Non-Crossing Matching of Online Points

by

Arezoo Sajadpour

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

MASTER OF COMPUTE SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg

Contents

List of Figures

ACKNOWLEDGEMENTS

I would like acknowledge my academic advisor, Dr. Shahin Kamali for his mentorship during my master study. Because of his trust, I have had a chance to experience in various aspect of my life, and expand my knowledge. I would be always sincerely thankful because of these opportunities.

I would like to thank Dr. Stephane Durocher and Dr. Parimala Thulasiraman for accepting to be a member of my advisory committee. Their valuable advice and academic support assisted me to conduct this thesis.

In the end, I would like to thank my research team member Pooya Nikbakht, and all the others who have helped me to successfully finish my master's studies.

DEDICATION

For my beloved family who has always believed in me and support me.

ABSTRACT

In this thesis, we consider the non-crossing matching problem in the online setting. In the monochromatic setting, a sequence of n points in general position in the plane is revealed in an online manner. 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 decision to match each arriving point is 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. Each matched pair then consists of one red point and one blue point.

In a deterministic setting, where randomization is not allowed, we show that a simple greedy algorithm matches roughly $2n/3$ points in the monochromatic case, which is the best that any deterministic algorithm can achieve in the worst-case scenario. For the bichromatic variant, we prove that for every deterministic algorithm ALG there is a set of $n/2$ red points and $n/2$ blue points such that ALG matches at most $O(\log n)$ points, and there exist algorithms that match $\Omega(\log n)$ points for any set of $n/2$ red points and $n/2$ blue points. We also study the problem under the advice model, where the online algorithm receives some bits of advice about the input sequence. We prove upper and lower bounds for the number of advice bits sufficient and necessary to match all points.

We show that randomization helps in the monochromatic matching of the problem. That is, we introduce an algorithm that matches roughly $235n/351$ $(> n/3)$ points on expectation, which is an improvement over what deterministic algorithms can achieve. We also prove a lower bound that shows a linear number of points stay unmatched by any randomized algorithm. More precisely, we show that any randomized algorithm is expected to leave at least $0.0738n$ points unmatched in the worst-case scenarios.

Finally, we provide experimental results for the typical performance of online algorithms. We report the number of unmatched points when the coordinates of the input points are independently and identically distributed random variables.

Chapter 1

Introduction

Matching points in the plane is a fundamental topic in computational geometry and has applications in the real world ranging from image processing [11] to bioinformatics [15]. The input to the most basic version of the matching problem is a set of points in general position. There are different ways to define an objective to optimize. One might be interested in minimizing the maximum matching distance between any pair of matched points, e.g., as in the bottleneck matching problem [14], or in maximizing the number of matched points [3]. Points might be monochromatic, where any pair of points can be matched, or they might be coloured red or blue, and the matching can be made between the points of either different or the same colour, depending on the problem's definition.

In the non-crossing matching points problem, the goal is to find the maximum number of matched points where the line segments between the matched pairs do not cross. Ideally, it is desirable to achieve a perfect matching, where all points are matched (except possibly one if the number of points is odd).

In the offline setting of the problem, where all points are given in advance, it is possible to achieve a perfect matching for every set of points. The offline variant can be trivially solved to achieve a perfect matching in $O(n \log n)$ time for an input of size n. For example, in the monochromatic setting, one can sort points in non-decreasing order, say by the x- or the y-coordinate, and match consecutive points to achieve a perfect matching.

In the online setting, the points are revealed one by one, and each point should be matched to previously revealed points or left unmatched without any knowledge about the forthcoming points. As we will argue later, in general, it is not possible to achieve a perfect matching in the online setting. Instead, one should consider maximizing the number of matched points.

In this thesis, we are interested in the online or semi-online variants of the matching problem under different settings and models. As mentioned above, an online algorithm does not have any prior information about the input. Once a point is revealed, the algorithm decides to match it with an unmatched point or leave it unmatched. The decision of the online algorithm is irrevocable, meaning that if two points are matched, they remain matched. We are particularly interested in worst-case settings, where the input is generated in an adversarial manner.

It is possible to study online matching algorithms under a deterministic or randomized setting. In the deterministic setting, the input is formed by an adversary that knows the decisions made by the online algorithm. In the randomized setting [4], the online algorithm can use random bits to make decisions about matching points. Under this setting, an *oblivious adversary* knows how the algorithm works, that is, it has access to the code of the algorithm, but it does not know what points are matched, given that the online algorithm uses random bits that are not known to the adversary. As such, randomization is expected to improve the worst-case performance of the online algorithms compared to deterministic algorithms.

Applications of the matching problem

We study the non-crossing matching problem from a theoretical point of view. Regardless, matching geometric objects has many applications in practice [11, 15, 18, 21, 30, 22], some of which can be listed as follows.

- Image matching problem in computer vision is to compare two images and identify similar sub-regions. Cohen [11] studied the *pattern problem*, a relaxed version of the image matching problem, and provided a framework to find a similar pattern between an image and a query pattern. The pattern has both color and shape attributes.

- Another application of matching algorithms in computer vision is to search a database for images that are similar to a given image [18]. To this aim, features of each image in the database are extracted to form a set of weighted points that are matched with similar points from the queried image.

- In computational biology, pattern matching is a practical operation to find the locations of particular DNA sub-sequences in a DNA sequence [34], or small parts of molecules in a complete protein [15].

Analysis measures

Worst-case (competitive) analysis

We study online algorithms in the worst-case scenarios, where the input is generated in an adversarial manner. To illustrate the online matching problem in an adversarial setting, consider the following example: assume two points with the same x-coordinate appear one after the other. If an online algorithm does not match the two points, its solution is already sub-optimal for this input of size two. If the two points are matched, the adversary generates two more points on opposite sides of the line segment between the matched points. The new points cannot be matched, and the solution is now sub-optimal for this input of size four. This is because the optimal solution matches all four points.

The worst-case analysis is normally used to evaluate and compare online algorithms. Under the worst-case analysis, the number of unmatched points of an online algorithm is compared with the number of unmatched points of an optimal offline algorithm, Opt, which knows the input in advance. The *competitive ratio* of an online algorithm is then the maximum ratio between the number of matched points by Opt and that of an online algorithm for the same input. In the non-crossing matching problem, where all points are matched in the offline setting, the number of points matched by the online algorithm is compared to the input size (usually denoted by n). Note that an online algorithm by nature is not aware of the number of points in the input sequence, and its solution is evaluated for any sub-sequence of the input.

Worst-case analysis for randomized algorithms

As pointed out earlier, the online matching problem can be studied under randomized settings as well. A randomized algorithm uses random bits to make its decision. Therefore, there is no well-defined worst-case input for the randomized algorithm as it depends on the random bits used by the algorithm. As such, the performance of randomized algorithms on the worst-case input is measured by taking an expectation over all possibilities of random bits.

In a way, randomization helps an algorithm to “hide" its worst possible input from an adversary that generates the input. Randomization has been widely used to improve the worst-case performance of online algorithms. In particular, we will show that randomization can be helpful in the monochromatic setting of the non-crossing matching problem. We assume an adversary that is oblivious to the algorithm's random bits generates the input sequence. Our measure of performance is then the expected number of points that stay unmatched by the online algorithm.

Experimental analysis

In the last part of our study, we consider the average-case performance of the online algorithm. Unlike the worst-case analysis, where the input is generated by an adversary, under the average-case performance, the input is generated randomly. More precisely, we assume the $x-$ and $y-$coordinates of points are random variables that are generated independently at random, following identical probability distribution. The measure of performance is then the expected number of unmatched points. Clearly, it is expected that a smaller number of points stay unmatched when the input is randomly generated (under the average-case performance) compared to when it is generated by an adversary.

1.1 Advice model for online algorithms

In the purely online setting, an online algorithm has no prior information about the input sequence. Under the advice model, this restriction is relaxed, and an online algorithm receives some bits of "advice" about the input sequence. The advice bits can encode *any* information about the input sequence. In particular, they can encode the entire input (thus, reducing the problem to an offline problem). In general, the larger is the size of advice, the better is the performance attainable by an online algorithm. In the context of the non-crossing matching problem, it is particularly interesting to study how many bits of advice are sufficient (or necessary) to achieve an optimal matching.

1.2 Roadmap and contribution

This thesis is made up of seven chapters. Chapter 2 defines the problem in detail. Chapter 3 reviews previous work on the geometric matching problems under different settings and models.

The main contributions of this thesis can be divided into the following three parts.

- The first part, presented in Chapter 4, is devoted to deterministic algorithms. We provide upper- and lower-bounds for the number of unmatched points to prove that a simple greedy family of algorithms has the best worst-case performance among all deterministic algorithms (for both monochromatic and bichromatic settings). One takeaway is that the best algorithm for the monochromatic setting leaves roughly one-third of points unmatched, while the best bichromatic algorithm leaves almost all points ($n - o(n)$ points) unmatched in the worst-case scenarios. We also prove upper- and lower-bounds for the number of advice bits sufficient and necessary to match all points (in both monochromatic and bichromatic settings).

- The results in the second part of the thesis, presented in Chapter 5, concern the power of randomization for the non-crossing matching problems. We analyze how the randomization attribute improves the results of the matching points problem in the worst-case scenarios. We show that randomization helps in the monochromatic setting, as a smaller number of points are expected to stay unmatched if the algorithms are allowed to use randomization. For the bichromatic setting, however, randomization does not help, as almost all points (exactly $n-o(n)$ points) are expected to stay unmatched by any randomized algorithm. We also provide a lower bound for the expected number of unmatched points by any online algorithm.

- Finally, in the third part of the thesis, presented in Chapter 6, we experimentally study the typical performance of various algorithms when the input points are generated independently at random from an identical distribution. One takeaway from our experiments is that if the number of points, n, is known to the online algorithm, it is best to leave the first $f(n)$ points unmatched before starting to match points, where $f(n)$ is a linear function of n, e.g., $f(n) = n/5$.

Chapter 7 concludes the thesis and lists a few directions for future work.

Chapter 2

Problem statement

2.1 Input of the problem

The input of the geometric matching problems is a set of points in a general position in the plane that need to be matched. These points might represent an entire, or a part of the object, such as an image [18], a map [11], or a protein structure [15]. In some cases, points represent two distinct classes of objects, and it is needed to match points of opposite colors. That is, the input is divided into subsets of different colors [24] (see also [29]). This variation of the input is called bichromatic matching. The setting in which any two points can be matched is called monochromatic matching. We study both bichromatic and monochromatic cases in this thesis.

We assume the set of points that form the input are located in a general position in a plane, and are generated in an online manner. In the monochromatic case, all points have the same color, black. In the bichromatic case, half of the points are red, and the rest are blue.

2.2 Objective of the problem

The objective of the matching points problem is to match all points such that there is no unmatched point in an offline setting (except possibly one point if there is an odd number of points). Such matching is called a "perfect matching". Perfect matching for the offline non-crossing matching problem can be efficiently computed in $O(n \log n)$ time in both monochromatic or bichromatic cases. To achieve a perfect matching in the monochromatic setting, one can sort points based on their x-coordinate and then match consecutive pairs. For the bichromatic case, an optimal offline algorithm finds the ham-sandwich line that bisects the blue and red points in $O(n)$ time [31], and applies a divide-and-conquer approach.

In the online setting, where the input points are revealed in an online and sequential manner, it is not possible to match all points. As such, the main objective is to achieve matchings that are as close as possible to a perfect matching. That is, the goal is to minimize the number of unmatched points (alternatively, to maximize the number of matched points).

In the non-chromatic setting, we define the online non-crossing matching problem as follows.

