Online Hypergraph Matching with Delays

Marco Pavone[1], Amin Saberi[1], Maximilian Schiffer[2], and Matthew Tsao[1]

[1] Stanford University, Stanford CA, 94305 USA
{pavone,saberi,mwtsao}@stanford.edu
[2] Technical University of Munich, Munich 80333, Germany
schiffer@tum.de

Abstract. We study an online hypergraph matching problem with delays, motivated by ridesharing applications. In this model, users enter a marketplace sequentially, and are willing to wait up to d timesteps to be matched, after which they will leave the system in favor of an outside option. A platform can match groups of up to k users together, indicating that they will share a ride. Each group of users yields a match value depending on how compatible are with one another. As an example, in ridesharing, k is the capacity of the service vehicles, and d is the amount of time a user is willing to wait for a driver to be matched to them.
We present results for both the utility maximization and cost minimization variants of the problem. In the utility maximization setting, the optimal competitive ratio is $\frac{1}{d}$ whenever $k \geq 3$, and is achievable in polynomial-time for any fixed k. In the cost minimization variation, when $k = 2$, the optimal competitive ratio for deterministic algorithms is $\frac{3}{2}$ and is achieved by a polynomial-time thresholding algorithm. When $k > 2$, we show that a polynomial-time randomized batching algorithm is $(2 - \frac{1}{d}) \log k$-competitive, and it is NP-hard to achieve a competitive ratio better than $\log k - O(\log \log k)$.

Keywords: Online algorithms · competitive analysis · ridesharing.

1 Introduction

The United States loses hundreds of billions of dollars annually through time spent waiting in traffic. Ride-hailing services such as Uber and Lyft are significant contributors to metropolitan congestion, since the convenience and privacy offered by their services has led to increased travel demand [11]. At the same time, they also have the potential to reduce congestion by the way of *shared rides*, whereby travelers can share rides instead of each having their own vehicle on the road. A recent study [18] suggests that ridesharing can reduce congestion when deployed together with congestion pricing and road tolls.

With this as motivation, we study the ridesharing problem through the lens of online hypergraph matching. In our model, the problem instance is represented as a hypergraph. Vertices represent users who arrive to a marketplace over time and hyperedges joining multiple users signify that those users can participate in a shared service. To model disutility for waiting, users will wait in the system for at most d timesteps, after which they will leave the system in favor of an outside option.

This model captures several characteristics of modern ridesharing. First, hyperedges represent groups of potentially more than 2 people sharing a ride, i.e. a service vehicle can carry up to k customers at a time. This is a generalization of previous work [5] where at most 2 people were allowed to share a ride. Similar to recent works [5,2], the spatial nature of ridesharing is captured by weights of the hyperedges, where the weight of a hyperedge measures the match efficiency of the associated requests. The online aspect of our model captures the temporal uncertainty faced in ridehailing systems where decisions must be made without knowledge of future demand.

We study both the utility maximization and cost minimization variants of the problem. In the utility maximization setting, edge weights represent the utility gained by the associated users sharing a service, so the goal is to find a matching in the hypergraph with large total weight. In the cost minimization setting, edge weights correspond to the cost required to serve all associated vertices together. Vertices also have

weights, representing the cost of serving the vertex individually. The goal in this setting is to find a matching which minimizes the total edge cost plus the total cost of unmatched vertices.

1.1   Contributions

We study `Online-`(k, d)`-Max-Matching` and `Online-`(k, d)`-Min-Matching`, which are the utility maximization and cost minimization variants of the problem respectively. For `Online-`(k, d)`-Max-Matching` we show that whenever $k \geq 3$, the optimal competitive ratio is $\frac{1}{d}$, and we present a polynomial-time randomized batching algorithm which is $\frac{1}{d}$-competitive.

The cost minimization version of the problem will help us incorporate additional structures. We will assume the costs are monotone and subadditive. That will allow us to prove the following set of results. For $k = 2$, the optimal competitive ratio is $\frac{3}{2}$ for deterministic algorithms. We present a polynomial-time deterministic thresholding algorithm which is $\frac{3}{2}$-competitive. For $k > 2$, there exists a randomized batching algorithm which is $2 - \frac{1}{d}$ competitive. This algorithm, however, is not polynomial-time. Using a reduction from set cover, we show that it is NP-hard to have a competitive ratio better than $\log k - O(\log \log k)$. Leveraging the reduction, we also construct a randomized greedy batching procedure and show that it is $(2 - \frac{1}{d}) \log k$-competitive, establishing the optimal polynomial-time competitive ratio up to a factor of $2 - \frac{1}{d}$.

1.2   Organization

The remainder of this paper is structured as follows. In Section 2 we formally define our online hypergraph matching model with deadlines. We discuss related work in Section 3. In Section 4 we study the utility maximization variant of the model wherein we characterize the optimal competitive ratio. We study the cost minimization variant of the model in Section 5. Section 6 presents conclusions and directions for future work.

2   Model

A total of n requests will arrive to the platform sequentially, one per timestep. Each request is represented by a vertex, and the vertices are ordered corresponding to the order in which requests arrive in the system. A request must be matched within d timesteps of its arrival, i.e. a request arriving at timestep i must be matched by timestep $i + d - 1$ at the latest. A vertex is **critical** if it will leave the system in the next timestep. Concretely, a vertex arriving at timestep i will become critical at timestep $i + d - 1$.

The system can serve the requests in groups of size at most k, where k is a constant independent of n, d. For example, in ridesharing, k would be the capacity of a single service vehicle. The edge set E represents compatible matches. Namely, $e \in E$ implies that the set e of users can be served together.

The problem instance can thus be represented by a hypergraph $H = (V, E, w)$. The following definition is a generalization of shareability graphs from [2,5].

Definition 1 (Shareability Hypergraph). *A hypergraph $H = (V, E, w)$ is a Shareability Hypergraph with parameters (n, d, k) if and only if 1) The set of vertices is ordered $V = [n]$, and vertex i arrives at timestep i, 2) $diam(e) < d$ for all $e \in E$, where $diam(e) := \max_{i,j \in e} |i - j|$ 3) The rank of H is at most k, i.e. $|e| \leq k$ for all $e \in E$ and 4) $w : E \to \mathbb{R}_+$, i.e. all hyperedge weights $w(e)$ are non-negative.*

In the utility maximization setting, the weight $w(e)$ of an edge e is the utility gained by having the vertices in e be served together. An optimal sharing strategy in a utility maximization problem corresponds to a maximum weight matching in H. In the cost minimization setting, the weight $w(e)$ of an edge e represents the cost incurred when the vertices in e are served together. An optimal sharing strategy in a cost minimization problem thus corresponds to a minimum weight matching in H, where the system also pays a cost for each unmatched vertex. In this model we assume that all edge weights are non-negative.

We now formally define the utility maximization version of online hypergraph matching with deadlines, which we will call the `Online-`(k, d)`-Max-Matching` problem.

Definition 2 (`Online-`(k, d)`-Max-Matching`)**.** *The vertices of a shareability hypergraph $H = (V, E, w)$ with parameters (n, d, k) are revealed sequentially, one per timestep. The vertex that arrives at timestep t will disappear at timestep $t + d$. A hyperedge (along with its weight) $e \in E$ is revealed once all of its vertices have arrived. The `Online-`(k, d)`-Max-Matching` problem is to maximize over matchings of H the weight of the matching, subject to the constraint that a hyperedge can only be included after it is revealed but before any of its vertices disappear, and hyperedge inclusion is irrevocable.*

The offline version of the problem is as follows:

Definition 3 (k-`Max-Matching`)**.** *We use k-`Max-Matching` to refer to the offline version of `Online-`(k, d)`-Max-Matching` where $d > n$ so the whole hypergraph can be observed before computing a matching.*

The cost minimization version, which we call the `Online-`(k, d)`-Min-Matching` problem can be defined similarly. In this setting, to ensure there is a feasible solution, we assume each customer can be served individually, i.e. the shareability hypergraph H satisfies $\{i\} \in E$ for every $i \in V$. In other words, the platform must decide by time $i + d - 1$ whether to serve request i individually or have it share a service with other vertices.

Definition 4 (`Online-`(k, d)`-Min-Matching`)**.** *The vertices of a shareability hypergraph $H = (V, E, w)$ with parameters (n, d, k) are revealed sequentially, one per timestep. Each vertex has a weight, which represents the cost of serving this vertex individually. The vertex that arrives at timestep t will disappear at timestep $t + d$. A hyperedge (along with its weight) $e \in E$ is revealed once all of its vertices have arrived. The `Online-`(k, d)`-Min-Matching` problem is to minimize over matchings of H the weight of the matching plus the weight of all unmatched vertices, subject to the constraint that a hyperedge can only be included after it is revealed but before any of its vertices disappear, and hyperedge inclusion is irrevocable.*

Definition 5 (k-`Min-Matching`)**.** *We use k-`Min-Matching` to refer to the offline version of `Online-`(k, d)`-Min-Matching` where $d > n$ so the whole hypergraph can be observed before computing a matching.*

3   Related Work

This paper is related to online resource allocation and online matching problems with delays. In the following, we survey some recent results in these fields and discuss how they relate to this work.

Online resource allocation has applications in advertising [15], network routing problems [17], and ridesharing [5]. In general, obtaining polynomial-time constant factor approximations for these problems is NP-hard, so [15,17] study special cases where users can request for at most k different resource types. This assumption is realistic in most settings as a single user will not need more than a constant number of resources. [15] presents a sample-and-price algorithm and shows that it is $\frac{1}{k^2}$-competitive if the arrival order of requests is uniformly random. If the full distribution over user requests is known to the network operator, [17] show via approximate dynamic programming that a competitive ratio of $\frac{1}{k+1}$ is possible.

The above line of work does not model the trade-off between waiting costs and market thickness. The option of waiting for a thicker market is studied in [4,10,1,3,5] in different contexts. In these works, requests arrive sequentially in time, and the platform can match pairs of requests together. The objective in [4,10,1,3] is cost minimzation, and the objective in [5] is utility maximization.

In [4,10], the platform can defer matching decisions but must pay a cost proportional to the total time requests wait before being matched. In [1,3] requests arrive to the system according to some known distribution. Requests will stay in the system for a random and unkown amount of time, so the platform operator risks users leaving the system if it chooses to wait for a thicker market. Online matching with deadlines was studied in [5]. Requests enter the system sequentially and are willing to wait up to d timesteps to be matched, after which they will leave the system. Here d is known to the platform. The authors present a $\frac{1}{4}$-competitive algorithm and also show that no algorithm can have a competitive ratio better than $\frac{1}{2}$.

This paper contributes generalizations to these aforementioned works in two ways. With respect to the online matching literature, we consider a model where $k > 2$ users can be matched together. This is motivated by ridesharing applications where service vehicles can hold more than 2 customers at any given time. With respect to the online resource allocation literature, we present results that do not assume any distributional information on the input instances. Our model also allows for waiting so that the platform has d units of time after a request appears to determine whether to accept it or reject it. This is in contrast to the works of [15,17] where the platform must immediately decide between accepting or rejecting upon a request's arrival.

4 Utility Maximization: `Online-`(k, d)`-Max-Matching`

In this section, we characterize the optimal competitive ratio for `Online-`(k, d)`-Max-Matching` by first proving an upper bound on achievable competitive ratios, and then presenting a polynomial-time algorithm which achieves this upper bound.

4.1 Upper Bounds

In this section we show that if the matching capacity k is at least 3, then no online algorithm can have a better competitive ratio than $\frac{1}{d}$, with d being the number of timesteps a customer stays in the system.

Theorem 1 (Upper bound on the competitive ratio). *If $k \geq 3$, then no online algorithm can have a competitive ratio better than $\frac{1}{d}$ for `Online-`(k, d)`-Max-Matching`.*

Proof. Consider a family $\mathcal{G}$ of shareability hypergraphs where $G \in \mathcal{G}$ is of the form $G = (V, E, w)$ with $V = \{0, 1, ..., 2d - 2\}$, $w \in \mathbb{R}_+^{|E|}$, and $E = \{e_t\}_{t=0}^{d-1}$ with $e_0 := \{0, d - 2, d - 1\}$, $e_{d-1} := \{d - 1, d, 2d - 2\}$ and

$$e_t := \{t, d - 1, t + d - 1\} \text{ for all } 0 < t < d - 1.$$

First, note that any matching can have at most one hyperedge since $e_t \cap e_{t'} \neq \emptyset$ for any t, t' since in particular, all edges contain the vertex $d - 1$. Second, every edge e_t can only be chosen in the timestep in which its weight is revealed. This is because the earliest vertex in e_t is t so e_t expires at time $t + d$ and the latest vertex in e_t is $t + d - 1$, so the earliest time e_t can be chosen is $t + d - 1$. So `Online-`(k, d)`-Max-Matching` restricted to instances in $\mathcal{G}$ is equivalent to the Adversarial Secretary Problem with d applicants, which is defined below. See Appendix B.1 for a visualization of graphs in $\mathcal{G}$ and the relationship to d-`ASP`.

Definition 6 (Adversarial Secretary Problem with d applicants (d-`ASP`)). *Adversarial Secretary Problem with d applicants, which we denote by d-`ASP`, has the following setting: There are d applicants for a secretary position, with corresponding non-negative aptitude scores $\{w_i\}_{i=1}^{d}$. At most 1 applicant can be hired and decisions are irrevocable. Time is discretized into d timesteps. The aptitude w_i of the ith applicant is revealed at time i, and the ith applicant can only be hired at time i. The goal is to maximize the aptitude of the hired secretary.*

Contrary to the canonical secretary problem where the arrival order of candidates is uniformly random, d-`ASP` makes no distributional assumption on the ordering of the candidates. Next we establish a hardness result for d-`ASP`.

Lemma 1 (Adversarial Secretary Problem). *The best possible competitive ratio for the adversarial secretary problem with d applicants is $\frac{1}{d}$.*

See Appendix B.2 for a proof of Lemma 1. Since d-`ASP` is a special case of `Online-`(k, d)`-Max-Matching`, this proves that the best possible competitive ratio of the latter is at most $\frac{1}{d}$.   □

4.2   Algorithms

In this section we present a polynomial-time $\frac{1}{d}$-competitive algorithm for `Online-`(k,d)`-Max-Matching` by adapting techniques from k`-Max-Matching`, the offline version of the problem. Specifically, we analyze the `Randomized-Batching` algorithm described in Algorithm 1 which runs a k`-Max-Matching` algorithm $\mathcal{A}$ every d timesteps, effectively treating the online problem as n/d separate offline problems. We refer to sets of d consecutive vertices on which $\mathcal{A}$ is applied as batches. Figure 1 illustrates a batching procedure. The start and end times for the batches are determined by a random shift $Z \in \{0, 1, ..., d-1\}$.

The random shift Z in `Randomized-Batching` is essential to the competitiveness of batching. Figure 2 illustrates why batching without randomization can do arbitrarily worse than the optimal, i.e. it has a competitive ratio of zero.

