Operations Research Letters 51 (2023) 591–594

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

A randomized algorithm for online metric b-matching

Bala Kalyanasundaram [a], Kirk Pruhs [b], Cliff Stein [c,*]

[a] Department of Computer Science, Georgetown University, Washington D.C., 20056, DC, USA
[b] Computer Science Department, University of Pittsburgh, Pittsburgh, 15260, PA, USA
[c] Department of Industrial Engineering and Operations Research, Columbia University, New York, 10027, NY, USA

ARTICLE INFO

Article history:
Available online 25 September 2023

Keywords:
Matching
Online algorithms
Metric matching

ABSTRACT

We give a randomized algorithm for online metric b-matching that is $O(\log^2 k)$ competitive, where k is the number of server locations, by giving a black box reduction from b-matching on a hierarchically separated tree to a uniform metric space.

1. Introduction

We consider the classic online metric matching problem, which we explain in the context of an application to the efficient parking of cars. The problem is set in a metric space G, with distance function $d(\cdot,\cdot)$, containing n parking spaces (servers). Let $S = s_1, \ldots, s_n$ be the collection of parking space locations in G. Then a sequence of (up to) n cars (requests) arrive at various locations in G. Let r_i denote the location that the i^{th} arriving car arrived, and let $R = r_1, \ldots, r_n$ be the sequence of all such locations. When car i arrives, the online algorithm must irrevocably assign/match car i to an available parking space of its choice, which we will denote by $s_{\alpha(i)}$. A parking space is available if it has not been previously assigned/matched to a car. The objective is to minimize the total distance moved by the cars (which we will view as a cost), namely $\min \sum_{i=1}^{n} d(r_i, s_{\alpha(i)})$. We will evaluate online algorithms based on their competitiveness. In general the competitiveness achievable by an online algorithm depends on the metric space G, but our focus will be on general metric spaces. So the competitiveness $c(|P|)$ of an online algorithm A as a function of some parameter P will be the maximum, over instances (G, R, S) where the value of parameter P is $|P|$, of the ratio of the cost that A incurs on request sequence R with parking spaces located at S in G to the cost of the minimum-cost perfect bipartite matching between R and S.

The classical competitiveness results for online metrical matching on general metric spaces are parameterized by $|P| = n$. The natural greedy algorithm, which matches each car to the nearest available parking space, is $2^n - 1$ competitive, and a worst-case metric where this competitiveness can be achieved is the line metric [3]. The optimal deterministic competitive ratio for a general metric space is $2n - 1$ [3,5], which is achieved by an algorithm called Permutation in [3]. The $2n - 1$ lower bound is achieved for a star metric (where this is a unique vertex, called the center, of distance one from all other vertices, and all other distances are two). The optimal deterministic competitive ratio for a uniform metric space (where all distances are one) is n. The expected competitiveness of every randomized algorithm (against an oblivious adversary) is $\Omega(\log n)$, and a matching upper bound is achieved on a uniform metric space. By embedding a metric space into a hierarchically separated tree (HST) one can obtain competitiveness $O(\log^2 n)$ for a general metric space [1,6].

In this paper, we consider the situation where the number k of distinct locations for the parking spaces is much less than the number of total parking spaces, say for example because there are a small collection of parking garages that each contain many parking spaces. The special case that each garage has the same number b of parking spaces, which one can assume with minimal loss of generality, is sometimes called b-matching in the literature. In [3] it is observed that the competitiveness of the natural greedy algorithm could be reparameterized using the smaller parameter k, namely that the competitiveness of the natural greedy algorithm is $2^k - 1$ for a general metric space [3]. In contrast, [4] observed that the competitiveness of Permutation can not be bounded by any function of k. Still [4] conjectured that the standard competitiveness results could be reparameterized in terms of the parameter k, and more precisely that the optimal deterministic competitive ratio is $2k - 1$.

