Edge Weighted Online Windowed Matching

Itai Ashlagi[*] Maximilien Burq[†] Chinmoy Dutta[‡] Patrick Jaillet[§]
Amin Saberi[¶] Chris Sholley[‖]

[*]Stanford, email: *iashlagi@stanford.edu*
[†]Verily Life Sciences, email: *burq.maximilien@gmail.com*
[‡]Turing Research, email: *chinmoy@turingres.com*
[§]MIT, email: *jaillet@mit.edu*
[¶]Stanford, email: *saberi@stanford.edu*
[‖]Lyft, email: *chris@lyft.com*

Abstract

Consider a matching problem, in which agents arrive to a marketplace over time and leave after d time periods. Agents can only be matched while present in the marketplace. Each pair of agents can yield a different match value and a social planner seeks to maximize the total value from matches over a finite time horizon.

First we study the case in which vertices arrive in an adversarial order. We provide a randomized $1/4$-competitive algorithm building on a result by Feldman et al. (2009a) and Lehmann et al. (2006). When departure times are drawn independently from a distribution with non-decreasing hazard rate, for which we establish a $1/8$-competitive algorithm. When the arrival order is chosen uniformly at random, a batching algorithm which computes a maximum-weighted matching every $(d+1)$ periods, is shown to be 0.279-competitive.

Keywords: Edge-weighted online matching, non-bipartite matching, windowed matching, adversarial arrivals, random order arrivals.

1 Introduction

We study the following online weighted matching problem. Agents, represented as vertices in a general (not necessarily bipartite) graph, arrive sequentially in a market over n time periods. Potential matches, represented by non-directed edges, have heterogeneous weights (match values). Each agent that is not matched within d time periods leaves the market unmatched. The edges between two vertices along with its weight is observed only when both the agents are in the market. The goal is to design an online matching algorithm that generates a matching with as large total weight as possible. After d time period since arrival, a vertex is said to be *critical* at which point the algorithm must decide either to match it with one of its existing neighbors or leave it unmatched.

Our problem is partly inspired by the challenge ride-sharing platforms face when attempting to carpool passengers. Agents in our problem can be viewed as passengers who request to share a ride. Leaving unmatched after d periods can be interpreted as sending a passenger who requests to carpool on a single trip instead of being matched with another passenger. Needless to say, the problem in consideration focuses on the matching angle and abstracts away from relevant carpooling features, such as prices (Vickrey, 1965), costs, travel destinations and even possible predictions over future demand.

We begin by studying the setting in which vertices arrive in an **adversarial order**. We introduce a $1/4$-competitive algorithm termed POSTPONED GREEDY (PG). We also show a hardness result that no algorithm achieves a competitive ratio that is higher than $1/2$.

The key idea behind PG is to consider a virtual bipartite graph, in which each vertex is duplicated into a *buyer* and a *seller* copy. Next we proceed in a manner similar to Feldman et al. (2009a): tentatively match each arriving buyer copy to the seller copy that maximizes its margin, i.e. the difference between the weight of its edge with the seller and the weight of the seller's current matched edge. We enforce that the seller copy does not match before the vertex becomes critical. This allows to postpone the matching decision and learn more about the graph structure and the likely matchings.

We extend the model to the case in which departure of vertices are determined stochastically according to a memoryless distribution. If departure times of vertices are revealed to the algorithm just when they become critical and are about to leave the market, the PG algorithm can be adapted to achieve a competitive ratio of $1/8$.

Next we study the setting in which vertices arrive in a **random order**. We analyze a BATCHING algorithm which, every $d+1$ time steps, computes a maximum weighted matching among the last $d+1$ arrivals.[1] Vertices that do not match within the batch, leave and remain unmatched forever. We show that when the number of vertices is sufficiently large, batching is 0.279-competitive. We note that in contrast to our result, Aouad and Saritac (2020) show that BATCHING can be arbitrarily bad in a different setting where vertices have widely heterogeneous sojourn times in the market. (Also note that their negative result holds in the cost minimization version of the problem.)

The analysis of the BATCHING algorithm proceeds in three steps. First, we show that the competitive ratio is bounded by the solution to a graph covering problem. Second, we show how covers for small graphs can be extended to covers for larger graphs. Finally, we establish a reduction that allows us to consider only a finite set of values for d. The proof concludes with a computer-aided argument for graphs in the finite family.

Note that in several models of online matching with adversarial arrivals, it is not possible to obtain any constant competitive ratio for edge-weighted graphs. At a high level, the difficulty stems from the inherent trade-off between matching a vertex with a current neighbor (in which case it foregoes a potential high value future match) and waiting in anticipation of future arrivals (in which case it foregoes current matches and a future match may never arrive). In the case of classic bipartite matching, Feldman et al. (2009a) circumvented this difficulty by imposing the *free disposal* assumption which lets an offline vertex tentatively match an arriving online vertex (so as to not forego the current possibility) while keeping the option open to revoke the match in future for a better one (so as to not forego a future better possibility). In our model, the difficulty is circumvented because of the crucial guarantee that vertices depart in order of arrivals (deterministically or stochastically). This allows us to commit the match for a vertex only after learning all its match possibilities with constant probability. Our techniques of making separate copies of arriving vertices and postponing match decisions to better learn the graph structure might find use in other matching problems where this crucial property of relatively homogeneous sojourn times and orderly departures is guaranteed.

Note that our bounds do not specifically depend on the parameter d which control departures. For the adversarial order of arrivals, our results essentially depend upon the guarantee that for any two vertices, there is a constant probability that the vertex that arrives earlier departs earlier as well. (See proposition 7.) This is in particular guaranteed when each vertex stays in the system for exactly d time steps. For the random order of arrivals, the parameter

[1]Batching is commonly used in ride-sharing platforms as well as in numerous kidney exchange platforms (from personal communications).

d essentially defines the structure of the algorithm by setting the batching window.

Related Literature This paper contributes to the literature on online matching. In the classic problem, introduced in Karp et al. (1990), the graph is bipartite with vertices on one side waiting, while those on the other side arriving sequentially that have to be matched (or left unmatched forever) *immediately and irrevocably* upon arrival. This work has numerous extensions, for example to stochastic arrivals and in the adwords context Mehta et al. (2007); Goel and Mehta (2008); Feldman et al. (2009b); Manshadi et al. (2011); Jaillet and Lu (2013). See Mehta (2013) for a detailed survey. Our work contributes to this literature in three ways. First, our graph can be non-bipartite, which is the case in applications such as ride-sharing and kidney exchange. Second, all vertices arrive over time and remain for some given time until they are matched or hit their deadline and depart. Third, we provide algorithms that perform well on edge-weighted graphs. Closely related are Huang et al. (2018, 2019), which study a similar model to ours in the non-weighted case, but allow departure times to be adversarial. The results in Huang et al. (2018) are obtained by extending the primal-dual analysis technique of Devanur et al. (2013) with a novel gain sharing and compensation mechanism. However, those techniques do not extend to our edge-weighted case. Another related work is Aouad and Saritac (2020) which studies dynamic stochastic matching on edge-weighted graphs where vertex arrivals and departures are stochastic and heterogeneous. In particular, vertices of different types arrive as independent Poisson processes and abandon with different rates. The weight of an edge between two vertices is a function of their types. In the reward maximization version of their problem, the authors give a matching algorithm with an approximation ratio of $\frac{e-1}{4e}$. The analysis of the algorithm is based on a novel linear programming based fluid relaxation of their problem.

Several papers consider the problem of dynamic matching in the edge-weighted case. Feldman et al. (2009a) find that in the classic online bipartite setting, no algorithm achieves a constant approximation. They introduce a *free disposal* assumption, which allows an offline vertex to discard its existing match vertex in favor of a new arriving vertex. They show, based on an algorithm by Lehmann et al. (2006), that a greedy algorithm that matches a vertex to the vertex with highest marginal utility, is 0.5-competitive. We build on this result for a special class of bipartite graphs. Ezra et al. (2020) studied edge weighted matching in general graphs under the vertex and edge arrival models in the secretary setting. Their vertex arrival model somewhat resembles the random order arrival in our model but vertices do not depart the system in their model. They showed a competitive ratio of $\frac{5}{12}$ for their case. In the adversarial setting, Emek et al. (2016); Ashlagi et al. (2017) study the problem of minimizing the sum of distances between matched vertices and the sum of their waiting times. In their model no vertex leaves unmatched. Few papers consider the stochastic setting (Baccara et al., 2015; Ozkan and Ward, 2020; Hu and Zhou, 2020). These papers find that some waiting before matching is beneficial for improving efficiency.

The paper is also related to several other streams of literature. One stream of papers is concerned with job or packet scheduling. Jobs arrive online to a buffer, and reveal upon arrival the deadline by which they need to be scheduled. The algorithm can schedule at most one job per time and the value of scheduling a job is independent of the time slot. Constant approximation algorithms are given by Chin et al. (2006) and Li et al. (2005).