Definition 1. *The input to the* ***monochromatic*** *setting of the* online non-crossing matching problem *is a sequence of points in general position in the plane that are revealed in an online, sequential manner. Note that the number of points, n, is known to the online algorithm. When a point p is revealed, an online algorithm can match p 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 p (hoping to match it later with a forthcoming point). The objective is to create a matching in which a maximum number of points are matched. Equivalently, the goal is to minimize the number of unmatched points.*

Similarly, the bichromatic setting is defined as follows:

Definition 2. *The input to the* ***bichromatic*** *setting of the* online non-crossing matching problem *is a sequence of points in general position in the plane that are revealed online. Half of the points are blue, and half are red. When a point p is revealed, an online algorithm can match p with an existing unmatched point of the opposite color, provided that the line segment between them does not cross previous line segments added to the matching. Alternatively, the algorithm can leave p (hoping to match it later with a forthcoming point). The objective is to create a matching in which a maximum number of points are matched. Equivalently, the goal is to minimize the number of unmatched points.*

From the perspective of an online algorithm, it is much harder to generate quality solutions for the bichromatic setting when compared to the monochromatic setting. As such, when studying the bichromatic setting, we sometimes consider a relaxed setting in which all red points appear before the first blue point. Clearly, this restricted input is easier to handle, given that for the first half of the input, an online algorithm does not need to make any decision (no two red points can be matched).

Chapter 3

Literature review

Overview

This chapter of the thesis reviews the existing literature on the geometric matching problems, as well as different models and settings that are relevant to our purpose. In Section 3.1, the previous works on some of the related matching problems are reviewed. Online algorithms under the advice model are reviewed in Section 3.2. Finally, some of the relevant results on randomized algorithms are reviewed in Section 3.3.

3.1 Relevant problems

Due to the importance of the matching problem in computational geometry and graph theory, there has been a wide range of previous works on this topic. The following is a list of problems that we find most related to our work.

- The bichromatic non-crossing matching problem was first defined by Atallah [3]. It was assumed that the input was formed by n red and n blue points, and the goal is to find a perfect, non-crossing matching between red and blue points. To this aim, a deterministic algorithm achieves a perfect matching in $O(n \log^2 n)$ time as presented. This problem was later popularized as the “Ghosts and Ghostbusters" problem in the book by Coffman et al. [12].

- Aloupis et al. [1] considered an offline problem in which input is formed by a pair (P, O), where P is a set of points, and O is a set of geometric objects such as convex polygons, line segments, and lines. The goal is to create a non-crossing matching between the points in P and objects in O. On the other hand, specific instances of the problem, e.g., where P is a set of line-segments positioned on the same line, are solvable in polynomial time [1].

- Vaidya [36] considered the problem of a minimum-weight perfect matching in a geometric graph G defined by a set of points as follows. Given a set P of n points, form a complete graph that has a vertex associated with each point. Let the weight of the edge connecting each two vertices (points) as the pairwise Manhattan (or Euclidean) distance between the points. Given that G is a complete graph, it is easy to find a perfect matching in G. In the minimum-weight perfect matching, however, the goal is to find a perfect matching with the smallest weight [36]. In other word, the problem asks for a matching between a set of points so as the maximum length of the line segments that connect matched pairs is minimized. Vaidya shows that such matching can be found in $O(n^{2.5}(\log n)^4)$ time, if the distance is measured by Manhattan or Euclidean distance. In the bichromatic case, half of the points are red, and half are blue. In this case, the goal is to create a matching between them to minimize the maximum distance between any pair of matched points (the underlying graph can be thought of as a complete bipartite graph). The optimal matching can be found in $O(n^{2.5}\log n)$ time for the Euclidean distance and in $O(n^2(\log n)^3)$ time for the Manhattan distance.

- Many graph matching problems have been studied in the online setting. In the online bipartite matching problem, vertices in one side of a bipartite graph are available at the beginning, and vertices on the other side appear in an online manner (along with their connections to the offline part). The goal is to match vertices of the second part with their neighbors in the first part to maximize the number of matched pairs. It is needless to say that the decision of the online algorithm is irrevocable and once a pair of vertices are matched, they stay matched throughout. It is easy to see that a greedy algorithm that matches a new vertex whenever possible has a competitive ratio of 2, and it is the best that a deterministic algorithm can achieve. Karp et al. [25] presented the randomized algorithm Rank, which assigns a random ordering to the vertices that are present at the beginning. Upon arrival of a vertex in the online side, it is matched with an unmatched neighbor with the highest priority (if any neighbor is unmatched). Karp et al. [25] showed that Rank has a competitive ratio of 1.58, which is the best a randomized algorithm can achieve. Khuller et al. [26] studied an extension of the problem to the weighted bipartite graphs, where edges are weighted, and the goal is to maximize the total weight of edges between matched pairs. There are many other works that are focused on online matching in graphs with certain structures, see, e.g., [16, 27, 28].

We also note that the offline matching point problem has several applications in different fields of science and engineering, like image processing [11], pattern recognition [37], mapping problems [14], and bioinformatics [15].

3.2 Advice model

In general, an online algorithm serves each request at the arriving time without any knowledge about forthcoming requests. In practice, however, online algorithms might have some information, in the form of some bits of advice, about future input, which improves the performance of the algorithms. Under the advice model, we study the number of advice bits that is necessary and sufficient for online algorithms to perform optimally (i.e., achieve a competitive ratio of 1). The standard advice model, introduced by Böckenhauer et al. [6], assumes the advice bits are written on an advice tape which is available to the online algorithm before starting to serve the input sequence. Moreover, it is assumed that the algorithms understand the meaning of advice bits (i.e., it is not needed to include information about the meaning of advice bits in the tape). We refer to the survey by Boyar et al. [9] for different advice models and problems that are studied under this setting. In particular, the online bipartite matching problem has been studied under the advice model [33], where it is proved that $O(n!)$ advice bits are necessary and sufficient to achieve an optimal matching for a graph with partition size n.

The classic advice model assumes that the offline oracle that generates advice is *trusted* in the sense that the advice that it generates is always consistent with what it represents. Angelopoulos et al. [2] introduced *untrusted* advice model, where the advice is generated by an untrusted oracle who provided untrusted advice. In this case, an online algorithm should be *consistent*, that is, it performs better than a purely online algorithm in case the advice is correct, and also *robust*, in the sense that if the advice is wrong, its competitive ratio is not worse than the purely online algorithm.

In this thesis, we study the matching online point problem under the assumption that the advice bits are written on an advice tape, and that the advice is trusted.

3.3 Randomization

As mentioned earlier, randomization can help improve the competitive ratio of online algorithms. For example, the competitive ratio of the deterministic greedy algorithms for the bipartite matching problem can be improved from 2 to 1.58 with the randomized algorithm Rank that associates priorities to vertices in the offline partition [25].

To analyze the algorithm, we are interested in the worst-case input generated by an *adversary*. An adversary can be “adaptive" or “oblivious". An adaptive adversary either knows the entire random bits used by the online algorithm beforehand (an adaptive offline adversary) or learns about the random bits sequentially (an adaptive online adversary). Regardless, an adaptive adversary can “adapt" its worst-case input based on the outcomes of the random bits used by the algorithm. In contrast, an oblivious adversary is unaware of random bits and has to create a worst-case input solely based on its knowledge of (the code of) the algorithm, but not the outcomes of the random bits. Throughout this thesis, we assume the adversary is oblivious and analyze randomized algorithms accordingly. This is because adaptive adversaries are too powerful in terms of learning random bits, and it is often not possible to derive results better than deterministic algorithms against such adversaries [5].

The advice model and randomization are closely related, e.g., Dürr et al. [13] compared randomization and advice settings for the bipartite matching problem and provided a method that takes advantage of both advice and random choices at the same time. Mikkelsen [32] introduced a technique to interpret randomized online algorithms to deterministic online algorithms with a linear number of advice bits. The given method is applied on all problems modeled as Metrical Task System (MTS), e.g., paging, list update, k-server, dynamic binary search tree, and metric matching problems.

Chapter 4

Deterministic algorithms for the online non-crossing matching problem[1]

Overview

In this chapter, we study deterministic online algorithms for the non-crossing matching problem. Recall that, under 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 maximal matching of these points such that the line segments connecting pairs of matched points do not cross. 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.

Contribution. Our contributions in this chapter can be summarized as follows.

- For the monochromatic setting (Section 4.1):
  - 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.

[1]A summary of the results in this chapter is published in the proceedings of the 32nd Canadian Conference in Computational Geometry (CCCG 2020) [7].

  - 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 (Section 4.2):
  - We introduce an algorithm that matches at least $\log n - o(\log n)$ points for any input sequence formed by n red and n blue points in a relaxed setting in which all red points appear before the first blue point.
  - We show this algorithm is optimal as 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.

- We study the problem under the advice setting (Section 4.3):
  - For the monochromatic version of the problem, 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.

4.1 Monochromatic non-crossing matching

This section contributes in providing tight bounds on the number of matched points in monochromatic non-crossing matching problem. The input consists of a set of one color points that are generated by an adversary in an online manner.

4.1.1 Upper bound

An online algorithm is said to have a *greedy property* if and only if it never leaves a point unmatched if it has an option to match it. We provide an upper bound for the number of unmatched points by an algorithm with greedy property. Equivalently, this gives a lower bound for the number of matched points by such an algorithm.

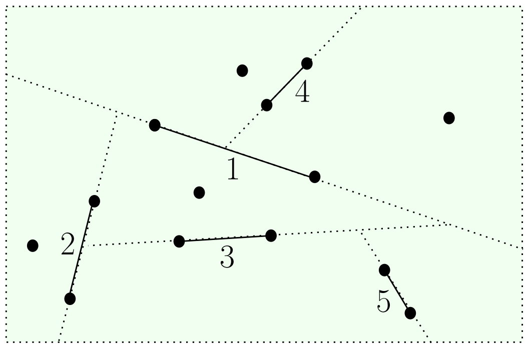


Figure 4.1: A partition of the plane into convex regions in the analysis of a greedy algorithm. The numbers on the line segments indicate the order they are processed in the analysis.

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 4.1). For detailed geometric properties of this convex subdivision, see, e.g., [8, 23]. 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 have $u \leq m+1$. The statement follows from the fact that $u + 2m = n$. $\square$

4.1.2 Lower bound

Next, we provide a lower bound for the number of unmatched points by *any* online algorithms in the worst-case scenarios. Equivalently, this would be an upper bound for the number of matched points by any online algorithm.

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. The critical region is updated as soon as the algorithm matches two points a and b. 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 4.5). Since 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 $u + 2m = n$. □

Given that the upper bound of Theorem 1 for the number of unmatched points of a greedy algorithm matches the lower bound of Theorem 2 for any online algorithm, we conclude that the greedy algorithm is the optimal deterministic algorithm for the non-crossing matching problem.

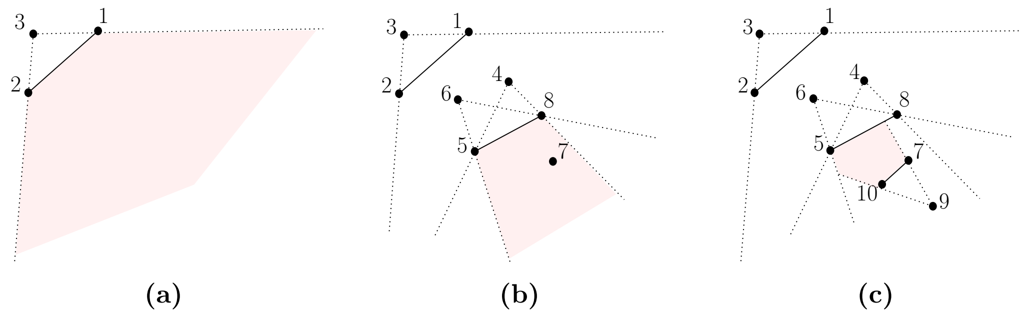


Figure 4.2: An illustration of updating the critical region (pink region) by the adversary in the proof of Theorem 2. The numbers on the points indicate their index in the input sequence. (a) Once points 1 and 2 are matched, there is no unmatched point in the critical region. The adversary generates 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.