Here we substantiate the conjecture from [4] for randomized algorithms, that is, that current randomized results can be reparameterized by k instead of n. We show that the $O(\log^2 n)$ competitive algorithm from [1] can be adapted to obtain an $O(\log^2 k)$ compet-

* Corresponding author at: Department of IEOR, Columbia University, New York, NY, 10027, USA.
E-mail addresses: bk38@georgetown.edu (B. Kalyanasundaram), kirk@cs.pitt.edu (K. Pruhs), cliff@ieor.columbia.edu (C. Stein).

https://doi.org/10.1016/j.orl.2023.09.002
0167-6377/

itive algorithm. Our algorithm design and analysis differs from the algorithm design and analysis [1] in some reasonably substantial ways. [1] gave an algorithm for an HST, whose analysis was similar to the analysis for a uniform metric space. Instead we give a black box reduction from an HST to a uniform metric space, that is, we show that a c-competitive algorithm for metric b-matching on a uniform metric space can be converted into an $O(c)$-competitive algorithm for an $O(1)$-HST. So we believe that one ancillary benefit of our analysis is to make more formal the intuition from [1] that embedding into an HST is reducing the problem on a general metric space to the problem on a uniform metric space. We also show how to solve the problem on a uniform metric space, which has its own technical challenges.

2. Randomized algorithms

In subsection 2.1 we define an analyze a randomized parking process that we call the tourist parking process. The tourist parking process actually allows cars to move once they are parked. In subsection 2.2 we explain how an online algorithm can emulate this tourist parking process, without actually moving any already parked cars, to obtain an $O(\log k)$-competitive algorithm for online metric matching in a uniform metric space. Finally in subsection 2.3 we give a reduction from a c-competitive algorithm for the uniform metric space to an $O(c)$-competitive algorithm for a 2-HST. Combining this with the now standard technique of embedding an arbitrary metric space into an HST, as in [1,6], we can obtain an $O(\log^2 k)$ randomized algorithm for online metric matching in a general metric space.

2.1. Tourist parking process

In the tourist parking process, n cars arrive over time in a uniform metric space on $k+1$ garages. Let b_j be the number of parking spaces in garage g_j, $j \in [k]$. We call garage g_{k+1} the *center* and set $b_{k+1} = 0$.

Let $c_j(i)$ be the total number of cars that have initially arrived at garage g_j in the first i requests. We set $e_j(i) = \max(c_j(i) - b_j, 0)$ and call this the *excess* at garage g_j after car i has be parked. We finally let $e(i) = \sum_{j=1}^{k+1} e_j(i)$ be the aggregate excess. Let E be an integer such that $e(n) \le E \le 2e(n)$. We assume for now that the arriving cars know E, and remove that assumption in Section 2.2.

We assume that the parking spaces in garage g_j are numbered 1 to b_j. Parking spaces 1 through $\max(b_j - E, 0)$ are called early bird parking spaces, and the remaining parking spaces are called regular parking spaces.

All cars are classified as *early birds*, *regulars*, or *tourists* when they arrive. The cars that arrive at a garage g_j are classified in the following manner:

- The first $\max(0, b_j - E)$ cars to arrive are classified as early birds.
- The next $b_j - \max(0, b_j - E)$ cars to arrive are classified as regulars.
- All subsequent cars to arrive are classified as tourists.

When an early bird arrives, it picks the reserved parking spot in the parking garage. After early bird arrivals, let m be the total number of regulars and tourists, namely $m = \sum_{j=1}^{k} \min(E, b_j)$.

If a regular arrives at garage g_j, and this car is the h^{th} arriving car at g_j among regular cars, then this car parks in parking space $\max(0, b_j - E) + h$ in g_j. If a tourist was previously parked in parking space h, which could be the case if the new car is a regular, then the tourist randomly picks a new parking space using the tourist selection method.

When a tourist arrives, it picks a parking space using what we call the *tourist selection method*. If the tourist is the $i+1$st in total of m requests among regulars and tourists, the tourist picks a garage g_j with probability proportional to the current number of available parking spaces for tourists and regulars in g_j. The tourist then picks an available regular parking space to park in g_j uniformly at random.

There are $e(n)$ tourist cars in total, and $m - e(n)$ regular cars. The following lemma states the probability distribution for where the tourists are parked.