Algorithm 1: `Randomized-Batching`$(H; \mathcal{A})$

---

1 Input: Shareability Graph $H = (V, E, w)$, Offline matching algorithm $\mathcal{A}$;
2 Output: Matching M;
3 $M \leftarrow \emptyset$;
4 Draw Z uniformly at random from $\{0, 1, 2, ..., d-1\}$;
5 for $1 \leq i \leq n$ **do**
6   Vertex i is revealed;
7   **if** $(i + 1 \equiv Z \mod d)$ *or* $(i = n)$ **then**
8   Use $\mathcal{A}$ to compute a matching M_0 for nodes appearing at times in $\{i - d + t\}_{t=1}^{d}$;
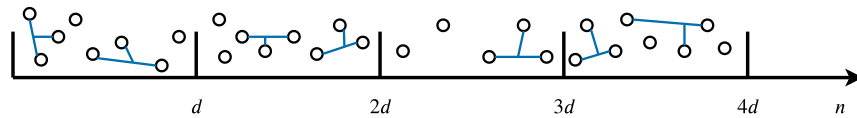**9** $M \leftarrow M \cup M_0$;
10 Return M

Fig. 1: An example of batching for the online matching problem. The time horizon is partitioned into batches of length d, and an offline algorithm is used to compute a matching (blue hyperedges) for each batch.
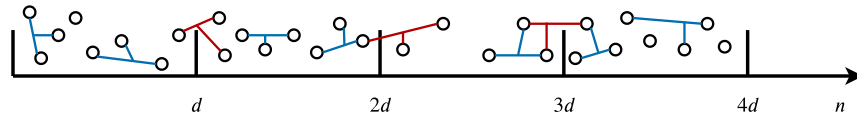

Fig. 2: Consider a deterministic batching algorithm where the ith batch consists of vertices $\{id + t\}_{t=0}^{d-1}$. The batching algorithm is unable to select the red edges, since they are not fully contained in any batch. If the red edges each have weight 1 and all other edges have weight ϵ, then the weight of the matching obtained by batching is at most $\epsilon |E|$, whereas the optimal solution has weight at least 3. Hence the competitive ratio of batching is at most $\frac{\epsilon |E|}{3}$, which converges to zero as $\epsilon \to 0$.

The following result provides a performance guarantee for `Randomized-Batching`.

Theorem 2 (Performance of `Randomized-Batching`). *Let $\mathcal{A}$ be an algorithm for `k-Max-Matching` with competitive ratio $\rho(\mathcal{A})$. For any matching $\widetilde{M}$ in H, the matching M returned by Algorithm 1 using $\mathcal{A}$ satisfies*

$$\mathbb{E}\left[w\left(M\right)\right] \geq \frac{\rho\left(\mathcal{A}\right)}{d} \sum_{e \in \widetilde{M}} w(e)\left(d - diam(e)\right).$$

Proof. Let $\mathcal{A}$ be an offline matching algorithm for `k-Max-Matching` with competitive ratio $\rho(\mathcal{A})$ and consider an instance $H = (V, E, w)$ of `Online-(k,d)-Max-Matching`. For each $z \in \{0, 1, ..., d-1\}$ we define the batch $W_{z,i} = (V_{z,i}, E_{z,i}, w)$ where

1. $V_{z,i} := \{z + id + t\}_{t=0}^{d-1}$, i.e. all vertices arriving between timestesp $z + id$ and $z + (i+1)d - 1$.
2. $E_{z,i} := \{e \in E : e \subset V_{z,i}\}$, i.e. all edges whose vertices are all in $V_{z,i}$.

We will also define $E_z := \cup_i E_{z,i}$. With this notation, `Randomized-Batching` returns a matching $M := \cup_i M_{z,i}$, where $M_{z,i}$ is obtained by applying $\mathcal{A}$ to $W_{z,i}$. We will also use $M_{z,i}^*$ to denote a maximum weight matching in $W_{z,i}$. We will make use of the following observation.

Observation 1 *For any $e \in E$ we have $\sum_{z=0}^{d-1} \mathbb{1}_{[e \in E_z]} \geq d - diam(e)$.*

Proof. Fix any $e \in E$ and let t be the smallest arrival time of any vertex in e. Without loss of generality (by a linear change of coordinates) we assume that $t = 0$. By definition of $diam(e)$, all vertices of e appear between time 0 and $diam(e)$. Any edge that is a subset of $\{z, z+1, ..., z+d-1\}$ will belong to E_z. Next, note that for any z satisfying $diam(e) - d + 1 \leq z \leq 0$, we have $z \leq 0$ and $z + d - 1 \geq diam(e)$, and hence $e \subset \{z, z+1, ..., z+d-1\}$. Therefore, $e \in E_z$ for every $z \in \{diam(e) - d + 1, ..., -1, 0\} \mod d$. In particular, $e \in E_z$ for at least $d - diam(e)$ values of z. $\qquad\square$

Let $\widetilde{M}$ be a matching for H, and let M_z^* be an optimal matching for $H_z := (V, E_z, w)$. Given $Z = z$, we have

$$w(M) = \sum_i w(M_{z,i}) \overset{(a)}{\geq} \sum_i \rho(\mathcal{A})w(M_{z,i}^*) = \rho(\mathcal{A})w(M_z^*). \tag{1}$$

Where (a) is because $\mathcal{A}$ is $\rho(\mathcal{A})$-competitive, and $M_{z,i}$ is obtained by applying $\mathcal{A}$ to $W_{z,i}$. Because Z is uniformly distributed over $\{0, 1, ..., d-1\}$, we have

$$\mathbb{E}_Z\left[w(M)\right] = \sum_{z=0}^{d-1} \mathbb{P}\left[Z = z\right]\mathbb{E}\left[w(M)|Z = z\right] \overset{(a)}{\geq} \frac{1}{d}\sum_{z=0}^{d-1}\rho(\mathcal{A})\sum_{e^* \in M_z^*} w(e^*) \overset{(b)}{\geq} \frac{\rho(\mathcal{A})}{d}\sum_{z=0}^{d-1}\sum_{e^* \in \widetilde{M} \cap E_z} w(e^*)$$

$$= \frac{\rho(\mathcal{A})}{d}\sum_{z=0}^{d-1}\sum_{e^* \in \widetilde{M}} w(e^*)\mathbb{1}_{[e \in E_z]} = \frac{\rho(\mathcal{A})}{d}\sum_{e^* \in \widetilde{M}} w(e^*)\sum_{z=0}^{d-1}\mathbb{1}_{[e \in E_z]} \overset{(c)}{\geq} \frac{\rho(\mathcal{A})}{d}\sum_{e^* \in \widetilde{M}} w(e^*)\left(d - diam(e)\right).$$

Where (a) is due to (1), (b) is because and $\widetilde{M} \cap E_z$ is a matching in H_z and M_z^* is a maximum weight matching in H_z. Finally, (c) is due to Observation 1. $\qquad\square$

The following is a straightforward consequence of Theorem 2 by choosing $\widetilde{M}$ to be an optimal matching, and noting that $d - diam(e) \geq 1$ for all $e \in E$ since H is a shareability hypergraph.

Lemma 2 (Competitive Ratio of `Randomized-Batching`). *Let $\mathcal{A}$ be an offline matching algorithm for `k-Max-Matching` with competitive ratio $\rho(\mathcal{A})$. Algorithm 1 using $\mathcal{A}$ is $\frac{\rho(\mathcal{A})}{d}$-competitive for `Online-(k,d)-Max-Matching`.*

We now briefly explore the implications of Lemma 2 for different choices of $\mathcal{A}$. The best known polynomial-time algorithms for k-`Max-Matching` are $O(\frac{1}{k})$-competitive [6,8], and in particular `Greedy` is $\frac{1}{k}$-competitive.[3]

Corollary 1. *Choosing $\mathcal{A}$ to be `Greedy` when running Algorithm 1 gives a polynomial-time algorithm with competitive ratio of $\frac{1}{kd}$ for `Online-`(k,d)`-Max-Matching`.*

We can also consider an exact procedure $\mathcal{A}^*$ for k-`Max-Matching` which satisfies $\rho(\mathcal{A}^*) = 1$. Applying Lemma 2 with such an $\mathcal{A}^*$ establishes $\frac{1}{d}$ as the optimal competitive ratio for `Online-`(k,d)`-Max-Matching`.

Corollary 2. *If $\mathcal{A}^*$ is an exact procedure for k-`Max-Matching`, i.e., $\rho(\mathcal{A}^*) = 1$, then Algorithm 1 using $\mathcal{A}^*$ as a subroutine achieves a competitive ratio of $\frac{1}{d}$ for `Online-`(k,d)`-Max-Matching`.*

Corollary 2 in conjunction with Theorem 1 shows that the optimal competitive ratio of $\frac{1}{d}$ is achievable in finite time. However, it is known from [13] that achieving $\rho(\mathcal{A}) = \Omega\left(\frac{\log k}{k}\right)$ for k-`Max-Matching` is NP-hard, and thus no exact procedure $\mathcal{A}^*$ can run in polynomial-time unless P = NP. For this reason, Corollary 2 does not show that a competitive ratio of $\frac{1}{d}$ is achievable in *polynomial-time*.

To obtain a polynomial-time $\frac{1}{d}$-competitive algorithm, we will replace `Greedy` by `Depth-`k `Greedy` as the choice for $\mathcal{A}$. `Depth-`k `Greedy` is described in Algorithm 2. Making this replacement leads to the following result.

Theorem 3. *`Randomized-Batching` using $\mathcal{A} =$`Depth-`k `Greedy` as a subroutine is $\frac{1}{d}$-competitive for `Online-`(k,d)`-Max-Matching`. Furthermore, its running time is $O\left(|E|^{k+1}\right)$ which is polynomial in the size of the shareability hypergraph H for any fixed k.*

Algorithm 2: `Depth-`k `Greedy`(H)

---

1 Input: Hypergraph $H = (V, E, w)$ where the vertices are ordered;
2 Output: A matching M for H ;
3 Define L to be the first k vertices;
4 Define R to be the last k vertices;
5 Define $\mathcal{M}_{L,R}(H) := \{M \text{ is a matching} : e \cap L \neq \emptyset, e \cap R \neq \emptyset \ \forall e \in M\}$;
6 Enumerate all matchings in $\mathcal{M}_{L,R}(H)$ as $\{M_i\}_{i=1}^{|\mathcal{M}_{L,R}(H)|}$;
7 for $1 \leq i \leq |\mathcal{M}_{L,R}(H)|$ **do**
8 $V_i \leftarrow \{i \in V : i \text{ is matched in } M_i\}$;
9 $\widetilde{M}_i \leftarrow$ `Greedy`$((V \setminus V_i, E))$
10 $i^* \leftarrow \arg\max_i w(M_i \cup \widetilde{M}_i)$;
11 $M \leftarrow M_{i^*} \cup \widetilde{M}_{i^*}$;
12 Return M

In the remainder of this section, we sketch the proof of Theorem 3. See Appendix B.3 for a full proof of Theorem 3.

To understand how to construct a polynomial-time algorithm for `Online-`(k,d)`-Max-Matching` with the optimal competitive ratio of $\frac{1}{d}$, we first need to understand why Corollary 1 falls short: let M^* be a maximum weight matching for a shareability hypergraph $H = (V, E, w)$. We can partition M^* into two sets: *short hyperedges* M_s^* and *long hyperedges* M_ℓ^* as follows:

$$M_s^* := \{e \in M^* : \operatorname{diam}(e) \leq d - k\} \text{ and } M_\ell^* := \{e \in M^* : \operatorname{diam}(e) > d - k\}.$$

---

[3] See Appendix A.1 for further details and existing results for k-`Max-Matching`.

As the names suggest, M_s^* contains all hyperedges of M^* with diameter at most $d - k$, and M_ℓ^* contains all hyperedges whose diameters are larger than $d - k$. By Theorem 2, running `Randomized-Batching` with the `Greedy` subroutine results in a matching whose weight satisfies

$$\mathbb{E}\left[w\left(M\right)\right] \geq \frac{\rho\left(\mathcal{A}\right)}{d} \sum_{e \in M^*} w(e)\left(d - \mathrm{diam}(e)\right) = \frac{1}{kd} \sum_{e \in M_s^*} w(e)\left(d - \mathrm{diam}(e)\right) + \frac{1}{kd} \sum_{e \in M_\ell^*} w(e)\left(d - \mathrm{diam}(e)\right)$$

$$\geq \frac{1}{kd} \sum_{e \in M_s^*} w(e)k + \frac{1}{kd} \sum_{e \in M_\ell^*} w(e) = \frac{1}{d} w(M_s^*) + \frac{1}{kd} w(M_\ell^*).$$

From this, we see that the approximation factor for M_s^* and M_ℓ^* are different.

Observation 2 *Since M_s^* is already contributing a $\frac{1}{d}$ fraction of its weight to M, we just need to improve the contribution of M_ℓ^* from $\frac{1}{kd}$ to $\frac{1}{d}$ in order to achieve a competitive ratio of $\frac{1}{d}$.*

Since all edges in M_ℓ^* have large diameter, we can show that $|M_\ell^*|$ cannot be too large. Specifically, for any batch (i.e. subgraph on d consecutive vertices) (V', E'), we can show that $|M_\ell^* \cap E'| \leq k$. We can then show that there are at most $O\left(\binom{d}{k}^k\right)$ possibilities for $M_\ell^* \cap E'$. Since k is a constant, this is polynomial in the batch size. Finally, we extend all of these possibilities for $M_\ell^* \cap E'$ to maximal matchings in (V', E') via `Greedy` and return the maximal matching with the highest total weight. This can be done in polynomial-time, and improves the contribution of $w(M_\ell^*)$ from $\frac{1}{kd}$ to $\frac{1}{d}$. This motivates `Depth-k Greedy` described in Algorithm 2 as an alternative to `Greedy`.

5   Cost Minimization: `Online-`(k, d)`-Min-Matching`

In this section we study `Online-`(k, d)`-Min-Matching` specialized to the ridesharing problem. We first present additional assumptions that we make to utilize the structure of ridesharing problems. For $k = 2$ we prove that the optimal competitive ratio for deterministic algorithms is $\frac{3}{2}$ and is achieved by a thresholding algorithm. For $k > 2$, we characterize both the optimal competitive ratio and the optimal competitive ratio attainable with polynomial-time algorithms up to a factor of $2 - \frac{1}{d}$.

5.1   Ridesharing Assumptions

Let $H = (V, E, w)$ be the shareability hypergraph for an instance of `Online-`(k, d)`-Min-Matching`. For the application to ridesharing, we leverage the following two assumptions.

Assumption 1 (Monotonicity) *For any $e \in E$, and any $e' \subseteq e$, we have $e' \in E$ and $w(e') \leq w(e)$.*

Assumption 1 is realistic in ridesharing problems for the following reason. The cost $w(e)$ is the minimum cost required to service the customers in e, i.e. taking them from their pickup point to their dropoff point. Let p be such a minimum cost path. Note that p also serves all requests in e', and therefore, the minimum cost required to serve e' can only be smaller than $w(e)$. Thus we see that $w(e') \leq w(e)$.

