Minimum Online Metric Matching: Monotone Randomized Algorithms on Trees are Priceable

Anupam Gupta[1], Kirk Pruhs[2], Aditya Krishnan[1], and Maximillian Converse Wood Bender[2]

[1]Department of Computer Science, Carnegie Mellon University
[2]Department of Computer Science, University of Pittsburgh

August 2017

Abstract

The Minimum Online Metric Matching problem is defined on an underlying metric graph with $k \in \mathbb{N}$ servers whose locations are known *a priori*. Requests arrive in an online fashion in adversarially chosen locations. The goal is to match requests as they arrive to yet unmatched servers while minimizing the overall cost of the matching. We show a characterization of all algorithms that can be simulated via pricing algorithms on tree metrics. We show that that priceability, first introduced in [3], can be characterized completely by the monotonicity property of an algorithm. More importantly, we give an explicit algorithm to construct a pricing scheme from a monotone randomized algorithm.

1 Introduction

1.1 The Problem

The Minimum Online Metric Matching problem was first introduced independently by Kalyanasundaram and Pruhs in [5] and Khuller, Mitchell and Vazirani in [6]. The problem is defined on an underlying metric that has $k \in \mathbb{N}$ servers, given by nodes in set S, placed in some locations known *a priori*. Requests, given by nodes in the set R, arrive in an online fashion in adversarially chosen locations on the metric space. The objective is to match requests to yet unmatched servers while minimizing the cumulative cost of the matching. The cost of matching a request to a server is simply the distance between the request and the server as given by the underlying metric.

1.2 Previous Work

Both Pruhs et al. and Vazirani et al. gave $(2k-1)$-competitive deterministic algorithms for the problem. This happens to be tight for deterministic algorithms. Most approaches for general metric spaces proceed by embedding the original metric space in a *Hierarchally Well-Separated Tree* (HST), first introduced by Bartal in [2]. They then solve the problem on the HST and lift the solution to the original space. Bartal showed that a β-competitive algorithm on an α-HST gives a $\alpha\beta\log(n)$-competitive algorithm on the original metric space. The first sublinearly competitive algorithm was given by Myerson et al. in [7] who gave a $\log(k)$-competitive algorithm on a $\log(n)$-HST, translating to a $\log^3(k)$-competitive algorithm on general metric spaces. Bansal et al. then improved this result by giving a $\log(k)$-competitive algorithm on a 2-HST in [1].

Gupta and Lewi gave a $\log(k)$-competitive algorithm for line metrics in [4]. Cohen et al. then showed in [3] that this algorithm can be "priced". Pricing based algorithms for the problem define prices for each server before each request arrives. The price defined at timestep t is a function $p_t : S \to \mathbb{R}$. When the request r_t arrives the request is matched to a server given by $\mathrm{argmin}_{s\in S}\, p_t(s) + d(r_t, s)$ where $d(u, v)$ is the distance between points u, v on the metric.

1.3 Our Work

We characterize the pricing based approach for tree metrics. We show that any algorithm that can be priced has the property that it is *monotone*.

A **monotone** algorithm $\mathcal{A}$ is an algorithm which satisfies the following:

- For any $u, v \in V$ and $s \in S$ where v lies on the path from u to s, it holds that $\mathbb{P}(u \to_{\mathcal{A}} s) \leq \mathbb{P}(v \to_{\mathcal{A}} s)$.

Where V is the vertex set of the tree T. And $\mathbb{P}(u \to_{\mathcal{A}} s)$ denotes the probability that a request that lands on u matches to server s under $\mathcal{A}$.

We characterize these algorithms completely by showing that pricing scheme algorithms and monotone algorithms are equivalent. More importantly, we give an explicit algorithm via an induction argument for constructing this randomized pricing scheme to simulate a monotone randomized algorithm. We do this by first giving a pricing scheme to simulate a given deterministic monotone algorithm and subsequently show how to build the distribution over monotone deterministic algorithms that gives the desired randomized monotone algorithm.