This paper is motivated by ride-sharing challenges. It is worth noting that carpooling, as argued by Ostrovsky and Schwarz (2018), is a major technological advancement towards reducing congestion and traffic costs (and, as they quote from the US census bureau, more than 75% of the US population still drive alone to work). Santi et al. (2014) finds that about 80% of rides in Manhattan could be shared by two passengers. Most studies focus on rebalancing or dispatching problems without pooling (Pavone et al., 2012; Zhang and Pavone, 2014; Santi et al., 2014; Spieser et al., 2016; Banerjee et al., 2018; Kanoria and Qian, 2019). Alonso-Mora et al. (2017) studies real-time high-capacity ride-sharing. However, these papers do not consider a graph-theoretic online matching problem.

Finally, several papers study dynamic matching problems that are motivated by kidney exchange challenges (Ünver, 2010; Anderson et al., 2015; Dickerson et al., 2013; Ashlagi et al., 2019; Blum and Mansour, 2020). These papers mostly focus on random graphs with no weights. Closer to our paper is Akbarpour et al. (2020), which finds that in a sparse random graph, knowledge about the departure time of a vertex is beneficial and matching a vertex only when it becomes critical performs well. We deviate from these papers in two ways; we consider the edge-weighted case, and, we make no assumption on the graph structure.

2 Model

Let $[n] = \{1, \ldots, n\}$. Consider an edge-weighted undirected graph G with n vertices indexed by $i \in [n]$. We will let $v_{ij} \geq 0$ denote the weight (or value) of the undirected edge between vertices i and j.

The vertices arrive sequentially over n time periods, denoted by $t \in [n]$. Denote by $\sigma(i)$ the arrival time of vertex i. For any two vertices i and j with $\sigma(i) < \sigma(j)$, the weight on the edge between i and j is observed only after vertex j has arrived.

For $d \geq 1$, the **online graph with deadline** d, denoted by $G_{d,\sigma}$, has the same vertices as G, and an edge between vertices i and j in G exists if an only if $|\sigma(i) - \sigma(j)| \leq d$. We say that i becomes **critical** at time $\sigma(i) + d$.

An *online matching algorithm* receives as input a graph $G_{d,\sigma}$ and determines at each period which vertices to match with each other (if at all); vertices can be matched only between the time they arrived and the time they become critical.

Given the order σ, the value of the maximum weight matching that can be generated is given by the integer program (Offline Matching).

$$\begin{aligned}
\text{maximize} \quad & \sum_{i,j\in[n]:\ i<j,\ |\sigma(i)-\sigma(j)|\leq d} x_{ij}v_{ij} & & \\
\text{subject to} \quad & \sum_{j\in[n]:\ i<j} x_{ij} + \sum_{j\in[n]:\ j<i} x_{ji} & \leq 1 \quad & \forall i \in [n], \\
& x_{ij} & \in \{0,1\} \quad & \forall i < j,\ i,j \in [n].
\end{aligned} \tag{Offline Matching}$$

The goal is to develop an online algorithm that for each graph $G_{d,\sigma}$ outputs a matching with a large total weight. More precisely, we seek to design a randomized online algorithm that obtains in expectation a high fraction of the expected maximum-weight of a matching over $G_{d,\sigma}$.

Two models for arrivals of vertices will be considered in this paper. First is the adversarial Order(AO) setting, in which $\sigma(i) = i$. Second is the random order (RO) setting, in which σ is sampled uniformly at random from S_n, the set of all possible permutations over $[n] = \{1, \dots, n\}$.

To illustrate a natural trade-off, consider the example in Figure 1 for $d = 1$. At time 2, the online algorithm can either match vertices 1 and 2 or let vertex 1 remain unmatched. This simple example shows that no deterministic algorithm can obtain a constant competitive ratio. Furthermore, no algorithm can achieve a competitive ratio higher than $1/2$.

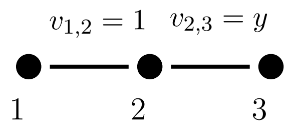


Figure 1: Let $d = 1$. Therefore, there is no edge between vertices 1 and 3. The algorithm needs to decide whether to match 1 with 2 and collect $v_{1,2}$ without knowing y.

3 Adversarial Order of Arrivals

It is instructive to first consider a special case which we call the constrained bipartite case. Here, the underlying input graph is bipartite. For exposition purposes, we refer to vertices on one side of the bipartition as *buyers* and on the other side as *sellers*. The online graph is further constrained so that there is no edge between a buyer and a seller if the buyer arrives before the seller. For such graphs, we will show that a greedy algorithm given by Feldman et al. (2009a) is 0.5-competitive.

We will then build on this algorithm to design a randomized $1/4$-competitive algorithm for arbitrary graphs. The algorithm we construct will transform the input graph into a constrained bipartite graph in an online manner by making two copies, a seller copy and a buyer copy, of each arriving vertex. We will prevent both copies of a vertex to be matched by assigning vertices the status of a seller or a buyer in a synchronized random fashion.

3.1 Constrained Bipartite Case

Let G be a bipartite graph and σ be the order of arrivals. The online graph with deadline d, $G_{d,\sigma}$, is called **constrained bipartite** if for every seller s and buyer b, there is no edge between s and b if $\sigma(b) < \sigma(s)$, i.e. b and s cannot match if b arrives before s.

Consider the following greedy algorithm, which attempts to match buyers in their arriving order. The marginal value of a seller for an arriving buyer is defined as the increment in the match value for the seller if matched with the arriving buyer, i.e. the weight of the edge between the seller and the arriving buyer minus the weight of the current matched edge of the seller, if any. An arriving buyer b is matched to the seller with the highest marginal value for the buyer if the marginal value is positive. If the seller is already matched with another buyer b', b' becomes unmatched and never matches again. Note that a buyer gets only one chance to get matched immediately on arrival but may subsequently get unmatched. The algorithm is formally presented as Algorihm 1.

ALGORITHM 1: GREEDY (Feldman et al., 2009a)

```
• Input: constrained bipartite online graph with deadline $d$, $G_{d,\sigma}$.
• Let $S$ denote the set of sellers that are still in the market. Set $S = \emptyset$.
• For each time period $t = 1, \dots, n$, process events in the following way:
    1. Vertex $j$ arrives:
        (a) If $j$ is a seller:
            i. Initialize $m(j) = null$ and $p(j) = 0$.
            ii. $S \leftarrow S \cup \{j\}$.
        (b) If $j$ is a buyer:
            i. Set $s = \text{argmax}_{s' \in S}\{v_{s'j} - p(s')\}$.
            ii. If $v_{sj} - p(s) > 0$, set $m(s) = j$ and $p(s) = v_{sj}$.
    2. Vertex $i$ becomes critical:
        (a) If $i$ is a seller:
            i. match $i$ to buyer $b = m(i)$ if $m(i) \neq null$.
            ii. $S \leftarrow S \setminus \{i\}$.
```

Proposition 1 (Feldman et al. (2009a))**.** *GREEDY is 1/2-competitive for constrained bipartite online graphs.*

Feldman et al. (2009a) prove that this algorithm is 1/2-competitive for an online matching problem with *free disposal*. In their setting, all seller exists in the system at the outset and buyers arrive one at a time. Buyers needs to be matched immediately on arrival or left unmatched forever. A matched seller is allowed to forego its previous match and form a new match at any point (in which case the buyer which was its previous match becomes unmatched). The algorithm provides the same guarantees for a constrained bipartite graph since, by definition of such a graph, any seller with whom an arriving buyer has an edge is guaranteed to have already arrived. Thus, from the point view of the algorithm, it makes no difference to assume that all the sellers are present in the system at the outset. The result, in fact, follows from a result by Lehmann et al. (2006) who study combinatorial auctions with submodular valuations. The key behind the proof is that the marginal value of a seller for an arriving buyer is submodular. For completeness, we include a formal proof below.

Proof of Proposition 1. Let $p^f(s)$ be the final match value for a seller node s. Let $q(b)$ be the largest positive marginal value of an already arrived seller that is still in the market when buyer b arrives: $q(b) = \max\{\max_s\{v_{sb} - p(s)\}, 0\}$, where $p(s)$ is the match value of seller s when buyer b arrives. We call $q(b)$ the margin for buyer b. Since every increase in the match value of a seller is associated with the margin of a buyer, we have

$$\sum_{s:\text{seller}} p_f(s) = \sum_{b:\text{buyer}} q(b).$$

Let ALG be the weight of the matching constructed by GREEDY. We have ALG $= \sum_s p_f(s)$. Let OFF be the maximum weight of any matching that can be constructed. To obtain an upper bound on OFF, consider the dual of the offline matching linear program (Offline Matching).

$$\begin{array}{lll} \text{minimize} & \sum_{i\in[n]} \lambda_i & \\ \text{subject to} & \lambda_i + \lambda_j \geq v_{ij} & \forall i, j \in [n],\ \text{s.t. } i < j,\ |\sigma(i) - \sigma(j)| \leq d \\ & \lambda_i \geq 0 & \forall i \in [n]. \end{array} \tag{Offline Dual}$$

Consider an edge between seller s and buyer b. When buyer b arrives, we have $q(b) \geq v_{sb} - p(s)$. Since the value of a seller's match never decreases during the run of the algorithm, we also have $p^f(s) \geq p(s)$. Thus, $p^f(s) + q(b) \geq v_{sb}$, and $\{p^f(s)\}_{s:\text{seller}} \cup \{q(b)\}_{b:\text{buyer}}$ is a feasible solution to (Offline Dual).