Assumption 2 (Subadditivity) *For any $e \in E$, and any partition $e = \sqcup_{j=1}^{\ell} e_j$, we have $w(e) \leq \sum_{j=1}^{\ell} w(e_j)$.*

The above assumption is also quite reasonable. If there exists $e \in E$ which does not satisfy Assumption 2, then any matching that includes e can be converted to a matching with lower cost by replacing e by $\{e_j\}_{j=1}^{\ell}$. Therefore e will never be in an optimal matching, and so we can remove e from E. Finally, we will consider the single occupancy matching M_{so} as a benchmark.

Definition 7 (Single Occupancy Matching). *The single occupancy matching serves each customer individually, i.e. $M_{so} = \emptyset$. It is clear from Assumption 2 that the competitive ratio of M_{so} is k.*

5.2   The $k = 2$ Case

We first study the $k = 2$ case where the shareability hypergraph is in fact a shareability graph. In this case, single occupancy is a 2-competitive algorithm. We will thus limit our search to algorithms whose competitive ratio is smaller than 2. We will use the following notation:

- For $i \in V$, we use $w_i \in \mathbb{R}_+$ to denote the cost of serving i individually.
- For $(i, j) \in E$, we use $w(i, j) \in \mathbb{R}_+$ to denote the cost of serving i and j together.
- Given a matching M, a vertex i is unmatched under M if it is not an endpoint of any edge in M.
- We use $w(M)$ to represent the cost of a matching M, defined via $w(M) := \sum_{(i,j) \in M} w(i, j) + \sum_{i \in U} w_i$ where U is the set of unmatched vertices in M.

The following lemma gives a lower bound on the competitive ratio of all deterministic and randomized algorithms and is proved in Appendix C.1.

Lemma 3 (Lower Bound on the Competitive Ratio). *When $k = 2$, no deterministic algorithm can have a competitive ratio smaller than $\frac{3}{2}$ and no randomized algorithm can have a competitive ratio smaller than $\frac{5}{4}$.*

Before presenting our algorithm, let us discuss a motivating example to highlight one of the key challenges in `Online-`(k, d)`-Min-Matching`. Consider the following shareability graph where $w_A = 1$, $w_B = x > 1$, and $w(A, B) = x$. There is also a third vertex C with $w_C \in \{0, x\}$ and $w(B, C) = x$.
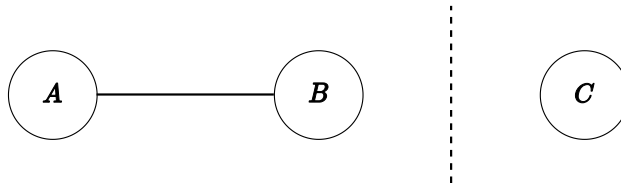
Suppose A becomes critical before C arrives. In this case, our algorithm needs to decide whether to match (A, B), or wait and match (B, C) instead. The optimal matching depends on w_C. In particular, if $w_C = 0$, then the minimum cost is $w(A, B) + w_C = x$ and is achieved by matching (A, B). However, if $w_C = x$, then the minimum cost is $w_A + w(B, C) = 1 + x$. Since we are interested in the worst-case error, we make the following two observations.

1. If $w_C = 0$, and we choose not to match (A, B), then our resulting cost will be $w_A + w(B, C) = 1 + x$. Recall that the minimum cost is x, so our matching has a cost that is $\frac{1+x}{x}$ times as large as the minimum cost.
2. If $w_C = x$ and we choose to match (A, B), then our resulting cost will be $w(A, B) + w_C = 2x$. Recall that the minimum cost in this case is $1 + x$, so our matching has a cost that is $\frac{2x}{1+x}$ times as large as the minimum cost.

Since our goal is to minimize the worst case performance, this shows that we should match (A, B) if $\frac{2x}{1+x} \leq \frac{1+x}{x}$, and not match A otherwise. Equivalently, we should match (A, B) if and only if $x \leq \frac{1}{\sqrt{2}-1}$.

This motivates thresholding algorithms which only accept matches that are sufficiently high quality as a way of balancing immediate reward with possible future reward. Algorithm 4 describes the `Risk-Threshold` algorithm, which matches critical vertices to the most appealing partner according to a risk score. We also present `Risk-Threshold-ag` in Algorithm 3, which is an agnostic version of `Risk-Threshold` that does not need to know when vertices become critical, and can thus be used in settings where users' willingness to wait is not observable. The competitive ratios of these algorithms are presented in the following results.

Theorem 4. *Using $\theta = \frac{\sqrt{5}-1}{2}$, the competitive ratio of `Risk-Threshold-ag`$(\cdot, \frac{\sqrt{5}-1}{2})$ is $\frac{\sqrt{5}+1}{2} \approx 1.61$.*

Algorithm 3: `Risk-Threshold-ag`(H, θ)

---

1 Input: Shareability Graph $H = (V, E, w)$, Risk parameter $\theta \in [\frac{1}{2}, 1]$;
2 Output: Matching M;
3 Define $\theta_{ij} := \frac{w(i,j)}{w_i + w_j}$ for $(i, j) \in E$;
4 $M \leftarrow \emptyset$;
5 while *There are vertices in the system* **do**
6   **if** $\exists i, j : \theta_{ij} \leq \theta$ **then**
7 $M \leftarrow M \cup \{(i, j)\}$;
8 Return M

Algorithm 4: `Risk-Threshold`(H, θ)

---

1 Input: Shareability Graph $H = (V, E, w)$, Risk parameter $\theta \in [\frac{1}{2}, 1]$;
2 Output: Matching M;
3 Define $\theta_{ij} := \frac{w(i,j)}{w_i + w_j}$ for $(i, j) \in E$;
4 $M \leftarrow \emptyset$;
5 while *There are vertices in the system* **do**
6   **if** i *is critical* **then**
7 $j^* \leftarrow \arg\min_j \theta_{ij}$;
8   **if** $\theta_{ij^*} \leq \theta$ **then**
9 $M \leftarrow M \cup \{(i, j^*)\}$
10 Return M

Theorem 5. *Using* $\theta = \frac{2}{3}$, `Risk-Threshold`$(\cdot, \theta)$ *is* $\frac{3}{2}$-*competitive.*

Theorem 5 in conjunction with Lemma 3 shows that `Risk-Threshold` has the optimal competitive ratio among deterministic algorithms.

While every customer is willing to wait d timesteps in our model, `Risk-Threshold-ag` and `Risk-Threshold` can handle heterogeneous deadlines. The algorithms do not require homogeneous deadlines, and the proofs do not need this property either. Concretely, Theorems 4 and 5 hold in the more general setting where the platform has until timestep $i + d_i$ to decide how to serve a user who arrives at timestep i, where the d_i need not be the same.

Another interesting observation is that knowledge of willingness to wait in this model is not very valuable for deterministic algorithms. Indeed, `Risk-Threshold-ag` is 1.61-competitive without knowledge of d, and Lemma 3 shows that knowledge of d can only improve the competitive ratio by 0.11.

We now prove Theorem 4. The proof of Theorem 5 is similar, but more technically involved, and can be found in Appendix C.2.

Proof (Theorem 4). For a given instance H, let M^* denote a minimum weight matching, and let U^* be the set of vertices unmatched under M^*. For each $(i, j) \in E$, we denote $\theta_{ij} := \frac{w(i,j)}{w_i + w_j}$. For $\theta \in [\frac{1}{2}, 1]$, define the set $E_\theta := \{(i, j) \in E : \theta_{ij} \leq \theta\}$. Next, let M be a matching produced by `Risk-Threshold-ag`(H, θ), and let U be the set of unmatched vertices under M. Note that M is a maximal matching in (V, E_θ, w). To compare $w(M)$ to $w(M^*)$, we assign to each vertex a score v according to M by the following rule:

$$v_i := \begin{cases} w_i & \text{if } i \in U \\ \theta_{ij} w_i & \text{if } (i, j) \in M \text{ for some } j \in V \end{cases}$$

Note that by construction, $\sum_{i \in V} v_i = w(M)$. One way to interpret v is that v_i is the contribution of vertex i to the total cost of the matching. With this notation, we can now write

$$w(M) = \sum_{i \in V} v_i \stackrel{(a)}{=} \underbrace{\sum_{(i,j) \in M^*} (v_i + v_j) + \sum_{i \in U^*} v_i}_{} = \underbrace{\sum_{(i,j) \in M^* \cap E_\theta} (v_i + v_j)}_{\text{term 1}} + \underbrace{\sum_{(i,j) \in M^* \setminus E_\theta} (v_i + v_j)}_{\text{term 2}} + \underbrace{\sum_{i \in U^*} v_i}_{\text{term 3}},$$

where (a) is due to the fact that the set of matched nodes under M^* and the set of unmatched nodes under M^* form a partition of V. For term 1, note that $M^* \cap E_\theta$ is a matching in (V, E_θ, w). Since M is a maximal matching in (V, E_θ, w), this means that for every $(i,j) \in M^* \cap E_\theta$, at least one of i or j is also matched in M. Without loss of generality, suppose i is matched. Then we have

$$v_i + v_j \le \theta_{ij} w_i + w_j \le \theta_{ij} \max(w_i, w_j) + \max(w_i, w_j) \le (1+\theta) \max(w_i, w_j) \stackrel{(a)}{\le} (1+\theta) w(i,j).$$

Where (a) is due to Assumption 1. For term 2, note that $(i,j) \notin E_\theta$ means that $\theta_{ij} > \theta$. Thus for any $(i,j) \in M^* \setminus E_\theta$, we have:

$$v_i + v_j \le w_i + w_j = \frac{1}{\theta_{ij}} w(i,j) \le \frac{1}{\theta} w(i,j).$$

Finally, for term 3, note that $v_i \le w_i$ is always true. Putting these observations together, we see that:

$$\begin{aligned}
w(M) &= \sum_{(i,j) \in M^* \cap E_\theta} (v_i + v_j) + \sum_{(i,j) \in M^* \setminus E_\theta} (v_i + v_j) + \sum_{i \in U^*} v_i \\
&\le \sum_{(i,j) \in M^* \cap E_\theta} (1+\theta) w(i,j) + \sum_{(i,j) \in M^* \setminus E_\theta} \frac{1}{\theta} w(i,j) + \sum_{i \in U^*} w_i \\
&\le \max\left(1+\theta, \frac{1}{\theta}\right) \sum_{(i,j) \in M^* \cap E_\theta} w(i,j) + \max\left(1+\theta, \frac{1}{\theta}\right) \sum_{(i,j) \in M^* \setminus E_\theta} w(i,j) + \max\left(1+\theta, \frac{1}{\theta}\right) \sum_{i \in U^*} w_i \\
&= \max\left(1+\theta, \frac{1}{\theta}\right) \left(\sum_{(i,j) \in M^* \cap E_\theta} w(i,j) + \sum_{(i,j) \in M^* \setminus E_\theta} w(i,j) + \sum_{i \in U^*} w_i \right) = \max\left(1+\theta, \frac{1}{\theta}\right) w(M^*).
\end{aligned}$$

Thus `Risk-Threshold`$(\cdot, \theta)$ is a $\max\left(1+\theta, \frac{1}{\theta}\right)$-competitive algorithm. To finish, we now observe this value is minimized when $\theta = \frac{\sqrt{5}-1}{2}$, which implies that `Risk-Threshold`$\left(\cdot, \frac{\sqrt{5}-1}{2}\right)$ is a $\frac{\sqrt{5}+1}{2}$-competitive algorithm for `Online-`(k,d)`-Min-Matching` when $k = 2$. $\qquad\square$

5.3 The $k > 2$ Case

This case can be addressed by a combination of ideas that we have discussed earlier. Therefore, we just state the results, sketch the main ideas, and refer the reader to the Appendix for the details.

First, observe that the offline version of the problem k`-Min-Matching` is essentially a weighted set cover problem (see Appendix A.2). To prove the equivalence, one needs to make sure Assumption 1 does not make the problem strictly easier. This observation allows us to leverage hardness results from set cover [19] to prove hardness results for `Online-`(k,d)`-Min-Matching`.

We can also use the idea of batching with a random shift to get a postive result with loss of factor 2. Putting them all together gives the following:

Theorem 6. *Let $\mathcal{A}$ be an algorithm for k-`WSC` with approximation ratio $\rho(\mathcal{A})$. Then `Randomized-Batching`$(\cdot, \mathcal{A})$ is $\left(2 - \frac{1}{d}\right) \rho(\mathcal{A})$-competitive for `Online-`$(k,d)$`-Min-Matching`. In particular, `Randomized-Batching`$(\cdot, \mathcal{A})$ with $\mathcal{A} = $`Weighted-Greedy` is $\left(2 - \frac{1}{d}\right)(1 + \log k)$-competitive for `Online-`$(k,d)$`-Min-Matching`.*

Furthermore, no algorithm with running time polynomial in n and k can obtain a better ratio than `Online-`(k,d)`-Min-Matching` within $\log k - O(\log \log k)$ unless $P = NP$.

See Appendix C.4 for a proof of Theorem 6.

6   Conclusions

In this paper we studied online hypergraph matching with delays motivated by applications to high capacity ridesharing. We studied both the utility maximization and cost minimization variants of the problem. For utility maximization, we presented a polynomial-time randomized batching algorithm and proved that it attains the optimal competitive ratio. We studied cost minimization under additional assumptions specialized to the ridesharing problem. For $k = 2$ we introduced a thresholding algorithm and proved that it achieves the optimal competitive ratio for deterministic algorithms. For $k > 2$, we characterize both the optimal competitive ratio and the optimal polynomial-time competitive ratio up to a factor of $2 - \frac{1}{d}$.

There are several interesting directions for future work. With regards to `Online-`(k,d)`-Max-Matching`, adding additional application-specific assumptions could enable better approximation guarantees. For `Online-`(k,d)`-Min-Matching`, closing the $(2 - \frac{1}{d})$ gap between the achievability and impossibility results for $k > 2$ and finding a randomized algorithm that beats the $\frac{3}{2}$ threshold for $k = 2$ would both be interesting. For both models, extensions to heterogeneous and unobserved willingness to wait for $k > 2$ would provide compelling insight for practitioners and real-world deployment.