Lemma 1. *Assume that at some point in the tourist parking process r regulars have arrived and e tourists have arrived. Then tourists are equally likely to be parked at each of the $\binom{m-r}{e}$ possible collections of e of the $m-r$ available regular parking spaces. Thus the probability that a particular available regular parking space contains a tourist is $\frac{e}{m-r}$.*

Lemma 2. *The expected number of times that the tourist selection method is invoked is $O(e(n)\log k)$.*

Proof. There are $e(n)$ invocations of the tourist selection method when the $e(n)$ tourists arrive. Thus we need only bound the number of invocations when regulars arrive. Let x_i be an indicator random variable that is 1 if the i^{th} arrival of a regular caused a tourist to move, and 0 others. Summing over the arrivals of regulars, and letting $H(x)$ denote the x^{th} harmonic number, we get that the expected number of tourist cars that have to move due to the arrival of a regular is:

$$\begin{aligned}
\sum_{i=1}^{m-e(n)} \mathbb{E}[x_i] &= \sum_{i=1}^{m-e(n)} \mathbb{P}[x_i = 1] \\
&= \sum_{i=1}^{m-e(n)} \frac{e(i)}{(m-(i-1))} \\
&\le \sum_{i=1}^{m-e(n)} \frac{e(n)}{(m-(i-1))} \\
&= e(n) \sum_{i=1}^{m-e(n)} \frac{1}{(m-(i-1))} \\
&= e(n) \sum_{i=e(n)+1}^{m} \frac{1}{i} \\
&= e(n)(H(m) - H(e(n))) \\
&= e(n)(H(\sum_{j=1}^{k} \min(E, b_j)) - H(e(n))) \\
&\le e(n)(H(kE) - H(e(n))) \\
&\le e(n)(H(2e(n)k) - H(e(n))) \\
&\le e(n)(1 + \ln(2e(n)k) - \ln e(n)) \\
&\le e(n)(\ln e(n) + \ln k + \ln 2 - \ln e(n)) \\
&= O(e(n)\ln k)\,.
\end{aligned}$$

The second equality follows from Lemma 1. The third to last inequality follows the assumption that $E \le 2e(n)$, and the last inequality follows from the fact that $\ln x < H(x) \le 1 + \ln(x)$. □

2.2. Uniform metric space

We now turn to the original parking problem in a uniform metric space. All the points in the metric space not containing a garage are collapsed to a single point, which is designated as the center and indexed by $k+1$. Unfortunately E is not known at the beginning but will be known at the end. We will use standard doubling trick to guess E.

Uniform Algorithm Description: The algorithm mimics the tourist parking process, initially with $E=1$. A car arriving at garage g_j is parked in garage g_h with the probability that the tourist parking process moves a car from garage g_j to garage g_h in response to the arrival of this car. So the cars are never parked at the same garage twice, and it is easy to see the equivalence, we just move the later car to arrive to the spot where the uniform parking process sent the earlier car.

The standard doubling method is used in the case that arrival of a car i causes the number of tourists to become equal to E. That is, the parameter E is doubled, say from x to $2x$. Let $C'_f(i)$ be the number of cars that would be parked in garage g_f if the tourist parking process with $E=2x$ had been run on the cars $1,\ldots,h-1$. Let $C_f(h)$ be the number of cars that are actually parked in garage g_f after car $h-1$ is parked. The uniform algorithm then continues simulating the tourist parking process for request i onwards using $E=2x$ and assuming configuration C'_{i-1}. If the tourist parking process tries to park a car i in a garage g_f, where $C'_i(f) \geq C_i(f)$ then car i actually parks a car in garage g_f. Otherwise if the tourist parking process tries to park a car i in a garage g_f, where $C'_i(f) < C_i(f)$ then car i actually parks a car in an arbitrary garage g_a where $C'_i(f) > C_i(f)$. Such a garage g_a always exists since the number of cars parked in each configuration is the same. So intuitively this causes the actual configuration to drift toward the desired configuration C'_i.

Lemma 3. *The uniform algorithm is $O(\log k)$-competitive for matching in a uniform metric space.*

Proof. If E were fixed then the cost of the uniform algorithm would be exactly the cost of the tourist parking process, which by Lemma 2 is $O(E \log k)$. We assume $E_1 = 1$ and the last doubling leads to $E = E_\ell = 2^{\ell-1}$. When the estimate E doubles, from say $E_i = x$ to $E_{i+1} = 2x$, the number of cars parked in different spaces in C and C' is at most $2x$. Thus the cost to the uniform algorithm due to being in configuration C instead of configuration C' is at most $2x$. And thus the cost to the uniform algorithm is

$$O\left(\sum_{i=1}^{\ell}(E_i \ln k + E_i)\right) = O(E_\ell \log k)\,,$$

where here E_ℓ is the final value of E. As E is at most twice the optimal cost the result follows. □