We conclude OFF $\leq \sum_s p^f(s) + \sum_b q(b) = 2\sum_s p^f(s) = 2$ALG. □

3.2 General Case

In this section, we extend the GREEDY algorithm for the constrained bipartite case to the general case. A naive way to generate an constrained bipartite online graph from an arbitrary input graph is to randomly assign each vertex to be either a seller or a buyer, independently and with probability $1/2$. Then only keep the edges between each buyer and all the sellers who arrived before her. Observe that for vertices i, j with $\sigma(i) < \sigma(j)$, edge (i, j) in the original graph remains in the generated constrained bipartite online graph with probability $1/4$ (if i is a seller and j is a buyer). We then use Proposition 1 to prove that naive greedy is $1/8$-competitive.

Corollary 1. *NAIVE GREEDY is $1/8$-competitive for arbitrary online graphs.*

One source of inefficiency in the NAIVE GREEDY algorithm is that the decision whether a vertex becomes a seller or a buyer is done independently at random and without taking the graph structure into consideration. We next introduce the POSTPONED GREEDY

ALGORITHM 2: NAIVE GREEDY

```
• Input: an online graph with deadline $d$, $G_{d,\sigma}$.
• For each time period $t = 1, \dots, n$:
    1. Toss a fair coin to decide whether arriving vertex $j$ is a *seller* or a *buyer*.
       Construct the online constrained bipartite graph $\tilde{G}(d, \sigma)$ by only keeping the
       edges between each buyer and the sellers who arrived before her.
    2. Process vertex arrival and vertex becoming critical according to the GREEDY
       algorithm on $\tilde{G}(d, \sigma)$.
```

algorithm that defers these decisions as long as possible in order to construct the constrained bipartite graph more carefully.

When a vertex j arrives, we add two copies of j to a virtual graph: a seller copy s_j and a buyer copy b_j. Let S_t be the set of seller copies that have already arrived and still in the market at period t. On arrival of vertex j at time period j, the seller copy s_j does not have any edge, and the buyer copy b_j has edges with every vertex $s_r \in S_j$ with value v_{rj}. (We assume $v_{rj} = 0$ if (r, j) is not an edge in the original graph.) Then we run the GREEDY algorithm with the virtual graph as input. When a vertex i becomes critical, s_i becomes critical in the virtual graph, and we compute its matches generated by GREEDY.

Both the seller and the buyer copies of a vertex can be matched in this process. If we were to honor both matches, the outcome would correspond to a 2-matching, in which each vertex has degree at most 2. Now observe that because of the structure of the constrained bipartite graph, this 2-matching does not have any cycles; it is just a collection of disjoint paths. We decompose each path into two disjoint matchings and choose each matching with probability 1/2.

In order to do that, the algorithm must determine, for each original vertex i, whether the seller copy s_i or the buyer copy b_i will be used in the final matching. We say that i is a *seller* or a *buyer* depending on which copy is used. The vertex i has its status *undetermined* until the algorithm has determined which copy will be used. When an undetermined vertex becomes critical, the algorithm flips a fair coin to decide whether it is a seller or buyer. This decision is then propagated to the next vertex in the 2-matching: if i is a *seller* then the next vertex will be a *buyer* and vice-versa. This mechanism ensures that assignments are correlated and saves a factor 2 compared to uncorrelated assignments in the NAIVE GREEDY algorithm.

Theorem 1. *POSTPONED GREEDY is $1/4$-competitive for arbitrary online graphs.*

Proof. Let PG denote the expected weight of the matching constructed by the postponed greedy algorithm. For a vertex i, let $p^f(s_i)$ denote the final value of its seller copy s_i's match. If the status of i is a seller, then PG collects $p^f(s_i)$. Note that a vertex gets the status of a seller with probability exactly 1/2 independently of the 2-matching.

$$\mathrm{PG} \;= \mathbb{E}\left[\sum_{i \text{ is a seller}} p^f(s_i)\right] = \frac{1}{2}\sum_{i\in[n]} p^f(s_i).$$

For a vertex j, we define the margin of its buyer copy to be the largest positive marginal value of the seller copy of any previously arrived vertex that is still in the market when j arrives: $q(b_j) = \max\{\max_{s_r \in S_j} v_{s_r b_j} - p(s_r), 0\}$. Here S_j demote the set of seller copies that are still in the market at time step j (when vertex j arrives). Note that every increase in a seller copy's match value corresponds to a buyer copy's margin. This implies

$$\sum_{i\in[n]} p^f(s_i) = \sum_{j\in[n]} q(b_j).$$

Let OFF be the value of the maximum weight matching that can be generated. Similar to the proof of proposition 1, we obtain an upper bound on OFF by considering the dual (Offline Dual) of the offline matching linear program (Offline Matching).

Note that $\sigma(i) = i$ for the adversarial arrival order. Let i and j be two vertices with $i < j$ and $j - i \leq d$. When j arrives, we have $q(b_j) \geq v_{ij} - p(s_i)$. Together with the fact that $p(s_i)$ does not decrease over time, this implies that $\{p^f(s_i) + q(b_i)\}_{i\in[n]}$ is a feasible solution to (Offline Dual).

We can conclude that $\mathrm{OFF} \leq \sum_i p^f(s_i) + q(b_i) = 2\sum_i p^f(s_i) = 4\mathrm{PG}$. □

ALGORITHM 3: POSTPONED GREEDY

```
• Input: an online graph with deadline $d$, $G_{d,\sigma}$.
• Let $S$ and $B$ respectively denote the set of seller and buyer copies of vertices that are
  still in the market. Set $S = \emptyset$ and $B = \emptyset$.
• For each time period $t = 1, \dots, n$, process events in the following way:
    1. Vertex $j$ arrives:
        (a) Set the status of $j$ to be *undetermined*.
        (b) (*Add a seller copy.*) $S \leftarrow S \cup \{s_j\}$, $m(s_j) = null$ and $p(s_j) = 0$.
        (c) (*Add a buyer copy.*) $B \leftarrow B \cup \{b_j\}$.
        (d) (*Add edges between existing seller copies and arriving buyer copy.*)
            $v_{s_r b_j} = v_{rj}\ \forall s_r \in S$.
        (e) (*Find a seller copy with highest marginal utility for the arriving buyer copy.*)
            $s_k = \text{argmax}_{s_r \in S} v_{s_r b_j} - p(s_r)$.
        (f) (*Tentatively match if marginal utility is positive.*) If $v_{s_k b_j} - p(s_k) > 0$, set
            $m(s_k) = b_j$ and $p(s_k) = v_{s_k b_j}$.
    2. Vertex $i$ becomes critical:
        (a) If status of vertex $i$ is *undetermined*, set it to be either *seller* or *buyer* with
            probability 1/2 each.
        (b) If $m(s_i) \neq null$:
            i. Let $m(s_i) = b_l$.
            ii. If $i$ is a seller, finalize the matching of vertex $i$ with vertex $l$. Set the
                status of vertex $l$ to be a buyer.
            iii. If $i$ is a buyer: Set the status of vertex $l$ to be a seller.
        (c) (*Remove the seller and buyer copies.*) Set $S \leftarrow S \setminus \{s_i\}$ and $B \leftarrow B \setminus \{b_i\}$.
```

3.3 Hardness Results

This section establishes hardness results for the adversarial order setting.

Claim 1. *When the input is a constrained bipartite graph:*

1. *No deterministic algorithm can obtain a competitive ratio above* $\frac{\sqrt{5}-1}{2} \approx 0.618$.
2. *No randomized algorithm can obtain a competitive ratio above* $\frac{4}{5}$.

Proof. For the first part, consider the example on the left of Figure 2. When seller 1 becomes critical, the algorithm either matches her with buyer 3, or lets 1 depart unmatched. The adversary then chooses x accordingly. Thus the competitive ratio cannot exceed:

$$\max\left(\min_{x\in\{0,1\}}\frac{\frac{\sqrt{5}-1}{2}+x}{\rho(x)},\ \min_{x\in\{0,1\}}\frac{1}{\rho(x)}\right)=\frac{\sqrt{5}-1}{2},$$

where $\rho(x)=\max(\frac{\sqrt{5}-1}{2}+x,1)$.

For the second part consider the example on the right of Figure 2. Similar to the first part, when seller 1 becomes critical, the algorithm decides to match her with 3 with probability p. The adversary then chooses x accordingly. Thus the competitive ratio cannot exceed:

$$\max_{p\in[0,1]}\min_{x\in\{0,1\}}\frac{p(1/2+x)+(1-p)}{\max(1/2+x,1)}=4/5.$$

□

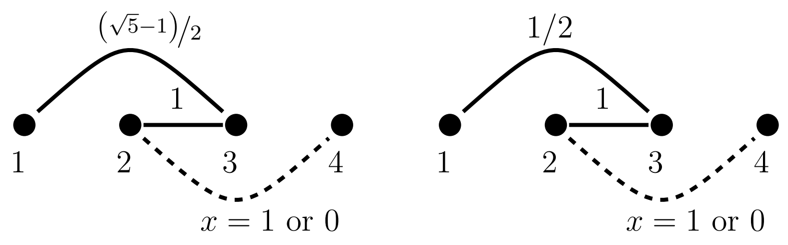


Figure 2: Constrained bipartite graph with deadline d where $S=\{1,2\}$ and $B=\{3,4\}$, with $d=2$. Vertex 1 becomes critical before 4 arrives. The adversary is allowed to choose the weight of edge $(2,4)$ to be either 1 or 0. Left: Instance for the deterministic case. Right: Instance for the randomized case.

