Non-Crossing Matching of Online Points

Prosenjit Bose* Paz Carmi† Stephane Durocher‡ Shahin Kamali‡ Arezoo Sajadpour‡

Abstract

We consider the non-crossing matching problem in the online setting. In the monochromatic setting, a sequence of points in general position in the plane is revealed in an online manner, and the goal is to create a maximum matching of these points such that the line segments connecting pairs of matched points do not cross. The problem is online in the sense that the decisions to match each arriving point are irrevocable and should be taken without prior knowledge about forthcoming points. The bichromatic setting is defined similarly, except that half of the points are red and the rest are blue, and each matched pair consists of one red point and one blue point. Inspired by the online bipartite matching problem [15], where vertices on one side of a bipartite graph appear in an online manner, we assume red points are given a priory and blue points arrive in an online manner.

In the offline setting, both the monochromatic and bichromatic problems can be solved optimally with all pairs matched [11]. In the online setting of the monochromatic version, we show that a greedy family of algorithms matches $2\lceil (n-1)/3 \rceil$ points, where n is the number of input points. Meanwhile, we prove that no deterministic online algorithm can match more than $2\lceil (n-1)/3 \rceil$ points, i.e., the greedy strategy is optimal. In the bichromatic setting, we introduce an algorithm that matches $\log n - o(\log n)$ points for instances consisting of n red and n blue points, and show that no deterministic algorithm can do better. We also consider the problem under the advice setting, where an online algorithm receives some bits of advice about the input sequence, and provide lower and upper bounds for the amount of advice that is required and sufficient to match all points.

1 Introduction

Matching is an important topic in combinatorics, particularly in graph theory (see, e.g., the book by Lovász and Plummer [17]). When it comes to computational geometry, matching of points in the plane has applications that range from circuit design [12] to colour-based image retrieval [1]. In a monochromatic setting, a collection of points in general position is given and the goal is to match a maximum number of points, provided they adhere to some constraints. In the bichromatic variant, each point is either blue or red, and must be matched to a point of the opposite color. Variants of the problems have been considered. For example, one might be interested in minimizing the total length or the maximum length of segments in the matching, also known as bottleneck matching (e.g., [19, 8]). Kaneko and Kano survey some of the results related to this setting [14].

In the *non-crossing matching problem*, the goal is to find a maximum matching so that the segments between pairs of matched points do not intersect. In the offline setting, where all points are given as input in advance, the problem can be easily solved in both the monochromatic and the bichromatic settings. In the case of monochromatic points, one can sort them by their x-coordinate and match consecutive pairs in the sorted sequence. This will match all points except potentially the last one (if the number of points is odd). For the bichromatic variant [2], also known as the "Ghosts and Ghostbusters" problem [7], one can find the ham-sandwich line that bisects the blue and red points in $O(n)$ time [16], and apply a divide-and-conquer approach. Both algorithms run in $O(n \log n)$ time for an input of size n, which is best possible [9]. Assuming the number of red and blue points are equal, all points are matched. A slightly different approach, with the same running time, is presented in [11]. We also note that a minimum-length matching is noncrossing. In summary, we can match all points optimally in $O(n \log n)$ time in the offline setting. Other variants of non-crossing matching have been studied (see [17]). For example, Aloupis et al. [1] considered the computational complexity of finding non-crossing matching of a set of points with a set of geometric objects, where an object can be a convex polygon, a line, or a line segment.

In this article, we are interested in the online variant of non-crossing matching problems.

Definition 1 *The input to the* ***monochromatic online non-crossing matching problem*** *is a set of points in general position in the plane that appear in an online, sequential manner. When a point arrives,*

*Carleton University, Ottawa, Canada. jit@scs.carleton.ca

†Ben-Gurion University of the Negev, Beer-Sheva, Israel. carmip@cs.bgu.ac.il

‡University of Manitoba, Winnipeg, Canada. {durocher,shahin.kamali}@cs.umanitoba.ca, sajadpoa@myumanitoba.ca

an online algorithm can match it with an existing unmatched point, provided that the line segment between them does not cross previous line segments added to the matching. Alternatively, the algorithm can leave the point unmatched to be matched later. In taking these decisions, the algorithm has no information about the forthcoming points or the length of the input. The algorithm's decisions are irrevocable in the sense that once a pair of points is matched, that pair cannot subsequently be removed from the matching. The objective is to find a maximum matching. In the ***bichromatic variant*** *of the problem, half of the points are red and half are blue. The red points are given in advance, while the blue points appear in an online manner. Upon arrival of a blue point, an online algorithm either matches it with a red point or leaves it unmatched. The goal is to find a maximum matching in which the line segments between matched pairs do not cross.*

