieves over 95% of the maximum possible travel distance saved from pooling. GRE, on the other hand, performs worse relative to hindsight optimal as density increases—despite the fact that the travel distance saved by GRE increases with density (see Figure 14b in Appendix E.1.1).

4.2.2. Comparison with forecast-aware heuristics. We now compare PB with the two forecast-aware benchmarks, HD and AD. Under HD, for each of the 100 instances generated for each market condition (n, d), we compute a shadow price for each job via the dual LP. As discussed in Subsection 4.1, this algorithm is not practically feasible, and we present its performance here mainly to illustrate the value of this hindsight information used in this particular way. For AD, for each (n, d), we first generate 400 *historical instances* in the same way that the original 100 instances were drawn. We then compute optimal dual variables for each historical instance, and then estimate average shadow prices $\bar{\lambda}$ for a uniform discretization of the type space $[0, 1]$ into 100 intervals.

Figure 4 compares the average regret and reward ratio on the original 100 instances. We can see that PB performs on par with AD across all density parameters, without relying on any forecast/historical data. Notably, even with access to hindsight information, HD is consistently dominated by both AD and PB, with the performance gap narrowing as d increases.

4.3. Order-Level Data from Meituan Platform

In this subsection, we test the algorithms and benchmarks using data from the Meituan platform, made public for the 2024 INFORMS TSL Data-Driven Research Challenge (Zhao et al. 2024). See Appendix D for more details on the dataset and high-level illustrations of market dynamics over both space and time.

The dataset provides detailed information on various aspects of the platform's delivery operations, for one city and a total of 8 days in October 2022. For each of the 569 million delivery orders, the dataset provides the pick-up and drop-off locations (the latitudes and longitudes were shifted for privacy considerations), and the timestamps at which orders are created, pushed into the dispatch system, etc. This allows us to (i) construct instances of job arrivals and (ii) calculate travel distances and pooling opportunities.

Since the pick-up and drop-off locations now reside in a two-dimensional (2D) space, we first extend our definition of pooling reward to the 2D space, derive the potential of different job types, and provide the notion of a pooling window (i.e. the sojourn time) that determines which jobs can be pooled together.

Pooling reward. Each job j has type $\theta_j = (O_j, D_j)$, with $O_j,\ D_j \in \mathbb{R}^2$ corresponding to the coordinates of the job origin and destination. The travel distance saved by pooling two orders together is the difference between (i) the travel distance required to fulfill the jobs in two separate trips, and (ii) the minimum travel distance for fulfilling the two requests in a single, pooled trip. For the pooled trip, the travel distance always includes the distance between the two origins and that between the two destinations (both orders must be picked up before either of them is dropped off; otherwise the orders are effectively not pooled). The remaining distance depends on the sequence in which the jobs are picked up and dropped off, resulting in four possible routes. The minimum travel distance is then determined by evaluating the four routes and selecting the smallest travel distance. Formally, let $\|\cdot\|$ denote the Euclidean norm, the reward function is

$$\begin{aligned} r(\theta,\theta') = &\|D-O\| + \|D'-O'\| \\ &- (\|O-O'\| + \rho(\theta,\theta') + \|D-D'\|)\,, \end{aligned} \tag{15}$$

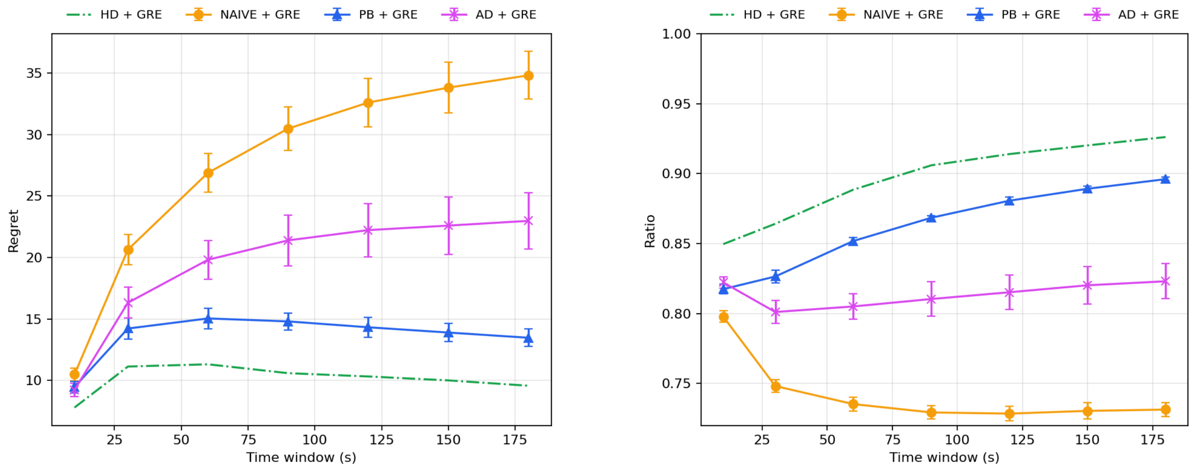
where $\rho(\theta,\theta') = \min\{\|D-O\|\,, \|D'-O'\|\,, \|D-O'\|\,, \|D'-O'\|\}$. Note that unlike the 1D common origin setting, the pooling reward in 2D space with heterogeneous origins is not always positive. As a result, we assume that all index-based greedy algorithms only pool jobs together when the reward is non-negative.

Potential. For a given job type $\theta = (O, D)$, the maximum reward a platform can achieve by pooling it with another job is the distance of the job, $\|D - O\|$. To see this, first observe that when a job $\theta = (O, D)$ is pooled with another job of identical type, the travel distance saved is precisely $\|D - O\|$. On the other hand, let m be the minimum in (15). Then, by triangle inequality $\|O' - D'\| \leq \|O - O'\| + \|D - O\| + \|D - D'\|$, and thus $r(\theta, \theta') \leq 2\|D - O\| - m \leq \|D - O\|$, since $m \leq \|D - O\|$. To achieve an even higher pooling reward, it must be the case that the second part of the pooling reward (15) is strictly smaller than $\|D' - O'\|$. This implies that the total distance traveled to visit all four locations is smaller than the distance between two of them, which is impossible. As a consequence, the potential of a job θ is $p(\theta) = \|D - O\| / 2$. This is aligned with our 1D theoretical model, in which longer deliveries have higher potential reward from being pooled with other jobs in the future.

Pooling window. Instead of assuming that each job is available to be pooled for a fixed number of future arrivals, we take the actual timestamps of the jobs, and assume that the jobs remain available for pooling for a fixed time window, at the end of which the order becomes critical. Intuitively, the length of the pooling window corresponds to the "patience level" of the jobs. This deviates from our assumptions for the theoretical model, but it is straightforward to see how our algorithm as well as the benchmarks we compare with can be generalized and applied.

4.3.1. Shadow Price Comparison. We focus our analysis on a three-hour lunch period (10:30am to 1:30pm) for the 8 days of data provided by the challenge. During this lunch period, there are approximately 130 orders per minute on average, thus $n \approx 24,000$ orders in total (see Figure 10 in Appendix D for an illustration of order volume over time). If the platform allows a time window of one minute for each order before it has to be dispatched, the average density would be around 130 jobs.

To simulate the *hindsight-dual* algorithm HD for each instance associated to each day, we compute the shadow prices via the dual LP and then compute the "online" pooling outcomes using pooling reward minus these shadow prices as the index function. To estimate historical average shadow prices for the AD algorithm, for each of the 8 days, we (i) use the remaining 7 days as the historical data, and (ii) discretize the 2D space into discrete types based on job origin and destination using H3, the hexagonal spatial indexing

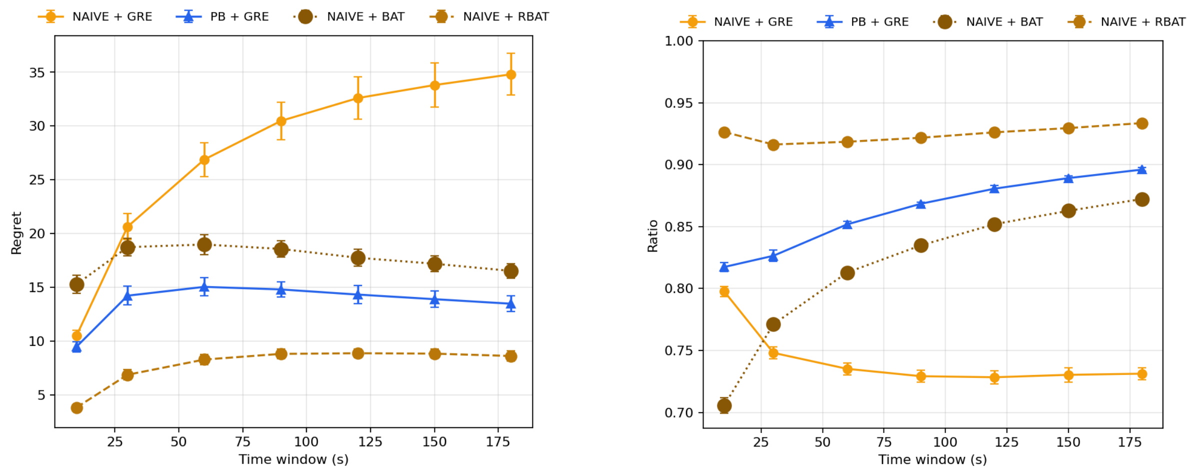


(a) Average regret. (b) Average ratio.

Figure 5 Average regret and reward ratio for greedy heuristics in Meituan data.

system developed by Uber (Uber 2018). The system supports sixteen resolution levels, each tessellating the earth using hexagons of a particular size. For a given resolution, the set of all origin-destination (OD) hexagon pairs represents the partition of the type space Θ, as described in Subsection 4.1. To compute the average shadow price for a particular OD hexagon pair, we compute the average of optimal dual variables associated with all historical jobs that share the same hexagon pair. If there are no jobs for a given pair, we consider the hexagon pairs at coarser resolutions (moving one or more levels up in the H3 hierarchy, with each level increasing the size of the hexagons by a factor of 7) until there exist associated historical jobs. There is a tradeoff between granularity and sparsity when we choose the baseline resolution level, as finer resolutions only average over trips with close-by origins and destinations, but may lack sufficient data for accurate estimation (see Figure 13 for the distribution of order volume by OD hexagon pairs). After testing all available resolution levels, we found that levels 8 or 9 perform the best, depending on the time window. The results presented in this section report the result corresponding to the best resolution for each time window.