4.2 Bichromatic non-crossing matching

In this section, we study deterministic online algorithms for the bichromatic non-crossing matching problem.

In the light of the above result, we consider a setting in which the input is formed by n red points 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. Under this relaxed setting, we provide tight upper and lower bounds for the number of unmatched points in the worst-case scenarios.

4.2.1 Upper bound

Consider an online algorithm, named the *Greedy Median* (GM) algorithm, that works as follows. Upon the 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 order 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 $M(n) \geq \log(n) - o(\log n)$. □

4.2.2 Lower bound

Although it is not difficult to match $\log n - o(\log n)$ points, as GM does, the following theorem shows that no online algorithm can guarantee to match more than $\log n - o(\log n)$ points.

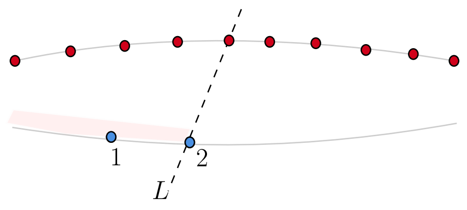


Figure 4.3: Updating the critical region by the adversary in the proof of Theorem 4. In the beginning, the critical region is the entire lower arc. Assume ALG does not match the first blue point but the second one is matched. The majority of red points appear on the right of the line L passing through the matched pair. As such, the adversary updates the critical region to be the left of L.

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 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) cross L (see Figure 4.3). 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 $\log n - o(\log n)$. □

4.3 Non-crossing matching with advice

In this section, we study the non-crossing matching problem under the advice model. Recall that 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.

4.3.1 Monochromatic setting

The first step is to show that $O(n)$ bits of advice is sufficient to match all the points. Next, we prove that the advice of size $O(n)$ is required to match all points.

Upper bound on the advice size

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 [19].

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 the 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 on the advice size

In what follows, we show that advice of size $\Omega(\log n)$ bits is required 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 4.4). 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.

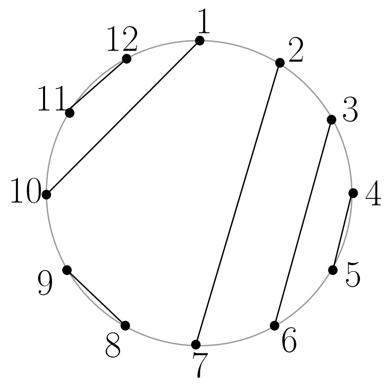


Figure 4.4: When points are located on the circumference of a circle, an offline algorithm can match all points by matching even-indexed points with odd-indexed points. The parentheses sequence associated with this matching is $(_1\ (_2\ (_3\ (_4\ )_5\ )_6\ )_7\ (_8\ )_9\ )_{10}\ (_{11}\ )_{12}$.

Lemma 1. 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. If more than two points are unmatched, some will stay unmatched at the end since only two more points from each sequence are 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 4.5a). 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 4.5b). 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 4.5c). □

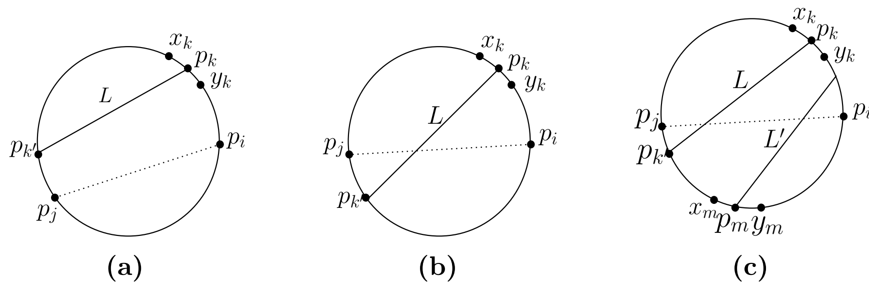


Figure 4.5: An illustration of Lemma 1. We have a partial matching $S_{i,j}$ where all points except p_i and p_j are matched. (a) if the line segment between p_i and p_j does not cross existing segments in $S_{i,j}$, it is not possible to match all points of any sequence σ_k for $k \notin \{i, j\}$. (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.

Using Lemma 1, we can prove the following lower bound on the size of advice required to match all points.

Theorem 6. 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 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 1, 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. □

4.3.2 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 monochromatic setting) and, at the same time, simplifies our lower and upper bound arguments.

Theorem 7. 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 bichromatic 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 4.3). 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. $\square$

Chapter 5

Randomized algorithms for the online non-crossing matching problem

Overview

This chapter of the thesis discusses the non-crossing matching problem under the randomization model for both monochromatic and bichromatic cases. We study worst-case scenarios, where the input is generated by an adversary that is oblivious to the random choices made by the algorithm, but it is aware of how the algorithm works (that is, the code of the algorithm). There are two main components in our randomized algorithm for the monochromatic setting.

Contribution. Our contributions can be summarized as follows:

- For the monochromatic setting:
  - We present a randomized algorithm that leaves at most $116n/351+202/351 \approx 0.3304n+0.5754$ unmatched points on expectation. This shows the advantage of randomized algorithms over deterministic algorithms, which leave at least roughly $0.\bar{3}\bar{3}n$ points in the worst case.
  - We show that a linear number of points are expected to remain unmatched by any randomized algorithms. Precisely, we show that any randomized algorithm leaves more than $(9-\sqrt{57})n/(3\sqrt{57}-3) \gtrapprox 0.0738\text{n}$ points unmatched on expectation for the sequences generated by an oblivious adversary.

- For the bichromatic setting, we show that randomization provides little improvement over deterministic algorithms. In particular, no randomized algorithm can match more than $O(\log n)$ points on expectation, even under a relaxed setting where all red points appear before the first blue point.

Overview of techniques. Our algorithm for the monochromatic setting has two attributes. First, it maintains a convex partitioning of the plane, and matches two points only if they appear in the same partition. This is followed by updating the partitioning by extending the edge between the matched pair. This partitioning enables us to use a simple inductive argument to analyze the algorithm. Second, the algorithm deviates from the greedy strategy. In particular, the algorithm gives a chance for an incoming point x to stay unmatched even if there are one or two points in the same convex region that it can match. As we will see, this will be essential for any improvement over deterministic algorithms. The lower bounds for both monochromatic and bichromatic settings are based on Yao's principle [38], where adversarial arguments are used for defining a probability distribution for input sequences. We will explain how an adversary can define such distributions to maximize the number of unmatched points. In both lower-bound arguments, the input points appear on the circumference of a circle.

5.1 Monochromatic Setting

In this section, we first present a randomized algorithm for the single-pair non-crossing matching problem (Section 5.1.1). This is followed by a lower bound that shows any randomized algorithm for the monochromatic non-crossing matching problem is expected to leave a linear number of points unmatched in the worst case (Section 5.1.2).

5.1.1 Upper bound

In this section, we present a randomized algorithm for the monochromatic non-crossing matching problem. In what follows, we use L_{ab} to denote the line passing through a and b, and S_{ab} to denote the line segment between a and b.

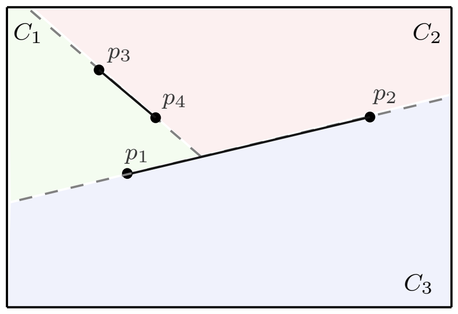
Algorithm's description

The algorithm maintains a partitioning of the plane into convex regions, and matches points only if they belong to the same region. At the beginning, there is only one region that is formed by the entire plane. After four points appear inside a convex region, one or two pairs of points are matched, and the convex region is partitioned into two or three convex regions by extending the line segments passing through the matched pairs.

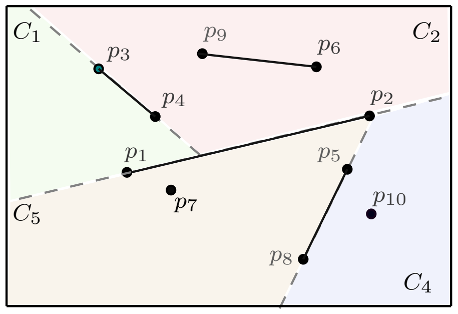
Let x, y, z, and w be the first four points inside a convex region C (in the same order). In what follows, we describe how these four points are treated.

- Upon the arrival of x, there is no decision to make, given that there is no point inside C to be matched with x.
- Upon the arrival of y, it is matched with x with a chance of $1/2$, and stays unmatched with a chance of $1/2$.
- Upon the arrival of z, if the pair (x, y) is already matched, then there is no decision to make. Otherwise, z is matched with x with a chance of $1/3$, with y with a chance of $1/3$, and stays unmatched with a chance of $1/3$.
- Upon the arrival of w, there are two possibilities to consider:
  - First, suppose a pair of points $a, b \in \{x, y, z\}$ is already matched, while a third point $c \in \{x, y, z\}/\{a, b\}$ is unmatched. If it is possible to match w with c (that is, S_{wc} does not cross S_{ab}), then w is matched with c; otherwise, when S_{wc} and S_{ab} cross, there is no decision to make.
  - Second, suppose no pair of the first three points are matched. Then w is matched with a point $a \in \{x, y, z\}$ so that the two points $b, c \in \{x, y, z\}/\{a\}$ appear on different sides of the line L_{aw} (if there is more than one such point, w is matched with z).

After the arrival of four points inside C, either all points are matched into two pairs, in which case we say a "double-pair" is realized, or only two points are matched while the other two appear on different sides of the matched pair, in which case we say a "single-pair" is realized. If a single-pair is realized, the line segment between the matched pair is extended until it hits the boundary of C; in this case, C is partitioned into two convex regions. If a double-pair is realized, the line segment between any of the matched pairs is extended until it hits the boundary of C or the (non-extended) segment between the other matched pair. This is followed by extending the line segment between the second pair until it hits the boundary of C, or extending the line that passes through the first matched pair. When a double-pair is realized, C is



(a) The state of the algorithm after processing $p_1, \dots, p_4$.



(b) The state of the algorithm after processing $p_1, \dots, p_{10}$.

Figure 5.1: One possible output of the algorithm when the input is a sequence of 10 points labeled as $p_1, \dots, p_{10}$ in the order of their appearance.

partitioned into three convex regions. The following example indicates the algorithm's steps.