In the online setting, it is not always possible to match all points to achieve an optimal solution. As an example, consider two points with the same x-coordinate appear at the beginning. If the online algorithm does not pair them, its solution is sub-optimal for an input formed only by these two points. If the algorithms does pair the first two points, the sequence might be followed by one point on the left and one on the right of the line segment between the matched pair. The new points cannot be matched and hence the solution is sub-optimal for an input formed by the four points.

We study the online matching problem in the worst-case scenario, where the input is generated by an adversary. This is consistent with the standard framework of competitive analysis [18]. The *competitive ratio* of an online algorithm is the maximum ratio between the number of pairs in an optimal offline solution and that of the online algorithm for sufficiently long sequences. Since an offline algorithm always matches all points (except potentially one), we prefer to express our results in terms of the number of matched/unmatched points. Throughout the paper, we assume the length n of the online sequence is sufficiently large.

1.1 Contribution

For the monochromatic variant of the problem, we consider greedy algorithms with the following *greedy property*: the algorithm never leaves an incoming point unmatched if it can be matched with some existing point. We prove that a greedy algorithm can match at least $\lceil 2(n-1)/3 \rceil$ points for any input of n points. Moreover, we prove optimality since no deterministic algorithm can match more than $\lceil 2(n-1)/3 \rceil$ points in the worst case.

For the bichromatic variant, we introduce an algorithm that matches at least $\log n - o(\log n)$ points for any input formed by n red and n blue points. Further, we prove optimality since no deterministic algorithm can match more points in the worst case. Our results indicate that the bichromatic variant is more difficult than the monochromatic variant in the online setting.

In addition to the purely online setting, we study the problem in a relaxed setting where the online algorithm is provided with some bits of *advice* about the input. The advice is generated by an offline oracle, and is available to the algorithm before the sequence is revealed (see [3, 5, 6] for a precise definition of advice). For the monochromatic variant, we show that advice of size $2n$ is sufficient to match all n points, and advice of size $\lfloor \log((n-2)/3) \rfloor$ is necessary. For the bichromatic variant, we show advice of size $\Theta(n \log n)$ is both sufficient and necessary to match all points; precisely $n\lceil \log n \rceil$ bits are sufficient and $\lceil \log n! \rceil$ bits are necessary.

2 Monochromatic Non-crossing Matching

In this section, we provide tight upper and lower bounds for the number of points that can be matched in the monochromatic non-crossing matching problem.

An online algorithm is said to have the *greedy property* if it never leaves a point unmatched when it has the option to match it.

Theorem 1 *Any online algorithm with the greedy property matches at least $2\lceil (n-1)/3 \rceil$ points in any instance of the online monochromatic non-crossing matching problem on n points.*

Proof. Let Gr be a greedy algorithm. The proof works by partitioning the plane into a set of convex regions such that each region, except one, is mapped to a pair of matched vertices. For that, we process the line segments between matched pairs of Gr in an arbitrary order. Initially, there is only one part, formed by a bounding box of the entire point set; this part has no pair associated with it. Extend each line segment until it intersects an existing line in the current partition. Note that the extended segment divides one convex region into two smaller convex regions, out of which we associate one with the pair that has been processed, and the other to the pair that was previously associated with the partitioned convex region. Repeating this process for all line segments results in $k+1$ convex regions in the final partition, where k is the number of matched pairs (see Figure 1). For detailed geometric properties of this convex subdivision, see, e.g., [4, 13]. Since Gr has the greedy property, there is at most one unmatched point inside each convex region.

To summarize, the number of unmatched points u is no more than the number of convex regions, which is one more than the number of matched pairs m. So, we

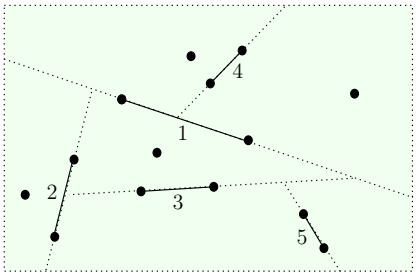


Figure 1: A partition of the plane into convex regions in the analysis of a greedy algorithm. The numbers on the line segments indicate the order they are processed in the analysis.

have $u \leq m + 1$. The statement follows from the fact that $u + 2m = n$. □

Next, we show that no online algorithm matches more points than does a greedy algorithm in the worst case.