References

1. Akbarpour, M., Li, S., Gharan, S.O.: Thickness and information in dynamic matching markets. Journal of Political Economy **128**(3), 783–815 (2020)
2. Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D.: On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proc. Natl. Acad. Sci. U.S.A. **114**(3), 462–467 (2017)
3. Aouad, A., Saritac, O.: Dynamic stochastic matching under limited time (2019), `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3497624`
4. Ashlagi, I., Azar, Y., Charikar, M., Chiplunkar, A., Geri, O., Kaplan, H., Makhijani, R.M., Wang, Y., Wattenhofer, R.: Min-cost bipartite perfect matching with delays. In: Jansen, K., Rolim, J.D.P., Williamson, D., Vempala, S.S. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA. LIPIcs, vol. 81, pp. 1:1–1:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
5. Ashlagi, I., Burq, M., Dutta, C., Jaillet, P., Saberi, A., Sholley, C.: Edge weighted online windowed matching. In: Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019. pp. 729–742 (2019)
6. Berman, P.: A d/2 approximation for maximum weight independent set in d-claw free graphs. Nord. J. Comput. **7**(3), 178–184 (2000)
7. Chan, Y.H., Lau, L.C.: On linear and semidefinite programming relaxations for hypergraph matching. Math. Program. **135**(1-2), 123–148 (2012)
8. Chandra, B., Halldórsson, M.M.: Greedy local improvement and weighted set packing approximation. Journal of Algorithms **39**(2), 223–240 (2001)
9. Chvalal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research **4**, 233–235 (1979)
10. Emek, Y., Kutten, S., Wattenhofer, R.: Online matching: haste makes waste! In: Wichs, D., Mansour, Y. (eds.) Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016. pp. 333–344. ACM (2016)
11. Erhardt, G.D., Roy, S., Cooper, D., Sana, B., Chen, M., Castiglione, J.: Do transportation network companies decrease or increase congestion? Science Advances **5**(5) (2019)
12. Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM **45**(4), 634–652 (1998)
13. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k-set packing. Comput. Complex. **15**(1), 20–39 (May 2006)
14. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences **9**(3), 256–278 (1974)

15. Korula, N., Pál, M.: Algorithms for secretary problems on graphs and hypergraphs. In: Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II. pp. 508–520 (2009)
16. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Mathematics **13**(4), 383–390 (1975)
17. Ma, Y., Rusmevichientong, P., Sumida, M., Topaloglu, H.: A constant-factor approximation algorithm for network revenue management (2018)
18. Ostrovsky, M., Schwarz, M.: Carpooling and the economics of self-driving cars. In: Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019. pp. 581–582 (2019)
19. Trevisan, L.: Non-approximability results for optimization problems on bounded degree instances. In: Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece. pp. 453–461. ACM (2001)

A   Appendix: Background

In this section we introduce the hypergraph matching and set cover problems along with existing achievability and hardness results for these problems. Hypergraph matching and set cover play an important role in our results on utility maximization and cost minimization respectively.

A.1   Hypergraphs

In this section we introduce hypergraphs and the hypergraph matching problem[4].

Definition 8 (Hypergraph). *A hypergraph is a generalization of a graph where edges can join any number of vertices. A weighted undirected hypergraph $H = (V, E, w)$ is defined by its vertex set V, its hyperedge set E, and its hyperedge weights w. The hyperedge weights $w : E \to \mathbb{R}$ assign to each hyperedge a real value. For any $E' \subset E$, we define the weight of E' as $w(E') := \sum_{e \in E'} w(e)$.*

Below we list some definitions related to hypergraphs that will be used throughout this paper.

Definition 9 (Hypergraph Rank). *The rank of a hypergraph is the maximum cardinality of a hyperedge in the hypergraph, i.e. the hyperedges of a rank k hypergraph have at most k vertices.*

Definition 10 (Matching). *For a graph or hypergraph $H = (V, E)$, a matching M in H is a set of edges $M \subseteq E$ where all edges are disjoint.*

Definition 11 (Maximal Matching). *A matching M in H is maximal if it is not a strict subset of any other matching.*

Definition 12 (Maximum Matching). *A matching M in H is maximum if it there is no matching with greater cardinality.*

Definition 13 (Edge Diameter of Graphs and Hypergraphs). *In a graph or hypergraph $H = (V, E, w)$ with ordered vertices $V = [n]$, we define the diameter of an edge to be $diam(e) := \max_{i,j \in e} i - j$. Furthermore, we define the edge diameter of a graph G to be $diam(E) := \max_{e \in E} diam(e)$, where E is its edge set.*

Finding matchings in rank k hypergraphs We now define the k-`Max-Matching` problem, which plays a key role in our discussion on online hypergraph matching.

Definition 14 (k-Max-Matching). *The k-`Max-Matching` problem is to find a maximum weight matching in a rank k hypergraph H. The maximum weight matching problem in graphs is a special case where $k = 2$.*

The best known algorithms for k-`Max-Matching` [6,8] are combinatorial in nature and are $\frac{2}{k}$-competitive. Convex relaxations are surveyed in [7] who show that the standard linear programming relaxation of k-`Max-Matching` can be rounded to produce a $\frac{1}{k-1+\frac{1}{k}}$-competitive algorithm. Conversely, [13] proves that it is NP-hard to approximate k-`Max-Matching` better than $\Omega\left(\frac{\log k}{k}\right)$, showing that existing methods are optimal up to log factors.

Among all of the existing methods, we focus our discussion on a `Greedy` algorithm outlined in Algorithm 5. As described in Lemma 4, `Greedy` is appealing because its competitive ratio is within a constant factor of the best known result and also runs in nearly linear time.

Lemma 4 (`Greedy` is $\frac{1}{k}$-competitive). *The competitive ratio of Algorithm 5 is at least $\frac{1}{k}$ for k-`Max-Matching`. Furthermore, its runtime is $O\left(|E| \log |E|\right)$.*

---

[4] also known as set packing in the computer science literature

---
Algorithm 5: `Greedy`

---
1 **Input:** Weighted Hypergraph $H = (V, E, w)$;
2 **Output:** A matching M in H;
3 Set $M \leftarrow \emptyset$;
4 Sort edges in $E = \{e_j\}_{j=1}^{|E|}$ according to their weights, in decreasing order to obtain $\{\widetilde{e}_j\}_{j=1}^{|E|}$;
5 **for** $1 \leq j \leq |E|$ **do**
6   **if** $M \cup \{\widetilde{e}_j\}$ *is a matching* **then**
7 $M \leftarrow M \cup \{\widetilde{e}_j\}$;
8 Return M

A.2   Set Cover

Crucial to our discussion on minimum cost matchings will be the set cover problem and its variants, which we define below.

Definition 15 (Weighted Set Cover (WSC)). *Given a set X, a collection $\mathcal{S} \subset 2^X$ of subsets of X satisfying $\cup_{S \in \mathcal{S}} S = X$ and a weight function $w : \mathcal{S} \to \mathbb{R}_+$, the weighted set cover problem $\mathit{WSC}(X, \mathcal{S}, w)$ asks the following minimization problem*

$$\underset{\mathcal{E} \subset \mathcal{S}}{minimize} \sum_{S \in \mathcal{E}} w(S)$$

$$s.t. \ \cup_{S \in \mathcal{E}} S = X.$$

Definition 16 (Set Cover (SC)). *An instance of $\mathit{WSC}(X, \mathcal{S}, w)$ is an unweighted set cover problem if $w(S) = 1$ for every $S \in \mathcal{S}$. We denote such an instance as $\mathit{SC}(X, \mathcal{S})$.*

Definition 17 (Set Cover with a rank constraint (k-WSC)). *An instance $\mathit{WSC}(X, \mathcal{S}, w)$ is rank k if $|S| \leq k$ for every $S \in \mathcal{S}$. We denote the rank k set cover problem as $k\text{-}\mathit{WSC}(X, \mathcal{S}, w)$. If in addition, $w(S) = 1$ for every $S \in \mathcal{S}$, then this is an instance of rank k unweighted set cover, which we denote as $k\text{-}\mathit{SC}(X, \mathcal{S})$.*

The set cover problem and its variants have been extensively studied in the computer science literature. A simple greedy algorithm was shown in [14,16] to be $1 + \log n$ competitive for SC, where $n = |X|$. It was later shown by [12] that this performance is optimal up to lower order terms. Namely, they showed that a polynomial-time $(1-\epsilon) \log n$ competitive ratio is achievable only if NP has quasi-polynomial-time algorithms.

For k-SC, the greedy algorithm presented in [14,16] is $1 + \log k$ competitive. [9] extended this result to the weighted version of the problem k-WSC. Trevisan showed that this performance is optimal up to lower order terms in [19] by showing that it is NP-hard to approximate k-SC better than $\log k - O(\log \log k)$.

B   Appendix: Proofs for Utility Maximization

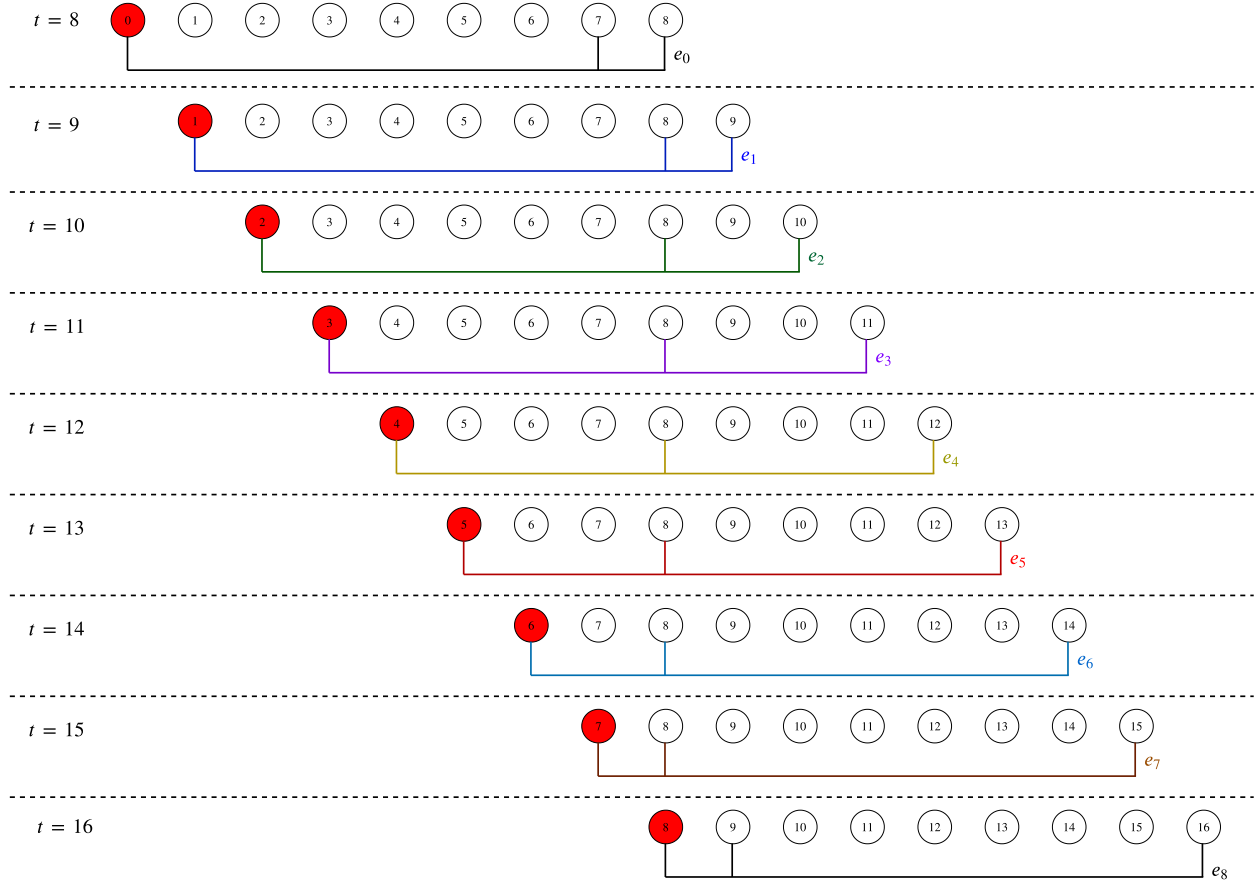## B.1   Visualizing the reduction from d-`ASP` to `Online-`(k, d)`-Max-Matching`

Fig. 3: An example of a shareability hypergraph $G \in \mathcal{G}$ evolving in time. This example has $d = 9$. Vertices arrive one per timestep on the right, and depart one per timestep from the left. A vertex is red during the last timestep it is in the system. For each t, e_t is revealed at time $t + d - 1$, when its last vertex appears. It also becomes critical at this time, since $t \in e_t$ is about to leave the system. Therefore, from timestep $d - 1$ onward, one edge, along with its weight, is revealed each timestep, and can only be included into the matching at that timestep. Furthermore, at most one edge can be in the matching, since all edges contain the node $d - 1 = 8$. Hence, the family $\mathcal{G}$ encodes the d-`ASP` problem.

B.2   Proof of Lemma 1

For any (possibly randomized) algorithm $\mathcal{A}$, we use the random variable $W(\mathcal{A})$ to denote the aptitude of the candidate chosen by $\mathcal{A}$. Let $\mathcal{A}_t$ be any algorithm with the optimal competitive ratio $\rho(\mathcal{A}_t)$ for t-`ASP`. First, $\rho(\mathcal{A}_t) \geq \frac{1}{t}$ by the trivial algorithm $\mathcal{A}_t^*$ which ignores all scores, and commits to one of the t secretaries

uniformly at random. Indeed,

$$\mathbb{E}\left[W(\mathcal{A}_t^*)\right] = \frac{1}{t}\sum_{\tau=1}^{t} w_\tau = \frac{||w||_1}{t} \geq \frac{||w||_\infty}{t} = \frac{\text{OPT}}{t}.$$

Therefore, all we need to show is that $\rho(\mathcal{A}_t) \leq \frac{1}{t}$, which we will do by induction.

Base Case: For $t = 1$ the problem is trivial; there is only one candidate, so hiring the candidate gives the optimal solution and the optimal competitive ratio is $\frac{1}{1} = \frac{1}{t}$.

Induction Step: Suppose that $\rho(\mathcal{A}_t) = \frac{1}{t}$ for some $t \geq 1$. For the $(t+1)$-`ASP`, at the first timestep, $\mathcal{A}_{t+1}$ decides to hire the first candidate with some probability p. If the first candidate is hired, the algorithm collects utility w_1. If the first candidate is not hired, then the algorithm must solve a secretary problem on the remaining t candidates. Therefore,

$$\mathbb{E}\left[W(\mathcal{A}_{t+1})\right] = pw_1 + (1-p)\mathbb{E}\left[W(\mathcal{A}_t)\right].$$

Therefore the competitive ratio of $\mathcal{A}_{t+1}$ is

$$\rho(\mathcal{A}_{t+1}) = \min_{w \in \mathbb{R}_+^{t+1}} \frac{pw_1}{||w||_\infty} + (1-p)\frac{\mathbb{E}[W(\mathcal{A}_t)]}{||w||_\infty} \tag{2}$$

Let $z \in \mathbb{R}_+^t$ be a worst-case instance for $\mathcal{A}_t$ so that $\mathbb{E}[W(\mathcal{A}_t)] = \frac{\text{OPT}}{t}$. Now let $[w_2, w_3, ..., w_{t+1}]^\top = \alpha z$ for some $\alpha > 0$. Sending $w_1 \to \infty$, (2) converges to p. Sending $\alpha \to \infty$, eventually we have $||w||_\infty = ||z||_\infty$, and (2) converges to $(1-p)\rho(\mathcal{A}_t) = \frac{1-p}{t}$. Therefore,

$$\rho(\mathcal{A}_{t+1}) \leq \min\left(p, \frac{1-p}{t}\right).$$

The best choice of p to maximize this upper bound is $p = \frac{1}{t+1}$, which yields $\rho(\mathcal{A}_{t+1}) = \frac{1}{t+1}$. This completes the induction step, and thus completes the proof.