Consider an input formed by 10 points labeled from p_1 to p_{10} in the order of their appearance, as depicted in Figure 5.1. The convex regions maintained by the algorithm are highlighted in different colors. Initially, the entire plane is a convex region C_0, where the points p_1, p_2, p_3, and p_4 appear. Upon the arrival of p_2, the algorithm match it with p_1 with a chance of $1/2$. Suppose (p_1, p_2) are matched. Then, there is no decision to be made for p_3. Upon the arrival of p_4, the line segments $S_{p_1p_2}$ and $S_{p_3p_4}$ do not cross. Therefore, p_4 is matched with p_3. At this point, four points have appeared in C_0 and a double-pair (p_1, p_2) and (p_3, p_4) has been realized. Therefore, C_0 is partitioned into three smaller convex regions C_1, C_2, and C_3 by extending S_{p_1,p_2} and then S_{p_3,p_4} (Figure 5.1a). Points p_5 and p_6 appear respectively in C_3 and C_2. Since these are the first points in their respective regions, there is no decision to be made, and they stay unmatched. Subsequently, p_7 appears in C_3 and the algorithm might match it to p_5 with a chance of $1/2$. Suppose these two points are not matched. Upon the arrival of p_8 in C_3, it is matched with p_5 or p_7, each with a chance of $1/3$, and is left unmatched with a chance of $1/3$. Suppose (p_5, p_8) are matched. Next, point p_9 appears in C_2 and is matched with p_6 with a chance of $1/2$, and stays unmatched with a chance of $1/2$. Suppose (p_6, p_9) are matched. Finally, point p_{10} appears on C_3. Given that the $S_{p_7p_{10}}$ crosses $S_{p_5p_8}$, there is no decision to be made, and p_{10} stays unmatched. At this point, four points have appeared in C_3, and a single-pair (p_5, p_8) has been realized. Therefore, C_3 is partitioned into two smaller convex regions C_4 and C_5 by extending S_{p_5,p_8} (Figure 5.1b).

Algorithm's analysis

Let $f(n)$ denote the expected number of unmatched points left by the algorithm when input is formed by n items. We use an inductive argument to find an upper bound for $f(n)$. First, we prove the following lemma, which is used when establishing the base of the induction.

Lemma 2. After four points arrived in the convex region C, with a chance of at least $1/3$, a double-pair is realized, and with a chance of at most $2/3$, a single-pair is realized.

Proof. Let x, y, z, and w denote the four points in the same order they appear. There are two cases to consider:

- Suppose S_{xy} crosses S_{wz}. With a chance of $1/2$, x and y are not matched. After that, with a chance of $2/3$, z is matched to x or y. Without loss of generality, assume z is matched with x. Given that S_{xy} crosses S_{wz}, line segments S_{xz} and S_{yw} will not cross, implying that w is matched to y, and a double-pair is realized. So, with a chance of at least $1/2 \cdot 2/3 = 1/3$, all points are matched, and a double-pair is realized.

- Suppose S_{xy} does not cross S_{xz}. Then, (x, y) are matched with a chance of $1/2$, and after that, (w, z) are matched, and a double-pair is realized.

□

Using Lemma 2, we can establish an upper bound for $f(n)$ for small values of n.

Lemma 3. We have $f(0) = 0$, $f(1) = 1$, $f(2) = 1$, $f(3) = 4/3$, $f(4) \leq 4/3, f(5) \leq 5/3, f(6) \leq 20/9$, and $f(7) \leq 52/18$.

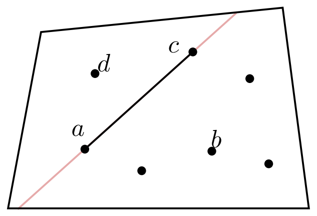
Proof. Suppose n items appear in a convex region C. The proof is trivial for $n \leq 2$. In what follows, we prove the lemma for other values of n.

- For $n = 3$, it is possible that all points stay unmatched, which happens when the second point is not matched with the first one (with a chance of $1/2$), and then the third point is not matched with any of the first two points (with a chance of $1/3$). Therefore, with a chance of $1/6$, all three points stay unmatched, and one point stays unmatched with a chance of $5/6$. We can write $f(3) = 1/6 \cdot 3 + 5/6 \cdot 1 = 4/3$.

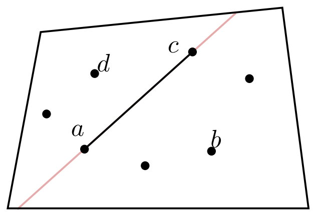
- For $n = 4$, using Lemma 2, we can write $f(4) \leq 1/3 \cdot 0 + 2/3 \cdot 2 = 4/3$.
- For $n = 5$, after the first four points appeared, either a single-pair or a double-pair is realized:
  - Suppose a single-pair is realized. Then, C is partitioned into two regions, one containing one point and the other one containing two points. Therefore, it is expected that $f(1) + f(2) = 2$ points stay unmatched.
  - Suppose a double-pair is realized. Then, the first four points are matched, and only the fifth point stays unmatched.

  By Lemma 2, with a chance of at least 1/3, a double-pair is realized, and with a chance of at most 2/3, a single-pair is realized. Therefore, we can write $f(5) \leq 1/3 \cdot 1 + 2/3 \cdot 2 = 5/3$.
- For $n = 6$, after the first four points appeared, either a single-pair or a double-pair is realized:
  - Suppose a single-pair is realized. Then, C is partitioned into two regions. Either (i) the fifth or the sixth points appear on the same region, in which case one region will have one point, and the other one will have three points, or (ii) the fifth and the sixth points appear in different regions, in which case each region contains two points. Therefore, it is expected that at most $\max\{f(1) + f(3), f(2) + f(2)\} = 7/3$ points stay unmatched.
  - Suppose a double-pair is realized. Then, at most 2 points (the last two points) stay unmatched.

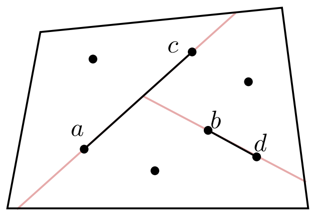
  By Lemma 2, with a chance of at least 1/3, a double-pair is realized, and with a chance of at most 2/3, a single-pair is realized. Therefore, we can write $f(6) \leq 1/3 \cdot 2 + 2/3 \cdot 7/3 = 20/9$.
- For $n = 7$, after the first four points appeared, either a single-pair or a double-pair is realized:
  - Suppose a single-pair is realized. Then, C is partitioned into two regions. Either (i) the fifth, the sixth, and the seventh points all appear in the same region, in which case one region has one point, and the other one has four points (Figure 5.2a), or (ii) one of these points appear in one region, and the



(a) The case where a single-pair is realized, and the last three points appear in different regions.



(b) The case where a single-pair is realized, and the last three points appear in different regions.



(c) The case where a double-pair is realized, and the last three points appear in different regions.

Figure 5.2: The cases used in the calculation of $f(7)$; $a, b, c, d \in \{x, y, z, w\}$ where x, y, z, and w are the first four points in the same order of their appearance.

other two appear in the other region, in which case one region contains two points, and the other region contains three points (Figure 5.2b). Therefore, it is expected that at most $\max\{f(1)+f(4), f(2)+f(3)\} \leq \max\{1+4/3, 1+4/3\} = 7/3$ points stay unmatched.

– Suppose a double-pair is realized. Then, at most three points stay unmatched, which happens when any of the three regions formed by partitioning of the first four points includes a single point (see Figure 5.2c).

Unlike other cases, here, the expected number of unmatched points is larger when a double-pair is realized, and hence we cannot use Lemma 2. Instead, we note that the chance of a single-pair being realized is at least $1/6$ This is because a single-pair is realized if either (i) the first two points are matched with a chance of $1/2$, and the other two points appear on opposite sides of the line passing through the matched points, happening with a total chance of $1/2$, (ii) the first two points are not matched with a chance of $1/2$, and the third point is matched to either of the first points with a chance of $1/3$, and the fourth point appears on the side of the matched line that the other unmatched point is not on, happening with a total chance of $1/6$, or (iii) the first three points stay unmatched with a chance of $1/2 \cdot 1/3 = 1/6$, and then the fourth point gets match to the point that bisects the unmatched points, happening with a total chance of $1/6$. Therefore, we can write $f(7) \leq 5/6 \cdot 3 + 1/6 \cdot 7/3 = 52/18$ (see Figure 5.2).

□

In what follows, we show that for $n \geq 2$, we have $f(n) \leq cn + d$ where $c =$

$116/351 \approx 0.3304$ and $d = 32c - 10 = 202/351 \approx 0.5754$.

To prove this claim, we use an inductive argument. For the base of the induction, we prove the following lemma.

Lemma 4. For $n \in [2, 7]$, it holds that $f(n) \le cn + d$ where $c = 116/351$ and $d = 202/351$.

Proof. The proof follows from Lemma 3. For $n = 2$, we have $f(2) = 1 < 2c+d$ (since $2c + d > 1.2362$). For $n = 3$, we have $f(3) = 4/3 = 3c + d$ (since $3c + d > 1.5669$). For $n = 4$, we have $f(4) \le 4/3 < 4c + d$ (since $4c + d > 1.8974$). For $n = 5$, we have $f(5) \le 5/3 < 5c+d$ (since $5c+d > 2.2279$). For $n = 6$, we have $f(6) \le 20/9 < 6c+d$ (since $6c + d > 2.5584$). For $n = 7$, we have $f(7) \le 52/18 = 7c + d$ (note that $7c + d = 52/18$). □

Assume $n \ge 8$. A single-pair is "good" if after the appearance of all n points, both of the two regions resulted from extending the line segment of the matching contain at least 2 points, and it is "bad" otherwise. A double-pair is said to be "good" if after the appearance of all n points, one of the three regions formed by extending the line segments of the two matchings is empty; otherwise, it is "bad".

Lemma 5. For $n \ge 8$, after serving the first four points inside a convex region, at least one of the followings hold:

- There is a good single-pair, and it is realized with a chance of at least $1/6$
- There is a good double-pair, and it is realized with a chance of at least $1/6$.

Proof. Let x, y, z, and w denote the first four points in the same order that they appear.

First, suppose the convex hull formed by the four points is a triangle Δ which includes the fourth point inside it. We consider the following two cases:

- Assume w is the point that is inside Δ. Then the pairs (x, y) and (w, z) form a double-pair which is realized with a chance of $1/2$. This is because the pair (x, y) is matched with a chance of $1/2$, and then the pair (w, z) is matched with a chance of 1. Meanwhile, (w, z) is a single-pair which is realized with a chance of $1/6$. This is because, with a chance of $1/6$, the first three points stay unmatched, and then the algorithm matches w to z with a chance of 1. Now, if the double pair formed by the pairs (x, y) and (w, z) is bad, then there should be at least one future point on each side of the line passing through (w, z), which means (w, z) is a good single-pair (see Figure 5.3a).

- Assume w is a vertex of Δ and another point $c \in \{x, y, z\}$ is inside Δ. Let a, b be the other two points in $\{x, y, z\}$. Then, the pairs (a, b) and (c, w) form a double-pair which is realized with a chance of at least $1/6$. This is because the pair (a, b) is matched with a chance of at least $1/6$ (the pair (a, b) is matched with a chance of $1/2$ if $z \notin \{a, b\}$, and with a chance of $1/6$ if $z \in \{a, b\}$), and then w is matched with c with a chance of 1. Meanwhile, the pair (c, w) is a single-pair which is realized with a chance of $1/6$. Similar to the previous case, if the double pair formed by the pairs (a, b) and (c, w) is bad, then there should be at least one future point on each side of (a, b), which means (a, b) is a good single-pair (see Figure 5.3b).