Figure 5 compares the index-based greedy heuristics with different shadow prices, as the pooling window increases from 30 seconds to 5 minutes. We can see that a single minute is sufficient for PB to outperform every other realistic algorithm, achieving around 80% of the hindsight optimal reward. Moreover, the performance of PB improves steadily as the allowed matching window increases. This range is aligned with



(a) Average regret.

(b) Average ratio.

Figure 6 Average regret and reward ratio for batching-based heuristics in Meituan data.

the timeframe for Meituan's dispatch system, where batches of orders are evaluated at time intervals of less than a minute, but assignments can be strategically delayed (Zhao et al. 2024). In fact, Meituan data shows that orders are typically assigned to couriers around 5 minutes after creation (see Figure 11).

In contrast to the synthetic setting from Subsection 4.2, the unrealistic HD benchmark now incurs the smallest regret at all density levels. This performance difference appears to be driven by the highly non-uniform spatial distribution of jobs .

The worse performance of AD in comparison to PB also highlights the challenges of estimating the opportunity costs of pooling different jobs from historical data in real-world settings. Intuitively, taking historical data into consideration should be valuable in scenarios with highly non-uniform spatial distributions. In this case, however, the sparsity of the data for many if not most origin-destination pairs makes it difficult to properly estimate the opportunity costs, or requires a very coarse granularity for location. PB bypasses this sparsity-granularity tradeoff by only considering the reward topology, i.e. the space of all possible types.

4.3.2. Comparison with batching-based heuristics. We now compare our algorithm with the batching-based heuristics on data from the same three-hour lunch period, 10:30am to 1:30pm.

The results in Figure 7 show that BAT performs noticeably worse than PB. This is primarily because BAT dispatches many non-critical jobs too early, forfeiting the possibility of more efficient future pairings. In contrast, R – BAT achieves high efficiency, as the optimal matching over available jobs implicitly captures some near-future value. However, this comes at a steep computational cost: $\mathsf{R-BAT}$ is approximately 100× slower than the greedy algorithms (see Figure 7a), which would make it impractical in large-scale real-time environments such as Meituan's, where millions of orders arrive every hour.

Despite this simplicity–complexity gap, the potential-based greedy rule PB closes most of the performance gap between naive greedy and rolling batching.

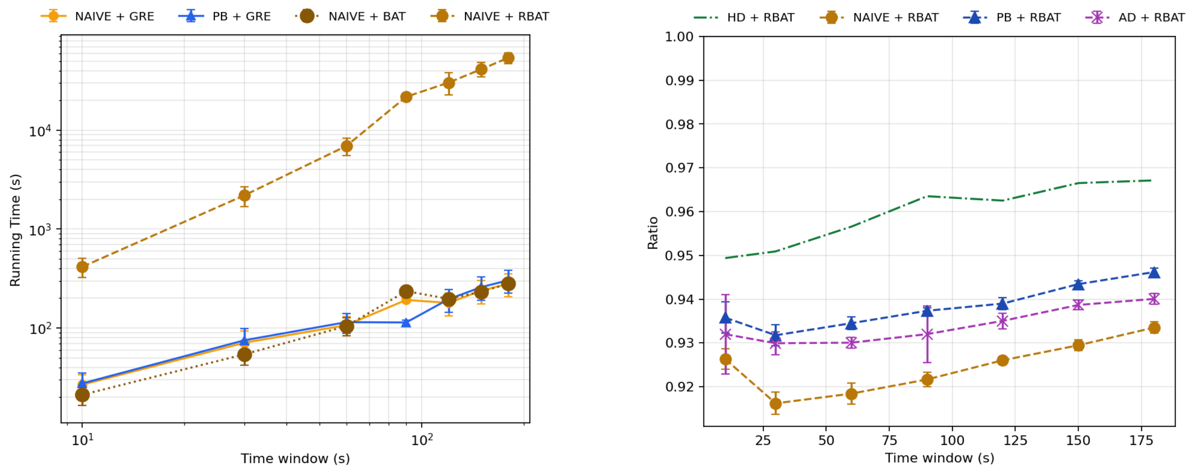
Across all waiting windows tested, PB attains nearly the same efficiency as $\mathsf{R-BAT}$ while being two orders of magnitude faster to compute.

To assess whether explicitly incorporating opportunity costs can further improve the rolling-batching framework, we modify $\mathsf{R-BAT}$ by subtracting a shadow price from the reward of each non-critical job, scaled by a parameter $\gamma \in (0,1)$. Because rolling batching already accounts for part of the jobs' future value, γ is restricted below one. We tune γ on a subset of the data for each policy and waiting-window configuration, and then evaluate performance on the full dataset.

Figure 7b summarizes the results. Introducing these shadow-price adjustments yields modest but consistent improvements over the naive $\mathsf{R-BAT}$ baseline. In particular, using our simple forecast-free *potential* as the shadow price achieves an additional 1–2% increase in total distance saved, at virtually no additional computational or estimation cost. These findings suggest that the potential provides a lightweight yet effective proxy for future value even within more sophisticated optimization-based dispatch frameworks.

5. Concluding Remarks and Future Work

In this work, we study dynamic non-bipartite matching in the context of pooling delivery orders. We exploit the reward structure for this particular problem to design a simple yet effective potential-based greedy algorithm (PB). This PB algorithm uses the length of a job as its opportunity cost, leading to longer-distance jobs being strategically held in the system for better future pooling opportunities. We demonstrate the effectiveness of this insight both theoretically, via worst-case regret bounds, and empirically, where it outperforms several commonly used benchmark algorithms on both synthetic data and real data from the Meituan platform (made public by the INFORMS TSL data-driven research challenge). Notably, our algorithm achieves this without relying on any forecast or complex computation, making it efficient, robust, and easy to implement in practice.



(a) Running time.
(b) Average ratio.

Figure 7 Average running time and reward ratio (with shadow price) for batching-based heuristics in Meituan data.

We provide intuition behind the success of this approach by showing through theory and experiments that various notions of opportunity cost are intimately related to our definition of potential, especially for markets with sufficient density. We believe that the main insight of longer-distance jobs being more valuable to hold should apply even when pooling more than two orders is possible (see Wei et al. (2023)), or when there is no hard deadline to dispatching any order but keeping the customers waiting is costly to the platform. Moreover, our notion of potential (i.e., the highest possible matching reward) could be relevant on non-metric reward structures, where it is a measure of e.g. the popularity of a volunteer position (see Manshadi et al. (2022)). Formalizing and generalizing these results is an interesting avenue for future research.

References

Aouad A, Saritaç Ö (2020) Dynamic stochastic matching under limited time. *Proceedings of the 21st ACM Conference on Economics and Computation*, 789–790.

Ashlagi I, Burq M, Dutta C, Jaillet P, Saberi A, Sholley C (2019) Edge weighted online windowed matching. *Proceedings of the 2019 ACM Conference on Economics and Computation*, 729–742.

Ashlagi I, Roth AE (2021) Kidney exchange: An operations perspective. *Management Science* 67(9):5455–5478.

Balkanski E, Faenza Y, Périvier N (2023) The power of greedy for online minimum cost matching on the line. *Proceedings of the 24th ACM Conference on Economics and Computation*, 185–205.

Cachon GP, Besbes O (2023) The fast and affordable delivery problem. *Available at SSRN 4825608* .

Chen M, Hu M (2024) Courier dispatch in on-demand delivery. *Management Science* 70(6):3789–3807.

Chen Y, Kanoria Y, Kumar A, Zhang W (2023) Feature based dynamic matching. *Available at SSRN 4451799* .

Cohen-Addad V, Eden A, Feldman M, Fiat A (2016) The invisible hand of dynamic market pricing. *Proceedings of the 2016 ACM Conference on Economics and Computation*, 383–400.

Curry D (2025) Doordash revenue and usage statistics. URL `https://www.businessofapps.com/data/doordash-statistics/`, accessed January 26, 2025.

DoorDash (2024) Batched offers, explained. URL `https://dasher.doordash.com/en-us/blog/batched-offers-explained`, accessed on January 30, 2025.

Eom M, Toriello A (2023) Batching and greedy policies: How good are they in dynamic matching? *SSRN 4357375* .

Feng Y, Niazadeh R (2024) Batching and optimal multistage bipartite allocations. *Management Science* .

Huang Z, Kang N, Tang ZG, Wu X, Zhang Y, Zhu X (2018) How to match when all vertices arrive online. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of computing*, 17–29.

Jiang B (2024) Meituan's fourth-quarter revenue jumps on strong orders, while losses from new initiatives narrow further. URL `https://www.scmp.com/tech/big-tech/article/3256388/meituans-fourth-quarter-revenue-jumps-strong-orders-while-losses-new-initiatives-narrow-further?module=perpetual_scroll_0&pgtype=article`, accessed on January 26, 2025.

Kanoria Y (2021) Dynamic spatial matching. *arXiv preprint arXiv:2105.07329* .

Kerimov S, Ashlagi I, Gurvich I (2024) Dynamic matching: Characterizing and achieving constant regret. *Management Science* 70(5):2799–2822.

Kerimov S, Ashlagi I, Gurvich I (2025) On the optimality of greedy policies in dynamic matching. *Operations Research* 73(1):560–582.

Manshadi V, Rodilitz S, Saban D, Suresh A (2022) Online algorithms for matching platforms with multi-channel traffic. *Proceedings of the 23rd ACM Conference on Economics and Computation*, 986–987.

Reddit (2022) How do you multi-app ? URL `https://www.reddit.com/r/grubhubdrivers/comments/14gwyrz/how_do_you_multiapp/`, accessed on January 17, 2025.

Reddit (2023) Tips on multi apping effectively and safely. URL https://www.reddit.com/r/doordash/comments/y3j1jw/tips_on_multi_apping_effectively_and_safely/, accessed on January 17, 2025.

Reingold EM, Tarjan RE (1981) On a greedy heuristic for complete matching. *SIAM Journal on Computing* 10(4):676–681.

Reyes D, Erera A, Savelsbergh M, Sahasrabudhe S, O'Neil R (2018) The meal delivery routing problem. *Optimization Online* 6571:2018.

Statista (2024) Platform delivery - worldwide. URL https://www.statista.com/outlook/emo/online-food-delivery/meal-delivery/platform-delivery/worldwide, accessed January 26, 2025.

Uber (2018) H3: Uber's system for efficient spatial indexing. URL https://www.uber.com/blog/h3/.

UberEats (2025) Delivering multiple orders. URL https://www.uber.com/us/en/deliver/basics/making-deliveries/delivering-multiple-orders/, accessed on January 30, 2025.