The next result shows that the analysis for POSTPONED GREEDY is tight.

Claim 2. *There exists a constrained bipartite graph for which POSTPONED GREEDY is* $^1/_{(4-\epsilon)}$ *-competitive.*

Proof. Consider the input graph in Figure 3. Seller 2 gets temporarily matched with buyer 3, and seller 1 departs unmatched. When seller 2 becomes critical, with probability $^1/_2$, she is determined to be a *buyer* and departs unmatched. Therefore, PG collects $^1/_2$ in expectation while the offline algorithm collects $2-{}^\epsilon/_2$. □

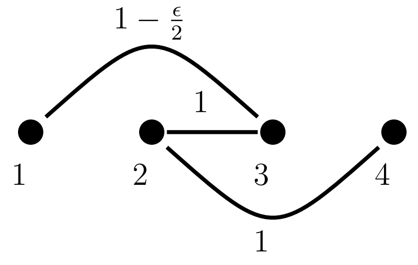


Figure 3: Constrained bipartite graph with deadline $d = 2$ where $S = \{1, 2\}$ and $B = \{3, 4\}$.

3.4 Time-Decreasing Edge Weights

In several real-life matching markets, an online decision maker can make better decisions if she waits longer for better matches to arrive. In doing so, the decision maker faces two issues. One is the possibility of agents departing from the market unmatched which is considered in this paper. Another is the fact that the perceived value of a match decreases over time even when the agents are still in the market and can be matched. For example, ride-sharing platforms face a trade-off between efficiency gains from better matches by delaying the matching decision and deterioration of user experience with increased wait-times. Our model ignores this fact and assumes edge-weights stay the same as long as agents are still in the market. In this section, we show that the POSTPONED GREEDY algorithm loses its guarantee if edges weights decrease over time.

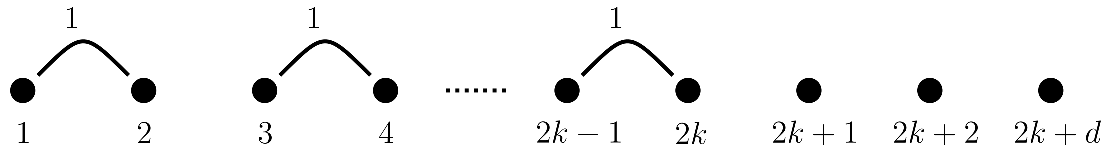


Figure 4: $n = 2k+d$ vertices arrive sequentially. Vertex $2j$ reveals an edge with vertex $2j-1$ on arrival, for $j = 1, 2, \ldots, k$. Other vertices reveal no edges on arrival. The weight of each revealed edge is 1 initially and decreases by a discount factor $0 < \alpha < 1$ at each time step.

Consider the example in Figure 4. Vertices arrive online in $n = 2k + d$ steps with vertex i arriving in time step i. For $j = 1, \ldots, k$, vertex $2j$ reveals an edge with vertex $2j - 1$ on arrival. Vertices $2j - 1$ for $j = 1, \ldots k$, and vertices $2k + i$ for $i = 1, \ldots, d$ reveal no edges to previously arrived vertices on arrival. The weight of an edge is 1 when first revealed and decreases by a discount factor $0 < \alpha < 1$ at each time step. The optimal online algorithm matches vertex $2j - 1$ with vertex $2j$ at time instant $2j$, for $j = 1, \ldots, k$, and collects a total match weight of k. On the other hand, POSTPONED GREEDY matches vertex $2j-1$ with vertex $2j$ with probability $1/2$ at time step $2j - 1 + d$, for $j = 1, \ldots, k$, and collects an expected match weight of $k\alpha^{d-1}/2$. The competitive ratio of $\alpha^{d-1}/2$ can be made smaller than any constant by increasing d, proving that POSTPONED GREEDY cannot guarantee a constant competitive ratio with adversarial arrival and time-decreasing edge weights.

4 Random Order of Arrivals

The assumption that vertices arrive in adversarial order and the edge weights are chosen adversarially is strong. This section considers the *Random Order (RO)* arrival model, in which the adversary chooses the weights, but the vertices arrive in a random order.

We study a simple and natural algorithm called *BATCHING* under the random order model and achieve a competitive ratio that improves upon the $1/4$ ratio achieved for the adversarial arrival model. In Section 4.2, we show that no algorithm can achieve a competitive ratio better than $1/2$ under the random order model.

4.1 Batching

The *BATCHING* algorithm computes and selects maximum-weight matchings periodically, namely, every $d+1$ time steps. Every vertex in the selected matching is then matched, and all other vertices are discarded.

Batching is used often in practice. In the context of kidney exchange, for example, most programs operate on a fixed schedule (every few weeks or months). In ride-hailing and ride-sharing applications, platforms often match passengers with rides in batches.

We conjecture that the competitive ratio of BATCHING is indeed $\frac{1}{2}$. In what follows, we prove that it is lower bound bounded by 0.279 for all n and d.

Theorem 2. *The competitive ratio of BATCHING is at least* $(0.279 + O(1/n))$.

The proof of Theorem 2 requires multiple steps. First, we reduce the analysis of the competitive ratio of BATCHING to a graph covering problem. More precisely, we show that it is enough to show that C_n^d, the cycle with n vertices to the power d, can be covered by a small number of cliques. Recall that raising a graph to the power d results in a graph with the same set of vertices and paths of at most d hops in the original graph as edges. Second, we show how a cover for small n can be extended to larger values of n at the cost of a small rounding error. Finally, we establish a reduction that allows to consider only a finite set of values for d. We conclude with a computer-aided argument for graphs in the finite family. The reminder of this subsection provides the proof.

4.1.1 Reduction to a Graph Theoretic Problem

The first step is to reduce the analysis of the performance of BATCHING to the problem of fractionally covering graphs from the family C_n^d, using ensembles of $\frac{n}{d+1}$ cliques of size $d+1$.

Given that the weights are arbitrary, we can assume that the underlying graph G is complete. For any d and any arrival sequence $\sigma \in S_n$, define the *path* graph $P_n^d(\sigma)$ with edge-weight $v_{ij} = 1$ if $|\sigma(i) - \sigma(j)| \le d$, and $v_{ij} = 0$ otherwise. In other words, $P_n^d(\sigma)$ is essentially the path $(\sigma(1), \sigma(2)), (\sigma(2), \sigma(3)), \cdots, (\sigma(n-1), \sigma(n))$ taken to the power d.

Every batch in the algorithm has $d+1$ vertices except the last batch which may have fewer vertices. Let $b_i(\sigma, d)$ be the batch of vertex i under permutation σ and batch size $d+1$: $b_i(\sigma, d)$ is the unique integer such that $(d+1)(b_i - 1) < \sigma(i) \le (d+1)b_i$. We define the *batched* graph $B_n^d(\sigma)$ with edge-weight $v_{ij} = 1$ if i and j are in the same batch (i.e. $b_i(\sigma, d) = b_j(\sigma, d)$), and $v_{ij} = 0$ otherwise. Observe that $B_n^d(\sigma)$ is a collection of disjoint $(d+1)$-cliques. The following two definitions are crucial.

Definition 1 (Graph operations)**.** *For any two graphs H and H' with vertices $1, 2, \ldots, n$ and respective edge weights v_{ij}, v'_{ij}, we define the following:*

(i) The linear combination $aH + bH'$ denotes the graph with edge weights $av_{ij} + bv'_{ij}$.

(ii) The product $H * H'$ denotes the graph with edge weights $v_{ij} * v'_{ij}$.

(iii) We say that H is a cover *of H' if for all i, j, $v_{i,j} \geq v'_{ij}$.*

Definition 2 ((α, d)-cover)**.** *Let F be an unweighted graph with n vertices. We say that a set of permutations $\{\sigma_1, \ldots, \sigma_K\} \in S_n$ forms an (α, d)-cover of F if there exist values $\lambda_1, \ldots, \lambda_K \in [0, 1]$ such that*

(i) $\sum_{k \leq K} \lambda_k B_n^d(\sigma_k)$ is a cover of F.

(ii) $\sum_{k \leq K} \lambda_k = \alpha$.

The next proposition will allow us to rephrase the competitive ratio of BATCHING in terms of a graph covering problem. Note that the reduction abstracts away from the weights that are chosen by the adversary.

Proposition 2. *If there exists an (α, d)-cover of C_n^d, then BATCHING is $1/\alpha$-competitive.*

Proof. For any graph H, let $m(H)$ denote the value of a maximum-weight matching over H. Observe that when the arrival sequence is σ, the graph $G(d, \sigma) = P_n^d(\sigma) * G$, and therefore the offline algorithm collects weight equal to $m(P_n^d(\sigma) * G)$. Note that BATCHING collects $m(B_n^d(\sigma) * G)$.