Next, suppose the convex hull formed by the four points is a quadrilateral and includes all of them. Consider the two single-pairs formed by the diagonals of the convex hull. Any of these pairs can be realized with a chance of at least $1/6$. Specifically, the diagonal involving w is realized when no pair of points from $\{x, y, z\}$ are matched, which takes place with a chance of $1/6$. The other diagonal is either between x and y, which is realized with a chance of $1/2$, or between z and $a \in \{x, y\}$, which is realized with a chance of $1/6$. Therefore, if any of the two diagonal forms a good single-pair, the statement of the lemma holds, and we are done (see Figure 5.3c). If

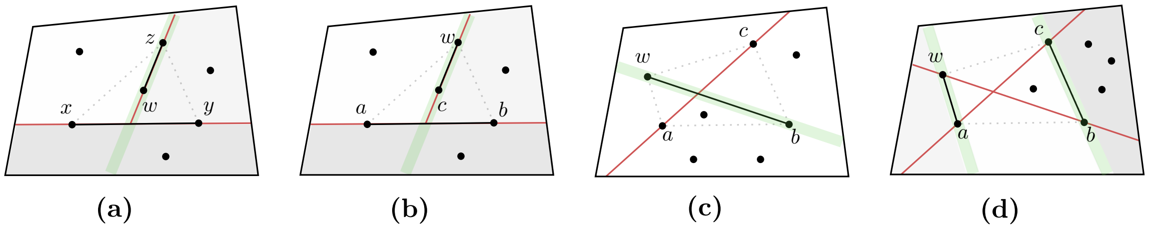


Figure 5.3: An illustration of the proof of Lemma 5. (a) when w is inside the triangle Δ, either the single-pair formed by (w, z) is a good single-pair, or the double-pair formed by $(x, y), (w, z)$ is a good double-pair. (b) when $c \in \{x, y, z\}$ is inside the triangle Δ, either the double pair formed by $(a, b), (w, c)$ is a good double-pair, or the single-pair formed by (w, c) is a good single-pair. (c) the case when at least one of the diagonals of the convex hull formed by the four points (here (w, b)) forms a good single-pair (d) when none of the single-pairs formed by the diagonals of the convex hull are good, all remaining points appear in one of the quarter-planes formed by extending these diagonals; therefore, the pair of points on the boundary of the quarter-plane (here (b, c)) and the pair of points outside the quarter-planes (here (w, a)) form a good double-pair.

none of the two diagonals is good, then all the remaining points in the input sequence should appear in one of the quarter-planes formed by extending these diagonals (see Figure 5.3d). Then, the double-pair formed by the pair of points on the boundary of the quarter-plane (points b and c in Figure 5.3d) and the pair of points outside of the quarter-plain (points w and a in Figure 5.3d) form a good double-pair. The chance of such a double-pair to be realized is at least $1/6$. This is because one of the pairs in the double-pair involves two of the first three points. If these points are (x, y), the double-pair is realized with a chance of $1/2$; otherwise, it is realized with a chance of $1/6$. □

We are now ready to prove the main result.

Theorem 8. Our randomized algorithm, for any input formed by $n \geq 2$ points, leaves at most $cn + d$ points unmatched, where $c = 116/351$ and $d = 202/351$.

Proof. We use an inductive argument. For $n \leq 7$, the claim holds by Lemma 3. Suppose $n \geq 8$, and assume that for any $m < n$, it holds that $f(m) \leq cm + d$.

First, we claim that the number of unmatched points is at most $cn + d + (2 - 6c)$ when a bad single-pair is realized, or a bad double-pair is realized after the first four points appear. If a bad single-pair is realized, then either (I) there is one point on one side of the matched pair and $n - 3 > 2$ points on the other side, or (II) there is no point on one side of the matched pair and $n - 2 > 2$ points on the other side. For (I), by the induction hypothesis, the number of unmatched points on the side with $n - 3$ points will be at most $f(n - 3) \leq cn - 3c + d$. Therefore, the number of unmatched points is at most $f(n - 3) + 1 \leq cn - 3c + d + 1 < cn + d + (2 - 6c)$. The last inequality holds because $c < 1/3$. For (II), the number of unmatched points will be at most $f(n - 2) \leq cn + d - 2c < cn + d + (2 - 6c)$. If a double-pair is realized which is not good, then one of the followings holds for the three regions formed by extending the line segments between the matched pairs:

i) One region contains $n - 6$ points, and the other two regions each contains one point. Note that $n - 6 \geq 2$ since $n \geq 8$. By the induction hypothesis, the number of unmatched points is at most $2 + f(n - 6) = cn + d + (2 - 6c)$.

ii) One region contains $m \geq 2$ points, another region contains one point, and the third region contains $n - m - 5 \geq 2$ points. The number of unmatched points is at most $f(m) + f(n - m - 5) + 1 \leq cn - 5c + 2d + 1 < cn + d + (2 - 6c)$. The last inequality holds because $c + d < 1$.

iii) one region contains $m_1 \geq 2$ points, one region contains $m_2 \geq 2$ points, and the third region contains $m_3 = n - m_1 - m_2 - 4 \geq 2$ points. The number of unmatched points is at most $f(m_1) + f(m_2) + f(m_3) \leq cn - 4c + 3d < cn + d + (2 - 6c)$. The last inequality holds because $c + d < 1$.

In summary, if a bad single-pair or a bad double-pair is realized, the number of unmatched points is at most $cn + d + (2 - 6c)$, and the claim holds.

By Lemma 5, after the appearance of the first four points, either a) a good pair or b) a good double-pair can be realized with a chance of at least $1/6$.

Suppose case a) holds, that is, a good single-pair is realized with a chance of at least $1/6$, which implies a bad single-pair or double-pair is realized with a chance of at most $5/6$. In case the good single-pair is realized, there will be $m \geq 2$ points on one side of the line segment connecting matched pair, and $n-m-2 \geq 2$ points on the other side. Therefore, the number of unmatched points will be at most $f(m) + f(n - m - 2) \leq cn+2d-2c = (cn+d)+(d-2c)$. On expectation, the number of unmatched points will be at most $1/6((cn+d)+(d-2c))+5/6(cn+d+(2-6c)) = cn+d+1/6(d-32c+10) = cn + d$. The last equality holds because $d = 32c - 10$.

Next, suppose case b) holds, that is, a good double-pair is realized with a chance of at least $1/6$, which implies a bad single-pair or double-pair is realized with a chance of at most $5/6$. In case the good double-pair is realized, by definition, at least one of the three convex regions formed by extending the double-pair will be empty. For the other two regions, we have the following cases:

i) One region is empty, and the other contains $n-4 \geq 2$ points, in which case the number of unmatched points becomes $f(n-4) \leq cn+d-4c < cn+d+(1-5c)$. The last inequality holds because $c < 1$.

ii) One region contains a single point, and the other one contains $n-5 \geq 2$ points. The number of unmatched points will be at most $f(n-5)+1 \leq cn+d+(1-5c)$.

iii) Both regions include $m \geq 2$ and $n-m-4 \geq 2$ points. In this case, the number of unmatched points will be at most $f(m)+f(n-m-4) \leq cn+d+(d-4c) < cn + d + (1 - 5c)$. The last inequality holds because $c + d < 1$.

Therefore, as long as the good double-pair is realized, the number of unmatched points will be at most $cn + d + (1 - 5c)$. On expectation, we can write $f(n) \leq 1/6((cn+d)+(1-5c))+5/6((cn+d)+(2-6c)) = cn+d+1/6(11-35c) < cn+d$. The last inequality holds since $c > 11/35$. $\square$

5.1.2 Lower bound

This section describes a probability distribution for the input sequence of size n and shows that any deterministic algorithm is expected to leave at least $0.0738n$ points unmatched. By Yao's principle, the expected number of unmatched points by any randomized algorithm on the worst-case input is then no better than $0.0738n$. Throughout, we assume n is divisible by 3.

Input distribution

Let S be a random sequence formed $n/3$ pairs (i, j) of integers, where $i = 1$ with a chance of $p = (9 - \sqrt{57})/4 \approx 0.362$, $i = 2$ with a chance of $1 - 2p$, and $i = 3$ with a chance of p. The values of j are generated uniformly and independently at random from the range $[1, 4]$. Note that there are $12^{n/3}$ possibilities for S. From S, we generate an input sequence $\sigma(S)$ for the monochromatic matching as follows. All points in $\sigma(S)$ appear on the circumference of a semicircle C, which is positioned horizontally with diameter up. There are $n/3$ *phases* in $\sigma(S)$, each formed by 3 points. The points of each phase are generated on a *critical arc* which is an arc in C. Initially, the entire circumference of C is the critical arc. At the end of each phase, the critical arc shrinks and becomes a sub-arc of what it used to be. Let C_t denote the critical arc at the beginning of phase t, and let (i, j) denote the t'th pair in S. The first two points of phase t appear on arbitrary positions in C_t. We denote these two points by a_t and b_t so that, without loss of generality, b_t is located on the right side of a_t. The location of the third point c_t is decided by the value of i. Precisely, c_t appears on an arbitrary position on the left of a_t when $i = 1$, on an arbitrary position between a_t and b_t when $i = 2$, and on an arbitrary position on the right of b_t when $i = 3$. After the three points are revealed, depending on the value of j, the critical arc is updated. That is, c_{t+1} is defined to be its sub-arc on the left of the leftmost point in the phase when $j = 1$, the sub-arc between the leftmost point and the middle point when $j = 2$, the sub-arc between the middle point and the rightmost points when $j = 3$, and the sub-arc on the right of the rightmost point when $j = 4$. Figure 5.4 shows the input sequence for $S = (1, 2), (3, 4), (2, 2)$.

Proof outline and basic observations

We will prove a lower bound of $pn/(6-3p) = (9-\sqrt{57})n/(3\sqrt{57}-3) \gtrapprox 0.0738\text{n}$ for the expected number of unmatched points left by any randomized algorithm. By Yao's

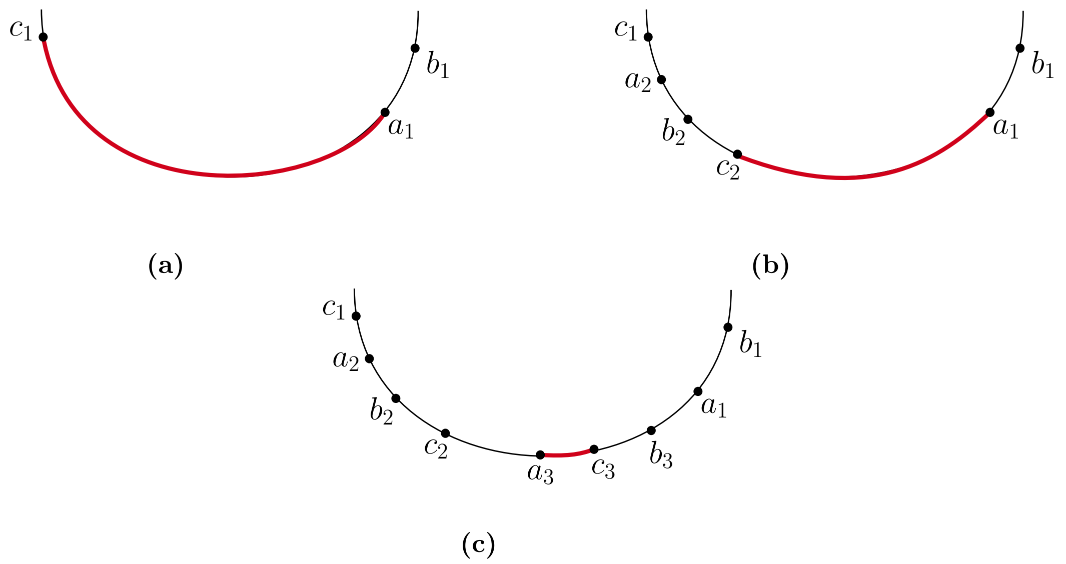


Figure 5.4: An illustration of the input sequence $\sigma(S)$ for $S = (1,2),(3,4),(2,2)$. (a) phase 1, where c_1 appears on the left side of a_1 (since $i = 1$), and the critical arc is updated to the area between c_1 and a_1 (since $j = 2$). (b) phase 2, where c_2 appears on the right side of b_2 (since $i = 3$), and the critical arc is updated to the rightmost area (since $j = 4$). (c) phase 3, where c_3 is between a_3 and b_3 (since $i = 2$), and the critical arc is updated to the area between a_3 and c_3 (since $j = 2$).

principle, it suffices to show that the expected number of unmatched points by any deterministic algorithm for a random input $\sigma(S)$ of n points is at least $pn/(6-3p)$.