2 Deterministic Algorithms

In this section we describe the process of transforming deterministic monotone algorithms on trees into pricing schemes.

A **partitioning scheme** on a tree is a set of disjoint connected subgraphs of the tree whose union is the whole tree. Deterministic monotone algorithms and partitioning schemes share the following relation.

Lemma 2.1. *Any monotone deterministic algorithm $\mathcal{A}$, at every time step, induces a partitioning scheme P that satisfies:*

1. *In each nonempty part $A \in P$ there is a single server designated as leader to which everyone matches to.*
2. *Each part is connected with respect to the tree.*

Proof. Let the sets $Q_1, ..., Q_t$ be defined by $Q_i = \{v : v \to_{\mathcal{A}} s_i\}$ and set $P = \{Q_i : Q_i \neq \emptyset\}$. Since $\mathcal{A}$ is a deterministic algorithm, these sets must be disjoint and their union must be V. For any $Q_i \neq \emptyset$, it follows that $s_i \in Q_i$. To see this, let $u \in Q_i$: since s_i lies on the path from u to s_i, it must be that $s_i \in Q_i$. Hence for any nonempty Q_i we define s_i to be its leader. Further, if $u, v \in Q_i$ then $\mathbb{P}(u \to_{\mathcal{A}} s_i) = \mathbb{P}(v \to_{\mathcal{A}} s_i) = 1$ and thus any vertex between u and v must also match to s_i. Thus these components are connected. $\square$

We will call partitioning schemes that satisfy Lemma 2.1 **monotone partitions**. Note that monotone partitions also induce monotone deterministic algorithms and are hence synonymous with them.

We now show how to transform a monotone partition $P = \{Q_1, ..., Q_t\}$ into a pricing scheme $p : S \to \mathbb{R}$. Set the price of the leader s_1 of Q_1 to be any constant $p(s_1) = c$. We then proceed iteratively as follows: let Q_j be any part with leader s_j adjacent to a part already priced. Then there exists a $v \in Q_j$ adjacent to a $u \in Q_i$ such that u's leader s_i has already been priced.

We set the price $p(s_j) = p(s_i) + d(u, s_i) - d(v, s_j)$ to satisfy

1. $d(v, s_j) + p(s_j) < d(v, s_i) + p(s_i)$
2. $d(u, s_i) + p(s_i) < d(u, s_j) + p(s_j)$.

When all leaders have been priced, price any non leaders at ∞. This pricing scheme acts exactly as the monotone partition.

Theorem 2.2. *For any monotone partition scheme P, the pricing scheme p described above simulates P. That is, if $v \to_P s_i$, then $d(v, s_i) + p(s_i) < d(v, s_j) + p(s_j)$ for all $j \neq i$.*

Proof. Suppose otherwise, that there exists an s_j and v with leader s_i satisfying that $d(v, s_i) + p(s_i) \geq d(v, s_j) + p(s_j)$. First, consider the case where Q_j and Q_i are adjacent parts. Then there exists an edge (u_i, u_j) that crosses over from Q_i to Q_j. But we know that $d(v, s_i) \leq d(v, u_i) + d(u_i, s_i)$ and that $d(v, s_j) = d(v, u_i) + d(u_i, s_j)$. But then by substituting into our assumption we have that $d(v, u_i) + d(u_i, s_i) + p(s_i) \geq d(v, u_i) + d(u_i, s_j) + p(s_j) \implies d(u_i, s_i) + p(s_i) \geq d(u_i, s_j) + p(s_j)$, a contradiction to the second observation of our pricing scheme above.