Remark 1. *Observe that for any graphs H, H', G and any $a, b \in \mathbb{R}$, we have:*

- *$m(aH + bH') \leq am(H) + bm(H')$.*
- *If H is a cover of H', then, $m(H' * G) \leq m(H * G)$.*

Let id be the identity permutation. Let $\{\sigma_1, \ldots, \sigma_K\}$ be an (α, d)-cover of C_n^d. Fix an arrival sequence $\sigma \in S_n$. We first claim that $\{\sigma_1 \circ \sigma, \ldots, \sigma_K \circ \sigma\}$ is an (α, d)-cover of $P_n^d(\sigma)$.

For any $\sigma \in S_n$, let us denote $\beta_{i,j}(\sigma)$ and $\rho_{i,j}(\sigma)$ to be the weights of edge (i, j) in $B_n^d(\sigma)$ and $P_n^d(\sigma)$ respectively. Consider $(i, j) \in P_n^d(\sigma)$: $|\sigma(i) - \sigma(j)| \leq d$:

$$\begin{aligned}
\sum_k \lambda_k \beta_{i,j}(\sigma_k \circ \sigma) &= \sum_k \lambda_k \mathbb{I}[b_i(\sigma_k \circ \sigma, d) = b_j(\sigma_k \circ \sigma, d)] \\
&= \sum_k \lambda_k \mathbb{I}[b_{\sigma(i)}(\sigma_k, d) = b_{\sigma(j)}(\sigma_k, d)] \\
&\geq \rho(\mathrm{id})_{\sigma(i), \sigma(j)} = 1,
\end{aligned}$$

where the last inequality is implied by the fact that $\{\sigma_1, \dots, \sigma_K\}$ is an (α, d)-cover of C_n^d and therefore of $P_n^d(\mathrm{id})$. Therefore the claim holds using Remark 1.

Denote by BAT the value collected by the BATCHING algorithm and OFF the value collected by the offline algorithm. Observe that

$$\begin{aligned}\mathrm{OFF} &= \frac{1}{n!}\sum_{\sigma\in S_n} m(P_n^d(\sigma) * G)\\ &\leq \frac{1}{n!}\sum_{\sigma\in S_n}\sum_{k}\lambda_k m(B_n^d(\sigma_k\circ\sigma) * G)\\ &= \frac{1}{n!}\sum_{k}\lambda_k\sum_{\sigma'\in S_n} m(B_n^d(\sigma') * G)\\ &= \alpha\mathrm{BAT},\end{aligned}$$

where we used the change of variable $\sigma' = \sigma_k \circ \sigma$ and the fact that the application $\mathcal{A}_k : \sigma \mapsto \sigma_k \circ \sigma$ is a bijection. □

We have reduced the analysis of Batching to a graph-theoretic problem without edge weights. In what follows, we will show that we can reduce the problem further to find covers of C_n^d for only small values of n and d.

4.1.2 Reducing n: Periodic Covers

We now wish to find (α, d)-covers for C_n^d for every n and d. In Proposition 3, we show that it is sufficient to find periodic covers for small values of n.

Definition 3 (Periodic permutation)**.** *For $p < n$ such that p divides n, we say that a permutation $\sigma \in S_n$ is p-periodic if for all $i \in [1, n-p]$, $\sigma(i+p) \equiv \sigma(i) + p \mod n$.*

A permutation σ is periodic if there exists p such that σ is p-periodic. Furthermore, an (α, d)-cover $\{\sigma_1, \dots, \sigma_K\}$ is p-periodic if for all $1 \leq k \leq K$, σ_k is p-periodic.

Proposition 3. *Let p be a multiple of $d+1$, and n_1 a multiple of p. Any p-periodic (α, d)-cover of $C_{n_1}^d$ can be extended into an $(\alpha + O(p/n), d)$-cover of C_n^d for any $n \geq n_1$.*

Proof. Let $\{\sigma_1, ..., \sigma_K\}$ be a p-periodic (α, d)-cover of $C_{n_1}^d$. We will show that it can be extended into an (α, d)-cover of C_n^d.

Assume first now that n is a multiple of p. Let σ'_k be the p-periodic permutation over $1, ..., n$ such that for all $i \in [1, p]$, $\sigma'_k(i) = \sigma_k(i)$. Take $i', j' \in [1, n]$ such that $|i' - j'| \leq d$. Because $n_1 > p$ is a multiple of p, there exist $i, j \in [1, n_1]$ such that $i \equiv i' \mod p$, $j \equiv j' \mod p$ and $|i - j| \leq d$.

For any graph H, let H_{ij} denote the weight v_{ij} in H. By p-periodicity of σ_k and σ'_k, we know that $B_n^d(\sigma'_k)_{i',j'} = B_{n_1}^d(\sigma_k)_{i,j}$. Thus we can conclude that $\{\sigma'_1, ..., \sigma'_K\}$ is an (α, d)-cover of C_n^d.

Consider next the general case. When n is not a multiple of p, let $v \in [1, p-1]$ be the remainder of the euclidian division of n by p, and u be such that $n = pu + v$. Let $\{\sigma_1, ..., \sigma_K\}$

be a p-periodic (α, d)-cover of $C^d_{n_1}$ with associated weights $\{\lambda_1, ..., \lambda_K\}$. We will show that it can be extended into an $(\alpha\left(u/u-2\right), d)$-cover of C^d_n.

We set $\widetilde{\sigma}_k$ to be the p-periodic permutation over $1, ..., pu$ such that for all $i \in [1, p]$, $\widetilde{\sigma}_k(i) = \sigma_k(i)$. Let x be an integer in the interval $[1, u+1]$. Define the permutation $\sigma'_{k,x}$ as follows:

$$\sigma'_{k,x}(i) = \begin{cases} \widetilde{\sigma}_k(i) & i \le px \\ i + (u-x)p & i \in [px+1, px+1+v] \\ \widetilde{\sigma}_k(i-v) & i > px+1+v. \end{cases} \tag{1}$$

Take $i', j' \in [1, px] \cup [px+1+v, n]$ such that $|i'-j'| \le d$. Because $n_1 > p$ is a multiple of p, there exist $i, j \in [1, n_1]$ such that $i \equiv i' \mod p$, $j \equiv j' \mod p$ and $|i-j| \le d$. By p-periodicity of σ_k and σ'_k, we know that edge (i', j') is in $B^d_n(\sigma'_k)$ iff (i, j) is in $B^d_{n_1}(\sigma_k)$. Thus we can conclude that $\sum_k \lambda_k B^d_n(\sigma'_{k,x})$ covers edge (i', j') of C^d_n.

Every edge is therefore covered for at least $u-2$ different values of x. Therefore, $\sum_k \sum_x \frac{u}{u-2} \lambda_k B^d_n(\sigma'_{k,x})$ covers C^d_n. This means that $(\sigma_{k,x})_{k,x}$ is an $(\alpha\left(u/u-2\right), d)$-cover of C^d_n. □

4.1.3 Reducing d: cycle contraction

In Proposition 3, we show that it is enough to find periodic (α, d)-covers of C^d_n for small values of n. Next, we provide a reduction that enables us to consider only a finite set of values for d. The key idea of the reduction is to contract vertices of C^d_n into n/u groups of u vertices. The resulting graph also happens to be a cycle $C^{(d+1)/u}_{n/u}$. In Proposition 4, we provide a way to expand an $(\alpha, u-1)$-cover on the contracted graph into an (α, d) cover on the original graph.

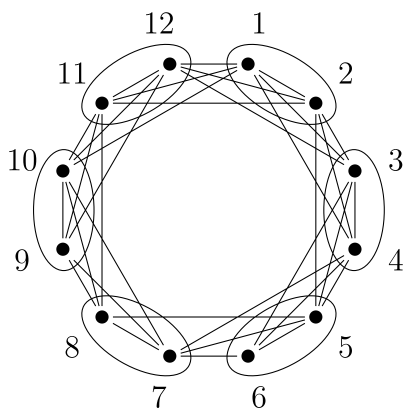
Definition 4 (Cycle contraction). *For any n, d and an integer u which divides n, we define the u-contraction $f_u(C^d_n)$ to be the graph with vertices $a_k = \{uk+1, ..., u(k+1)\}$ for $k \in [0, n/u - 1]$, and edges (a_k, a_l) if and only if there exist $i \in a_k$ and $j \in a_l$ with an edge (i, j) in C^d_n.*

Claim 3. *For any d, if $u > 1$ divides $d+1$ and $d+1$ divides n, then $f_u(C^d_n) = C^{(d+1)/u}_{n/u}$.*

Proof. We first prove that $C^{(d+1)/u}_{n/u}$ covers $f_k(C^d_n)$. Fix $k, l \in [0, n/u - 1]$, and assume that $k < l$. If $|l-k| \le (d+1)/u$, then let $i = u(k+1)$ and $j = ul+1$. We have $|j-i| = u(l-k-1)+1 \le d$, thus $(i, j) \in C^d_n$ and $(k, l) \in f_u(C^d_n)$.