Let A be any deterministic randomized algorithm. At any given time, the set of points that have appeared so far can be partitioned into the following three sets (see Figure 5.5 for an illustration):

- M: the set of points that are already matched by A.
- U_g: the set of points that are unmatched and are guaranteed to stay unmatched throughout the matching process. More precisely, the line segment between any point in U_g and any point in the critical arc C_t (any future point) crosses a line segment between a pair of points that are already matched.
- U_u: the set of points that are unmatched and undecided, that is they might end up being matched with some of the points that will appear in the future.

Let m_t and u_t respectively denote the expected increase in the size of M and U_g in phase t. Once a point is added to M and U_g, it will stay in that set till the end; therefore, we have $m_t, u_t \geq 0$. Note that $m_0 = u_0 = 0$. In the proof, we will show that if we have $m_t > 0$ for a phase t, then $u_t \geq m_t(p/(6-4p))$. The fraction of the unmatched points over all points at the end of the execution of A will be:

$$\begin{aligned}\frac{|U_g| + |U_u|}{|U_g| + |U_u| + |M|} \geq \frac{|U_g|}{|U_g| + |M|} &= \frac{\sum_t u_t}{\sum_t u_t + \sum_t m_t} \\ &\geq \frac{\sum_t m_t(p/(6-4p))}{\sum_t m_t(p/(6-4p)) + \sum_t m_t} = p/(6-3p)\end{aligned}$$

Side-arcs: Suppose A matches the first point a_t of phase t with some point a' (which appeared in one of the previous phases). Define the *side-arc* of a_t as follows. If a' appears on the left of a_t in the semi-circle C, then the side-arc of a_t will be the arc between a' and a_t; otherwise, it will be the entire sub-arc of C on the left of a_t. Symmetrically, if A matches b_t with some point b' from a previous phase, then the side-arc of b_t is the arc between b_t and b' if b' appears on the right of b_t and the entire sub-arc of C on the right of b_t otherwise. Figure 5.6 provides an illustration of side-arcs and the following observation.

Observation 1. The following holds when a deterministic algorithm A matches a_t and/or b_t at phase t with some unmatched point from the previous phases:

- Suppose A matches a_t with a point a', and there is an unmatched point x on the side-arc of a_t. If the critical arc is updated to be on the right of a_t, then the line segment between x and any point in the updated critical arc crosses the line segment between a_t and a'.

- Suppose A matches b_t with a point b', and there is an unmatched point y on

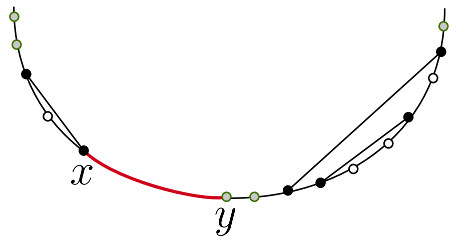


Figure 5.5: The state of a deterministic algorithm A after five phases. The critical arc is the arc between x and y. The set M contains the six matched points (black points). The set U_g contains the four white points. Given that all future points will appear in the critical arc, these points are guaranteed to stay unmatched. The set U_u is formed by the gray points, which are currently unmatched but might be matched with a point that is not revealed yet.

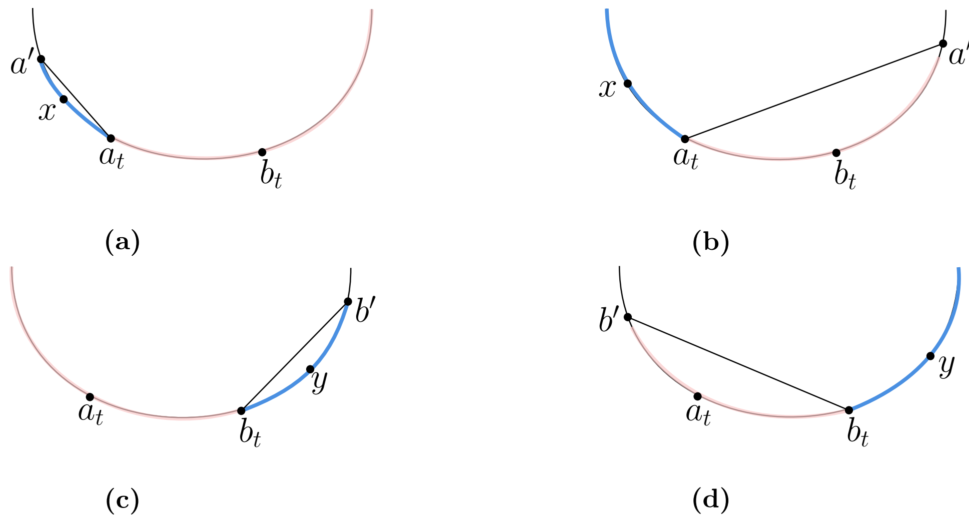


Figure 5.6: An illustration of Observation 1. The arcs highlighted in blue are the side-arcs of a_t (Figures (a) and (b)), and b_t (Figures (c) and (d)). Suppose a_t (respectively b_t) is matched with a point a' (respectively b') and the critical arc is updated to be within the arc highlighted in pink. Observation 1 implies that a point in the blue arc cannot be matched with a point in the pink arc.

the side-arc of b_t. If the critical arc is updated to be on the left of b_t, then the line segment between y and any point in the updated critical arc crosses the line segment between b_t and b'.

Main result via case analysis

We are now ready to prove the main result in this section.

Theorem 9. Any randomized algorithm for the online monochromatic non-crossing matching is expected to leave at least $(9-\sqrt{57})n/(3\sqrt{57}-3) \gtrapprox 0.0738\text{n}$ points unmatched in the worst-case.

Proof. In the light of the discussion in the previous section, it suffices to prove $u_t \geq m_t(p/(6-4p)) \approx 0.0795 m_t$ for any phase t. For that, we will use a case analysis that concerns how a_t and b_t are treated by A, and whether there is an unmatched point in U_u on the left of a_t or on the right of b_t.

In all cases, at most three pairs of points (involving a_t, b_t, and c_t) are added to M, that is $m_t \leq 6$. In some cases, however, we can derive $m_t \leq q$ for some $q < 6$, which is more desirable result when establishing our lower bound for the competitive ratio. We assume that at least one pair is matched at phase t; otherwise, we will have $m_t = u_t = 0$, and hence $u_t \geq m_t(p/(6-4p))$ holds.

- **Case I:** Suppose A matches a_t with a point a' and b_t with a point b', where a' and b' are two points from previous phases. We consider the following sub-cases. See Figure 5.7 for an illustration of Case I.
  - Sub-case I-a: Suppose there is at least one unmatched point $x \in U_u$ on the side-arc of a_t, and at least one unmatched point $y \in U_u$ on the side-arc of b_t. We consider the following possibilities, depending on the location of c_t relative to a_t and b_t.
    * Suppose c_t appears on the left of a_t. With a chance of 3/4, the critical arc is updated to be on the left of b_t. By Observation 1, any line segment between y and a point in the updated critical arc crosses the line segment between b_t and its b'. Therefore, y will be added to U_g. So, we can write $u_t \geq 3/4 \cdot 1 = 3/4$.
    * Suppose c_t appears on the right of b_t. With a symmetric argument as in the previous case, with a chance of 3/4, x is added to U_g, and we can write $u_t \geq 3/4$.
    * Suppose c_t appears between a_t and b_t. With a chance of 1/4, the critical arc is updated to be on the left of a_t, and by Observation 1, y will be added to U_g. Symmetrically, by a chance of 1/4, the critical arc is updated to the right of b_t, and x will be added to U_g. With a chance of 1/2, the critical arc is updated to the area between a_t and b_t. In this case, Observation 1 can be applied for both x and y, adding both of them to U_g. So, we will have $u_t \geq 1/4 \cdot 1 + 1/4 \cdot 1 + 1/2 \cdot 2 = 3/2$.

    The chance of c_t appearing on the left of a_t or on the right of b_t is each p, and the chance of c_t appearing between a_t and b_t is $1 - 2p$. So, on expectation, we can write $u_t \geq p \cdot 3/4 + p \cdot 3/4 + (1-2p) \cdot 3/2 = (3-3p)/2$. Given that $m_t \leq 6$, we will have $u_t/m_t \geq (1-p)/4$, which is larger than $p/(6-4p)$ (we have $(1-p)/4 \gtrapprox 0.15 > 0.0795$). Note that we cannot derive a tighter bound for m_t, given that it is possible that c_t is matched with some other points from previous phases.
  - Sub-case I-b: Suppose there is at least one unmatched point $x \in U_u$ on the side-arc of a_t while there is no point from U_u on the side-arc of b_t. We consider the following cases, depending on the location of c_t relative to a_t and b_t.

* Suppose c_t appears on the left of a_t. We can write $m_t \leq 6$, and $u_t \geq 0$ (this holds for any phase regardless of how it unfolds).
* Suppose c_t appears between a_t and b_t. With a chance of $3/4$, the critical arc is updated to be on the right of a_t, and by Observation 1, x will be added to U_g, that is $u_t \geq 3/4$. It also holds that $m_t \leq 6$.
* Suppose c_t appears on the right of b_t. Given that the side-arc of b_t is empty, c_t cannot be matched with any point at phase t. Therefore, only a_t, b_t, and their matched points are added to M, and we have $m_t \leq 4$. Now, if the critical arc is updated on the left of a_t (and hence left of b_t), then by Observation 1, c_t is added to U_g; this is because c_t is on the side-arc of b_t. If the critical arc is updated to be on the right of a_t, by Observation 1, point x will be added to U_g. Therefore, we will have $u_t \geq 1$ regardless of how the critical arc is updated.

Therefore, on expectation, we can write $m_t \leq p \cdot 6 + (1-2p) \cdot 6 + p \cdot 4 = 6 - 2p$ and $u_t \geq p \cdot 0 + (1-2p) \cdot 3/4 + p \cdot 1 = (3-2p)/4$. Therefore, we have $u_t/m_t \geq \frac{(3-2p)/4}{(6-2p)} = (3-2p)/(24-8p)$, which is larger than $p/(6-4p)$ (we have $(3-2p)/(24-8p) \gtrapprox 0.10 > 0.0795$).

– Sub-case I-c: Suppose there is at least one unmatched point $y \in U_u$ on the

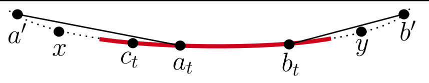
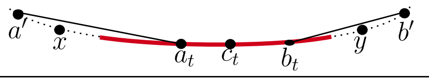
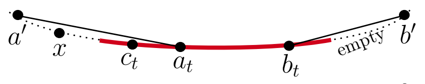
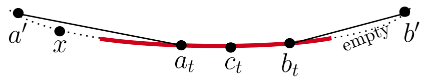
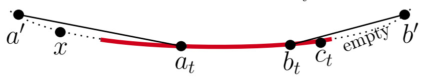
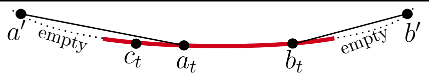
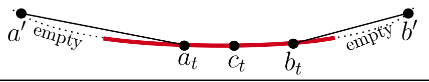
case		illustration	$m_t \leq$	$u_t \geq$	$u_t/m_t \geq$
I	I-a		6	3/4	$\frac{1-p}{4}$
			6	3/2	
	I-b		6	0	$\frac{3-2p}{24-8p}$
			6	3/4	
			4	1	
	I-d		4	1/2	$\frac{p}{6-4p}$
			6	0	

Figure 5.7: A summary of the analysis for Case I. The cases that are handled symmetrically are excluded from the figure.

side-arc of b_t while there is no point from U_u on the side-arc of a_t. This case is symmetric to the case I-b, and we have $u_t/m_t \geq (3-2p)/(24-8p) > p/(6-4p)$.