Theorem 2 *Let* Alg *be any deterministic online algorithm for the monochromatic non-crossing matching problem. There are sequences of n points for which* Alg *matches at most $2\lceil (n-1)/3 \rceil$ points.*

Proof. We form an input that is generated in an adversarial manner based on the actions of Alg. The adversary maintains a *critical region*, which is initially the entire plane and *shrinks* as the algorithm proceeds. The adversary keeps adding points to arbitrary positions in the critical region. As soon as the algorithm matches two points a and b, the critical region is updated as follows. Consider the two sides of the line passing through a and b. If there is a non-empty set S of unmatched points on any side of the line in the critical region, then the critical region is updated to be its sub-region that is not *visible* to any point in $x \in S$ assuming the line segment between a and b acts as an obstacle. This can be done by extending the line segments between x and a and b (see Figure 2). Since the points are in general position, the updated critical region is non-empty. Note that if both sides of the line passing through a and b include unmatched points, the adversary selects one side arbitrarily. In case no unmatched point exists in the critical region, the adversary first generates a point x in an arbitrary position in the critical region and updates the critical region as a sub-region not visible by x. This process continues by sending the subsequent points in the updated (smaller) critical region.

The main observation is that, after a critical region is updated, at least one point x remains unmatched since the line segment between x and any future point crosses the segment between a and b. In particular, we can assign at least one unmatched point x to a matched pair. After updating the critical region, the very first point generated in the updated region also remains unmatched. Let u and m denote the number of unmatched points and matched pairs, respectively. By the above observations, we have $u = m + 1$. The statement of the theorem follows from $u + 2m = n$. □

3 Bichromatic Non-crossing Matching

In this section, we study online algorithms for the bichromatic non-crossing matching problem and provide tight upper and lower bounds for the number of points that can be matched. Recall that the input is formed by n red points that are known to the algorithm from the beginning and n blue points that appear in an online manner and need to be matched with the red points.

We introduce an online algorithm, named the *Greedy Median* (Gm) algorithm, that works as follows. Upon arrival of a blue point a, Gm forms a set S of eligible red points that can be matched with a without crossing previous line segments. If S is non-empty, Gm matches a with the median of the points in S when arranged in angular order around a. The selection of angular ordering is arbitrary, and it can be replaced by any ordering as long as the line through a and the median of the points in S bisects S.

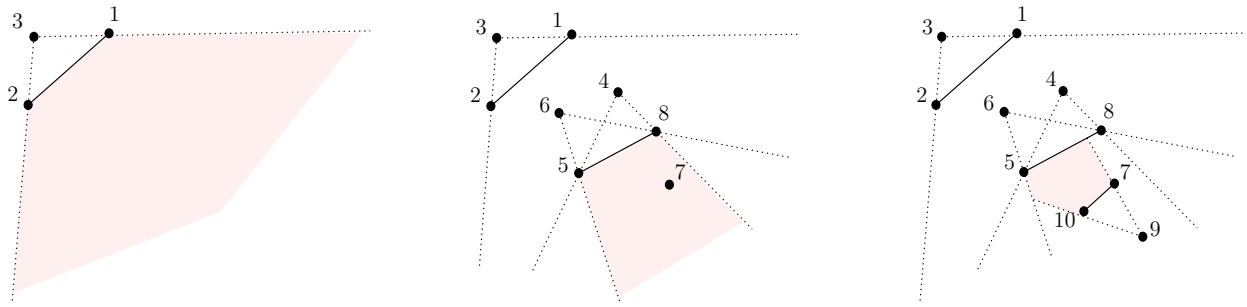
Theorem 3 *The Greedy Median (*Gm*) algorithm matches at least $\log(n) - o(\log n)$ pairs of points in any instance of the bichromatic non-crossing matching problem formed by a set of n red points and a sequence of n blue points.*