Thus Q_j cannot be adjacent to Q_i. However, let Q_k be the adjacent part to Q_j that must be crossed by the path from s_i to s_j. Since these two parts are adjacent, there exists an edge (u_k, u_j) the crosses from Q_k to Q_j. We also know that $d(u_k, s_k) + p(s_k) \leq d(u, s_j) + p(s_j)$ by design of the pricing scheme: however, this implies that $d(v, s_k) + p(s_k) \leq d(v, u_k) + d(u_k, s_k) + p(s_k) \leq d(v, u_k) + d(u_k, s_j) + p(s_j) = d(v, s_j) + p(s_j)$. This means that v should have instead matched to s_k, a contradiction. □

3 Randomized Algorithms

Now that we can price monotone deterministic algorithms, it remains to show that any monotone randomized algorithm can be written as a distribution over monotone deterministic algorithms and give an explicit way to go from randomized algorithms to deterministic algorithms. To do this, we will show that we can define a distribution $\mathcal{P}$ over monotone partitions that simulates the algorithm and give an inductive argument on how to build this distribution. For the remainder of this section, we will assume our algorithm $\mathcal{A}$ defines probability v_i for vertex $v \in V$ of matching to server s_i. We will use the notation $\mathbb{P}_T(D)$ to denote the probability that partition D is chosen on graph T and let $D_i^v = \{D | v \to_D s_i\}$.

3.1 Trees

Now we show by induction on the number of vertices in the tree T that any monotone algorithm on a tree induces a distribution over monotone partitions of T. Here, this means we are defining $\mathcal{P}$ such that $\sum_{D \in D_i^v} \mathbb{P}_T(D) = v_i$ for all vertices $v \in T$. Our induction will proceed by contracting leaves of the tree and any servers located on those leaves, building the partition on the smaller tree, and then extending those partitions back onto the larger tree.

When contracting downwards, the leaf vertex (server(s)) will inherit the distribution of the vertex onto which they're contracted. Thus if $s_1, \ldots, s_t$ are leaf servers and v is their neighbour, after contraction all of $s_1, \ldots, s_t$ will have distribution vector $(v_1, \ldots, v_k)$

Lemma 3.1. *For any monotone algorithm $\mathcal{A}$ on a single tree T there exists a distribution $\mathcal{P}$ over monotone partitions of the tree such that $\mathbb{P}(v \to_{\mathcal{P}} s_i) = \mathbb{P}(v \to_{\mathcal{A}} s_i)$ for all vertices $v \in T$ and all servers s_i.*

Proof. The base case of this induction will be the final contraction where all servers are collocated on one vertex v. In this case, each possible partition will just be that one point with one server chosen as a leader. For each such partition A_i with leader s_i, we will just assign the partition the probability v_i.

Let us call the original tree T and let us assume we are contracting t colocated servers on vertex u which is a leaf and is adjacent to some vertex v. Let us call the smaller tree T' where we contract the servers onto v.

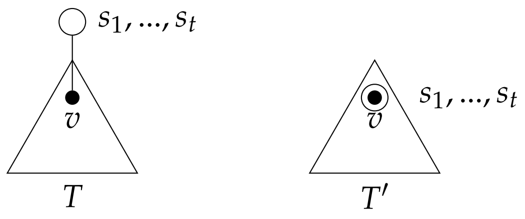

Figure 1: An example of a contraction

As in Figure 1, we will extend every partition on the contracted graph T' by uncontracting the edge to obtain the old graph T and defining new partition(s) with new probabilities. For any partitions $F_1, \ldots, F_t$ of T created from a partition F of T', we will maintain the constraint that

$$\sum_i \mathbb{P}_T(F_i) = \mathbb{P}_{T'}(F) \tag{1}$$

Subject to the above constraint we must assign probabilities to the new partitions such that $\mathbb{P}(v \to_{\mathcal{P}} s_i) = v_i$.

In particular, there are only two cases to consider for the partition F:

3.1.1 Case 1

One of $s_1, ..., s_t$ is a leader of F.

In this case, only one new partition F_1 will be created from F and will inherit its probability: $\mathbb{P}_T(F_1) = \mathbb{P}_{T'}(F)$ so as to satisfy constraint (1).