- Sub-case I-d: Suppose there is no unmatched point in U_u on the side-arc of a_t nor on the side-arc of b_t. Note that if c_t appears in these side-arcs, it cannot be matched during phase t.

  We consider the following cases, depending on the location of c_t relative to a_t and b_t.

  * Suppose c_t appears on the side-arc of a_t. Then, only a_t, b_t, and their matched points are added to M during phase t, and we will have $m_t = 4$. If the critical arc is updated to the right of a_t, which happens with a chance of $1/2$, then, the line segment between c_t and any point in the updated critical arc crosses the line segment between a_t and a'. Therefore, c_t is added to U_g. We can write $u_t \geq 1/2 \cdot 1 = 1/2$.
  * Suppose c_t appears on the side-arc of b_t. With a symmetric argument as in the previous case, we have $m_t = 4$ and $u_t \geq 1/2$.
  * Suppose c_t does not appear on the side-arc of a_t nor on the side-arc of b_t. Then, given that the critical arc is a continuous arc, c_t should appear between a_t and b_t (note that a' and b' do not belong to the critical arc and thus any point on the left of a' or on the right of b' does not belong to the critical arc as well). In this case, we can write $m_t \leq 6$ and $u_t \geq 0$.

  So, on expectation, we have $m_t = p \cdot 4 + p \cdot 4 + (1-2p) \cdot 6 = 6 - 4p$ and $u_t \geq p \cdot 1/2 + p \cdot 1/2 + (1-2p) \cdot 0 = p$. Therefore, we have $u_t/m_t \geq \frac{6-4p}{p}$.

- **Case II**: Suppose A matches a_t with some point a' from previous phases while b_t is unmatched when c_t appears. In this case, a_t and its matched point are added to M while b_t belongs to U_u when c_t appears. Note that c_t might be matched with b_t or another member of U_u. Regardless, the number of matched pairs at this phase will be at most 2, and we have $m_t \leq 4$. See Figure 5.8 for an illustration of Case II.

  - Sub-case II-a: Suppose there is at least one unmatched point $x \in U_u$ on the side-arc of a_t. We consider the following cases, depending on the location of c_t relative to a_t and b_t.

  * Suppose c_t appears on the left of a_t. If the critical arc is updated to be on the left of a_t, which happens with a chance of $1/2$, then the line segment between b_t and any point in the updated critical arc crosses the line segment between a_t and its matched pair. Therefore, b_t is added to U_g. We can write $u_t \geq 1/2 \cdot 1 = 1/2$.
  * Suppose c_t appears on the right of a_t. If the critical arc is updated to be on the right of a_t, which happens with a chance of $3/4$, then the line segment between x and any point in the updated critical arc would cross the line segment between a_t and its matched pair. Therefore x is added to U_g. We can write $u_t \geq 3/4 \cdot 1 = 3/4$.

  The chances of c_t appearing on the left and right of a_t are respectively p and $(1-2p)+(p) = 1-p$. So, on expectation, we can write $u_t \geq p \cdot 1/2 + (1-p) \cdot 3/4 = (3-p)/4$. Recall that $m_t \leq 4$ in this case. Therefore, we have $u_t/m_t \geq \frac{(3-p)/4}{4} = (3-p)/16$, which is larger than $p/(6-4p)$ (we have $(3-p)/16 \gtrapprox 0.16 > 0.0795$).

– Sub-case II-b: Suppose there is no unmatched point in U_u on the side-arc of a_t. We consider the following cases, depending on the location of c_t relative to a_t and b_t.

  * Suppose c_t appears on the left a_t. In this case, c_t cannot be matched with any point during the phase t (note that there is still a chance for c_t to get matched with a point in the future phases if the critical arc is updated to be on the left of a_t). So at phase t, at most two points, a_t and its matched pair, are added to M, and we have $m_t = 2$. As for u_t, if the critical arc is updated to be on the left (respectively right) of a_t, then the line segment between b_t (respectively c_t) and any point on the updated critical arc crosses the line segment between a_t and its matched pair, and b_t (respectively c_t) is added to U_g. Therefore, one point from $\{b_t, c_t\}$ is added to U_g, and we have $u_t \geq 1$
  * Suppose c_t appears on the right of a_t. Then it is possible that no point is added to U_g (when c_t is matched with b_t).

  The chances of c_t appearing on the left and right of a_t are respectively p and $1-p$. So, on expectation, we can write $m_t \leq p \cdot 2 + (1-p) \cdot 4 = 4-2p$, and $u_t \geq p \cdot 1 + (1-p) \cdot 0 = p$. Therefore, we have $u_t/m_t \geq \frac{p}{4-2p}$, which is larger than $p/(6-4p)$ (we have $p/(4-2p) \gtrapprox 0.11 > 0.0795$).

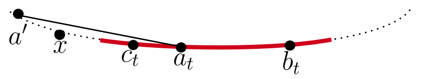
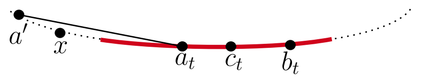
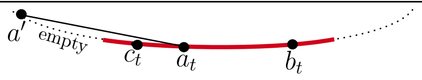
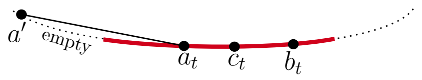
case		illustration	$m_t \leq$	$u_t \geq$	$u_t/m_t \geq$
II	II-a		4	1/2	$\frac{3-p}{16}$
			4	3/4	
	II-b		2	1	$\frac{p}{4-2p}$
			4	0	

Figure 5.8: A summary of the analysis for Case II. Case III is symmetric to Case II.

- **Case III**: Suppose a_t is unmatched when c_t appears while A matches b_t with some point from previous phases. This case is symmetric to case II, and we have $u_t/m_t \geq \min\{(3-p)/16, p/(4-2p)\} > p/(6-4p)$.

- **Case IV:** Suppose that A matches (a_t, b_t). See Figure 5.9 for an illustration of Case IV.

  - Sub-case IV-a: Suppose the set U_u is non-empty, that is, there is a point $x \in U_u$. Note that before the three points of the current phase t are being revealed, the updated critical arc is empty, and hence x appears either on the left of a_t or on the right of b_t. We consider the following cases, depending on the location of c_t relative to a_t and b_t.

    * Suppose c_t appears on the left of a_t or on the right of b_t. In both cases, it is possible that no point is added to U_g, given that c_t can be matched with x. There will be up to two pairs of matched points (that is, (a_t, b_t) and a possible pair involving c_t, e.g., (c_t, x)) which are added to M. So we have $m_t \leq 4$ and $u_t \geq 0$.

    * Suppose c_t appears between a_t and b_t. Given that all points in U_u are located on either on the left of a_t or on the right of b_t, the point c_t cannot be matched with an unmatched point in U_u. Therefore, the only points added to M during phase t is (a_t, b_t), and we have $m_t = 2$. As for u_t, if the critical arc is updated to be on the left of a_t or on the right of b_t (with a chance of $1/4 + 1/4 = 1/2$), then the line segment between c_t and any point in the updated critical arc crosses the line segment between (a_t, b_t). Therefore, c_t is added to U_g. If the critical

arc is updated to be between (a_t, b_t) (with a chance of $1/2$), then the line segment between x and any point in the critical arc crosses the line segment between (a_t, b_t). Therefore, x is added to U_g. So, we can write $u_t \geq 1/2 \cdot 1 + 1/2 \cdot 1 = 1$.

The chance of c_t appearing on the left of a_t or on the right of b_t is $2p$, and the chance of it being in between a_t and b_t is $1-2p$. So, on expectation, we can write $m_t \leq 2p{\cdot}4+(1-2p){\cdot}2 = 2+4p$, and $u_t \geq 2p{\cdot}0+(1-2p){\cdot}1 = 1-2p$. Therefore, we have $u_t/m_t \geq \frac{1-2p}{2+4p}$, which is equal to $p/(6-4p)$. This is because p is set to be the answer to equation $(1-2p)/(2+4p) = p/(6-4p)$.

– Sub-case IV-b: Suppose U_u is empty. Given that there is no point in U_u that c_t can be matched to and that a_t and b_t are already matched, c_t will not be matched with another point at phase t. Therefore, the only pair of matched points added to M are (a_t, b_t), that is, $m_t = 2$. We consider the following cases, depending on the location of c_t relative to a_t and b_t.

  * Suppose c_t appears on the left of a_t or on the right of b_t.
    If the critical arc is updated to be between a_t and b_t, which happens with a chance of $1/4$, then the line segment between c_t and any point in the critical arc crosses the line segment between (a_t, b_t). Therefore, c_t is added to U_g, and we have $u_t \geq 1/4 \cdot 1 = 1/4$.
  * Suppose c_t appears between a_t and b_t. If the critical arc is updated to be on the left of a_t or on the right of b_t, which happens with a chance of $1/4 + 1/4 = 1/2$, then the line segment between c_t and any point in the updated critical arc crosses the line segment between (a_t, b_t). Therefore, c_t is added to U_g.
    We can write $u_t \geq 1/2 \cdot 1 = 1/2$.

The chance of c_t appearing on the left of a_t or on the right of b_t is $2p$, and the chance of it being in between a_t and b_t is $1-2p$. So, on expectation, we can write $u_t \geq 2p \cdot 1/4 + (1-2p) \cdot 1/2 = (1-p)/2$. Recall that $m_t = 2$ in Case IV-b. Therefore, we have $u_t/m_t \geq \frac{(1-p)/2}{2} = (1-p)/4$, which is larger than $p/(6-4p)$ (we have $(1-p)/4 \gtrapprox 0.15 > 0.0795$).

- **Case V:** Suppose both a_t and b_t are unmatched at the time c_t arrives. Given that at least one pair of points are matched at phase t (according to the assumption made throughout the case analysis), when c_t arrives, it should be matched

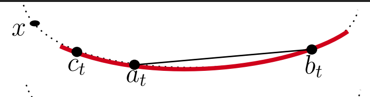
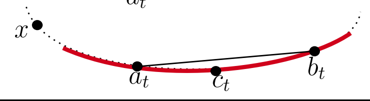
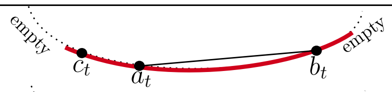
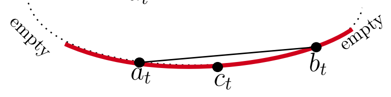
case		illustration	$m_t \leq$	$u_t \geq$	$u_t/m_t \geq$
IV	IV-a		4	0	$\frac{1-2p}{2+4p}$
			2	1	
	IV-b		2	1/4	$\frac{1-p}{4}$
			2	1/2	

Figure 5.9: A summary of the analysis for Case IV.

with some other point. Note that at phase t only this pair is added to M in Case V, that is $m_t = 2$.