Conversely, we now prove that $f_u(C^d_n)$ covers $C^{(d+1)/u}_{n/u}$. If there exist $i \in a_k$ and $j \in a_l$ such that $|j-i| \le d$, then $u(l-k) \le ul + 1 - u(k+1) \le d+1$ which implies that $(k, l) \in C^{(d+1)/u}_{n/u}$. □

Proposition 4. *Fix $d \ge 1$. For $d+1 > k \ge 1$, suppose that there is a periodic $(\alpha, k-1)$-cover of C^k_{rk}.*



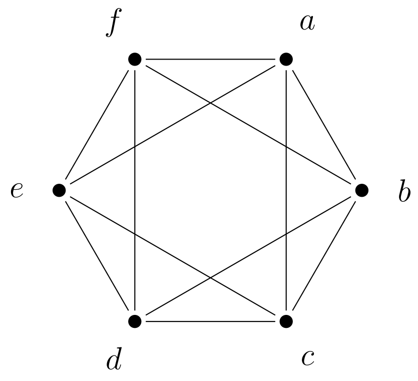


Figure 5: Left: C^3_{12}, with contraction for $u = 2$. Right: 2-Contracted graph $f_2(C^3_{12}) = C^2_6$ with vertices $a = \{1, 2\}$, $b = \{3, 4\}$, ... $f = \{11, 12\}$.

(i) For any integer r, if k divides $d+1$ then there exists a periodic (α, d)-cover of $C^d_{r(d+1)}$.

(ii) In general, if v is the remainder of the euclidian division of $d+1$ by k, then there exists a periodic $(\alpha(1 + {}^{v}/_{d+1-v})^2, d)$-cover of $C^d_{r(d+1)}$.

Proof of (i). Suppose that $d + 1 = ku$ and suppose that there exists p multiple of $d + 1$ such that we have a p-periodic $(\alpha, k - 1)$-cover $\{\sigma_1, ..., \sigma_K\}$ of $f_u\left(C^d_{r(d+1)}\right) = C^k_{rk}$. For any permutation $\sigma \in S_{rk}$ we can construct a permutation $\sigma' \in S_{r(d+1)}$ in the following way: if $i \in a_t$ then $\sigma'(i) = \frac{n}{k}\sigma(t) + i$. Because $B^k_{rk}(\sigma_i)$ is a cover of $B^d_{r(d+1)}$, we can conclude that $\sigma'_1, ..., \sigma'_K$ is an (α, d)-cover of $C^d_{r(d+1)}$.

$\square$

Proof of (ii). Suppose now that $d + 1 = ku + v$ with $1 \leq v < k$. We first select vertices in the following way: select a subset $\Phi \subset [1, d + 1]$ of $d + 1 - v$ vertices uniformly at random. Take $\Delta = \Phi + ku[1, r - 1] = \{a + kub | a \in \Phi, b \in [1, r - 1]\}$ and note that $|\Delta| = kur$.

We now contract vertices in Δ. This is the same as in Definition 4: for $t \in [1, u]$, a_t is the set of u smallest vertices of Δ that are not in $a_1 \cup ... \cup a_{t-1}$. Because $d + 1 - v$ is a multiple of u, we have $a_{i+k} = a_i + (d + 1)$. This implies that the contracted graph is $C^{(d+1)/u}_{n/u}$.

Similarly to the proof of case (i), we extend a cover for $C^{(d+1)/u}_{n/u}$ to cover every edge (i, j) for $i, j \in \Delta$. If we sum over all the possible ways to select subset Φ, we note that every edge $(i, j) \in C^d_{r(d+1)}$ is covered with probability at least $\left(\frac{d+1-v}{d+1}\right)^2$. $\square$

4.1.4 Final step: Computer-aided proof of factor 2.79

We will now apply Proposition 3 with $p = 2(d + 1)$ and $n_1 = 4(d + 1)$. Let Ω_d be the set of $2(d + 1)$-periodic permutations of $1, ..., 4(d + 1)$. We can find covers for $C^d_{4(d+1)}$ using the following linear program:

$$
\begin{aligned}
\min \quad & \sum_{\sigma \in \Omega_d} \lambda_\sigma \\
\text{s.t.} \quad & \sum_{\sigma \in \Omega_d} \lambda_\sigma \mathbb{I}[b_i(\sigma, d) = b_j(\sigma, d)] \geq 1, \quad \forall (i,j) \in C^d_{4(d+1)} \\
& \lambda_\sigma \in \mathbb{R}^+, \qquad \sigma \in \Omega_d
\end{aligned} \tag{LP$_d$}
$$

Proposition 5. *Let α_d be the solution to LP$_d$. Let $\alpha = \sup_{d \geq 1} \alpha_d$. Batching is $(1/\alpha + O(1/n))$-competitive.*

Proof. Follows from Propositions 2 and 3. □

The Linear program (LP$_d$) has $O(d!)$ variables, and solving it may not be computationally possible when d is large. Using Proposition 4, we now provide a way to find upper bounds on α_d by solving a different LP on a smaller graph. Recall that Ω_{k-1} is the set of $2k$-periodic permutations of $1, ..., 4k$. We define the problem of finding an $(\alpha, k-1)$-cover of the cycle C^k_{4k}.

$$
\begin{aligned}
\min \quad & \sum_{\sigma \in \Omega_{k-1}} \lambda_\sigma \\
\text{s.t.} \quad & \sum_{\sigma \in \Omega_{k-1}} \lambda_\sigma \mathbb{I}[b_i(\sigma, k-1) = b_j(\sigma, k-1)] \geq 1, \quad \forall (i,j) \in C^k_{4k} \\
& \lambda_\sigma \in \mathbb{R}^+, \qquad \sigma \in \Omega_{k-1}
\end{aligned} \tag{LP'$_k$}
$$

We denote by α'_k the solution to (LP'$_k$). Solving (LP'$_k$) numerically for $k = 4$ yields $\alpha'_4 \leq 3.17$ (See Table 1). For all $d \geq 52$ Proposition 4 therefore implies that, $\alpha_d \leq 3.17 * \left(\frac{51}{49}\right)^2 = 3.58$. [2]

It remains to check that for all $d \leq 50$, $\alpha_d \leq 3.58$. We either solve (LP$_d$) directly (see left Table 1) or use Proposition 4 (see right Table 1) to conclude. Observing that $2.79 \leq \frac{1}{3.58}$, this allows us to conclude that Batching is 0.279-competitive, and concludes the proof for Theorem 2. □

4.2 A Hardness Result

Proposition 6. *No algorithm is more than $\frac{1}{2}$-competitive even under the random arrival model.*

Proof. For the sake of contradiction, assume there is a $(1/2 + \epsilon)$-competitive algorithm $\mathcal{A}$ under the random arrival model for some $0 < \epsilon < 1/2$. Consider a graph with three vertices $\{1, 2, 3\}$, with $v_{12} = \alpha$ and $v_{23} = v_{13} = \epsilon'\alpha$, where $0 < \alpha$ and $0 < \epsilon' < 1$ are some constants. Let $d = 1$, i.e., vertices can only be matched to the ones arriving just before or after them. Let E denote the event that the arrival order is such that vertices 1 and 2 can be matched together. We have $\Pr[E] = 2/3$. Since OFF can collect α by matching vertices 1 and 2 when

[2] We note that our methodology can be extended to obtain a better factor. For instance, being able to solve (LP$_d$) for values higher than 50 should bring the competitive ratio closer to $\frac{1}{3}$.

d	α_d	α'_d
1	2	
2	2.33	4
3	2.5	3.45
4	2.64	3.17
5	2.71	3.15
6	2.75	3.12
7	2.79	3.09
8	2.83	3.08
9	2.99*	3.07
10	3.2*	3.20*
11	3.11*	3.153*
12		3.264*
13	3.23*	3.318*

d	$\alpha_d \leq$	k used
17	3.58	4
19	3.48	6
23	3.44	11
29	3.31	7
31	3.36	5
37	3.30	6
41	3.31	5
43	3.24	7
47	3.35	9

Table 1: Left: Numerical values for α_d and α'_d for small values of d. Starred elements were solved approximately (are therefore upper bounds on the actual value). Right: Upper bounds for α_d for prime values of d; derived from Proposition 4 using the following formula: $\alpha_d \leq \alpha_k \left(\frac{d+1-v}{d+1}\right)^2$ for $k \leq d$ and $v = d \bmod k$.

event E occurs, it collects an expected match weight of at least $\frac{2\alpha}{3}$. Given $\mathcal{A}$ is $(1/2 + \epsilon)$-competitive, it must collect an expected match weight of at least $\frac{\alpha}{3} + \frac{2\alpha\epsilon}{3}$. Since any algorithm can collect at most $\alpha\epsilon'$ when event E does not occur, we infer that, conditioned on E, the conditional expected match weight collected by $\mathcal{A}$ is at least $(\frac{\alpha}{3} + \frac{2\alpha\epsilon}{3} - \frac{\alpha\epsilon'}{3})/\frac{2}{3} = \frac{\alpha}{2} + \alpha\epsilon - \frac{\alpha\epsilon'}{2}$.

On the other hand, it is easy to see that conditioned on E, $\mathcal{A}$ essentially solves a secretary problem with two arrivals (corresponding to the two edge arrivals in the second and third time steps). Since no randomized algorithm can pick the larger weight edge with probability larger than $1/2$ in this case, the conditional expected match weight collected by $\mathcal{A}$, conditioned on E, is at most $\frac{\alpha}{2} + \alpha\epsilon'$.