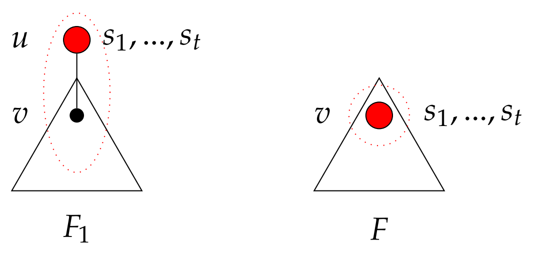

Figure 2: An example of a case 1 contraction: the red dotted region denotes a part in a partitioning with a red circle as its leader

3.1.2 Case 2

The leader of F is $s' \notin \{s_1, ..., s_t\}$. We will extend F into $t+1$ new partitions. For partitions $F_1, ..., F_t$, we will extend the partitioning to make s_i the leader of its own singleton part. The final partitioning F_{t+1} created from F extends the partition of s' to include u.

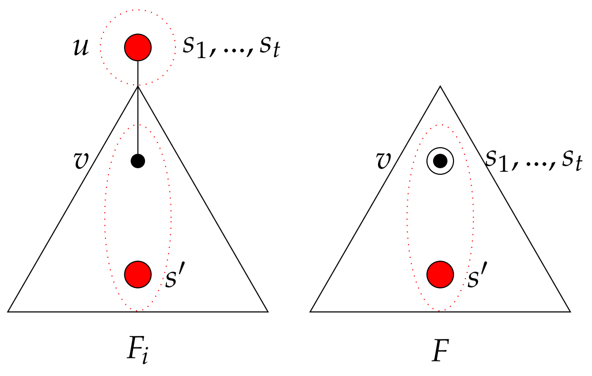

Figure 3: An example of case 2 for $1 \leq i \leq t$

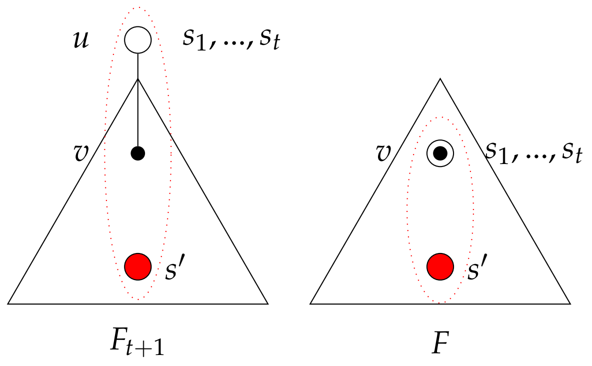

Figure 4: An example of case 2 for $i = t+1$

Notice that the way we're extending the partitions of T' to T and with constraint 1, we only need to make sure we're satisfying the distribution for u since all other vertices' distributions remain satisfied under our extension.

By our inductive constraint we know that for any $i \notin \{1, ..., t\}$ we get a distribution over partitions of T' such that $\sum_{D \in D_i^v} \mathbb{P}_{T'}(D) = v_i$, where $v_i \geq u_i$ by monotonicity. We will let $-\Delta_i = u_i - \sum_{D \in D_i^v} \mathbb{P}_{T'}(D)$.

Notice that each $D \in D_i^v$ gets extended to an equivalent partition in D_i^u.

Initially, we set $\mathbb{P}_T(F_{t+1}) = \mathbb{P}_{T'}(F)$, but then greedily remove Δ_i across all $F_{t+1} \in D_i^u$ without violating the constraint that all $F_{t+1} \in D_i^u$ must have non-negative probability. After completing this, we have that $\sum_{D \in D_i^u} \mathbb{P}_T(D) = u_i$ for all $i \notin \{1, ..., t\}$.

We now only need to show that we satisfy that $\sum_{D \in D_j^u} \mathbb{P}_T(D) = u_j$ for all $j \in \{1, ..., t\}$. Specifically, we want for each s_j:

$$\sum_{D \in D_j^u} \mathbb{P}_T(D) = \sum_{F \in D_j^v} \mathbb{P}_T(F_1) + \sum_{i > t} \sum_{F \in D_i^v} \mathbb{P}_T(F_j) = u_j. \tag{2}$$

The probabilities in the first summation in (2) have already been assigned in Case 1, so we must ensure that the probability we assign to each F_j satisfies both (1) and $\sum_{F \in D_j^v} \mathbb{P}_T(F_j) = u_j$.