Ulmer MW, Thomas BW, Campbell AM, Woyak N (2021) The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science* 55(1):75–100.

Wei Y, Xu J, Yu SH (2023) Constant regret primal-dual policy for multi-way dynamic matching. *Proceedings of the 2023 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 79–80.

Zhao L, Wang H, Liang Y, Li D, Zhao J, Ding X, Hao J, He R (2024) The first informs tsl data-driven research challenge (tsl-meituan 2024): Background and data description. URL https://github.com/meituan/meituan_informs_data, accessed: February 28, 2025.

Appendix

We provide in Appendix A proofs that are omitted from Section 3 of the paper. Theoretical guarantees under alternative reward topologies are stated and proved in Appendix B. Appendix C shows an interpretation of potential in terms of the marginal value of jobs. Finally, we include in Appendix D high-level descriptions of market-dynamics observed in Meituan data, and in Appendix E additional simulation results for settings studied in the body of the paper as well as more general settings with non-uniform spatial distributions and different reward topologies.

A. Deferred Proofs

A.1. Proof of Proposition 1

Let $0 < \varepsilon < 1/3$. Consider an instance $\boldsymbol{\theta} \in [0,1]^n$ such that $\theta_{n/2} < \theta_{n/2-1} < \ldots < \theta_1 < \varepsilon$ (low-type jobs) and $\theta_{n/2+1} = \theta_{n/2+2} = \ldots = \theta_n = 1$ (high-type jobs). Note that because of Remark 1, when the first job becomes critical, GRE chooses to match with a high-type job, i.e. $m(1) \in \{n/2+1, \ldots, n\}$, collecting $r(\theta_1, \theta_{m(1)}) \leq \varepsilon$ independent of the actual choice of $m(1)$. In particular, the second job is not matched and is then the next to become critical. Since there are still high-type jobs to be matched, then by the same argument $m(2) \in \{n/2+1, \ldots, n\}$ and $r(\theta_2, \theta_{m(2)}) \leq \varepsilon$. By induction, since there are as many high-type jobs as low-type jobs, every job $j \in \{1, \ldots, n/2\}$ is matched with $m(j) \in \{n/2+1, \ldots, n\}$ and $r(\theta_j, \theta_{m(j)}) \leq \varepsilon$. Therefore, $\mathsf{GRE}(\boldsymbol{\theta}, \infty) \leq \varepsilon n/2$.

On the other hand, $r(\theta_j, \theta_{j+1}) = 1$ for all $j \in \{n/2+1, \ldots, n-1\}$. Thus, an optimal matching solution would always match all $n/2$ high-type jobs together, collecting $\mathsf{OPT}(\boldsymbol{\theta}, \infty) \geq n/4$. Hence, $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta}, \infty) = \mathsf{OPT}(\boldsymbol{\theta}, \infty) - \mathsf{GRE}(\boldsymbol{\theta}, \infty) \geq \frac{(1-2\varepsilon)}{4} n$, independent of the tie-breaking rule. Since this inequality holds for all $0 < \varepsilon < 1/3$, the proof is complete. $\square$

A.2. Proof of Proposition 2

We construct a sequence of instances $\boldsymbol{\theta}(k) \in [0,1]^{n(k)}$ with $n(k) = 2^{k+3} - 4$ for every integer $k \geq 0$, and show that the regret gain at each iteration, $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(k+1), \infty) - \mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(k), \infty)$ is at least $1/4$. For $k = 0$, consider $\boldsymbol{\theta}(0) = (1/2, 0, 1, 0)$. We have that $\mathsf{OPT}(\boldsymbol{\theta}(0), \infty) = 1/2$. In turn, under PB, when the first job becomes critical, all available jobs are equally close, resulting in ties. Assume ties are broken so that the resulting matching is $\{(1,2), (3,4)\}$. Then, $\mathsf{PB}(\boldsymbol{\theta}(0), \infty) = 0$, and thus $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(0), \infty) = 1/2$. For $k \geq 1$, we inductively construct an instance $\boldsymbol{\theta}(k)$ by carefully adding 2^{k+2} new jobs to be processed before the instance considered in the previous step. In particular, we add 2^k copies of $\boldsymbol{\theta}(0)$, each scaled by $1/2^{k+1}$ and shifted in space so that PB matches within each copy.

Formally, for $s \in \{0, \dots, 2^k - 1\}$ and $r \in \{1, 2, 3, 4\}$, define

$$\boldsymbol{\theta}(k)_{4s+r} = \frac{\boldsymbol{\theta}(0)_r}{2^{k+1}} + \frac{s}{2^k}.$$

Lastly, define $\boldsymbol{\theta}(k)_j = \boldsymbol{\theta}(k-1)_{j-2^{k+2}}$ for $j \in \{2^{k+2}+1, \dots, n(k)\}$. By construction, for each $s \in \{0, \dots, 2^k - 1\}$, PB matches job $4s+1$ with $4s+2$ since it is the closest in space (see Remark 2), and then matches $4s+3$ with $4s+4$ for the same reason. After all these jobs are matched, the remaining jobs represent $\boldsymbol{\theta}(k-1)$ exactly. On the other hand, OPT is at least as good as processing OPT separately on each of the 2^k copies, and then on $\boldsymbol{\theta}(k-1)$. Hence, the regret gain $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(k), \infty) - \mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(k-1), \infty)$ is at least the sum of the regret of PB on each copy. Moreover, note the reward function $r(\theta, \theta') = \min\{\theta, \theta'\}$ satisfies $r(\alpha + \theta/\beta, \alpha\theta'/\beta) = r(\theta, \theta')/\beta$ for any scalars α, β such that $\alpha + \theta/\beta, \alpha\theta'/\beta \in [0, 1]$. Thus, the regret of PB on each copy is exactly $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(0), \infty)/2^{k+1}$, and then $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(k), \infty) - \mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(k-1), \infty) \geq 2^k \frac{\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}(0), \infty)}{2^{k+1}} \geq \frac{1}{4}$, completing the proof. □

A.3. Proof of Proposition 3

Note that if $(d+1)$ is divisible by 4, then for any number of jobs n divisible by $(d+1)$, we have an exact partition $b = n/(d+1) \in \{1, 2, \dots\}$ batches, each of size $(d+1)$. We construct an instance $\boldsymbol{\theta} \in [0, 1]^n$ that can be analyzed separately on each batch in the following sense. For each $t = 1, \dots, b$, let $\boldsymbol{\theta}^{(t)} = (\theta_j)_{j \in B_t} \in [0, 1]^{(d+1)}$. It is straightforward to check that $\mathsf{OPT}(\boldsymbol{\theta}, d) \geq \sum_{t=1}^b \mathsf{OPT}(\boldsymbol{\theta}^{(t)}, \infty)$, since the matching solution induced by the offline instances on the right-hand side is feasible in the online setting. Thus, if $\boldsymbol{\theta}$ is such that $\mathsf{GRE}(\boldsymbol{\theta}, d) = \sum_{t=1}^b \mathsf{GRE}(\boldsymbol{\theta}^{(t)}, \infty)$, then

$$\mathsf{OPT}(\boldsymbol{\theta}, d) - \mathsf{GRE}(\boldsymbol{\theta}, d) \geq \sum_{t=1}^{b} \mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta}^{(t)}, \infty).$$

Moreover, we construct $\boldsymbol{\theta}$ such that each $\boldsymbol{\theta}^{(t)} \in [0, 1]^{(d+1)}$ resembles the worst-case instance analyzed in Proposition 1. To this end, let $0 < \varepsilon < 1$. Consider $\boldsymbol{\theta} \in [0, 1]^n$ such that for each $t = 0, \dots, b-1$, $\varepsilon/2^{t+1} < \theta_{(d+1)t+(d+1)/2} < \dots < \theta_{(d+1)t+1} < \varepsilon/2^t$ (low-type jobs) and $\theta_{(d+1)t+(d+1)/2+1} = \dots = \theta_{(d+1)(t+1)} = 1$ (high-type jobs). By construction, when job 1 becomes critical, the available jobs $k \in A(1) \subseteq B_1$ have types $\theta_k < \theta_1$ or $\theta_k = 1$. Then, by Remark 1, GRE chooses to match with a high-type job $m(1) \in B_1$. As job 2 becomes critical next, the new set of available jobs is $A(2) = A(1) \cup \{(d+1)+1\} \setminus \{m(1)\}$, with $\theta_{(d+1)+1} < \varepsilon/2 < \theta_2$. Again, GRE chooses to match with a high-type job $m(2) \in B_2$. By repeating this argument inductively, GRE outputs $(C, \{m(j)\}_{j \in C})$ such that $C = \bigcup_{t=1}^{b-1} \{(d+1)t+1, \dots, (d+1)t+d+1)t+(d+1)/2\}$ and $m(j) \in B_t \setminus C$ for all $j \in C \cap B_t$. Thus, $\mathsf{GRE}(\boldsymbol{\theta}, d) = \sum_{t=1}^b \mathsf{GRE}(\boldsymbol{\theta}^{(t)}, \infty)$, and since $\boldsymbol{\theta}^{(t)} \in [0, 1]^{(d+1)}$ is specified as in the proof of Proposition 1, then $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}^{(t)}, \infty) \leq (d+1)(1-2\varepsilon)/4$. Hence, $\mathsf{OPT}(\boldsymbol{\theta}, d) - \mathsf{GRE}(\boldsymbol{\theta}, d) \geq b\frac{(d+1)(1-2\varepsilon)}{4} = \frac{n}{4}(1-2\varepsilon)$, completing the proof. □

A.4. Proof of Proposition 4

Since n is divisible by $(d+1)$, we have an exact partition of the set of jobs into $b = n/(d+1) \in \{1, 2, \ldots\}$ batches. As in Proposition 3, we construct an instance $\boldsymbol{\theta} \in [0,1]^n$ that can be analyzed separately on each batch, where we denote $\boldsymbol{\theta}^{(t)} = (\theta_j)_{j \in B_t}$ for $t = 1, \ldots, b$. By Proposition 2, there exists an instance $\boldsymbol{\theta}^{(0)} \in [0,1]^{(d+1)}$ such that $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}^{(0)}, \infty) \geq (\log(d+5) - 3)/4$. Then, for each $t = 1, \ldots, b$, let $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(0)} \cdot \frac{1}{3}$ if t is odd and $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(0)} \cdot \frac{1}{3} + \frac{2}{3}$ if t is even. Note that since each batch size is $d+1$ divisible by 4, it is possible to match within batch only. Moreover, by construction, every critical job has a matching candidate within batch, which is closer in space than any job on the next batch. Thus, because of Remark 2, PB only matches within batch, and therefore $\mathsf{PB}(\boldsymbol{\theta}, d) = \sum_{t=1}^{b} \mathsf{PB}(\boldsymbol{\theta}^{(t)}, \infty)$. As in Proposition 3, we arrive at $\mathsf{OPT}(\boldsymbol{\theta}, d) - \mathsf{PB}(\boldsymbol{\theta}, d) \geq \sum_{t=1}^{b} \mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}^{(t)}, \infty) = b \frac{\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}^{(0)}, \infty)}{3} \geq \frac{n}{3(d+1)}(\log(d+5) - 3)/4$, concluding the proof. □

B. Algorithmic Performance under Alternative Reward Topologies

In this section, we study the performance of both the naive greedy algorithm GRE and potential-based greedy algorithm PB, under the two alternative reward topologies defined in (3) and (4).