B.3   Proof of Theorem 3

Let M^* be a maximum weight matching for an instance $H = (V, E, w)$ of `Online-`(k, d)`-Max-Matching`. Partition M^* into two disjoint subsets $M_s^* \cup M_\ell^*$ defined via

$$M_s^* := \{e \in M^* : \text{diam}(e) \leq d - k\}$$
$$M_\ell^* := \{e \in M^* : \text{diam}(e) > d - k\},$$

where M_s^* contains all of the edges of M^* whose diameter is at most $d - k$, and M_ℓ^* contain the other edges. First note that $w(M^*) = w(M_s^*) + w(M_\ell^*)$.

Recall the definition of batches: For each $z \in \{0, 1, ..., d-1\}$ and $i \leq \frac{n}{d}$, we define the batch $W_{z,i} := (V_{z,i}, E_{z,i}, w)$ to be a subgraph of H in the following way:

1. $V_{z,i} = \{z + id + t\}_{t=0}^{d-1}$ is the set of nodes that arrive between times $z + id$ and $z + (i+1)d - 1$.
2. $E_{z,i} \subset E$ are all edges whose endpoints are a subset of $V_{z,i}$.

We then use $E_z := \bigcup_i E_{z,i}$.

Now suppose we have a procedure $\mathcal{A}$ that, for every pair z, i, can find a matching $M_{z,i} \subset E_{z,i}$ so that

$$w(M_{z,i}) \geq w(M_\ell^* \cap E_{z,i}) + \frac{1}{k} w(M_s^* \cap E_{z,i}). \tag{3}$$

Next we show that Algorithm 1 using such an $\mathcal{A}$ as a subroutine is $\frac{1}{d}$-competitive for Online-(k,d)-Max-Matching.

$$
\begin{aligned}
\mathbb{E}\left[w(M)\right] &= \sum_{z=0}^{d-1} \mathbb{P}\left[Z=z\right] \mathbb{E}\left[w(M)|Z=z\right] \\
&= \frac{1}{d} \sum_{z=0}^{d-1} \sum_{i=0}^{n/d} w(M_{z,i}) \\
&\overset{(a)}{\geq} \frac{1}{d} \sum_{z=0}^{d-1} \sum_{i=0}^{n/d} \left[w(M_\ell^* \cap E_{z,i}) + \frac{1}{k} w(M_s^* \cap E_{z,i}) \right] \\
&= \frac{1}{d} \sum_{z=0}^{d-1} \left[w(M_\ell^* \cap E_z) + \frac{1}{k} w(M_s^* \cap E_z) \right] \\
&= \frac{1}{d} \sum_{z=0}^{d-1} \sum_{e \in M_\ell^* \cap E_z} w(e) + \frac{1}{kd} \sum_{z=0}^{d-1} \sum_{e \in M_s^* \cap E_z} w(e) \\
&= \frac{1}{d} \sum_{e \in M_\ell^*} w(e) \sum_{z=0}^{d-1} \mathbb{1}_{[e \in E_z]} + \frac{1}{kd} \sum_{e \in M_s^*} w(e) \sum_{z=0}^{d-1} \mathbb{1}_{[e \in E_z]} \\
&\overset{(b)}{\geq} \frac{1}{d} \sum_{e \in M_\ell^*} w(e) + \frac{1}{kd} \sum_{e \in M_s^*} w(e)k = \frac{1}{d} \sum_{e \in M^*} w(e) = \frac{w(M^*)}{d}.
\end{aligned}
\tag{4}
$$

Where (a) is due to condition (3) and (b) is due to Observation 1; edges in M_s^* have diameter at most $d-k$ and edges in M_ℓ^* have diameter at most $d-1$. Therefore if we can find a $\mathcal{A}$ that runs in polynomial-time and satisfies (3) for all pairs z, i, then Algorithm 1 using $\mathcal{A}$ as a subroutine will achieve a competitive ratio of $\frac{1}{d}$ in expectation.

Finding a subroutine that satisfies (3): Let $W_{z,i} = (V_{z,i}, E_{z,i}, w)$ be a batch. Without loss of generality (by shifting the time indices) we will assume that $V_{z,i} = \{0, 1, ..., d-1\}$, i.e. $z = i = 0$. Recalling that Greedy is $\frac{1}{k}$-competitive for k-Max-Matching, for $M = \text{Greedy}(W_{z,i})$ we have

$$
\begin{aligned}
w(M) &\geq \frac{1}{k} w(M^* \cap E_{z,i}) \\
&= \frac{1}{k} w(M_\ell^* \cap E_{z,i}) + \frac{1}{k} w(M_s^* \cap E_{z,i}).
\end{aligned}
\tag{5}
$$

Following the same analysis as (4), running Algorithm 1 with Greedy gives

$$\mathbb{E}\left[w(M)\right] \geq \frac{1}{kd} \left(\sum_{e \in M_\ell^*} w(e) \sum_{z=0}^{d-1} \mathbb{1}_{[e \in E_z]} + \sum_{e \in M_s^*} w(e) \sum_{z=0}^{d-1} \mathbb{1}_{[e \in E_z]} \right) = \frac{1}{kd} w(M_\ell^*) + \frac{1}{d} w(M_s^*). \tag{6}$$

This is not sufficient for our purposes because the $\frac{1}{k}$ coefficient in front of $w(M_\ell^* \cap E_{z,i})$ in (5) leads to a $\frac{1}{kd} w(M_\ell^*)$ contribution to the bound in (6). Note that the reason we are able to tolerate the $\frac{1}{k}$ coefficient on

$w(M_s^* \cap E_{z,i})$ in (3) is because of Observation 1; each $e \in M_s^*$ is in E_z for at least k choices of z, which will cancel the $\frac{1}{k}$ coefficient. However, edges in M_ℓ^* are in E_z for less than k choices of z, and hence we cannot do the same cancellation. Thus, we need to find a way to modify `Greedy` to get a better approximation factor for $w(M_\ell^* \cap E_{z,i})$.

To this end, we have the following observation. Let $L := \{0, 1, .., k-1\}$ and $R := \{d-k, d-k+1, ..., d-1\}$ be the sets of the first and last k vertices of $W_{z,i}$ respectively.

Observation 3 *For $e \in M^* \cap E_{z,i}$, if $e \cap L = \emptyset$ or $e \cap R = \emptyset$, then $diam(e) \leq d-k$, and hence $e \notin M_\ell^*$. Equivalently, for any $e \in M_\ell^*$, we have $e \cap L \neq \emptyset$ and $e \cap R \neq \emptyset$.*

Hence $M_\ell^* \cap E_{z,i}$ is a member of the following set:

$$\mathcal{M}_{L,R}(W_{z,i}) := \{M \text{ a matching in } W_{z,i} : e \cap L \neq \emptyset, e \cap R \neq \emptyset \; \forall \, e \in M\}.$$

In other words, $M_\ell^* \cap E_{z,i}$ is a matching in $W_{z,i}$ that is restricted to L, R where we define restricted matchings below.

Definition 18 (Restricted Matchings). *Given a hypergraph $H = (V, E)$ and two disjoint subsets $L, R \subset V, L \cap R = \emptyset$, a collection of edges $M \subset E$ is a matching is restricted to L, R if*

1. *M is a matching in H.*
2. *For any $e \in M$, $e \cap L \neq \emptyset$ and $e \cap R \neq \emptyset$.*

Finally, we use $\mathcal{M}_{L,R}(H)$ to denote the set of all matchings in H that are restricted to L, R.

Since $|L| = |R| = k$ is a constant, $|\mathcal{M}_{L,R}(W_{z,i})|$ is small (polynomial in d). This motivates the `Depth-k Greedy` algorithm which does the following: For each $M \in \mathcal{M}_{L,R}(W_{z,i})$, extend M to a maximal matching by adding edges via `Greedy`, and then among the $|\mathcal{M}_{L,R}(W_{z,i})|$ resulting maximal matchings, return the one with the largest weight. First, we will show that `Depth-k Greedy` satisfies (3). To this end, for any matching M we use $V(M)$ to denote the vertices that are matched by M. If $M_{z,i}$ is the matching returned by `Depth-k Greedy`, we have

$$w(M_{z,i}) = \max_{M \in \mathcal{M}_{L,R}(W_{z,i})} w(M) + w(\text{Greedy}(V_{z,i} \setminus V(M), E_{z,i}))$$

$$\overset{(a)}{\geq} w(M_\ell^* \cap E_{z,i}) + w(\text{Greedy}(V_{z,i} \setminus V(M_\ell^* \cap E_{z,i}), E_{z,i}))$$

$$\overset{(b)}{\geq} w(M_\ell^* \cap E_{z,i}) + \frac{1}{k} \max_{\substack{M' \text{ s.t.} \\ M' \cup (M_\ell^* \cap E_{z,i}) \text{ is} \\ \text{a matching} \\ \text{in } W_{z,i}}} w(M')$$

$$\overset{(c)}{\geq} w(M_\ell^* \cap E_{z,i}) + \frac{1}{k} w(M_s^* \cap E_{z,i}).$$

Where (a) is due to $M_\ell^* \cap E_{z,i} \in \mathcal{M}_{L,R}(W_{z,i})$, (b) is due to `Greedy` being $\frac{1}{k}$-competitive for k-`Max-Matching`, and (c) is because $(M_\ell^* \cap E_{z,i}) \cup (M_s^* \cap E_{z,i}) = M^* \cap E_{z,i}$ is a matching in $W_{z,i}$. Hence `Depth-k Greedy` satisfies condition (3). To complete the proof, we next show that the runtime of `Depth-k Greedy` is polynomial in the size of the problem instance H.

Runtime of `Depth-k Randomized Batching`: Finally, we need to show that the runtime of `Depth-k Randomized Batching` is polynomial in the size of H. Since $|L| = k$, any matching in $\mathcal{M}_{L,R}(W_{u,i})$ can have at most k hyperedges. From this, we conclude that

$$|\mathcal{M}_{L,R}(W_{z,i})| \leq \sum_{\ell=1}^{k} \binom{|E_{z,i}|}{\ell} \leq k \binom{|E_{z,i}|}{k} \leq k \left(\frac{|E_{z,i}|}{ek}\right)^k$$

For a batch $W_{z,i}$, it takes $c\,|E_{z,i}|\log|E_{z,i}|$ time to sort the edges by weight, for some constant c. For each $M \in \mathcal{M}_{L,R}(W_{z,i})$, given that the edges are already sorted, extending M to a maximal matching via `Greedy` requires $O(k\,|E_{z,i}|)$ time, since for each edge, it takes $O(k)$ time to check if it is disjoint from the current matching. Doing this for all members of $\mathcal{M}_{L,R}(W_{z,i})$ would take

$$c\,|E_{z,i}|\log|E_{z,i}| + |\mathcal{M}_{L,R}(W_{z,i})| \cdot k\,|E| = c\,|E_{z,i}|\log|E_{z,i}| + \frac{k^2}{(ek)^k}\,|E_{z,i}|^{k+1}$$

time. Therefore the expected runtime of `Randomized-Batching`$(\cdot, \mathcal{A})$ with $\mathcal{A} = $ `Depth-k Greedy` is

$$\frac{1}{d}\sum_{z=0}^{d-1}\sum_{i}\left(c\,|E_{z,i}|\log|E_{z,i}| + \frac{k^2}{(ek)^k}\,|E_{z,i}|^{k+1}\right) = O\left(\frac{1}{d}\sum_{z=0}^{d-1}\sum_{i}|E_{z,i}|^{k+1}\right)$$

$$\leq O\left(\frac{1}{d}\sum_{z=0}^{d-1}|E_z|^{k+1}\right)$$

$$\leq O\left(\max_{0\leq z\leq d-1}|E_z|^{k+1}\right)$$

$$\leq O\left(|E|^{k+1}\right).$$

This completes the proof.

C   Appendix: Proofs for Cost Minimization in Ridesharing

C.1   Proof of Lemma 3

Consider the following pair of shareability graphs H_1, H_2, where the arrival order of vertices is A, B, C, D, the cost of each vertex is 1, and the cost of every edge is also 1. The minimum weight matchings are highlighted for each graph.

H_1 H_2
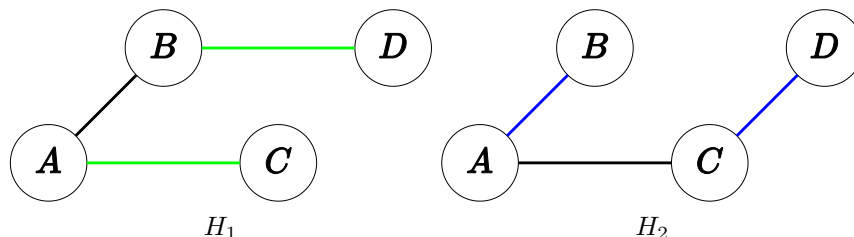
When A becomes critical, there are only 3 possible actions for a deterministic algorithm:

(a) Pair (A, B).
(b) Pair (A, C).
(c) Don't pair A with anything.

Proving a $\frac{3}{2}$ lower bound for deterministic algorithms Action (a) is optimal for H_2, but leads to a non-maximum matching for H_1. On the other hand, action (b) is optimal for H_1 but leads to a non-maximum matching for H_2. A non-maximum matching has either 0 edges and 4 unmatched vertices, which gives a cost of 4, or 1 edge and 2 unmatched vertices, which gives a cost of 3. A maximum matching has 2 edges, and a cost of 2. Therefore, every non-maximum matching has cost at least $\frac{3}{2}$ times as large as an optimal matching. Before D arrives, the instances H_1, H_2 look identical:
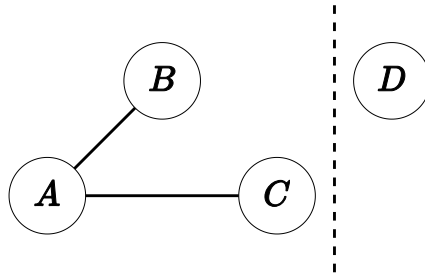
Therefore, if A becomes critical before D arrives, then there is no way to distinguish whether the instance is H_1 or H_2. Using action (a) will lead to a non-maximum matching if the instance is H_1, and using the action (b) leads to a non-maximum matching if the instance is H_2. Action (c) always leads to a non-maximum matching.