Let us define $\Delta_j = u_j - \sum_{F \in D_j^v} \mathbb{P}_T(F_1) = u_j - v_j$, which must be positive by monotonicity. We then greedily assign the Δ_j probability mass across all $\{F_j | F \in D_i^v, i \notin \{1, ..., t\}\}$ such that we don't violate (1) by assigning too much probability mass to some F_j.

By doing the above, we have satisfied that $u_i = \sum_{D \in D_i^u} \mathbb{P}_T(D)$ for all i. Note that the distribution of every other vertex was maintained just by the inductive hypothesis and (1). It is left to argue that we didn't violate (1) by greedily assigning the Δ_is and Δ_js. To this end, fix some $\ell \in \{1, ..., t\}$ and consider the distribution $\vec{p}_\ell$ over servers to match to if a request landed on s_ℓ, Consider the same distribution if a request landed on v given by the distribution vector $\vec{p}_v$. From before, the first t Δs were the increase in probability of matching to the servers on $s_1, \ldots, s_t$ (for a request on u) by uncontracting the edge, while the last $k - t$ Δs were the reduction in probability of u matching to some $s_j \notin \{s_1, ..., s_t\}$. Then

$$\vec{p}_\ell = \vec{p}_v + (\Delta_1, ..., \Delta_t, -\Delta_{t+1}, ..., -\Delta_k).$$

Since both $\vec{p}_v$ and $\vec{p}_\ell$ are probability distributions, their L_1 norm is 1. Thus,

$$\sum_{\ell=1}^{t} \Delta_\ell = \sum_{i=t+1}^{k} \Delta_i.$$

Thus, we know that assigning the Δs greedily the way we have will not violate constraint (1). □

Combining Lemma 3.1 and Theorem 2.2 gives us that every monotone randomized algorithm is priceable. Notice that the inductive argument is constructive and hence gave an explicit way to construct the distribution over deterministic monotone algorithms. Since we also gave an explicit way to price every deterministic algorithm, the result gives an explicit way to go between randomized monotone algorithms and pricing schemes.

We have shown that every monotone randomized algorithm has a pricing scheme but what about the other way around? It is also true that every pricing scheme is a randomized monotone algorithm. To see this, notice that it is sufficient to argue that every deterministic pricing scheme is monotone due to Yao's minimax principle.

Let $p : S \to \mathbb{R}$ be the pricing scheme given by the algorithm. Given this pricing scheme it is sufficient to show that we can build a monotone partition. For any vertex $u \in V$ that matches to $s_i \in S$ under p, it must be the case that $p(s_i) + d(u, s_i) < p(s_j) + d(u, s_j)$ for all $s_j \neq s_i$. If we have two vertices u, v that match to s_i under p, notice that all vertices w on the path between u, v must also match to s_i because of the previous condition. Hence the part corresponding to the vertices matching to s_i is connected.

4 Conclusion

Pricing schemes and monotone algorithms are equivalent on tree metrics but what about general metric graphs? It is not even clear what the right definition of monotonicity is for graphs with cycles since there isn't necessarily a unique path between two vertices. It would be interesting to characterize the condition required for priceability in general graphs and see how the notion of monotonicity extends to general graphs.

References

[1] N. Bansal, N. Buchbinder, A. Gupta, and J. Naor. A randomized o(log2 k)-competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.

[2] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 184–193, 1996.

[3] I. R. Cohen, A. Eden, A. Fiat, and L. Jez. Pricing online decisions: Beyond auctions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 73–91, 2015.

[4] A. Gupta and K. Lewi. The online metric matching problem for doubling metrics. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 424–435, 2012.

[5] B. Kalyanasundaram and K. Pruhs. Online weighted matching. *J. Algorithms*, 14 (3):478–488, 1993.

[6] S. Khuller, S. G. Mitchell, and V. V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127(2):255–267, 1994.

[7] A. Meyerson, A. Nanavati, and L. J. Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 954–959, 2006.