Proof. Let $M(n)$ denote the number of matched pairs by Gm in the worst case in an instance formed by n blue and n red points (we have $M(1) = 1$). The algorithm matches the first blue point with the median of the red points. Consider the two sides of the line that passes through the matched pair. One of the two sides contains at least half of the future blue points, i.e., at least $\lfloor (n-1)/2 \rfloor$ blue points. There are also $\lfloor (n-1)/2 \rfloor$ red points on the same side (since the line bisects the red points). So, we have $M(n) \geq 1 + M(\lfloor \frac{n-1}{2} \rfloor)$ for $n > 1$, which solves to $M(n) \geq \log(n) - o(\log n)$. □

Although it is not difficult to match $\log n - o(\log n)$ points, as we now show, no online algorithm can guarantee to match more than $\log n - o(\log n)$ points.

Theorem 4 *Let* Alg *be any deterministic online algorithm for the bichromatic non-crossing matching problem. There are inputs formed by a fixed set of n red points and a sequence of n blue points for which* Alg *matches at most $\log n - o(\log n)$ points.*

Proof. We create an adversarial input in which n red points are placed in arbitrary positions on an arc of a



(a) (b) (c)

Figure 2: An illustration of updating the critical region (pink region) by the adversary in the proof of Theorem 2. The numbers on the points indicate their index in the input sequence. (a) Once points 1 and 2 are matched, there is no unmatched point in the critical region; the adversary generates point 3 and updates the critical region to its subregion that is not visible to 3. (b) Assume the algorithm does not match the next points 4, 5, 6, and 7. When it matches points 5 and 8, points in $S = \{4, 6\}$ are unmatched on one side of the line passing through 5 and 8. The adversary updates the critical region to be its subregion not visible by any member of S. (c) Assume the algorithm does not match the next point 9. When it matches points 10 and 7, the set $S = \{9\}$ is unmatched on one side of the line. The critical region is updated to be its subregion not visible to 9.

large circle so that they seem collinear except that the corresponding arc slightly curves outwards. The blue points appear in an online manner below the red point on a similar arc that slightly curves inwards; this arc is referred to as a *critical region* at the beginning, and is updated as the algorithm matches points. Assume at some point ALG matches an incoming blue point with a red point, and let L be the line that passes through the matched pair. The number of red points on one side of L is at most $\lfloor (n-1)/2 \rfloor$. The adversary updates the critical region to only include this side of L. This ensures that at least $\lceil (n-1)/2 \rceil$ red points on the other side of L remain unmatched; this is because the line segments between these points and all future blue points (generated in the updated critical region) crosses L (see Figure 3). So, each time ALG matches two points, the number of red points that can still be matched decreases by a factor of at least 2. Consequently, the number of matched pairs is at most $\log n - o(\log n)$. □

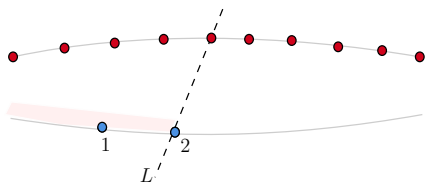


Figure 3: Updating the critical region by the adversary in the proof of Theorem 4. In the beginning, the critical region is the entire lower arc. Assume ALG does not match the first blue point but the second one is matched. The majority of red points appear on the right of the line L passing through the matched pair. As such, the adversary updates the critical region to be the left of L.

4 Non-Crossing Matching with Advice

In this section, we study the non-crossing matching problem under the advice model. We refer the reader to [5] for a survey on online algorithms with advice. Under the advice model, an online algorithm is provided with some bits of advice about the input sequence. The advice can encode any information about the input sequence, and is generated by a benevolent offline oracle that knows the entire input. A central question under the advice model asks for the number of advice bits necessary/sufficient to achieve an optimal solution. In the context of the non-crossing matching problem, this question translates to the number of advice bits needed to match all points.

4.1 Monochromatic setting

In this section, we study the monochromatic non-crossing matching problem under the advice setting. First, we show that $O(n)$ bits of advice is sufficient to match all the points.