Now we review the observations to prove a lower bound of $\frac{3}{2}$ for all deterministic algorithms. First, on the instances $\{H_1, H_2\}$, there are only 3 choices for the first action in any deterministic algorithm. Next, for each of the three possible first actions, there is an instance $H \in \{H_1, H_2\}$ for which this action leads to a non-maximum matching. Finally, for any $H \in \{H_1, H_2\}$, a non-maximum matching has cost at least $\frac{3}{2}$ times as large as the minimum cost matching. Therefore, for any deterministic algorithm, there is an instance for which the algorithm's cost is at least $\frac{3}{2}$ times that of the minimum cost matching.

Proving a $\frac{5}{4}$ lower bound for randomized algorithms We will prove a $\frac{5}{4}$ lower bound using Yao's minimax principle. When A becomes critical, the only actions for deterministic algorithms are $\{(a), (b), (c)\}$. Thus the behavior of any randomized algorithm at the time A becomes critical can be described as a distribution over $\{(a), (b), (c)\}$. Let p_a, p_b, p_c be the probabilities that actions $(a), (b), (c)$ are taken respectively. Since $p_a + p_b + p_c = 1$, at least one of the following must happen:

    Case 1: $p_a + p_c \geq \frac{1}{2}$.
    Case 2: $p_b + p_c \geq \frac{1}{2}$.

Note that actions $(a), (b)$ lead to non-maximum matchings for the instance H_1, and actions $(b), (c)$ lead to non-maximum matchings for the instance H_2. If Case 1 is true, then on the instance H_1, and letting M be the matching produced by the randomized algorithm,

$$\mathbb{E}\left[c(M)\right] = p_b \mathbb{E}\left[M | \text{first action is } (b)\right] + (p_a + p_c)\mathbb{E}\left[M | \text{first action } \in \{(a), (c)\}\right]$$
$$\geq p_b \cdot 2 + (p_a + p_c)3$$
$$\geq 1 + \frac{3}{2} = \frac{5}{2}.$$

Where the first inequality is because 2 is the minimum cost matching, so in particular it is a lower bound for $\mathbb{E}\left[M | \text{first action is } (b)\right]$, and because taking action (a) or (c) leads to a non-maximum matching in H_1, which has cost at least 3. Therefore, $\frac{c(M)}{c(M^*)} \geq \frac{5}{2} \div 2 = \frac{5}{4}$. The analysis if Case 2 is true is analogous; the only difference is we apply the analysis to the H_2 instance instead.

C.2   Proof of Theorem 5

For an instance $H = (V, E, w)$, let M^* be a minimum weight matching for H, and let U^* denote the set of vertices that are unmatched under M^*. For each edge $(i, j) \in E$, define $\theta_{ij} := \frac{w(i,j)}{w_i + w_j}$. For $\theta \in \left[\frac{1}{2}, 1\right]$, define

$$E_\theta := \{(i, j) \in E : \theta_{ij} \leq \theta\}.$$

Let M be the matching produced by `Risk-Threshold`(H, θ), and let U be the set of vertices that are unmatched under M. Note that M is a maximal matching in (V, E_θ, w). Define v^*, v in the following way:

$$v_i^* := \begin{cases} w_i & \text{if } i \in U^* \\ \theta_{ij} w_i & \text{if } (i, j) \in M^* \end{cases} \quad \text{and } v_i := \begin{cases} w_i & \text{if } i \in U \\ \theta_{ij} w_i & \text{if } (i, j) \in M. \end{cases}$$

Note that by construction, $w(M^*) = \sum_{i \in V} v_i^*$ and $w(M) = \sum_{i \in V} v_i$, so v_i, v_i^* denote the contribution of vertex i to the total cost under matchings M, M^* respectively.

Since M, M^* are matchings, their union $M \cup M^*$ is a disjoint union of cycles and paths, i.e. every connected component of $(V, M \cup M^*, w)$ falls into one of the three following categories:

1. Vertices of degree 0.
2. Cycles.
3. Maximal Paths.

Where the definition of a maximal path is the following.

Definition 19 (Maximal Paths). *We say a path P is maximal in a graph $G = (V, E, w)$ if it is not a strict subset of another path.*

To prove Theorem 5, we will prove the following claim.

Observation 4 *For every connected component in $M \cup M^*$, the cost of the vertices in this connected component under M is no more than $\frac{3}{2}$ times the cost of those same vertices under M^*.*

Once we establish this, Theorem 5 follows by the following implications. Suppose

$$M \cup M^* = \left(\bigcup_{j=1}^{p} C_j \right) \bigcup \left(\bigcup_{\ell=1}^{q} P_\ell \right)$$

is the partition of $M \cup M^*$ into cycles $\{C_j\}_{j=1}^{p}$ and maximal paths $\{P_\ell\}_{\ell=1}^{q}$. If for a collection of edges E' we use $V(E')$ to denote the set of vertices that are touched by edges in E', we have

$$V = \left(\bigcup_{j=1}^{p} V(C_j) \right) \bigcup \left(\bigcup_{\ell=1}^{q} V(P_\ell) \right) \bigcup (U \cap U^*)$$

where recall that $U \cap U^*$ denotes the vertices that are unmatched under both M and M^*. Finally we can see that:

$$
\begin{aligned}
w(M) &= \sum_{i \in V} v_i \\
&= \sum_{j=1}^{p} \sum_{i \in V(C_j)} v_i + \sum_{\ell=1}^{q} \sum_{i \in V(P_\ell)} v_i + \sum_{i \in U \cap U^*} v_i \\
&\overset{(a)}{\leq} \sum_{j=1}^{p} \frac{3}{2} \sum_{i \in V(C_j)} v_i^* + \sum_{\ell=1}^{q} \frac{3}{2} \sum_{i \in V(P_\ell)} v_i^* + \sum_{i \in U \cap U^*} v_i^* \\
&\leq \frac{3}{2} \left(\sum_{j=1}^{p} \sum_{i \in V(C_j)} v_i^* + \sum_{\ell=1}^{q} \sum_{i \in V(P_\ell)} v_i^* + \sum_{i \in U \cap U^*} v_i^* \right) \\
&= \frac{3}{2} \sum_{i \in V} v_i^* = \frac{3}{2} w(M^*).
\end{aligned}
$$

Where (a) is due to Observation 4. We will prove claim 3 by examining each of the 3 types of connected components of $(V, M \cup M^*, w)$.

Vertices of degree 0 If a vertex i has degree zero in the graph $(V, M \cup M^*)$, this means it is unmatched under both M and M^*, meaning that $v_i = w_i$ and $v_i^* = w_i$, therefore the cost contribution of i to M is the same as its cost contribution to M^*.

Cycles Note that any cycle in $(V, M \cup M^*, w)$ must have even length. A cycle having odd length would imply that there are two edges from the same matching that share a vertex. So consider an even cycle on vertices $\{1, 2, ..., 2\ell\}$ where the edges $\{(2k+1, 2k+2)\}_{k=0}^{\ell-1}$ belong to M^* and the edges $\{(2k, 2k+1)\}_{k=1}^{\ell}$ belong to M. Since every vertex in a cycle has degree 2, this means that each vertex in the cycle is matched in both M and M^*. We thus see that

$$\sum_{i=1}^{2\ell} v_i = \sum_{k=1}^{\ell} w(2k, 2k+1) \overset{(a)}{\leq} \sum_{k=1}^{\ell} \theta w_{2k} + \theta w_{2k+1}$$

$$\overset{(b)}{\leq} \sum_{k=1}^{\ell} 2\theta v_{2k}^* + 2\theta v_{2k+1}^* = 2\theta \sum_{i=1}^{2\ell} v_i^*.$$

Where (a) is due to the fact that $w(i, j) = \theta_{ij}(w_i + w_j)$, $\theta_e \leq \theta$ for every $e \in E_\theta$ and $M \subset E_\theta$. We get (b) because $\theta \geq \frac{1}{2}$ due to Assumption 1. We conclude this part by noting that for $\theta = \frac{2}{3}$, $2\theta = \frac{4}{3} \leq \frac{3}{2}$.

Paths To analyze the cost contributions of maximal paths, we will use the following definition of restricted cost.

Definition 20 (Restricted Cost). *Given a graph $G = (V, E, w)$, an edge sets $E' \subset E$, and a matching M, we define $w(M; E')$ to be the cost of $M \cap E'$ in the subgraph $(V(E'), E', w)$.*

To prove Observation 4 for paths, we will use the following lemma.

Lemma 5 (Path cost bound). *For any path $P \subset (V, M \cup M^*, w)$, we have:*

$$w(M; P) \leq \frac{3}{2} w(M^*; P).$$

See Section C.3 for a proof of Lemma 5. Once we establish this lemma, we can establish Observation 4 for maximal paths using the following observation.

Observation 5 *If P is a maximal path, $\sum_{i \in V(P)} v_i = w(M; P)$ and $\sum_{i \in V(P)} v_i^* = w(M^*; P)$.*

Proof (proof of Observation 5). Note that $\sum_{i \in V(P)} v_i \neq w(M; P)$ only if there is some vertex $i \in V(P)$ so that either

1. i is matched under $M \cap P$ in $(V(P), P, w)$ but is not matched under M in (V, E, w), or
2. i is not matched under $M \cap P$ in $(V(P), P, w)$ but is matched under M in (V, E, w).

The first situation can never happen because $M \cap P \subset M$, so every vertex matched by $M \cap P$ is also matched by M. If the second situation happens, this means that the degree of i is 1 in $(V(P), P, w)$ and there exists some $i' \in V \setminus V(P)$ with $(i, i') \in M$. But this implies that $P \cup \{(i, i')\}$ is a path in $M \cup M^*$, which contradicts the assumption that P is maximal. The proof for M^* is analogous.

Using Observation 5, we can prove Observation 4 for maximal paths P via

$$\sum_{i \in V(P)} v_i = w(M; P) \leq \frac{3}{2} w(M^*; P) = \sum_{i \in V(P)} v_i^*.$$

This completes the proof for Theorem 5.

C.3   Proof of Lemma 5

We partition the set of maximal paths in $(V, M \cup M^*, w)$ into four categories, and treat each of the categories separately.

1. Length 1 paths.
2. Type 1 paths.
3. Even length paths.
4. Type 2 paths of length at least 3.

Where type 1 and type 2 paths are defined as follows:

Definition 21 (Path Types). *A path in $(V, M \cup M^*, w)$ is type 1 if it has an odd length, and contains more edges from M than from M^*. A path is type 2 if it has odd length and contains more edges from M^* than from M.*

Proving Lemma 5 for paths in categories 1 and 2 are easy, so we will do this directly. For Categories 2 and 3, we will prove the lemma by induction on the path length.

Maximal paths of length 1 Consider a path of length 1 in $M \cup M^*$, i.e. an edge $e = (i, j) \in E$. There are two cases to consider: 1) $e \in M$ and 2) $e \in M^* \setminus M$.

Case 1: First note that either $(i, j) \in M^*$, or both i, j are unmatched under M^*. To see this, if $(i, j') \in M^*$ for some $j \neq j'$, then the path $\{(j, i)\}$ would not be maximal since in particular the path $\{(j, i), (i, j')\} \subset M \cup M^*$. From this we can conclude that that $v_i^* \geq \theta_{ij} w_i, v_j^* \geq \theta_{ij} w_j$. Finally,

$$v_i + v_j = w(i, j) = \theta_{ij} w_i + \theta_{ij} w_j \leq v_i^* + v_j^*.$$

Case 2: Now we consider the case where $e \in M^* \setminus M$. We argue by contradiction that $e \notin E_\theta$. Suppose $e \in E_\theta$. We can immediately say that neither i nor j matched under M, because if any of the vertices of e are matched under M, then this path would have more than 1 edge. Since none of the vertices of e are matched in M, then $M \cup \{e\}$ is a matching in (V, E_θ). However, this contradicts the fact that M is a maximal matching in (V, E_θ). Therefore if $e \in M^*$, it must be the case that $e \notin E_\theta$. From this we see that:

$$v_i + v_j \leq w_i + w_j = \frac{w(i, j)}{\theta_{ij}} = \frac{v_i^* + v_j^*}{\theta_{ij}} \leq \frac{v_i^* + v_j^*}{\theta} = \frac{3}{2}(v_i^* + v_j^*).$$

Where $\theta_{ij} \geq \theta$ is because $e \notin E_\theta$, and the last equality is due to $\theta = \frac{2}{3}$.

Type 1 Paths Let P be a type 1 path on the vertices $\{1, 2, ..., 2\ell\}$ where $\{(2k + 1, 2k + 2)\}_{k=0}^{\ell-1} \in M$ and $\{(2k, 2k + 1)\}_{k=1}^{\ell-1} \in M^*$. Since P contains more edges from M than M^*, the set of vertices in $V(P)$ matched by $M^* \cap P$ is a subset of vertices matched by $M \cap P$. Concretely, note that M matches all vertices $\{1, 2, ..., 2\ell\}$, whereas M^* matches all of the vertices except $\{1, 2\ell\}$. We thus see that

$$\sum_{k=1}^{2\ell} v_k = v_1 + v_{2\ell} + \sum_{k=2}^{2\ell-1} v_k$$

$$\leq v_1^* + v_{2\ell}^* + \sum_{k=2}^{2\ell-1} \theta w_k$$

$$\leq v_1^* + v_{2\ell}^* + \sum_{k=2}^{2\ell-1} 2\theta v_k^*$$

$$\leq 2\theta \sum_{k=1}^{2\ell} v_k^*.$$

We conclude by noting that $\theta = \frac{2}{3}$ implies that $2\theta = \frac{4}{3} \leq \frac{3}{2}$.