2.3. Hierarchically separated trees

We now turn to the general metric and give a reduction from the uniform metric to the case of Hierarchically Separated Trees, HST for short. Given a parameter $\alpha \geq 1$, an α-Hierarchically Separated Tree (α-HST) is a rooted tree $T=(V,E)$ along with a length function d on the edges which satisfies the following properties:

- For each node v, all the children of v are at the same distance from v.
- For any node v, if $p(v)$ is the parent of v and $c(v)$ is any child of v, then $d(p(v),v) = \alpha \cdot d(v,c(v))$.
- Each leaf has the same distance to its parent, for convenience, we assume that the distance between each leaf and its parent is 1.

We view an α-HST as a leveled tree, where all the leaves are at the same level, and the edge-lengths increase geometrically by a factor of α as we go up the tree from the leaves to the root. We set $\alpha = 2$.

HST Algorithm Description: The root node of the tree is different from every other node. Each nonleaf node v of T, with children $w_1,\ldots,w_h$, runs a copy U_v of a uniform algorithm on an $h+1$ node uniform metric space where the number of parking spaces in g_j, $j \in [h]$, is the number of parking spaces in the leaves of the subtree T_{w_j} rooted at w_j in T, and v is the center. When a car first arrives, it is given to the copy of the uniform algorithm running at the root of T which has x children and it runs a x node uniform algorithm.

We now explain how the algorithm works for an arbitrary nonleaf node v in T.

- Assume a car arrives at a garage in T_{w_j}. If in response to this arrival U_v parks this car in g_j then this car's arrival is passed to U_{w_j}.
- Otherwise if in response to this arrival U_v moves a car from g_j to a garage g_f, $f \neq j$ then a request w_f is passed to the copy of the uniform algorithm U_{w_f} running at w_f.

Any car arrival that is passed to a leaf in T is parked in the garage associated with that leaf.

Lemma 4. *If the uniform algorithm is c-competitive for online metric matching on a uniform metric space, then the HST algorithm is $O(c)$ competitive for metric matching on a 2-HST.*

Proof. A lower bound to the optimal cost is the sum over the nodes v in T of the number of tourists that U_v sees times $2^{h(v)-1}$, where $h(v)$ is the height of v in T in terms of hops. By design the cost of the HST algorithm is $O(c)$ times this amount since each time HST incurs $O(c)$ in cost. □

Theorem 1. *There is an $O(\log^2 k)$-competitive algorithm for online metric matching in an arbitrary metric space.*

Proof. Such an algorithm can be obtained by combining the uniform algorithm in the previous subsection, the HST algorithm above, and the now standard techniques for embedding an general metric space into an HST with distortion $O(\log k)$ [1,2,6]. □

Data availability

No data was used for the research described in the article.

Acknowledgements

B. Kalyanasundaram was supported in part by Craves Family Professorship. K. Pruhs was supported in part by NSF grants CCF-1907673, CCF-2036077, CCF-2209654 and an IBM Faculty Award. C. Stein was supported in part by NSF grants CCF-2218677 and ONR-13533312, and by the Wai T. Chang Chair in Industrial Engineering and Operations Research.

References

[1] N. Bansal, N. Buchbinder, A. Gupta, J. Naor, A randomized $o(log^2\ k)$-competitive algorithm for metric bipartite matching, Algorithmica 68 (2) (2014) 390–403.

[2] J. Fakcharoenphol, S. Rao, K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, J. Comput. Syst. Sci. 69 (3) (2004) 485–497.
[3] B. Kalyanasundaram, K. Pruhs, Online weighted matching, J. Algorithms 14 (3) (1993) 478–488.
[4] B. Kalyanasundaram, K. Pruhs, On-line network optimization problems, in: A. Fiat, G.J. Woeginger (Eds.), Online Algorithms, The State of the Art, The Book Grow Out of a Dagstuhl Seminar, June, 1996, in: Lecture Notes in Computer Science, vol. 1442, Springer, 1996, pp. 268–280.
[5] S. Khuller, S.G. Mitchell, V.V. Vazirani, On-line algorithms for weighted bipartite matching and stable marriages, Theor. Comput. Sci. 127 (2) (1994) 255–267.
[6] A. Meyerson, A. Nanavati, L.J. Poplawski, Randomized online algorithms for minimum metric bipartite matching, in: ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 954–959.