Theorem 5 *There is an online algorithm that receives* $(\log_2 3)\ n + o(n) \leq 1.59n$ *bits of advice and matches all points (except one if* n *is odd) in any instance of the online monochromatic non-crossing matching problem on* n *points.*

Proof. Consider an offline matching that sorts the points by their x-coordinate and matches consecutive pairs of points. Call these pairs of matched points "partners". Note that all points are matched by this offline algorithm (except one if n is odd). Now, for each point p, we generate an advice $f(p) \in \{0, 1, 2\}$, based on this offline matching, as follows:

- when the partner of p appears after p in the online sequence, we define $f(p) = 0$.
- when the partner of p appears before p and is located to the left of p, we define $f(p) = 1$.
- when the partner of p appears before p and is located to the right of p, we define $f(p) = 2$.

So, the advice forms a string of length n over an alphabet of size 3. This can be encoded in $(\log_2 3)\ n + o(n) < 1.59n$ bits using, e.g., a wavelet tree structure [10].

It remains to show how to match points using the advice. Assume a point p arrives. If the advice encoded for p is 0, the algorithm keeps it unmatched as its partner has not arrived yet. If the advice is 1 or 2, then, p should be matched with the point with closest x-coordinate on its left or right, respectively. Using this scheme, we obtain a matching that is the same as the optimal offline solution. □

Lower Bound. In what follows, we show that advice of size $\Omega(\log n)$ bits is required in order to match all points in a given sequence of n points (assume n is even). Our lower bound argument generates sequences in which all points are on the circumference of a circle. In the offline setting, we can index the points, in clockwise order, starting from an arbitrary position. Any matching of a point with an even index to a point with an odd index divides the problem into two even-sized sub-problems, which can be solved recursively. Any such matching is equivalent to a balanced parenthesis sequence (see Figure 4). Consequently, in the offline setting, there are $C_{n/2}$ different ways to match all points, where $C_{n/2}$ is the $(n/2)$th Catalan number.

In order to provide a lower bound for the size of advice bits required to match all points, we create a family of $n-2$ input sequences of length n, denoted by $\sigma_1, \sigma_2, \ldots, \sigma_{n-2}$. All these sequences start with a common prefix $p_1, p_2, \ldots, p_{n-2}$, where the p_i's appear in clockwise order on the circumference of a circle. The last two points of any sequence σ_i are x_i and y_i, where x_i is a point located between p_{i-1} and p_i, and y_i is a point located between p_i and p_{i+1}.

Assume an online algorithm Alg (with advice) is applied on a sequence σ_i. Define a *partial matching* as the (incomplete) solution of Alg for the common prefix of the sequences in the family (the first $n-2$ points). In the partial solution, some points are matched, call them *partners*, and some are unmatched. A partial matching is said to be *valid* for σ_i, iff it can be completed such that all points in σ_i are matched at the end.

Lemma 6 *Any partial matching is valid for at most two sequences from the family.*

Proof. A valid partial matching for any sequence in the family should have exactly two unmatched points.

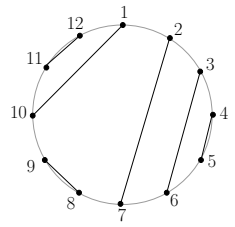


Figure 4: When points are located on the circumference of a circle, an offline algorithm can match all points by matching even-indexed points with odd-indexed points. The parentheses sequence associated with this matching is $(_1\ (_2\ (_3\ (_4\ )_5\ )_6\ )_7\ (_8\ )_9\ )_{10}\ (_{11}\ )_{12}$.

If more than two points are unmatched, some will stay unmatched at the end since only two more points from each sequence is left. If all points are matched, the last two points x_i and y_i in σ_i remain unmatched since the line segment between them crosses the line segment between p_i and its partner. So, we can consider a partial matching $S_{i,j}$ where two points p_i and p_j are unmatched. There are two cases to consider:

Case I: assume the line segment between p_i and p_j does not cross any line segment between matched pairs in $S_{i,j}$. We claim $S_{i,j}$ cannot be valid for any σ_k, where $k \notin \{i, j\}$. Consider a line L passing through p_k and its partner $p_{k'}$ in $S_{i,j}$. Both p_i and p_j appear on the same side of L. Among x_k and y_k, one appears on the same side of L while the other appears on the other side. In short, three unmatched points appear on one side of L and one on the other side (see Figure 5a). We cannot match all points without crossing L.