Even length paths: Base case We argue by induction on the length of even paths. The base case is when the path $P = \{(a,b), (b,c)\}$ has 2 edges where $(a,b) \in M, (b,c) \in M^*$. We will use the following fact:

Fact 1 (Fractions and convex combinations) *If $x > y > 0$ and $z > 0$, then $\frac{x+z}{y+z} \leq \frac{x}{y}$.*

Proof (Proof of Fact 1). This follows from the following algebra:

$$
\begin{aligned}
x > y > 0, z > 0 &\implies xz > yz \\
&\implies xy + xz > xy + yz \\
&\implies x(y+z) > y(x+z) \\
&\implies \frac{x}{y} > \frac{x+z}{y+z}.
\end{aligned}
$$

With this, observe that

$$
\frac{v_a + v_b + v_c}{v_a^* + v_b^* + v_c^*} = \frac{w(a,b) + w_c}{w_a + w(b,c)} \leq \frac{w(a,b) + w_c}{w_a + \max(w_b, w_c)} \tag{7}
$$

If the numerator in (7) is smaller than the denominator, we are done. Therefore from this point onward assume that the numerator is larger. There are two cases. If $w_c \leq w_b$, then we apply this bound to the numerator. If $w_b < w_c$, then we apply Fact 1 with $z = w_c - w_b$. In either case, we get

$$
\begin{aligned}
\frac{v_a + v_b + v_c}{v_a^* + v_b^* + v_c^*} &\leq \frac{w(a,b) + w_b}{w_a + w_b} \\
&= \frac{\theta_{ab} w_a + \theta_{ab} w_b + w_b}{w_a + w_b} \\
&= 1 + \frac{(\theta_{ab} - 1) w_a + \theta_{ab} w_b}{w_a + w_b} \\
&\overset{(a)}{\leq} 1 + \frac{(\theta_{ab} - 1)\left(\frac{1-\theta_{ab}}{\theta_{ab}}\right) w_b + \theta_{ab} w_b}{\left(\frac{1-\theta_{ab}}{\theta_{ab}}\right) w_b + w_b} \\
&= 1 + \frac{-(1-\theta_{ab})^2 w_b + \theta_{ab}^2 w_b}{w_b} \\
&= 1 + \theta_{ab}^2 - \left(1 - 2\theta_{ab} + \theta_{ab}^2\right) \\
&= 2\theta_{ab} \leq 2\theta = \frac{4}{3} \leq \frac{3}{2}.
\end{aligned}
$$

We get (a) because the numerator is a decreasing function of w_a and the denominator is a increasing function of w_a, which means the bound is maximized when w_a is minimized. Furthermore, $\theta_{ab} w_a + \theta_{ab} w_b = w(a,b) \geq w_b$ implies that $w_a \geq \left(\frac{1-\theta_{ab}}{\theta_{ab}}\right) w_b$. This establishes the base case for even length paths.

Induction Step on the Path Length Since we already bounded the cost contribution of type 1 paths in Section C.3, in this section we focus on paths that are not type 1. In particular, we focus on paths that have at least as many edges from M^* as they do from M. Note that any such path has the following property: Either the first edge in the path or the last edge in the path belong to M^*. In this section, we will without loss of generality assume that the first edge of the path belongs to M^*. If the first edge belongs to M, then

it must be the case that the last edge belongs to M^*, and we can just reverse the ordering of the edges.

Now we begin the induction step. Assume that for all paths P of length n whose vertices are $\{0, 1, ..., n\}$ and whose first edge belongs to M^*, we have

$$w(M; P) \leq \frac{3}{2} w(M^*; P).$$

Now let P' be a length $n + 2$ path whose vertices are $\{0, 1, 2, ..., n + 1, n + 2\}$ and whose first edge belongs to M^*. Define P to be the path containing the edges $\{(i, i + 1)\}_{i=2}^{n+1}$. Note that the first edge of P belongs to M^*, and the length of P is n. We can write

$$w(M; P') = \sum_{i=0}^{n+2} v_i = v_0 + v_1 + (v_2 - w_2) + w_2 + \sum_{i=3}^{n+2} v_i$$
$$= v_0 + v_1 + \underbrace{(v_2 - w_2)}_{\text{term 1}} + w(M; P)$$

Where term 1 is because of the following. Vertex 2 is unmatched in $P \cap M$, so it contributes a cost of w_2. However, $(1, 2) \in (P' \setminus P) \cap M$ reduces the cost contribution of vertex 2 from w_2 to v_2. Similarly, we can write

$$w(M^*; P') = v_0^* + v_1^* + \sum_{i=2}^{n+2} v_i^*$$
$$= v_0^* + v_1^* + w(M^*; P).$$

Since P is a length n path whose first edge belongs to M^*, we can use the induction hypothesis to conclude that

$$w(M; P) \leq \frac{3}{2} w(M^*; P).$$

Next, we will show that

$$\frac{v_0 + v_1 + (v_2 - w_2)}{v_0^* + v_1^*} \leq \frac{3}{2}. \tag{8}$$

Once we show (8), we can conclude the proof as follows:

$$w(M; P') = v_0 + v_1 + (v_2 - w_2) + w(M; P)$$
$$\leq \frac{3}{2} (v_0^* + v_1^*) + \frac{3}{2} w(M^*; P)$$
$$= \frac{3}{2} w(M^*; P')$$

Where the inequality is obtained by applying (8) and the induction hypothesis on P. Therefore, all that remains is to prove (8). Observe that

$$\frac{v_0 + v_1 + (v_2 - w_2)}{v_0^* + v_1^*} = \frac{w_0 + \theta_{12} w_1 + (\theta_{12} - 1) w_2}{\theta_{01} w_0 + \theta_{01} w_1} \leq \frac{w_0 + \theta_{12} w_1 + (\theta_{12} - 1) w_2}{\max (w_0, w_1)}.$$

We now use an approach similar to what was done in Section C.3. If the numerator is smaller than the denominator, then we are done. If it is not, then there are two cases. If $w_0 \leq w_1$, we apply this bound to the

numerator. If $w_1 < w_0$, then we apply Fact 1 with $z = w_0 - w_1$. In both cases, we get

$$
\frac{v_0 + v_1 + (v_2 - w_2)}{v_0^* + v_1^*} \leq \frac{w_1 + \theta_{12}w_1 + (\theta_{12} - 1)w_2}{w_1}
$$

$$
\overset{(a)}{\leq} \frac{w_1 + \theta_{12}w_1 + (\theta_{12} - 1)\left(\frac{1-\theta_{12}}{\theta_{12}}\right)w_1}{w_1}
$$

$$
= \frac{\theta_{12} + \theta_{12}^2 - (1-\theta_{12})^2}{\theta_{12}} = \frac{\theta_{12} + \theta_{12}^2 - (1 - 2\theta_{12} + \theta_{12}^2)}{\theta_{12}}
$$

$$
= \frac{3\theta_{12} - 1}{\theta_{12}}
$$

$$
\overset{(b)}{\leq} \frac{3\theta - 1}{\theta} = \frac{3}{2}.
$$

Where (a) is because the numerator is a decreasing function of w_2, so the bound is maximized when w_2 is minimized. Next, $\theta_{12}w_1 + \theta_{12}w_2 = w(1,2) \geq w_1$ implies that $w_2 \geq \left(\frac{1-\theta_{12}}{\theta_{12}}\right)w_1$. Finally, (b) is because $\theta_{12} \leq \theta$ since $(1,2) \in M \subset E_\theta$, and the function $\frac{3x-1}{x}$ is increasing on $[\frac{1}{2}, \frac{2}{3}]$.

Base case for type 2 paths of length at least 3 Our base case is a path $P = \{(a,b), (b,c), (c,d)\}$ where $(a,b), (c,d) \in M^*$ and $(b,c) \in M$. There are two cases we need to consider:

1. At most one of (a,b) or (c,d) is in E_θ.
2. $(a,b), (c,d) \in E_\theta$.

Remark 1. One may be tempted to use the result for type 2 paths of length 1 from Section C.3 together with the induction step in Section C.3 to establish a result for all type 2 paths. This, however, will not work. In section C.3 when discussing type 2 paths of length 1, we used the fact that the edge $e \in M^*$ cannot be in E_θ. Therefore applying the result from Section C.3 together with the induction step in Section C.3 will only prove the result for type 2 paths that contain at least one edge from $E \setminus E_\theta$. There can in general be type 2 paths whose edges are all contained in E_θ. Case 2 of this section addresses such paths. Once we address case 2, then we can apply the induction argument from Section C.3 to establish the desired result for all type 2 paths.

We now prove the desired result for each of the two cases. This section, together with the induction argument from Section C.3 establishes the desired result for all type 2 paths.

Case 1: In this case, either $(a,b) \notin E_\theta$ or $(c,d) \notin E_\theta$. Without loss of generality assume that $(a,b) \notin E_\theta$ (otherwise simply relabel the vertices). Since $(a,b) \notin E_\theta$, we have $\frac{w(a,b)}{w_a + w_b} > \theta$. We then observe

$$
v_a + v_b \leq w_a + w_b \leq \frac{1}{\theta}w(i,j) \leq \frac{v_a^* + v_j^*}{\theta} = \frac{3}{2}(v_a^* + v_j^*).
$$

We now have a path $P_0 = \{(a,b)\}$ where $w(M; P_0)$ is no more than $\frac{3}{2}w(M^*; P_0)$. Applying the induction argument from Section C.3, we can conclude that

$$
\frac{w(M; P)}{w(M^*; P)} \leq \frac{3}{2}.
$$

This completes the proof of Case 1.

Case 2: In this case we have a path $P = \{(a,b),(b,c),(c,d)\}$ where $(a,b),(c,d) \in M^* \cap E_\theta$ and $(b,c) \in M$. We first show that the first vertex among $\{a,b,c,d\}$ to become critical must be either b or c. Note that if a were the first vertex among $\{a,b,c,d\}$ to become critical, then in particular a becomes critical before either b,c becomes critical. Since $(b,c) \in M$, no vertex that becomes critical before b and c choose to match with either of them. Therefore, by the time a becomes critical, both b,c are available, meaning that a should be matched, since in particular it can match to b. However, since a is unmatched in M, it cannot be the case that a is the first vertex among $\{a,b,c,d\}$ to become critical. The same argument concludes that d cannot be the first to become critical. This means that the first among these vertices to become critical is either b or c. Thus without loss of generality assume that b is the first vertex to become critical. When b becomes critical, both a,c are available. Since b chooses to match with c, we can conclude that $\theta_{bc} \le \theta_{ab}$.

We now observe that

$$\frac{w(M;P)}{w(M^*;P)} = \frac{w_a + w(b,c) + w_d}{w(a,b) + w(c,d)}$$

$$= \frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_d}{\theta_{ab}w_a + \theta_{ab}w_b + \theta_{cd}w_c + \theta_{cd}w_d}$$

$$\le \frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_d}{\theta_{ab}w_a + \theta_{ab}w_b + \max(w_c, w_d)}$$

If the numerator is smaller than the denominator, then we are done. So from now on we assume that the numerator is larger. There are two cases to consider. If $w_c > w_d$, then we can bound the numerator by $w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_c$. If $w_d > w_c$, then applying Fact 1 with $z = w_d - w_c$, we get

$$\frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_d}{\theta_{ab}w_a + \theta_{ab}w_b + \max(w_c, w_d)} = \frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_c + (w_d - w_c)}{\theta_{ab}w_a + \theta_{ab}w_b + w_c + (w_d - w_c)} \le \frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}.$$

So in either case, we have

$$\frac{w(M;P)}{w(M^*;P)} \le \frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

There are two cases to consider. If $w_a \ge w_c$, then

$$\frac{w(M;P)}{w(M^*;P)} \overset{(a)}{\le} \frac{w_a + \theta_{ab}w_b + \theta_{ab}w_c + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

$$= 1 + \frac{(1 - \theta_{ab})w_a + \theta_{ab}w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

$$\le 1 + \frac{(1 - \theta_{ab})w_a + \theta_{ab}w_c}{w_a + w_c}$$

$$= 1 + (1 - \theta_{ab})\frac{w_a}{w_a + w_c} + \theta_{ab}\frac{w_c}{w_a + w_c}$$

$$\le 1 + \frac{1 - \theta_{ab}}{2} + \frac{\theta_{ab}}{2} = \frac{3}{2}.$$

Where (a) is due to $\theta_{bc} \le \theta_{ab}$. The other case is that $w_c > w_a$. In this case, we have

$$\frac{w(M;P)}{w(M^*;P)} = \frac{w_a + \theta_{bc}w_b + \theta_{bc}w_c + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c} \overset{(a)}{\le} \frac{w_a + \theta_{ab}w_b + \theta_{bc}w_c + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

$$= 1 + \frac{(1 - \theta_{ab})w_a + \theta_{bc}w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

$$= 1 + \underbrace{\frac{(1 - \theta_{ab})w_a}{\theta_{ab}w_a}\frac{\theta_{ab}w_a}{\theta_{ab}w_a + \theta_{ab}w_b + w_c} + \frac{\theta_{bc}w_c}{\theta_{ab}w_b + w_c}\frac{\theta_{ab}w_b + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}}_{\text{term } 1}$$

Where (a) is obtained by bounding $\theta_{bc}w_b \leq \theta_{ab}w_b$. We will show that term 1 is maximized when w_b takes its smallest possible value. To this end, note that term 1 is a convex combination of $\frac{1-\theta_{ab}}{\theta_{ab}}$ and $\frac{\theta_{ab}w_c}{\theta_{ab}w_b+w_c}$. Since $w_b \geq \frac{1-\theta_{bc}}{\theta_{bc}}w_c$ and $\theta_{bc} \leq \theta_{ab}$, we have

$$\frac{\theta_{bc}w_c}{\theta_{ab}w_b + w_c} \leq \frac{\theta_{bc}w_c}{\theta_{bc}w_b + w_c} \leq \frac{\theta_{bc}w_c}{\theta_{bc}\frac{1-\theta_{bc}}{\theta_{bc}}w_c + w_c} = \frac{\theta_{bc}}{2 - \theta_{bc}} \leq \frac{\theta_{ab}}{2 - \theta_{ab}}. \tag{9}$$

Next, we note that since $\theta_{ab} \leq \theta = \frac{2}{3}$, we have $\frac{\theta_{ab}}{2-\theta_{ab}} \leq \frac{1-\theta_{ab}}{\theta_{ab}}$. So $\frac{1-\theta_{ab}}{\theta_{ab}}$ is larger than $\frac{\theta_{ab}w_c}{\theta_{ab}w_b+w_c}$ for any valid values[5] of w_a, w_b, w_c. Since $\frac{\theta_{ab}w_a}{\theta_{ab}w_a+\theta_{ab}w_b+w_c}$ and $\frac{\theta_{ab}w_c}{\theta_{ab}w_b+w_c}$ are decreasing functions of w_b, we see that for any valid values of w_a, w_c, term 1 is maximized when w_b takes its smallest possible value, that is to say $w_b = \max\left(\frac{1-\theta_{ab}}{\theta_{ab}}w_a, \frac{1-\theta_{bc}}{\theta_{bc}}w_c\right)$. From this we get:

$$\frac{w(M;P)}{w(M^*;P)} = 1 + \frac{(1-\theta_{ab})w_a}{\theta_{ab}w_a} \frac{\theta_{ab}w_a}{\theta_{ab}w_a + \theta_{ab}w_b + w_c} + \frac{\theta_{bc}w_c}{\theta_{ab}w_b + w_c} \frac{\theta_{ab}w_b + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

$$\overset{(a)}{\leq} 1 + \frac{(1-\theta_{ab})w_a}{\theta_{ab}w_a} \frac{\theta_{ab}w_a}{\theta_{ab}w_a + \theta_{ab}w_b + w_c} + \frac{\theta_{ab}}{2-\theta_{ab}} \frac{\theta_{ab}w_b + w_c}{\theta_{ab}w_a + \theta_{ab}w_b + w_c}$$

$$\overset{(b)}{\leq} 1 + \frac{(1-\theta_{ab})}{\theta_{ab}} \frac{\theta_{ab}w_a}{\theta_{ab}w_a + \theta_{ab}\left(\frac{1-\theta_{ab}}{\theta_{ab}}\right)w_a + w_c} + \frac{\theta_{ab}}{2-\theta_{ab}}\left(1 - \frac{\theta_{ab}w_a}{\theta_{ab}w_a + \theta_{ab}\left(\frac{1-\theta_{ab}}{\theta_{ab}}\right)w_a + w_c}\right)$$

$$= 1 + \frac{(1-\theta_{ab})}{\theta_{ab}} \frac{\theta_{ab}w_a}{w_a + w_c} + \frac{\theta_{ab}}{2-\theta_{ab}}\left(1 - \frac{\theta_{ab}w_a}{w_a + w_c}\right)$$

$$\overset{(c)}{\leq} 1 + \frac{(1-\theta_{ab})}{\theta_{ab}} \frac{\theta_{ab}w_a}{w_a + w_a} + \frac{\theta_{ab}}{2-\theta_{ab}}\left(1 - \frac{\theta_{ab}w_a}{w_a + w_a}\right)$$

$$= 1 + \frac{1-\theta_{ab}}{2} + \frac{\theta_{ab}}{2-\theta_{ab}}\left(\frac{2-\theta_{ab}}{2}\right)$$

$$= 1 + \frac{1-\theta_{ab}}{2} + \frac{\theta_{ab}}{2} = \frac{3}{2}.$$

We get (a) from the chain of inequalities in (9). We get (b) and (c) by increasing the weight of $\frac{1-\theta_{ab}}{\theta_{ab}}$ in the convex combination and reduce the weight of $\frac{\theta_{ab}}{2-\theta_{ab}}$ by the same amount. This leads to an upper bound since $\frac{1-\theta_{ab}}{\theta_{ab}} \geq \frac{\theta_{ab}}{2-\theta_{ab}}$. This completes the base case for case 2.