B.1. Reward Function $r(\theta, \theta') = 1 - |\theta - \theta'|$

This reward function represents the goal to minimize the total distance between matched jobs. In this case, for any $\theta \in [0,1]$, the potential of a job of type θ is $p(\theta) = 1/2$ constant across job types. Thus, the description of PB is equivalent to GRE, since their index functions differ only in an additive constant.

Remark 3. Under the 1-dimensional type space $\Theta = [0,1]$ and the reward function $r(\theta, \theta') = 1 - |\theta - \theta'|$, both the naive greedy and potential-based greedy algorithm generate the same output $(C, m(\cdot))$ as described in Algorithm 1. In particular, both algorithms always match each critical job to an available job that is the closest in space.

Moreover, we show that their performance can be reduced to the one of PB under reward function $r(\theta, \theta') = \min\{\theta, \theta'\}$. In particular, we first show that when $d = \infty$, the regret is at least logarithmic in the number of jobs.

Proposition 5. *Under reward function $r(\theta, \theta') = 1 - |\theta - \theta'|$, if $n = 2^{k+3} - 4$ for some integer $k \geq 0$, then there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ for which $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta}, \infty) = \mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta}, \infty) \geq (\log_2(n+4) - 3)/2$.*

Proof. We prove the statement for GRE, since $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},\infty) = \mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},\infty)$ follows from Remark 3. Let $n = 2^{k+3} - 4$, for $k \in \{0, 1, \ldots\}$. From Proposition 2, there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ such that the regret of PB under $r(\theta,\theta') = \min\{\theta,\theta'\}$, which we denote by R from here on, is at least $(\log_2(n+4) - 3)/4$. We show that under $r(\theta,\theta') = 1 - |\theta - \theta'|$, we have $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},\infty) \geq 2R$. First, note that since $|\theta - \theta'| = \theta + \theta' - 2\min\{\theta,\theta'\}$, we have $\mathsf{GRE}(\boldsymbol{\theta},\infty) = \sum_{j\in C} 1 - |\theta_j - \theta_{m(j)}| = |C| - \sum_{j\in C}(\theta_j + \theta_{m(j)}) + 2\sum_{j\in C}\min\{\theta,\theta'\}$. Similarly, recall that $\mathcal{M}_{\mathsf{OPT}}$ is the set of matches in the hindsight optimal solution. Then, $\mathsf{OPT}(\boldsymbol{\theta},\infty) = |\mathcal{M}_{\mathsf{OPT}}| - \sum_{(j,k)\in\mathcal{M}_{\mathsf{OPT}}}(\theta_j + \theta_k) + 2\sum_{(j,k)\in\mathcal{M}_{\mathsf{OPT}}}\min\{\theta_j,\theta_k\}$. Since $r(\theta,\theta') \geq 0$ for any $\theta,\theta' \in \Theta$ and the total number of jobs n is even, both GRE and OPT match every job. Hence, $|C| = |\mathcal{M}_{\mathsf{OPT}}| = n/2$, and moreover $\sum_{(j,k)\in\mathcal{M}_{\mathsf{OPT}}}(\theta_j + \theta_k) = \sum_{j\in C}(\theta_j + \theta_{m(j)})$. Thus, $\mathsf{OPT}(\boldsymbol{\theta},\infty) - \mathsf{GRE}(\boldsymbol{\theta},\infty) \geq 2\left(\sum_{(j,k)\in\mathcal{M}_{\mathsf{OPT}}}\min\{\theta_j,\theta_k\} - \sum_{j\in C}\min\{\theta_j,\theta_{m(j)}\}\right)$. We claim that the term in large parenthesis is exactly the regret of PB under reward topology $r(\theta,\theta') = \min\{\theta,\theta'\}$, which we denote by R. Indeed, from Remark 3, $(C, m(\cdot))$ coincides with the resulting matching of PB under $r(\theta,\theta') = \min\{\theta,\theta'\}$, since it is specified to always match to the closest job (Remark 2). On the other hand, since all jobs are matched, the set of matches $\mathcal{M}_{\mathsf{OPT}}$ must induce disjoint intervals, which coincides with the optimal matching solution under reward function $r(\theta,\theta') = \min\{\theta,\theta'\}$. Thus, by Proposition 2, we get $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},\infty) \geq 2R \geq (\log_2(n+4) - 3)/2$, completing the proof. □

We now extend this result to the online setting ($d < n$).

PROPOSITION 6. *Under reward topology $r(\theta,\theta') = 1 - |\theta - \theta'|$, if $d + 1 = 2^{k+3} - 4$ for some integer $k \geq 0$, then for any number of jobs n divisible by $(d+1)$, there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ for which $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},d) = \mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},d) \geq \frac{n}{3(d+1)}(\log_2(d+5) - 3)/2$.*

Proof. This proof is analogous to the proof of Proposition 4, since (i) there exists an offline instance that achieves the desired regret for $d + 1 = n$ (Proposition 5), and (ii) the algorithm matches a critical job with the closest available job (Remark 3).

□

Finally, we directly show the tight regret upper bound (up to constants) as an immediate consequence of the proof of Theorem 2. Note that if $d + 1 = n$, then this yields the result for offline matching.

THEOREM 3. *Under reward topology $r(\theta,\theta') = 1 - |\theta - \theta'|$, we have $\mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},d) = \mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},d) \leq 1/2 + (\frac{n}{d+1} + 1)(1 + \log(d+2))$ for any n, and any instance $\boldsymbol{\theta} \in [0,1]^n$.*

Proof. Let $d \geq 1$ and $(C, m(\cdot))$ be the output of GRE on $\boldsymbol{\theta}$. Since $p(\theta) = 1/2$, for all θ, we have

$$\begin{aligned}\mathsf{OPT}(\boldsymbol{\theta}, d) - \mathsf{GRE}(\boldsymbol{\theta}, d) &\leq \sum_{j=1}^{n} p(\theta_j) - \mathsf{GRE}(\boldsymbol{\theta}, d) \\ &\leq \frac{1}{2} + \sum_{j \in C} |\theta_j - \theta_{m(j)}|,\end{aligned}$$

The second term is exactly the sum of distances between jobs matched by GRE. Recall that $(C, m(\cdot))$ coincides with the matching output of PB under $r(\theta, \theta') = \min\{\theta, \theta'\}$, both are specified to always match with the closest job (Remark 2, Remark 3). Thus, recalling (14) in the proof of Theorem 2, we have $\sum_{j \in C} |\theta_j - \theta_{m(j)}| = \sum_{t=1}^{b} \sum_{j \in C \cap B_t} |\theta_j - \theta_{m(j)}| \leq \left(\frac{n}{d+1} + 1\right)(1 + \log(d+2))$, and the result follows. □

B.2. Reward Function $r(\theta, \theta') = |\theta - \theta'|$

Under this reward topology, it is worst to match two jobs of the same type, contrasting the two settings studied in Section 3 and Appendix B.1. In this case, the potential of a job type $\theta \in [0,1]$ is $p(\theta) = \max\{\theta, 1-\theta\}/2$. We first show that *any* index-based matching policy must suffer regret constant regret per job in the offline setting.

PROPOSITION 7. *Under reward topology $r(\theta, \theta') = |\theta - \theta'|$, when the number of jobs n is divisible by 4, for any index-based greedy matching algorithm* ALG*, there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ for which $\mathsf{Reg}_{\mathsf{ALG}}(\boldsymbol{\theta}, \infty) \geq c_{\mathsf{ALG}} n$, for some $c_{\mathsf{ALG}} \in (0,1)$. In particular, $c_{\mathsf{GRE}} = c_{\mathsf{PB}} = 1/4$.*

Proof. Let n be divisible by 4. Let ALG be an index-based greedy matching algorithm specified by index function $q : [0,1]^2 \to \mathbb{R}$. Define $\theta^c = \sup\{\theta \in [0,1] : q(\theta, 1) \geq q(\theta, 0)\}$. We first construct an instance in the case $\theta^c > 0$ (e.g. $\theta^c = 1/2$ for GRE and PB), and analyze the alternative case separately in the end. Consider an instance $\boldsymbol{\theta} \in \Theta^n$ such that $\theta_1, \ldots, \theta_{n/4} = \theta^c$, $\theta_{n/4+1}, \ldots, \theta_{3n/4} = 0$, and $\theta_{3n/4+1}, \ldots, \theta_n = 1$. It is straightforward to check that $\mathsf{OPT}(\boldsymbol{\theta}, \infty) \geq (1 + \theta^c)n/4$. On the other hand, ALG chooses to match every $j = 1, \ldots, n/4$ to $m(j) \in \{3n/4 + 1, \ldots, n\}$. In the presence of ties, such an output is achieved by some tie-breaking rule. Then,

$$\mathsf{ALG}(\boldsymbol{\theta}, \infty) = 0 + \sum_{j=1}^{n/4} r(\theta_j, \theta_{3n/4+j}) = \frac{n(1-\theta^c)}{4}.$$

and thus $\mathsf{Reg}_{\mathsf{ALG}}(\boldsymbol{\theta}, \infty) = \mathsf{OPT}(\boldsymbol{\theta}, \infty) - \mathsf{ALG}(\boldsymbol{\theta}, \infty) \geq n\theta^c/2$, proving the statement with $c_{\mathsf{ALG}} = \theta^c/2$.