Case II: assume the line segment between p_i and p_j crosses a line segment L between p_k and its partner $p_{k'}$. So, p_i and p_j appear on different sides of L, which implies the remaining two points should be also on different sides of L to be matched with p_i and p_j. This is only possible for σ_k and $\sigma_{k'}$ (see Figure 5b). Note that if the line segment between p_i and p_j crosses more than one line segment in $S_{i,j}$, the same argument implies that the remaining points should be on the two sides of two existing line segments in $S_{i,j}$ at the same time, which is not possible (see Figure 5c). □

Using Lemma 6, we can prove the following lower bound on the size of advice required to match all points.

Theorem 7 *A deterministic algorithm requires advice of size at least $\lfloor \log((n-2)/3) \rfloor$ in order to guarantee matching all points in any instance of the online monochromatic non-crossing matching problem on n points.*

Proof. Assume, for the sake of a contradiction, that there is an algorithm Alg that matches all points

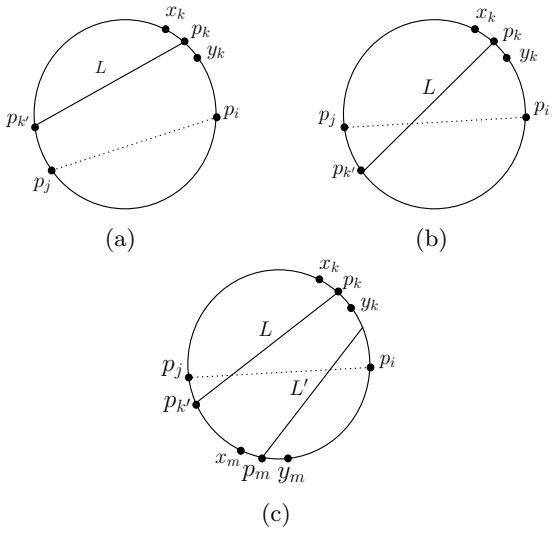


Figure 5: An illustration of Lemma 6. We have a partial matching $S_{i,j}$ where all points except p_i and p_j are matched. (a) if the line segment between p_i and p_j does not cross existing segments in $S_{i,j}$, it is not possible to match all points of any sequence σ_k for $k \notin \{i, j\}$. (b) if the line passing through p_i and p_j crosses a line segment between two matched points p_k and $p_{k'}$, then it might be possible to match the remaining points of σ_k and $\sigma_{k'}$. (c) if the line segment passing through p_i and p_j crosses two line segments L and L' between matched points, we cannot match the remaining two points.

in any instance of length n with less than $\alpha(n) = \lfloor \log((n-2)/3) \rfloor$ bits of advice. In particular, ALG should match all points for any sequence in the family $\{\sigma_1, \ldots, \sigma_{n-2}\}$ as we described above. We partition this set into $2^{\alpha(n)} \leq (n-2)/3$ *sub-families*, each formed by sequences that receive the same advice bits. Since there are $n-2$ sequences and at most $(n-2)/3$ sub-families, there is a sub-family with at least 3 sequences, that is, there are three sequences σ_a, σ_b, and σ_c that receive the same advice. Since these three sequences have the same common prefix and receive the same advice, ALG treats them similarly for the first $n-2$ points. That is, the partial matching of ALG is the same for all σ_a, σ_b, and σ_c. By Lemma 6, however, this partial matching is not valid for at least one of these sequences. We conclude that ALG cannot match all points for at least one sequence, a contradiction. □

4.2 Bichromatic setting

We show that advice of size $\Theta(n \log n)$ is both sufficient and necessary to match all points in the bichromatic setting. The more complicated nature of the bichromatic setting implies that advice of size of $\Theta(n)$ is insufficient (unlike the monochromatics setting) and, at the same time, simplifies our lower and upper bound arguments.

Theorem 8 *Consider any instance of the online bichromatic non-crossing matching problem with a sequence of n blue and a fixed set of n red points. There is a deterministic algorithm that receives $n\lceil \log n \rceil$ bits of advice and matches all points. Meanwhile, any deterministic algorithm requires advice of size at least $\lceil \log n! \rceil$ bits in order to match all points.*