We obtain $\frac{\alpha}{2} + \alpha\epsilon - \frac{\alpha\epsilon'}{2} < \frac{\alpha}{2} + \alpha\epsilon'$. Choosing $\epsilon' < \frac{2\epsilon}{3}$ gives us the desired contradiction. □

5 Extensions

5.1 Stochastic departures in the Adversarial Order

Note that the analysis of POSTPONED GREEDY did not assume that vertices depart exactly after d time steps since arrival. In particular, we can obtain a $1/4$-competitive algorithm as long as departures are in order of arrivals and we get to know when a vertex becomes critical. Here, we reconsider the adversarial order setting but relax the assumption that vertices depart in order of arrivals. In particular we assume that the departure time d_i of vertex i is sampled independently from a distribution $\mathcal{D}$. Moreover, for every vertex i, the realization d_i is observed at the time i becomes critical.

We can run the POSTPONED GREEDY algorithm even in this setting of stochastic departures. We need a natural modification to the algorithm. When vertex i becomes critical with $m(s_i) = b_l$, we finalize the matching of vertex i with vertex l if the status of vertex i is a seller *and* vertex l has not already departed.

Proposition 7. *Suppose that there exists $\alpha \in (0, 1)$ such that $\mathcal{D}$ satisfies the property that for all $i < j$,*

$$\mathbb{P}[i + d_i \leq j + d_j | i + d_i \geq j] \geq \alpha.$$

Then modified POSTPONED GREEDY is $^{\alpha}/_{4}$-competitive.

Proof. Recall that in the case of departures in order of arrivals, POSTPONED GREEDY ensures statuses of vertices are synchronized. In the case of stochastic departures however, when a vertex i becomes critical with $m(s_i) = b_l$, the statuses of vertices i and l can both turn out to be sellers. But if that happens, it implies that vertex l has departed when vertex i becomes critical and the modified algorithm will not match the two vertices together. Thus the algorithm always outputs a valid matching.

Note that a vertex assumes the status of a seller or buyer with probability $^1/_2$ each, independent of the realization of the departure times. Moreover, with probability at least α, vertex l is still present when vertex i becomes critical. Thus, when vertex i becomes critical, we collect $p^f(s_i)$ with probability at least $^{\alpha}/_{2}$. The rest of the proof follows similarly to that of Theorem 1. $\square$

$\mathcal{D}$ is called memory-less process, conditional on i still being present when j arrives, the probability that i departs first is exactly 1/2. Therefore, the above condition is satisfied with $\alpha = 1/2$.

Corollary 2. *POSTPONED GREEDY is $^1/_8$-competitive when $\mathcal{D}$ is memory-less.*

5.2 Look-ahead Under Random Order

Here, We assume that the online algorithm knows the vertices that will arrive in the next l time steps along with their adjacent edges. We can update the BATCHING Algorithm in the following way: every $d+l+1$ time steps, compute a maximum-weight matching on both the current vertices and the next l arrivals. Match vertices as they become critical according to the matching, and discard unmatched vertices. Note that this is the same as running Batching when the deadline is $d + l$.

Proposition 8. *There exists an $(\frac{d+l+1}{l+1}, d + l)$-cover of C_n^d.*

Proof. For $k \in [0, d + l]$, let $\sigma_k(i) = i + k \mod n$. Let i, j be such that $|i - j| \leq d$, then $b_i(\sigma_k, d) = b_j(\sigma_k, d)$ for at least $l + 1$ different values of k. We can conclude that $\sigma_0, ..., \sigma_{d+l}$ is a $(\frac{d+l+1}{l+1}, d + l)$-cover by taking $\lambda_0 = ... = \lambda_{d+l} = \frac{1}{l+1}$. $\square$

Corollary 3. *BATCHING with l-lookahead is $\frac{l+1}{d+l+1}$-competitive when n is large.*

5.3 Adversarial Order: Alternative to Greedy

Consider the adversarial order of arrival settings. Observe that under the greedy algorithm, once a buyer becomes unmatched it never matches again. In this section we present another algorithm, referred to as Dynamic Deferred Acceptance (DDA), that does not have this property.

The DDA will receive as input a constrained bipartite graph. The main idea is to maintain a tentative maximum-weight matching m at all times during the run of the algorithm. This tentative matching is updated according to an auction mechanism: every seller s is associated with a *price* p_s, which is initiated at zero upon arrival. Every buyer b that that already arrived and yet to become critical is associated with a *profit margin* q_b which corresponds to the value of matching to their most preferred seller minus the price associated with that seller. Every time a new buyer arrives, she bids on her most preferred seller at the current set of prices. This triggers a bidding process that terminates when no unmatched buyer can profitably bid on a seller.

When a seller becomes critical, she is irrevocably matched to her tentative match. A buyer is discarded only if she is unmatched and becomes critical.

At any point t throughout the algorithm, we maintain a set of sellers S_t, a set of buyers B_t, as well as a matching m, a price p_s for every seller $s \in S_t$, and a marginal profit q_b for every buyer $b \in B_t$.

ALGORITHM 4: Dynamic Deferred Acceptance

```
• Input: an online graph with deadline $d$, $G_{d,\sigma}$.
• Process each event in the following way:
    1. *Arrival of a seller s:* Initialize $p_s \leftarrow 0$ and $m(s) \leftarrow \emptyset$.
    2. *Arrival of a buyer b:* Start the following *ascending auction.*
       Repeat
       (a) Let $q_b \leftarrow \max_{s' \in S_t} v_{s',b} - p_{s'}$ and $s \leftarrow \text{argmax}_{s' \in S_t} v_{s',b} - p_{s'}$.
       (b) If $q_b > 0$ then
           i. $p_s \leftarrow p_s + \epsilon$.
           ii. $m(s) \leftarrow b$ (tentatively match $s$ to $b$)
           iii. Set $b$ to $\emptyset$ if $s$ was not matched before. Otherwise, let $b$ be the previous
                match of $s$.
       Until $q_b \leq 0$ or $b = \emptyset$.
    3. *Departure of a seller s:* If seller $s$ becomes critical and $m(s) \neq \emptyset$, finalize the
       matching of $s$ and $m(s)$ and collect the reward of $v_{s,m(s)}$.
```

Our algorithm bears similarities to the auction algorithm by Bertsekas (1988). Prices in this auction increase by ϵ to ensure termination, and optimality is proven through ϵ-complementary slackness conditions. For the analysis, we consider the limit $\epsilon \to 0$ and assume the auction phase terminates with the maximum weight matching. [3]

The auction phase is always initiated at the existing prices and profit margins. This, together with the fact that the graph is bipartite, ensures that prices never decrease and and marginal profits never increase throughout the algorithm. Furthermore, the prices and marginal profits of the sellers and buyers that are present in the "market" form an optimum dual for the matching linear program.

Lemma 1. *Consider the DDA algorithm on a constrained bipartite graph.*

1. *Sellers' prices never decrease, and buyers' profit margins never increase.*
2. *At the end of every ascending auction, prices of the sellers and the marginal profits of the buyers form an optimal solution to the dual of the matching linear program associated with buyers and sellers present at that particular time.*

Maintaining a maximum-weight matching along with optimum dual variables does not guarantee an efficient matching for the whole graph. The dual values are not always feasible for the offline problem. Indeed, the profit margin of some buyer b may decrease after some seller departs the market. This is because b may face increasing competition from new buyers, while the bidding process excludes sellers that have already departed the market (whether matched or not).

Proposition 9. *DDA is $1/2$-competitive for constrained bipartite graphs.*

Proof. The proof follows the primal-dual framework. First, observe that by complementary slackness, any seller s (buyer b) that departs unmatched has a final price $p_s^f = 0$ (final profit margin $q_b^f = 0$). When a seller s is critical and matches to b, we have $v_{s,b} = p_s^f + q_b^f$. Therefore, *DDA* collects a reward of $\mathcal{A} = \sum_{s \in S} p_s^f + \sum_{b \in B} q_b^f$.

Second, let us consider a buyer b and a seller $s \in [b-d, b)$ who has arrived before b but not more than d steps before. Because sellers do not finalize their matching before they are critical, we know that $s \in S_b$. An ascending auction may be triggered at the time of b's arrival, after which we have: $v_{s,b} \leq p_s(b) + q_b(b) \leq p_s^f + q_b^i$, where the second inequality follows from the definition that $q_b(b) = q_b^i$ and from the monotonicity of sellers' prices (Lemma 1). Thus, (p^f, q^i) is a feasible solution to the offline dual problem.

Finally, we observe that upon the arrival of a new buyer, the ascending auction does not change the sum of prices and margins for vertices who were already present:

Claim 4. *Consider an arriving buyer b, and let p, q (p', q') be the prices and margins before at the beginning (the end) the ascending auction phase (Step 2(a) in Algorithm 1). Then:*

$$\sum_{s \in S_t} p_s + \sum_{b \in B_t} q_b = \sum_{s \in S_t} p'_s + \sum_{b \in B_t} q'_b. \tag{2}$$

[3] One way to formalize this argument is through the Hungarian algorithm Kuhn (1955), where prices are increased simultaneously along an alternating path that only uses edges for which the dual constraint is tight.