Now, if $\theta^c \le 0$, then $q(\theta,0) < q(\theta,1)$ for all $\theta \in (0,1]$. In particular, for any $\varepsilon > 0$, consider the instance $\theta_1,\dots,\theta_{n/4} = \varepsilon$, $\theta_{n/4+1},\dots,\theta_{3n/4} = 1$, and $\theta_{3n/4+1},\dots,\theta_n = 0$. Then, $\mathsf{OPT}(\boldsymbol{\theta},\infty) \ge (1+(1-\varepsilon))n/4$, and $\mathsf{ALG}(\boldsymbol{\theta},\infty) = \varepsilon n/4$. Hence, $\mathsf{Reg}_{\mathsf{ALG}}(\boldsymbol{\theta},\infty) \ge (1-\varepsilon)/2$, and the statement is true with $c_{\mathsf{ALG}} = (1-\varepsilon)/2$ for every $\varepsilon > 0$. $\square$

We now show that under PB and GRE, this lower bound extends to the online setting, losing a factor of 2.

PROPOSITION 8. *Under reward topology $r(\theta,\theta') = |\theta-\theta'|$, if $(d+1)$ is divisible by 4, then for any number of jobs n divisible by $2(d+1)$, there exists an instance $\boldsymbol{\theta} \in [0,1]^n$ for which* $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},d) = \mathsf{Reg}_{\mathsf{PB}}(\boldsymbol{\theta},d) \ge n/8$.

Proof. If n is divisible by $2(d+1)$, we have an exact partition of the set of jobs into $b = n/(d+1) \in \{2,4,\dots\}$ batches, defined as $B_{t+1} = \{t(d+1)+1,\dots,(t+1)(d+1)\}$ for $t = 0,\dots,b-1$. We construct an instance $\boldsymbol{\theta} \in [0,1]^n$ for which GRE (and PB) "resets" every two batches, in the sense that all jobs in these two batches are matched among them. To formally specify the construction, first fix $\varepsilon \in (0,1/2)$. Define, for $t = 0,\dots,b/2-1$,

$$\theta_{2t(d+1)+j} = \begin{cases} \frac{1}{2}, & j \le \frac{d+1}{4}, \\ \varepsilon, & \frac{d+1}{4} < j \le \frac{3(d+1)}{4}, \\ 0, & d+1 < j \le \frac{3(d+1)}{2}, \\ 1, & \text{otherwise}. \end{cases}$$

Consider the notation $\boldsymbol{\theta}^{(0)} = (\theta_j)_{j \le 2(d+1)}$. It is easy to check that $\mathsf{OPT}(\boldsymbol{\theta}^{(0)},\infty) = \left(\frac{1}{2}-\varepsilon\right)\frac{d+1}{4} + (1-\varepsilon)\frac{d+1}{4} + \frac{d+1}{2} = \left(\frac{7-4\varepsilon}{8}\right)(d+1)$, and then $\mathsf{OPT}(\boldsymbol{\theta},d) \ge \frac{b}{2}\mathsf{OPT}(\boldsymbol{\theta}^{(0)},\infty) = \frac{7-4\varepsilon}{16}n$. On the other hand, $\mathsf{GRE}(\boldsymbol{\theta}^{(0)},\infty) = \frac{1}{2}\frac{d+1}{4} + \varepsilon\frac{d+1}{4} + (1-\varepsilon)\frac{d+1}{4} + \frac{d+1}{4} = \frac{5}{8}(d+1)$. We claim that $\mathsf{GRE}(\boldsymbol{\theta},d) = \frac{b}{2}\mathsf{GRE}(\boldsymbol{\theta}^{(0)},\infty) = \frac{5}{16}n$, and thus $\mathsf{Reg}_{\mathsf{GRE}}(\boldsymbol{\theta},d) \ge \frac{1-2\varepsilon}{8}n$. Indeed, let $(C,m(\cdot))$ be the output of GRE. Running GRE on $\boldsymbol{\theta}$ is as follows:

1. For every job $j \le (d+1)/4$ of type $\theta_j = 1/2$, and then $3(d+1)/4 < m(j) \le (d+1)$ with $r(\theta_j,\theta_{m(j)}) = 1/2$.
2. Then, for every job $\frac{d+1}{4} < j \le \frac{(d+1)}{2}$, the only available jobs are of type $\theta \in \{0,\varepsilon\}$, and then $d+1 < m(j) \le \frac{3(d+1)}{2}$ with $r(\theta_j,\theta_{m(j)}) = \varepsilon$.
3. In turn, jobs $\frac{(d+1)}{2} < j \le \frac{3(d+1)}{4}$ get to observe jobs of type $\theta = 1$, and then $\frac{3(d+1)}{2} < m(j) \le 2(d+1)$ with $r(\theta_j,\theta_{m(j)}) = 1-\varepsilon$.

4. Finally, the key observation is that the all remaining available jobs $j \le \frac{3(d+1)}{2}$ of type $\theta_j = 0$ observe jobs of type 1, as well as jobs on a "third" batch. However, by construction every job $2(d+1) < k \le 2(d+1) + \frac{3(d+1)}{2}$ are of type $\theta_k \in \{1/2, \varepsilon\}$, and then each job $j \le \frac{3(d+1)}{2}$ is matched (within batch) to $\frac{3(d+1)}{2} < m(j) \le 2(d+1)$ with $r(\theta_j, \theta_{m(j)}) = 1$.

The proof for PB is analogous, since the matching decisions coincide with GRE on this instance.

□

C. Interpretation of Potential

When a job j becomes critical, our potential-based greedy algorithm matches it to an available job k maximizing $r(\theta_j, \theta_k) - p(\theta_k)$, where $p(\theta_k) = \frac{1}{2}\sup_{\theta\in\Theta} r(\theta_k, \theta)$ can be interpreted as the opportunity cost of matching job k. This is an optimistic measure of opportunity cost because it assumes that job k would otherwise be matched to an "ideal" type $\theta \in \Theta$ maximizing $r(\theta_k, \theta)$ (with half of this ideal reward $\sup_{\theta\in\Theta} r(\theta_k, \theta)$ attributed to job k). We now prove that this ideal reward can indeed be achieved under asymptotically-large market thickness, for a stochastic model under our topology of interest.

Definition 1. Let $\boldsymbol{\theta} \in \Theta^n$. For any job $j \in [n]$, let $\boldsymbol{\theta}^{-j} \in \Theta^{n-1}$ be the same instance with the exception that job j is not present. Meanwhile, let $\boldsymbol{\theta}^{+j} \in \Theta^{n+1}$ be the same instance with the exception that an additional copy of job j is present. Consider the following definitions.

1. Marginal Loss: $\mathsf{ML}_j(\boldsymbol{\theta}) = \mathsf{OPT}(\boldsymbol{\theta}) - \mathsf{OPT}(\boldsymbol{\theta}^{-j})$
2. Marginal Gain: $\mathsf{MG}_j(\boldsymbol{\theta}) = \mathsf{OPT}(\boldsymbol{\theta}^{+j}) - \mathsf{OPT}(\boldsymbol{\theta})$

Note that definitions $\mathsf{ML}_j(\boldsymbol{\theta}), \mathsf{MG}_j(\boldsymbol{\theta})$ are based solely on offline matching, and we will use them as our definitions of opportunity cost if the future was known. One could alternatively use shadow prices from the LP relaxation of the offline matching problem, but we note that the LP is not integral. In either case, there is no ideal definition of opportunity cost that is guaranteed to lead to the optimal offline solution in matching problems (see Cohen-Addad et al. 2016).

We now establish the following lemma to help analyze the opportunity costs $\mathsf{ML}_j(\boldsymbol{\theta}), \mathsf{MG}_j(\boldsymbol{\theta})$.

Lemma 1. *Let $\Theta = [0,1]$ and $r(\theta, \theta') = \min\{\theta, \theta'\}$. Consider an instance $\boldsymbol{\theta} \in \Theta^n$ and re-label the indices to satisfy $\theta_1 \ge \theta_2 \ge \ldots \ge \theta_n$. Then, $\mathsf{ML}_j(\boldsymbol{\theta}) = \sum_{k\ge j, k \text{ even}} (\theta_k - \theta_{k+1})$, and $\mathsf{MG}_j(\boldsymbol{\theta}) = \sum_{k\ge j, k \text{ odd}} (\theta_k - \theta_{k+1})$, where we consider $\theta_{n+1} = 0$.*

Proof. Let $\boldsymbol{\theta} \in \Theta^n$ such that $\theta_1 \ge \theta_2 \ge \ldots \ge \theta_n$. Then, $\mathsf{OPT}(\boldsymbol{\theta}) = \sum_{k \text{ even}} \theta_k$, and moreover $\mathsf{OPT}(\boldsymbol{\theta}^{-j}) = \sum_{k<j, k \text{ even}} \theta_k + \sum_{k>j, k \text{ odd}} \theta_k$, and $\mathsf{OPT}(\boldsymbol{\theta}^{+j}) = \theta_j + \mathsf{OPT}(\boldsymbol{\theta}^{-j})$. Therefore, $\mathsf{ML}_j(\boldsymbol{\theta}) =$

$\sum_{k\geq j,k \text{ even}} \theta_k - \sum_{k>j,k \text{ odd}} \theta_k = \sum_{k\geq j,k \text{ even}}(\theta_k - \theta_{k+1})$, and $\mathsf{MG}_j(\boldsymbol{\theta}) = \theta_j - \sum_{k\geq j,k \text{ even}}(\theta_k - \theta_{k+1}) = \sum_{k\geq j,k \text{ odd}}(\theta_k - \theta_{k+1})$, completing the proof. □

Note that because $\mathsf{OPT}(\boldsymbol{\theta}^{+j}) = \mathsf{OPT}(\boldsymbol{\theta}^{-j}) + \theta_j$ and $p(\theta_j) = r(\theta_j, \theta_j)/2$ for this reward function, we immediately get the following corollary.