Proof.
Upper bound: The offline oracle creates an ordering of the red points (say ordered by x-coordinate and ties broken by y-coordinate) and computes an optimal biochromatic matching on these. Now, for each blue point x, it encodes an advice of size $\lceil \log n \rceil$ that indicates the label of the red point to which x is matched. The online algorithm can mimic the offline matching by forming the same ordering of red points and matching each blue point to the red point indicated in the advice.
Lower bound: Consider instances of the problem in which the n red points $r_1, r_2, \ldots, r_n$ are placed, from left to right, on an arc of a large circle so that they seem collinear. The blue points $b_1, b_2, \ldots, b_n$ appear below the red points on an arc that slightly curves inwards (similar to Figure 3). In order to match all points, the left-most red point (r_1) should be matched with the left-most blue point (b_1). Using an inductive argument, we can show there is a unique matching of all points, where r_i is matched with b_i. Consider a family of $n!$ sequences, each associated with a permutation of the blue points $b_1, \ldots, b_n$ that indicates the order at which they appear in the online sequence. Let ALG be a deterministic online algorithm with less than $\lceil \log n! \rceil$ bits of advice. This implies that two sequences σ and σ' in the family receive the same advice. Assume the permutations associated with σ and σ' differ for the first time at index i, and let x be the i'th point in the input sequence. In σ, the point x is b_k and in σ' it is $b_{k'}$ for some $k \neq k'$. Since ALG is deterministic and receives the same advice for σ and σ', it matches x with the same red point in both cases. Such a matching, however, is not consistent with the unique optimal matching for at least one of the two sequences. As such some points remain unmatched in either σ or σ', and hence ALG fails to match all points. □

5 Concluding Remarks

Theorems 5 and 7 indicate that advice of size $O(n)$ and $\Omega(\log n)$ are respectively sufficient and necessary to match all points in the monochromatic setting. Closing the gap between these bounds does not seem to be easy and requires alternative techniques.

All algorithms studied in this paper are deterministic. We expect that randomization can improve the expected number of matched points which we propose as a direction for future research.

Acknowledgement

We thank the anonymous reviewers for their useful suggestions. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). NSERC funds were used for the research visit that resulted in publication of this paper.

References

[1] G. Aloupis, J. Cardinal, S. Collette, E. D. Demaine, M. L. Demaine, M. Dulieu, R. F. Monroy, V. Hart, F. Hurtado, S. Langerman, M. Saumell, C. Seara, and P. Taslakian. Non-crossing matchings of points with geometric objects. *Comput. Geom.*, 46(1):78–92, 2013.

[2] M. J. Atallah. A matching problem in the plane. *J. Comput. Syst. Sci.*, 31(1):63–70, 1985.

[3] H. Böckenhauer, D. Komm, R. Královic, and R. Královic. On the advice complexity of the k-server problem. *J. Comput. Syst. Sci.*, 86:159–170, 2017.

[4] P. Bose, M. E. Houle, and G. T. Toussaint. Every set of disjoint line segments admits a binary tree. *Discret. Comput. Geom.*, 26(3):387–410, 2001.

[5] J. Boyar, L. M. Favrholdt, C. Kudahl, K. S. Larsen, and J. W. Mikkelsen. Online algorithms with advice: A survey. *ACM Comput. Surv.*, 50(2):19:1–19:34, 2017.

[6] J. Boyar, S. Kamali, K. S. Larsen, and A. López-Ortiz. Online bin packing with advice. *Algorithmica*, 74(1):507–527, 2016.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[8] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[9] J. Erickson. https://mathoverflow.net/questions/86906. https://stackexchange.com/. Accessed: 2020-07-21.

[10] R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proc. 14th Symp. on Discrete Algorithms (SODA)*, pages 841–850, 2003.

[11] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT Comput. Sci. Sect.*, 32(2):249–267, 1992.

[12] J. Hershberger and S. Suri. Efficient breakout routing in printed circuit boards. In J. Boissonnat, editor, *Proc. 13th Annual Symposium on Computational Geometry (SOCG)*, pages 460–462. ACM, 1997.

[13] M. Hoffmann, B. Speckmann, and C. D. Tóth. Pointed binary encompassing trees: Simple and optimal. *Comput. Geom.*, 43(1):35–41, 2010.

[14] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane—a survey. *Discrete & Computational Geometry*, 25:551–570, 2003.

[15] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In H. Ortiz, editor, *STOC90*, pages 352–358. ACM, 1990.

[16] C. Lo, J. Matousek, and W. L. Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11:433–452, 1994.

[17] L. Lovász and M. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. North-Holland, 2009.

[18] D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208, 1985.

[19] P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18(6):1201–1225, 1989.