C.4   Proof of Theorem 6

The proof proceeds in the following steps. First, we establish an equivalence between the offline version of the problem k-Min-Matching and the weighted set cover problem through Observation 6. Using this equivalence, we use existing results from set cover to prove that it is NP-hard to approximate Online-(k,d)-Min-Matching better than $\log k - O(\log\log k)$. This is done in Lemma 6. We conclude by presenting a randomized batching scheme that can convert any algorithm for weighted set cover into an algorithm for Online-(k,d)-Min-Matching with a $2 - \frac{1}{d}$ overhead in the competitive ratio. This allows us to establish both the optimal competitive ratio and optimal polynomial-time competitive ratio up to a factor of $2 - \frac{1}{d}$.

We will use $\mathtt{SC}(X,\mathcal{S})$ to refer to the set cover problem with universe X and a collection of sets $\mathcal{S}$. We use $\mathtt{WSC}(X,\mathcal{S},w)$ to denote the weighted version of the problem where $w : \mathcal{S} \to \mathbb{R}_+$. We use k-$\mathtt{SC}$ and k-$\mathtt{WSC}$ to denote instances where $|S| \leq k$ for all $S \in \mathcal{S}$. See Appendix A.2 for further details and existing results for set cover.

---

[5] By this we simply mean $\theta_{bc} \leq \theta_{ab} \leq \theta$.

Observation 6 (Equivalence between k-Min-Matching and Weighted Set Cover) *Due to Assumption 1, k-Min-Matching on an instance $H = (V, E, w)$ is equivalent to k-WSC(V, E, w)*

Proof. First we will show that from every matching M for Online-(k, d)-Min-Matching, we can construct a cover with the same cost for k-WSC(V, E, w). Given any matching M in H, let U be the set of vertices that are unmatched under M. Defining $\mathcal{U} := \cup_{i \in U} \{i\}$, note that $\mathcal{E} := M \cup \mathcal{U}$ covers V and

$$\sum_{S \in \mathcal{E}} w(E) = \sum_{S \in M} w(S) + \sum_{S \in \mathcal{U}} w(S) = \sum_{e \in M} w(e) + \sum_{i \in U} w(\{i\}).$$

Next, from every cover $\mathcal{E}$ of V, we construct a matching in H whose cost is at most the cost of $\mathcal{E}$ for k-WSC(V, E, w). Given a cover $\mathcal{E} := \{S_1, S_2, ..., S_m\}$, consider the edges $e_i := S_i \setminus \left(\cup_{j=1}^{i-1} S_j\right)$ and the set $M := \{e_i\}_{i=1}^m$. By construction, $\cup_{i=1}^m S_i = \cup_{i=1}^m e_i$, so in particular M is also a cover of V. Furthermore, since $e_i \subset S_i$, by Assumption 1, we have $w(e_i) \leq w(S_i)$, so the cost of M is no greater than the cost of $\mathcal{E}$. Finally, note that by construction, $e_i \cap e_j = \emptyset$ if $i \neq j$. Therefore M is a matching in H whose cost for k-Min-Matching is no more than the cost of $\mathcal{E}$ for k-WSC. □

This result allows us to leverage hardness results from set cover [19] to prove hardness results for Online-(k, d)-Min-Matching.

Lemma 6 (Computational Complexity of Online-(k, d)-Min-Matching). *Under Assumption 1, it is NP-hard to approximate Online-(k, d)-Min-Matching within $\log k - O(\log \log k)$.*

Proof. Observation 6 shows an equivalence between k-Min-Matching and k-WSC$(X, \mathcal{S}, w)$ with the additional condition that $\mathcal{S}$ is downward closed, i.e. $S \in \mathcal{S}$ implies that $S' \in \mathcal{S}$ for any subset S' of S, and $w(S') \leq w(S)$. This additional condition is due to Assumption 1. Let $\overline{\mathcal{S}}$ denote the downward closure of a collection of sets $\mathcal{S}$. We will show in every instance of k-SC, replacing $\mathcal{S}$ with $\overline{\mathcal{S}}$ does not change the optimal solution. This proves that all hardness results for k-SC also hold for k-Min-Matching. Since k-Min-Matching is a special case of Online-(k, d)-Min-Matching, this allows us to inherit hardness results from k-SC to Online-(k, d)-Min-Matching.

Now we prove that k-SC$(X, \mathcal{S})$ and k-SC$(X, \overline{\mathcal{S}})$ have the same optimal solution. Let $\mathcal{E}$ be any minimum cardinality set cover for k-SC$(X, \overline{\mathcal{S}})$. For any $S \in \overline{\mathcal{S}} \setminus \mathcal{S}$, there exists some $T_S \in \mathcal{S}$ with $S \subset T_S$. Therefore, by replacing every $S \in \mathcal{E}$ with the corresponding T_S, we obtain a set cover for k-SC$(X, \mathcal{S})$. Thus k-SC$(X, \overline{\mathcal{S}})$ cannot have a strictly smaller optimal solution than k-SC$(X, \mathcal{S})$. Hence requiring $\mathcal{S}$ to be downward closed does not make k-SC easier. Trevisan proved in [19] that it is NP-hard to approximate k-SC better than $\log k - O(\log \log k)$. Therefore, by the above reasoning, it is also NP-hard to approximate Online-(k, d)-Min-Matching better than $\log k - O(\log \log k)$. □

We now leverage the relation to set cover to design polynomial time algorithms for Online-(k, d)-Min-Matching. In Section 4.2 we presented Randomized-Batching$(\cdot, \mathcal{A})$ for Online-(k, d)-Max-Matching, which is described by Algorithm 1. We will re-purpose this algorithm for Online-(k, d)-Min-Matching. Recall that in Randomized-Batching, every d timesteps we use a k-Max-Matching algorithm $\mathcal{A}$ to compute a matching of the most recently arrived d vertices. We now prove the main result of Theorem 6:

Main Result: Let $\mathcal{A}$ be an algorithm for k-WSC with competitive ratio $\rho(\mathcal{A})$. Then Randomized-Batching$(\cdot, \mathcal{A})$ is $\left(2 - \frac{1}{d}\right) \rho(\mathcal{A})$-competitive for Online-$(k, d)$-Min-Matching.

Proof. Let $H = (V, E, w)$ be a shareability graph. For a given value of z, recall the definition the batches $\{W_{z,i}\}_{i=0}^{n/d}$ as $W_{z,i} := (V_{z,i}, E_{z,i}, w)$ where

$$V_{z,i} = \{z + i * d + j\}_{j=0}^{d-1}$$
$$E_{z,i} = \{e \in E : e \subset V_{z,i}\}.$$

Algorithm 6: `Weighted-Greedy`(H)

---

1 Input: Shareability Graph $H = (V, E, w)$;
2 Output: Matching M;
3 $M \leftarrow \emptyset$;
4 $U \leftarrow V$;
5 for $e \in E$ **do**
6 $\quad \theta_e \leftarrow \frac{w(e)}{|e|}$
7 while $U \neq \emptyset$ **do**
8 $\quad e^* \leftarrow \arg\min_{e \subset U} \theta_e$;
9 $\quad M \leftarrow M \cup e^*$;
10 $\quad U \leftarrow U \setminus e^*$
11 Return M

Let M^* be a minimum weight matching in H. Define $\widetilde{M}_z = \cup_{i=1}^{n/d} \widetilde{M}_{z,i}$ where

$$\widetilde{M}_{z,i} := \left\{ e \cap V_{z,i} : e \in M^* \right\}.$$

Observation 7 *Note that since H is a shareability hypergraph, for any $e \in E$, $\sum_{i=0}^{n/d} \mathbb{1}[e \cap V_{z,i} \neq \emptyset] \leq 2$, i.e. every edge $e \in E$ intersects at most two windows.*

Proof. We can show this easily by contradiction. Suppose there exists $e \in E$ that intersects more than 2 windows. Then define

$$j = \max \left\{ k : e \cap V_{z,k} \neq \emptyset \right\}$$
$$i = \min \left\{ k : e \cap V_{z,k} \neq \emptyset \right\}.$$

It must be the case that $i + 1 < j$. If this is not the case, i.e. if $j = i + 1$, then e intersects $V_{u,k}$ only if $k \in \{i, i+1\}$, and thus it does not intersect more than 2 windows. Therefore, we know that $i + 1 < j$. Finally, note that for any $a \in V_{z,i}, b \in V_{z,j}$, we have $b > a + d$. This implies that $\text{diam}(e) > d$, which is a contradiction to the fact that $\text{diam}(e) < d$ promised by H being a shareability hypergraph. $\square$

Now consider the matching produced by `Randomized-Batching`$(\cdot, \mathcal{A})$ given $Z = z$. This procedure computes a matching M as $\cup_{i=1}^{n/d} M_{z,i}$, where $M_{z,i}$ is a set cover of $W_{z,i}$ chosen by $\mathcal{A}$. Recall from Observation 6 that under Assumption 1, any set cover can be converted into a matching with the same cost. We thus see that

$$\mathbb{E}\left[w(M)|Z = z\right] = \sum_{i=0}^{n/d} \mathbb{E}\left[w(M_{z,i})|Z = z\right]$$

$$\stackrel{(a)}{\leq} \sum_{i=0}^{n/d} \rho(\mathcal{A}) w(\widetilde{M}_{z,i})$$

$$= \rho(\mathcal{A}) \sum_{i=0}^{n/d} \sum_{e \in M^*} w(e \cap V_{z,i})$$

$$\leq \rho(\mathcal{A}) \sum_{e \in M^*} w(e) \sum_{i=0}^{n/d} \mathbb{1}[e \cap V_{z,i} \neq \emptyset].$$

Where (a) is due to the following. Since $\widetilde{M}_{z,i}$ is a matching in $W_{z,i}$, from Observation 6 we know $w(M_{z,i})$ is at least as large as the minimum weight of a set cover of $(V_{z,i}, E_{z,i}, w)$. Since $M_{z,i}$ is the output of a $\rho(\mathcal{A})$-competitive k-WSC algorithm, we have $w(M_{z,i}) \leq \rho(\mathcal{A})w(\widetilde{M}_{z,i})$. We now take expectation over Z to bound $\mathbb{E}[w(M)]$:

$$\mathbb{E}\left[w(M)\right] = \mathbb{E}_Z\left[\mathbb{E}\left[w(M)|Z = z\right]\right]$$

$$\leq \mathbb{E}_Z\left[\rho(\mathcal{A})\sum_{e \in M^*}w(e)\sum_{i=0}^{n/d}\mathbb{1}[e \cap V_{Z,i} \neq \emptyset]\right]$$

$$= \rho(\mathcal{A})\sum_{e \in M^*}w(e)\mathbb{E}_Z\left[\sum_{i=0}^{n/d}\mathbb{1}[e \cap V_{Z,i} \neq \emptyset]\right]$$

For any edge $e \in E$, let i denote its earliest vertex. Note that when $Z = i \mod d$, $\sum_{i=0}^{n/d}\mathbb{1}[e \cap V_{z,i} \neq \emptyset] = 1$. Therefore, $\mathbb{P}\left(\sum_{i=0}^{n/d}\mathbb{1}[e \cap V_{z,i} \neq \emptyset] = 1\right) \geq \frac{1}{d}$. Since we already showed that $\sum_{i=0}^{n/d}\mathbb{1}[e \cap V_{z,i} \neq \emptyset]$ is always either 1 or 2, this means that the probability it is 2 is at most $1 - \frac{1}{d}$. Applying this gives:

$$\mathbb{E}\left[w(M)\right] \leq \rho(\mathcal{A})\sum_{e \in M^*}w(e)\mathbb{E}_Z\left[\sum_{i=0}^{n/d}\mathbb{1}[e \cap V_{z,i} \neq \emptyset]\right]$$

$$\leq \rho(\mathcal{A})\sum_{e \in M^*}w(e)\left[2\left(1 - \frac{1}{d}\right) + 1\frac{1}{d}\right]$$

$$= \left(2 - \frac{1}{d}\right)\sum_{e \in M^*}w(e) = \left(2 - \frac{1}{d}\right)\rho(\mathcal{A})w(M^*).$$

This completes the proof. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

We conclude with several remarks. If $\mathcal{A}$ computes a minimum weight set cover, i.e. $\rho(\mathcal{A}) = 1$, then `Randomized-Batching`$(\cdot, \mathcal{A})$ is $(2 - \frac{1}{d})$-competitive by Theorem 6. However, it is well known that k-WSC is an NP-hard problem, so there is likely no polynomial-time algorithm which can find minimum weight set covers.

The `Weighted-Greedy` algorithm shown in Algorithm 6 runs in polynomial-time and is $(1 + \log k)$-competitive for k-WSC. This is optimal up to lower order terms, given the $\log k - O(\log\log k)$ lower bound of Trevisan [19] for k-SC, which is a special case of k-WSC. The following result is obtained from Theorem 6 with $\mathcal{A} =$ `Weighted-Greedy`. Running Algorithm 1 with `Weighted-Greedy` as a subroutine gives a $\left(2 - \frac{1}{d}\right)\left(1 + \log k\right)$-competitive algorithm for `Online-`(k, d)`-Min-Matching`. Lemma 6 provides a complementary lower bound, establishing the optimal in polynomial-time competitive ratio for `Online-`(k, d)`-Min-Matching` to be $O(\log k)$.