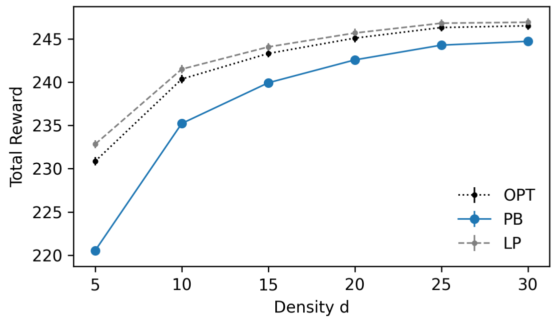
Corollary 1. *If* $\Theta = [0,1]$ *and* $r(\theta,\theta') = \min\{\theta,\theta'\}$, *then for all jobs* j,

$$p(\theta_j) = \frac{\mathsf{ML}_j(\boldsymbol{\theta}) + \mathsf{MG}_j(\boldsymbol{\theta})}{2}.$$

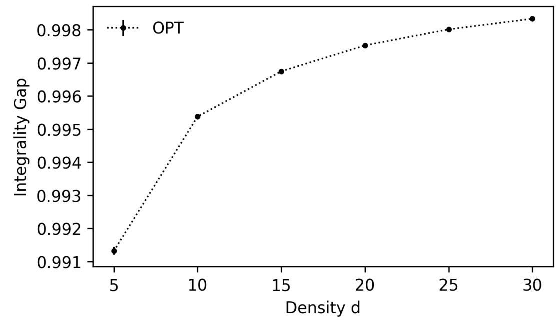
We are now ready to state our main result about the interpretation of potential, that the true opportunity costs $\mathsf{ML}_j(\boldsymbol{\theta}), \mathsf{MG}_j(\boldsymbol{\theta})$ concentrate around $p(\theta_j)$ in a random uniform instance, assuming the market is sufficiently thick. We without loss consider job $j = 1$ and fix its type θ_1.

Proposition 9. *Let* $\Theta = [0,1]$ *and* $r(\theta,\theta') = \min\{\theta,\theta'\}$. *Fix* $\theta_1 \in \Theta$ *and suppose* $\theta_2, \ldots, \theta_n$ *are drawn IID from the uniform distribution over* $[0,1]$, *forming a random instance* $\boldsymbol{\theta} \in \Theta^n$, *for some* $n \geq 2$. *Then, both* $\mathbb{E}[\mathsf{ML}_1(\boldsymbol{\theta})]$ *and* $\mathbb{E}[\mathsf{MG}_1(\boldsymbol{\theta})]$ *are within* $O\left(\frac{1}{n}\right)$ *of* $p(\theta_1)$ *and moreover* $\mathrm{Var}(\mathsf{ML}_1(\boldsymbol{\theta})) = \mathrm{Var}(\mathsf{MG}_1(\boldsymbol{\theta})) = O\left(\frac{1}{n}\right)$.

Proof. We prove the statement only for $\mathsf{ML}_1(\boldsymbol{\theta})$, and the analogous result for $\mathsf{MG}_1(\boldsymbol{\theta})$ follows from Corollary 1. If $\theta_1 = 0$, then $\mathsf{ML}_1(\boldsymbol{\theta}) = 0$ for any $\boldsymbol{\theta}$, coinciding with $p(\theta_1) = 0$. Then, for the remainder of the proof, assume that $\theta_1 > 0$. Consider the random variable $N = |\{j : \theta_j < \theta_1\}|$, which counts the number of points between 0 and θ_1 and has distribution $\mathrm{Binom}(n-1, \theta_1)$. These N points divide the interval $[0, \theta_1]$ into $N+1$ intervals with total length θ_1. From Lemma 1, the marginal loss $\mathsf{ML}_1(\boldsymbol{\theta})$ is determined by computing the total length of a subset of these intervals. It is known that if N random variables are drawn independently from $\mathrm{Unif}[0,1]$, then the joint distribution of the induced interval lengths is $\mathrm{Dirichlet}(1,1,\ldots,1)$ with $N+1$ parameters all equal to 1 (i.e., drawn uniformly from the simplex). In particular, since the Dirichlet distribution is symmetric, the sum of any k of these intervals is equal in distribution to the k-th smallest sample (k-th order statistic) of the N uniform random variables, whose distribution is known to be $\mathrm{Beta}(k, N+1-k)$. Thus, conditional on N, with $N \geq 1$, since the distribution of each of the N jobs to the left of θ_1 is $\mathrm{Unif}[0,\theta_1]$, the distribution of $\mathsf{ML}_1(\boldsymbol{\theta})/\theta_1$ is $\mathrm{Beta}(k_L, N+1-k_L)$, where k_L is either $\lceil \frac{N+1}{2} \rceil$ or $\lfloor \frac{N+1}{2} \rfloor$. To be precise, for $N \geq 1$, $k_L = \lfloor N/2 \rfloor + 1 = \lceil \frac{N+1}{2} \rceil$ if n is even, and $k_L = \lceil N/2 \rceil = \lfloor \frac{N+1}{2} \rfloor$ if n is odd. Moreover, if $N = 0$, then $\mathsf{ML}_1(\boldsymbol{\theta}) = \theta_1$ if n is even, and $\mathsf{MG}_1(\boldsymbol{\theta}) = 0$ if n is odd. Hence, $\mathbb{E}[\mathsf{ML}_1(\theta) \mid N] = \theta_1 \frac{k_L}{N+1}$. for all $N \in \{0, \ldots, n-1\}$. Since $|k_L - (N+1)/2| \leq 1$, then $|\mathbb{E}[\mathsf{ML}_1(\theta) \mid N] - \theta_1/2| \leq \theta_1/(N+1)$,



(a) Average Total Reward.



(b) Fraction of LP value achieved by OPT.

Figure 8 Comparison with LP value.

thus $\left|\mathbb{E}[\mathsf{ML}_1(\theta)] - \frac{\theta_1}{2}\right| \leq \theta_1 \mathbb{E}\left[\frac{1}{N+1}\right] = \theta_1 \sum_{k=0}^{n-1} \frac{1}{k+1}\binom{n-1}{k}\theta_1^k(1-\theta_1)^{n-1-k} = \frac{\theta_1}{n}\sum_{k=0}^{n-1}\binom{n}{k+1}\theta_1^k(1-\theta_1)^{n-1-k} = \frac{1}{n}\sum_{k=1}^{n}\binom{n}{k}\theta_1^k(1-\theta_1)^{n-k} = \frac{1-(1-\theta_1)^n}{n} = O\left(\frac{1}{n}\right)$.

This completes the proof of the statement about $\mathbb{E}[\mathsf{ML}_1(\boldsymbol{\theta})]$.

For the statement about variance, we know $\mathrm{Var}(\mathsf{ML}_1(\theta)) = \mathbb{E}[\mathrm{Var}(\mathsf{ML}_1(\theta) \mid N)] + \mathrm{Var}(\mathbb{E}[\mathsf{ML}_1(\theta) \mid N])$ by the law of total variance. For the first term, we have $\mathrm{Var}(\mathsf{ML}_1(\theta) \mid N) = \theta_1^2 \frac{k_L(N+1-k_L)}{(N+1)^2(N+2)}\mathbb{1}\{N \geq 1\} \leq \frac{\theta_1^2}{N+1}$, and then from the previous argument $\mathbb{E}[\mathrm{Var}(\mathsf{ML}_1(\theta) \mid N)] = O(1/n)$. For the second term, $\mathrm{Var}(\mathbb{E}[\mathsf{ML}_1(\theta) \mid N]) = \mathrm{Var}\left(\mathbb{E}[\mathsf{ML}_1(\theta) \mid N] - \frac{\theta_1}{2}\right) \leq \mathbb{E}\left[\left(\mathbb{E}[\mathsf{ML}_1(\theta) \mid N] - \frac{\theta_1}{2}\right)^2\right] \leq \mathbb{E}\left[\frac{\theta_1^2}{(N+1)^2}\right] \leq \theta_1^2\mathbb{E}\left[\frac{1}{N+1}\right] = O\left(\frac{1}{n}\right)$. Thus, $\mathrm{Var}(\mathsf{ML}_1(\boldsymbol{\theta})) = O(1/n)$, completing the proof. □

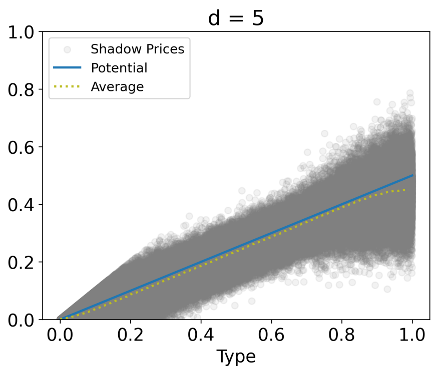
C.1. Linear Relaxation and Dual Variables

In this section, we provide details on the LP formulation and dual variables used for the HD and AD benchmarks. Given market density $d \geq 1$ and full information of the sequence of arrivals $\boldsymbol{\theta} \in \Theta$, we can define a linear program from (5) by relaxing the integrality constraints. It is known that the LP relaxation is integral only for bipartite graphs. However, we observe in Figure 8 that the integrality gap is small, and thus the value of the LP can be reasonably used as a benchmark.

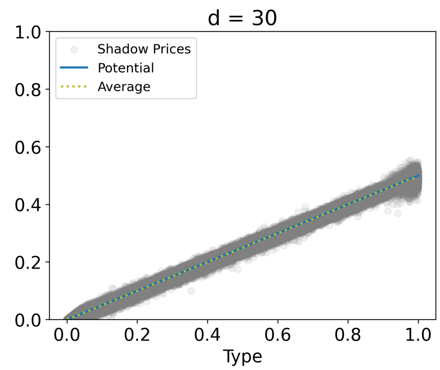
Also, note that the potential can always be used as a feasible (but not necessarily optimal) solution to the dual LP

$$\begin{aligned} \min_{\lambda} \quad & \sum_{j \in [n]} \lambda_j \\ \text{s.t.} \quad & \lambda_j + \lambda_k \geq r(\theta_j, \theta_k), \quad j \neq k, |j-k| \leq d, \\ & \lambda_j \geq 0, \quad j \in [n]. \end{aligned} \tag{16}$$

Indeed, $r \geq 0$ and $r(\theta, \theta') \leq \frac{1}{2}r(\theta,\theta') + \frac{1}{2}r(\theta,\theta') \leq p(\theta) + p(\theta')$, for any j, k, verifying the dual feasibility constraints.



(a) Density $d = 5$.



(b) Density $d = 30$.

Figure 9 Scatter plot of shadow prices, average, and potential.

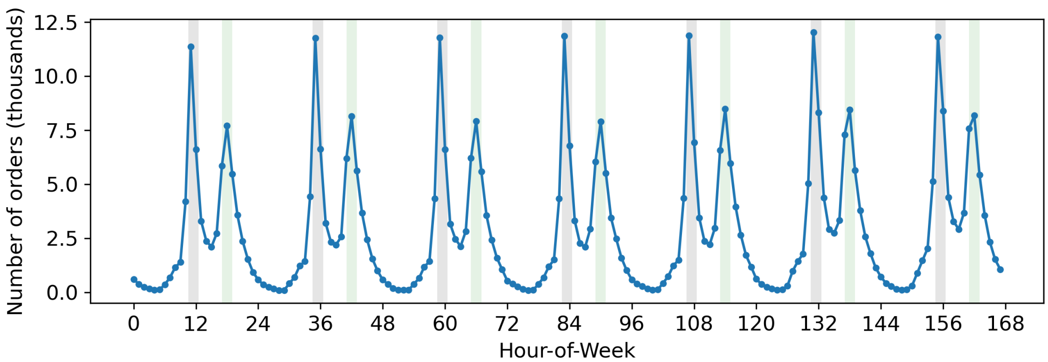


Figure 10 Average number of orders by hour-of-week in Meituan data.