Where S_t and B_t are the set of sellers and buyers present before b arrived.

Proof of Claim 4. Because auctions are always started by *buyers*, no previously matched seller will become unmatched. Thus, except for the buyer which started the auction, every increase in a seller's price is exactly matched by a buyer's decrease in margin. $\square$

By applying this equality iteratively after each arrival, we can relate the initial margins q^i to the final margins q^f and prices p^f:

Claim 5. $\sum_{s\in S} p_s^f + \sum_{b\in B} q_b^f = \sum_{b\in B} q_b^i$.

This completes the proof of Proposition 1 given that the offline algorithm achieves at most:

$$\mathcal{O} \le \sum_{s\in S} p_s^f + \sum_{b\in B} q_b^i \le 2\mathcal{A}.$$

$\square$

Proof of Claim 5. We iteratively apply the result of Claim 4 after any new arrival. Let $\widetilde{S}_t$ (resp. $\widetilde{B}_t$) be the set of sellers (buyers) who have departed, or already been matched before time t. We show by induction over $t \le T$ that:

$$\sum_{s\in \widetilde{S}_t} p_s^f + \sum_{b\in \widetilde{B}_t} q_b^f + \sum_{s\in S_t} p_s(t) + \sum_{b\in B_t} q_b(t) = \sum_{b\in \widetilde{B}_t} q_b^i + \sum_{b\in B_t} q_b^i. \tag{3}$$

This is obvious for $t = 1$. Suppose that it is true for $t \in [1, T-1]$. Note that departures do not affect (3). If the agent arrivint at $t+1$ is a seller, then for all other sellers s, $p_s(t+1) = p_s(t)$ and for all buyers b, $q_b(t+1) = q_b(t)$, thus (3), is clearly still satisfied. Suppose that vertex $t+1$ is a buyer. Using equation (2), we have:

$$\sum_{s\in S_{t+1}} p_s(t+1) + \sum_{b\in B_{t+1}} q_b(t+1) \tag{4}$$

$$= q_{t+1}(t+1) + \sum_{b\in B_t} q_b(t) + \sum_{s\in S_t} p_s(t) \tag{5}$$

$$= \sum_{b\in B_{t+1}} q_b^i. \tag{6}$$

Note that at time $T+d$, every vertex has departed. Thus, $\widetilde{S}_{T+d} = S$, $\widetilde{B}_{T+d} = B$ and $S_{T+d} = B_{T+d} = \emptyset$. This enables us to conclude our induction and the proof for (3). $\square$

6 Conclusion

This paper introduces a model for dynamic matching, in which agents arrive in the market over time and depart after being in the market for some amount of time. Match values are heterogeneous and the underlying graph may be non-bipartite. Online algorithms are established for settings in which vertices arrive either in an adversarial or in random order. The model imposes restrictions on the departure process and allows the algorithm to know when vertices become critical and are about to leave.

In the adversarial arrival case, we introduce two new 1/4-competitive algorithms when departures are deterministic and agents depart in order of arrival. In the random arrival case, we show that a batching algorithm is 0.279-competitive. Closing the gaps between the upper bound of $1/2$ and the achievable competitive ratios remain interesting open problems.

We also point out a few other interesting research directions. One is to consider an alternative objective that in addition to achieve a high total value from matches also seeks to match a large number of agents. The algorithmic techniques developed in this paper might prove useful in devising online matching strategies with better guarantees in this setting.

Another direction is to consider a stochastic setting with prior information over weights and future arrivals. Such information is available in several real-life situations. For example, in ride-sharing applications, stochastic information of future arrivals is usually available from past arrival data or other extraneous sources such as weather and events data. Designing algorithms that can harness stochastic information of future arrivals has been an active and fruitful area of research in a variety of matching models. Our algorithms in this paper do not exploit any stochastic information of future arrivals, and designing algorithmic techniques that can do so in our matching model is a very important research direction.

Lastly, our model assumes that matches retain the same value regardless of when they are made as long as the agents are still in the market. We showed that the POSTPONED GREEDY algorithm does not guarantee any constant competitive ratio if matches lose their value over time. Thus, an interesting research direction is to devise algorithms accounting for agents' waiting times in the matching process. This would require devising techniques to balance the trade-off between waiting longer for better matches and making matches faster to retain their value.

References

Akbarpour, M., Li, S., and Oveis Gharan, S. (2020). Thickness and information in dynamic matching markets. *Journal of Political Economy*, 128(3):783–815.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., and Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc Natl Acad Sci USA*.

Anderson, R., Ashlagi, I., Gamarnik, D., and Kanoria, Y. (2015). A dynamic model of barter exchange. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1925–1933. Society for Industrial and Applied Mathematics.

Aouad, A. and Saritac, O. (2020). Dynamic stochastic matching under limited time. In *Proceedings of the twenty-first ACM Conference on Economics and Computation*, pages 789–790.

Ashlagi, I., Azar, Y., Charikar, M., Chiplunkar, A., Geri, O., Kaplan, H., Makhijani, R., Wang, Y., and Wattenhofer, R. (2017). Min-cost bipartite perfect matching with delays. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 81. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Ashlagi, I., Burq, M., Jaillet, P., and Manshadi, V. (2019). On matching and thickness in heterogeneous dynamic markets. *Operations Research*, 67(4):927–949.

Baccara, M., Lee, S., and Yariv, L. (2015). Optimal dynamic matching. Working paper.

Banerjee, S., Kanoria, Y., and Qian, P. (2018). State dependent control of closed queueing networks. In *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, pages 2–4. ACM.

Bertsekas, D. P. (1988). The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of operations research*, 14(1):105–123.

Blum, A. and Mansour, Y. (2020). Kidney exchange and endless paths: On the optimal use of an altruistic donor. *arXiv preprint arXiv:2010.01645*.

Chin, F. Y., Chrobak, M., Fung, S. P., Jawor, W., Sgall, J., and Tichỳ, T. (2006). Online competitive algorithms for maximizing weighted throughput of unit jobs. *Journal of Discrete Algorithms*, 4(2):255–276.

Devanur, N. R., Jain, K., and Kleinberg, R. D. (2013). Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, page 101107.

Dickerson, J. P., Procaccia, A. D., and Sandholm, T. (2013). Failure-aware kidney exchange. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 323–340. ACM.

Emek, Y., Kutten, S., and Wattenhofer, R. (2016). Online matching: haste makes waste! In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 333–344. ACM.

Ezra, T., Feldman, M., Gravin, N., and Tang, Z. G. (2020). Secretary matching with general arrivals. *CoRR abs/2011.01559.pdf*.

Feldman, J., Korula, N., Mirrokni, V., Muthukrishnan, S., and Pál, M. (2009a). Online ad assignment with free disposal. In *International Workshop on Internet and Network Economics*, pages 374–385. Springer.

Feldman, J., Mehta, A., Mirrokni, V. S., and Muthukrishnan, S. (2009b). Online stochastic matching: Beating 1-1/e. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 117–126.

Goel, G. and Mehta, A. (2008). Online budgeted matching in random input models with applications to adwords. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 982–991.

Hu, M. and Zhou, Y. (2020). Dynamic type matching. *Manufacturing & Service Operations Management*.

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

Huang, Z., Peng, B., Tang, Z. G., Tao, R., Wu, X., and Zhang, Y. (2019). Tight competitive ratios of classic matching algorithms in the fully online model. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2875–2886.

Jaillet, P. and Lu, X. (2013). Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646.

Kanoria, Y. and Qian, P. (2019). Near optimal control of a ride-hailing platform via mirror backpressure. *arXiv*, pages arXiv–1903.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. (1990). An optimal algorithm for online bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing (STOC)*, pages 352–358.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

Lehmann, B., Lehmann, D., and Nisan, N. (2006). Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296.

Li, F., Sethuraman, J., and Stein, C. (2005). An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 801–802. Society for Industrial and Applied Mathematics.

Manshadi, V. H., Oveis-Gharan, S., and Saberi, A. (2011). Online stochastic matching: online actions based on offline statistics. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1285–1294.

Mehta, A. (2013). Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368.

Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2007). Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22.

Ostrovsky, M. and Schwarz, M. (2018). Carpooling and the economics of self-driving cars. Technical report, National Bureau of Economic Research.

Ozkan, E. and Ward, A. R. (2020). Dynamic matching for real-time ridesharing. *Stochastic Systems*, 10(1):29–70.

Pavone, M., Smith, S. L., Frazzoli, E., and Rus, D. (2012). Robotic load balancing for mobility-ondemand systems. *Int J Rob Res.*

Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H., and Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. In *Proc Natl Acad Sci USA*.

Spieser, K., Samaranayake, S., Gruel, W., and Frazzoli, E. (2016). Shared-vehicle mobility-ondemand systems: A fleet operators guide to rebalancing empty vehicles. In *Transportation Research Board 95th Annual Meeting.*

Ünver, M. U. (2010). Dynamic Kidney Exchange. *Review of Economic Studies*, 77(1):372–414.

Vickrey, W. (1965). Pricing as a tool in coordination of local transportation. In *Transportation economics*, pages 275–296. NBER.

Zhang, R. and Pavone, M. (2014). Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. In *Proceedings of Robotics: Science and Systems Conference.*