When we extract the shadow prices from the 400 instances in the historical data in Subsection 4.2, we can see that these concentrate around potential, as density increases (Figure 9). This is aligned with our results in Appendix C.

D. Meituan Data - Market Dynamics

In this appendix, we briefly describe the Meituan dataset made public by the INFORMS TSL Data-Driven Challenge, and provide high-level descriptions of the market-dynamics.

D.1. Temporal Dynamics

Figure 10 provides the average number of orders per hour by *hour-of-week* (HOW). HOW 0 corresponds to midnight to 1am on Mondays, and HOW 1 corresponds to 1am to 2am on Mondays, and so on. The gray bars indicate the peak lunch hours (10:30am-1:30pm); the green bars indicate peak dinner hours (5pm-8pm). We can see that the platform observes highest order volume during lunch time, averaging around 200 order per minute between noon and 1pm.

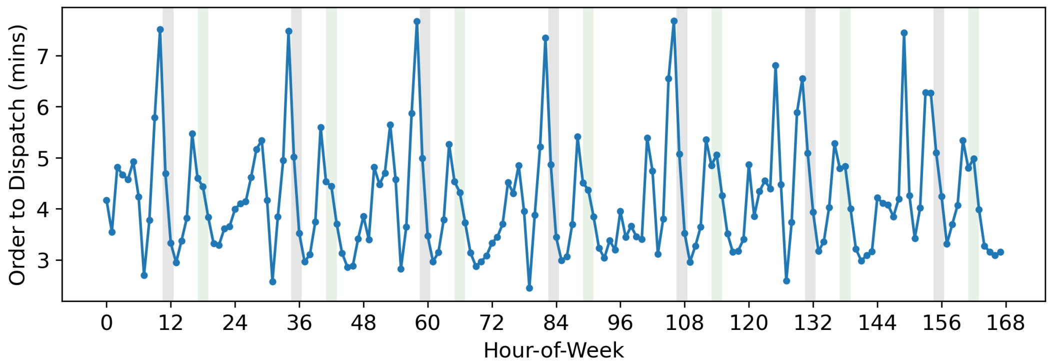


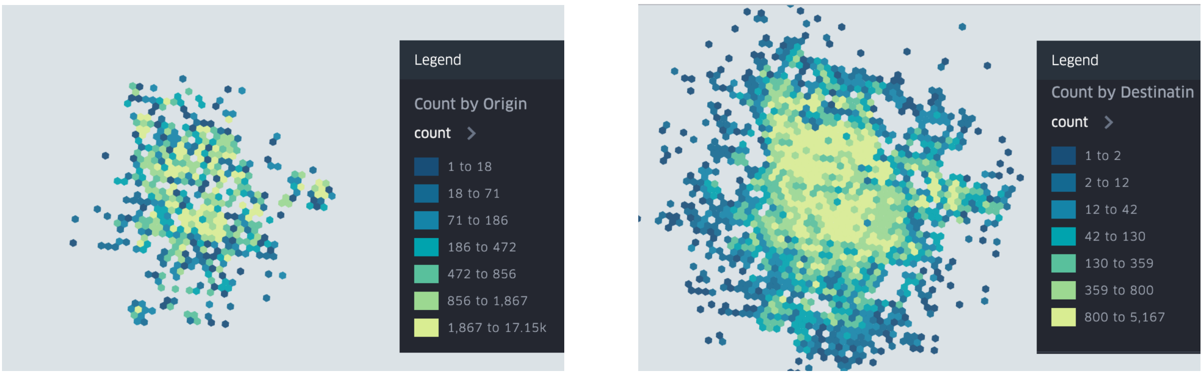
Figure 11 The average amount of time (minutes) that elapses between (i) when an order is placed by the customer, and (ii) when the order is dispatched to a delivery driver for the first time.

Figure 1 in Section 1 illustrates the fraction of orders placed during each hour-of-week that is pooled with at least one other order. This is computed by combining the order-level data (which contains the timestamp of each order and the order ID) and the "wave level" data, where each wave corresponds to a list of orders that were pooled and assigned to the same driver (Zhao et al. 2024). Each wave may contain more than two orders, though some orders may have been dropped off by the driver before some other orders were picked up. The orders that are not pooled with any other order correspond to waves with only one order, and Figure 1 illustrates 1 minus this fraction of orders that are not pooled.

Finally, Figure 11 illustrates the amount of time that elapses, between platform_order_time, i.e. when each order is placed by the customer, and first dispatch_time, i.e. when the order was offered to a delivery driver for the first time. Intuitively, this is the amount of time it takes the platform to start dispatching each order, and the average is around 4 minutes across all orders. Despite the fact that it takes an estimated 12 minutes for the restaurant to prepare the orders, the platform starts to dispatch the orders shortly after they are placed. This is potentially due to the fact that it takes time for drivers to get to the restaurants , so orders are dispatched before they are ready in order to reduce the total wait time experienced by the customers.

D.2. Spatial Distribution

We now visualize the distribution of orders in space, and provide the distribution of order volume by origin-destination hexagon pairs. We use resolution-9 hexagons from the h3 package, which is the resolution that performs the best for AD when the matching window is at least 2 minutes. The average size of each hexagon is 0.1km^2, and the average edge length is around 0.2km(Uber 2018).



(a) By Origin. (b) By Destination.

Figure 12 Number of orders by order origin hexagon (left) and order destination hexagon (right).

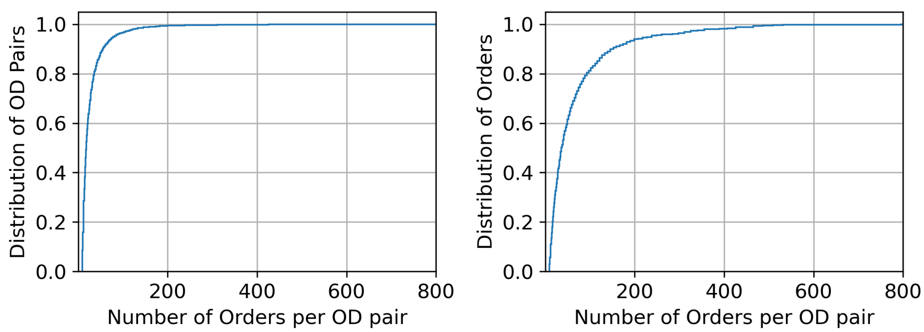


Figure 13 Distribution (CDF) of the number of orders per origin-destination hexagon pair (resolution-9), unweighted (left) and weighted by the number of orders (right).

First, Figure 12 provides heat-maps of trip volume by order origin and destination, respectively. We can see that the order origins are more concentrated than destinations, which is aligned with the more concentrated locations of restaurants in comparison to residential areas and office buildings. Note that both distributions are highly skewed. For hexagons with at least one originating order, the median, average, and maximum number of orders are 330, 960, and 17151, respectively. For hexagons that are the destination of at least one order, the median, average, and maximum order count are 71, 327 and 5167, respectively.

Finally, Figure 13 presents the distribution order volume by origin and destination hexagon pairs. From the unweighted CDF on the left, we can see that the vast majority of OD pairs have fewer than 100 orders. Out of all OD hexagon pairs with at least one order, the median, average, and maximum number of orders are 3, 8.22, and 808, respectively. The CDF on the right is weighted by order count. Overall, roughly 14% of orders are associated with OD pairs with at least 100 orders, and 72.8% of orders are from OD pairs with at least 10 orders.

E. Additional Simulation Results

We report in this section additional performance metrics for settings studied in Section 4 of the paper. Moreover, we also present numerical results for more general synthetic environments, including two-dimensional locations, non-uniform spatial distributions, and different reward topologies.

E.1. Match Rate and Saving Fraction

We consider in this section two additional performance metrics: the *match rate*, i.e. the fraction of jobs that were pooled instead of dispatched on their own, and the *saving fraction*, i.e. the fraction of the total distance that is reduced by pooling, relative to the total travel distance without any pooling (see Aouad and Saritaç 2020).

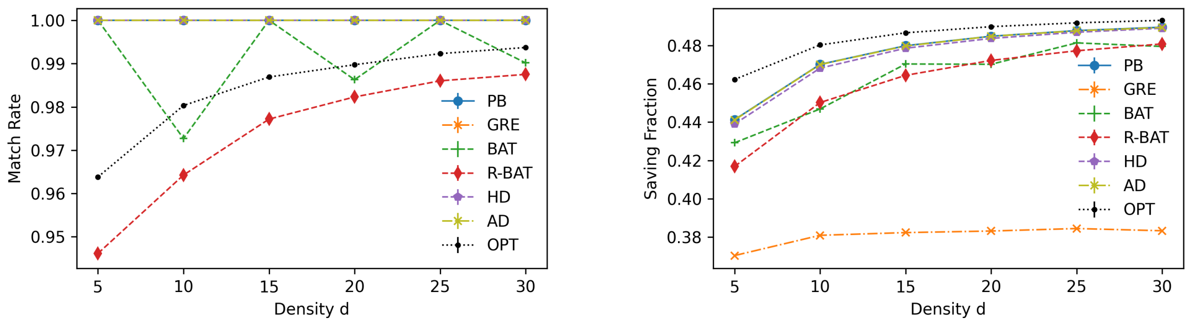
Both regret and reward ratio are performance metrics that compare pooling algorithm relative to the hindsight optimal pooling outcome. The saving fraction illustrates the benefit of delivery pooling in comparison to the total distance traveled, and is upper bounded by 0.5 (since reward from pooling a job cannot exceed its distance). The match rate, as shown in Figures 14a and 15a, highlights an additional desirable property of PB and, more generally, of *index-based greedy matching* algorithms: they pool as many jobs as possible. This is desirable in practice, since drivers who are offered just only one order from a platform may try to pool orders from competing platforms (see e.g. Reddit 2022, 2023), leading to poor service reliability for customers.

E.1.1. Uniform one-dimensional case. Figure 14 compares the match rate and the saving fraction achieved for the one-dimensional synthetic environment studied in Subsection 4.2. In this setting, all jobs share the same origin at 0 and destinations are drawn uniformly at random from $[0, 1]$.

Figure 14b shows that the reduction in travel distance can be remarkably high for this setting, with OPT saving more than 46% even at low density levels.

The match rate in Figure 14a illustrates that all *index-based greedy matching* algorithms match every one of the $n = 1000$ jobs. This is because (i) the reward from matching any two jobs is non-negative in this 1D setting, and (ii) the total number of jobs is even. On the other hand, BAT matches every job only if $d + 1$ is even, since every time it computes an optimal matching, the *batch size* is $d + 1$ and it dispatches *all* available jobs in pooled trips.

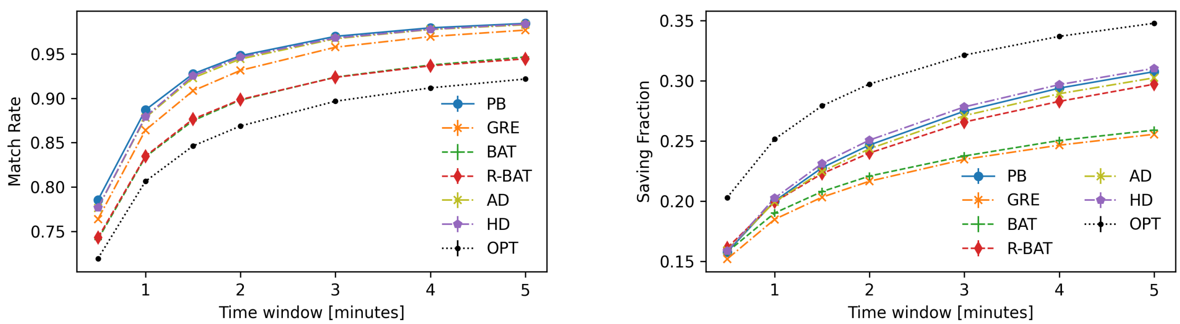
E.1.2. Meituan data. Figure 15 provides a comparison of the match rate and the saving fraction for the setting presented in Subsection 4.3. Recall that as we extend our definition of pooling reward to allow for 2D locations with heterogeneous origins, it is no longer guaranteed to be non-negative. Consequently, both the matching rate and saving fraction of all algorithms are lower than in the one-dimensional setting.



(a) Average match rate.

(b) Average saving fraction.

Figure 14 Match rate and saving fraction in random 1D instances.



(a) Average match rate.

(b) Average saving fraction.

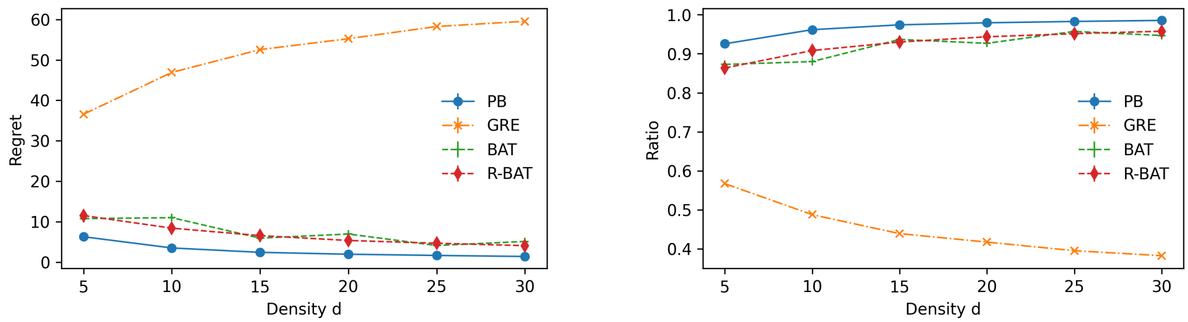
Figure 15 Match rate and saving fraction for Meituan Order-Level Data.

Nonetheless, we can see that if the platform allows a time window of at least 1 minute for each order before it has to be dispatched, PB achieves a reduction in distance traveled by over 20%. At 5 minutes, this fraction increases to roughly 30%, which is the best among all tested *practical* heuristics, i.e. excluding HD and OPT.

Figure 15a shows that greedy algorithms (GRE, PB, HD, AD) still achieve substantially higher match rates in comparison to BAT, R – BAT, and OPT. In particular, PB consistently pools 5-10% more jobs than OPT.

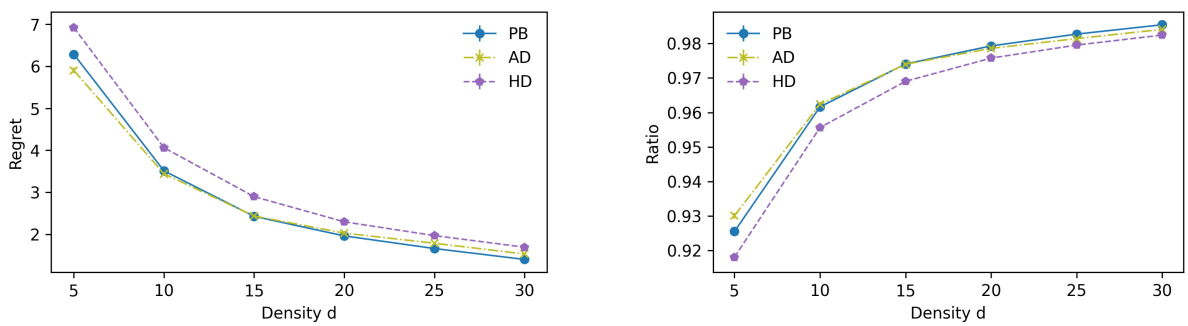
E.2. Non-Uniform One-Dimensional Case

In this section, we show that our experimental results in one-dimensional synthetic data are robust to non-uniform distribution of types. We consider an analogous setting from Subsection 4.2, but now each job type is drawn IID from a distribution Beta(0.5, 2). Overall, the results are consistent (see Figures 16 and 17), with the main difference being that PB performs slightly worse than AD for low densities. Nonetheless, PB ends up outperforming all benchmarks starting from $d \geq 15$.



(a) Average regret. (b) Average ratio.

Figure 16 Average regret and reward ratio for forecast-agnostic heuristics in non-uniform 1D instances.



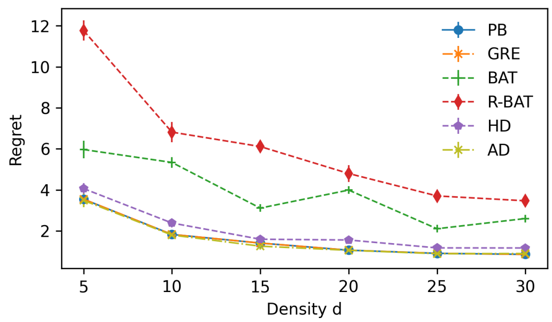
(a) Average regret. (b) Average ratio.

Figure 17 Average regret and reward ratio for forecast-aware heuristics in non-uniform 1D instances.

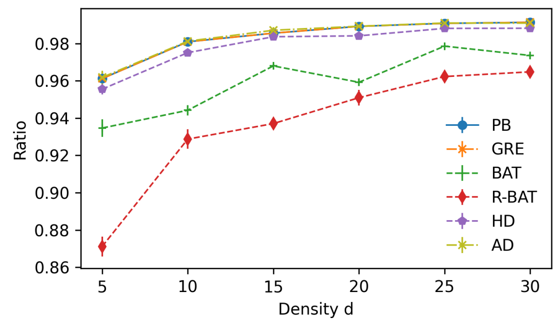
E.3. Different Reward Topology

In this appendix, we present numerical results under the two alternative reward topologies defined in (3) and (4). This illustrates the importance of reward topology in determining the performance of algorithms. As in Subsection 4.2, we generate instances of size $n = 1000$, where each type is drawn IID from a uniform distribution on $[0, 1]$. Overall, we observe that PB performs on par with the best realistic benchmarks (i.e., outside HD and OPT). This suggests that PB could be a leading heuristic even beyond the reward function $r(\theta, \theta') = \min\{\theta, \theta'\}$ from delivery pooling, although here we are still restricting to continuous distributions over (one-dimensional) metric spaces.

E.3.1. Reward function $r(\theta, \theta') = 1 - |\theta - \theta'|$. Recall that in this case the potential of a job is $1/2$, independent of the job type, and thus both PB and GRE are the same algorithm (see Appendix B.1). Figure 18 shows their superior performance compared to batching-based heuristics. Moreover, as in Subsection 4.2, forecast-aware heuristics are no better than PB.



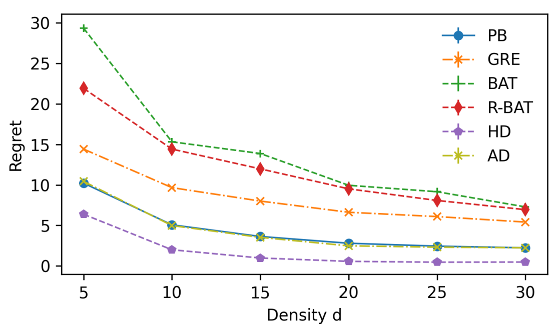
(a) Average regret.



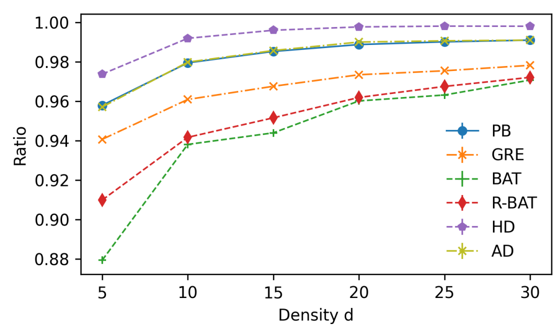
(b) Average ratio.

Figure 18 Average regret and reward ratio for forecast-agnostic heuristics in random 1D instances under $r(\theta,\theta') = 1 - |\theta - \theta'|$.

E.3.2. Reward function $r(\theta,\theta') = |\theta - \theta'|$. Consider $\Theta = [0,1]$ and $r(\theta,\theta') = |\theta - \theta'|$ (analyzed in Appendix B.2). Figure 19 highlights that greedy-like algorithms perform better than batching-based heuristics, with PB outperforming every practical matching algorithm (i.e., aside from HD and OPT that use hindsight information).



(a) Average regret.



(b) Average ratio.

Figure 19 Average regret and reward ratio for forecast-agnostic heuristics in random 1D instances under $r(\theta,\theta') = |\theta - \theta'|$.