– Sub-case V-a: Suppose c_t is matched with some $x \in U_u$ from previous phases ($x \notin \{a_t, b_t\}$). See Figure 5.10 for an illustration of Case V. We consider the following cases:

  * Sub-case V-a-1: Suppose c_t finds a_t and b_t on the same side of itself, say on its left side The case where a_t and b_t are both on the right side of c_t is handled symmetrically. If the critical arc is updated to be on the right of c_t, which happens with a chance of $1/4$, then the line segments between a_t or b_t and any point in the updated critical arc would cross the line segment between c_t and its matched pair. Therefore, both a_t and b_t are added to U_g. We can write $u_t \geq 1/4 \cdot 2 = 1/2$. Given that $m_t = 2$ in Case V, we have $u_t/m_t \geq \frac{1/2}{2} = 1/4$, which is larger than $p/(6-4p) \approx 0.0795$.
  * Sub-case V-a-2: Suppose c_t finds a_t on its left side and b_t on its right side (recall that a_t is always on the left of b_t by assumption). If the critical arc is updated to be on the left (respectively right) of c_t, then the line segment between b_t (respectively a_t) and any point in the updated critical arc crosses the line segment between c_t and its matched pair. Therefore, b_t (respectively a_t) is added to U_g. Thus, regardless of how the critical arc is updated, one of a_t or b_t is added to U_g, and we have $u_t \geq 1$. Given that $m_t = 2$, we have $u_t/m_t \geq 1/2$, which is larger than $p/(6-4p) \approx 0.0795$.

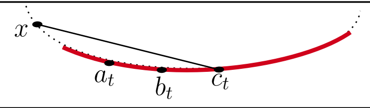
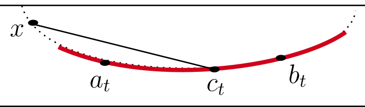
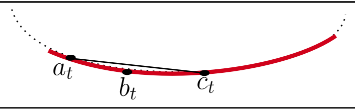
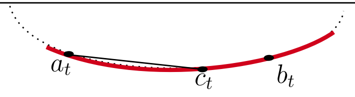
case			illustration	$m_t =$	$u_t \geq$	$u_t/m_t \geq$
V	V-a	V-a-1		2	1/2	1/4
		V-a-2		2	1	1/2
	V-b	V-b-1		2	1/2	1/4
		V-b-2		2	1/4	1/8

Figure 5.10: A summary of the analysis for Case V.

– Sub-case V-b: Suppose c_t is matched with either a_t or b_t. Without loss of generality, assume c_t is matched with a_t. The case where c_t is matched with b_t can be treated symmetrically. We consider the following cases:

  * Sub-case V-b-1 Suppose b_t is located between a_t and c_t If the critical arc is updated to be on the left of a_t or on the right of c_t, which happens a chance of $1/4 + 1/4 = 1/2$, then the line segment between b_t and any point in the updated critical arc crosses the line segment between (a_t, c_t). Therefore, b_t is added to U_g. We can write $u_t \geq 1/2$. Given that $m_t = 2$, we will have $u_t/m_t \geq 1/4$, which is larger than $p/(6-4p) \approx 0.0795$.

  * Sub-case V-b-2: Suppose b_t is located on the right side of both a_t and c_t. If the critical arc is updated to be between a_t and c_t, which happens with a chance of $1/4$, then the line segment between b_t and any point in the updated critical arc crosses the line segment between (a_t, c_t). Therefore, b_t is added to U_g. We can write $u_t \geq 1/4 \cdot 1 = 1/4$. Given that $m_t = 2$, we will have $u_t/m_t \geq 1/8$, which is larger than $p/(6-4p) \approx 0.0795$.

In summary, in all cases, we have $u_t/m_t \geq p/(6-4p)$, which completes the proof. Note that the lower bound for u_t/m_t takes its smallest value of $(6-4p)/p = (1-2p)/(2+4p)$ in cases I-d and IV-a. □

5.2 Randomized bichromatic non-crossing matching

In this section, we show that any randomized algorithm for the bichromatic non-crossing matching problem is expected to match only $O(\log n)$ points in the worst case, even under the relaxed setting where all red points appear before the first blue point. For that, we will describe a probability distribution for the input sequence of size n and show that any deterministic algorithm is expected to match at most $O(\log n)$ points for a random sequence generated using this distribution. By Yao's principle, the expected number of matched points by any randomized algorithm is no more than $O(\log n)$ in the worst case. Throughout, we assume n is divisible by 3.

Let B be a bit string of length $n - 1$ in which bit values are taken uniformly and independently at random from $\{0, 1\}$. From B, we generate an input sequence $\sigma(B)$ for the bichromatic matching as follows. All points in $\sigma(B)$ appear on the circumference of a circle B. First, n red points appear on some arbitrary positions in the top half of C. This is followed by n blue points that appear in the bottom half of C as follows. There is a critical arc that is initially the entire bottom-half of C. The first blue point p_1 appears in an arbitrary position on the critical arc. If the first bit of B is 0 (respectively 1), then the critical region is updated to be its sub-arc on the left (respectively on the right) of p_1. Similarly, after the t^{th} point p_t appears, the critical arc is updated to its sub-arc on the left (respectively on the right) of p_t. Figure 5.11 shows the input sequence for $B = 0110$.

Theorem 10. Consider the instances of the bichromatic non-crossing matching in which all n red points appear before the first (out of n) blue point appears. Any randomized algorithm is expected to match at most $O(\log n)$ points in the worst-case.

Proof. Let A be a randomized algorithm. The set of red points can be partitioned into sets $M \cup U_g \cup U_u$, where M is the set of matched points, U_u is the set of unmatched red points that can be potentially matched with a future blue point, and U_g is the set of unmatched red points that are guaranteed to stay unmatched, that is, the line segment between a point in U_g and any future point (any point on the critical arc) crosses the segment between a pair of matched points. Just before the first blue point appears, U_u contains all red points, while M and U_g are empty. As the blue points get revealed and matched with red points, U_u reduces to the red points that are “visible" by the critical region. To be more precise, let p_x be the rightmost blue point on the

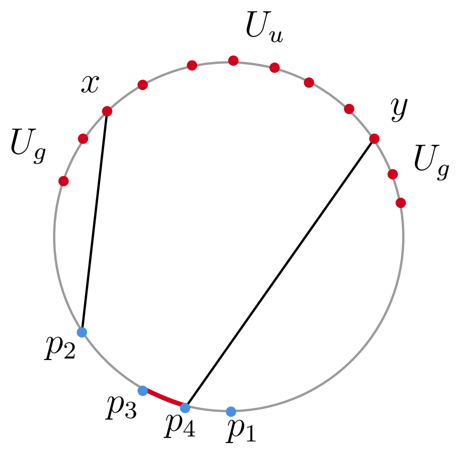


Figure 5.11: An illustration of the input sequence $\sigma(B)$ for a bit string B that starts with 0110. After p_4 is revealed, the critical arc is the red arc between p_3 and p_4. Given that (p_2, x) and (p_4, y) are matched, U_u includes the red points that are on the right of x and on the left of y (i.e., the red points on the undecided arc).

left of the critical arc such that p_x is matched with a red point (call that red point x), and p_y be the leftmost blue point on the right of the critical arc such that p_y is matched with a red point (call that red point y). Any red point on the left of x cannot be matched with any future blue point p_f because the line segment between p_f and a point in the critical region crosses the line segment between p_x and x. As such, the red points on the left of x belong to U_g. Similarly, the red points on the right of y belong to U_g. We conclude that U_u is formed by the red points between x and y, and call the arc between x and y the *undecided arc*.

Suppose A matches the blue point p_t with a red point z on the undecided arc. Let r denote the number of red points in U_u at the time p_t is revealed, and let $f(r)$ indicates the expected number of these r points that get matched by A. Assume there are i points from U_u on the left of z. Hence, $r - i - 1$ points from U_u on the right of z on the undecided arc. Now, if the critical arc is updated to the left of p_t, then the $r - i - 1$ red points on the right side of z on the undecided arc are removed from U_u and added to U_g. This is because any line segment from these points to a point in the critical region crosses the line segment between p_t and z. Therefore, the size of U_u is decreased from r to i, and we can write $f(r) \leq f(i)+1$. Similarly, if the critical arc is updated to the right of p_t, then the i red points on the left side of z on the undecided arc are removed from U_u and added to U_g. In this case, the size of U_u is decreased from r to $r-i-1$, and we can write $f(r) \leq f(r-i-1)+1$. So, on expectation, we can write $f(r) \leq 1/2 \cdot f(i)+1/2 \cdot f(r-i-1)+1 < 1/2 \cdot f((r-1)/2)+1/2 \cdot f(r)+1$. The last inequality holds because we have $\min\{i, r-i-1\} \leq (r-1)/2$ and $\max\{i, r-i-1\} < r$. Subsequently, we can write $f(r) \leq f(r/2)+2$, which gives $f(r) \leq 2\log r$. In particular, when p_t is the first blue point that is matched with a red point, all red points are in U_u, and we have $r = n$. Therefore, the total number of matched red points is $f(n) \leq 2\log n$.

$\square$

Chapter 6

Experimental analysis

Overview

In previous chapters, we studied the non-crossing matching problem in the worst-case scenario, where the input was generated by an adversary. In this chapter, we study algorithms under the average-case scenarios, where the x- and y-coordinates of the input points are random variables that are Identically and Independently Distributed (IID) [10]. We consider three different probability distributions, namely uniform, normal, and Zipfian distributions, to generate IID points on a plane. We implement and compare online algorithms with greedy properties. Unlike the worst-case setting, where randomization helps to “hide" some choice of an online algorithm from the adversary, in the average-case scenarios, there is little advantage in using randomized algorithms. Therefore, all algorithms that we study in this section are deterministic.

In what follows, we first explain the input distributions, implementation of algorithms, and the details of experiments are explained. At the end of this chapter, we conclude the results of our experiments.

6.1 Experimental set-up

6.1.1 Input distribution

This section explains different distributions that are considered to generate the coordinates of the input points. In the bichromatic setting, the colour of each point is selected to be red or blue with equal chances.

- Uniform distribution: Uniform distribution generates numbers in a specific range that are equally likely to occur. The minimum and maximum values are set to 0 and 1, that is points find their x- and y-coordinates as uniform random variables in the range $[0, 1]$. Figure 6.1 depicts 10,000 points that are uniformly distributed.

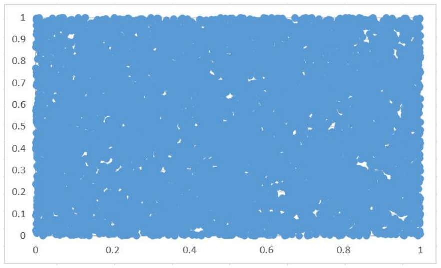


Figure 6.1: Uniform distribution for an input of size 10,000

- Normal distribution: In normal distribution, also known as the Gaussian distribution [17], instances that are close the mean are more frequent. Graph representation of normal distribution is a bell curve. Normal distribution is defined by two parameters. Standard deviation, which indicates how spread out numbers are, and mean (expected value), which specifies data center. In our experiments, the standard deviation is set to 0.1 and mean equals to 0.5. Figure 6.2 demonstrates an instance of size 10,000 that is normal distributed.

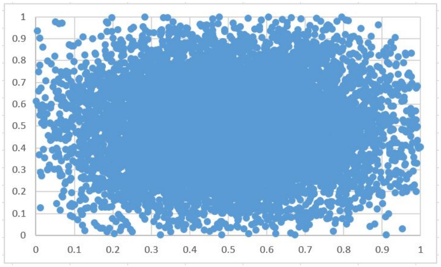


Figure 6.2: Normal distribution for an input of size 10,000, standard deviation 0.1, and mean 0.5

- Zipfian distribution: Zipfian distribution [35], also known as zeta/Zipf distribution, is using to model the popularity of few members of a population. Zipfian distribution has two parameters; a *shape* parameter and a *scale* parameter. The shape parameter defines the spread of item sizes: lower values indicate greater skew towards smaller values. The scale parameter, informally, has the effect of stretching out the probability density. To generate points with positive x-coordinate and y-coordinate, the scale is set to 1.001, and shape set as 1, which is the default values in the standard Python library. Figure 6.3 shows a sample of 10,000 points generated based on Zipfian distribution. Our interest in the Zipfian distribution is mostly due to the fact that it can be considered an “anti-uniform" distribution.

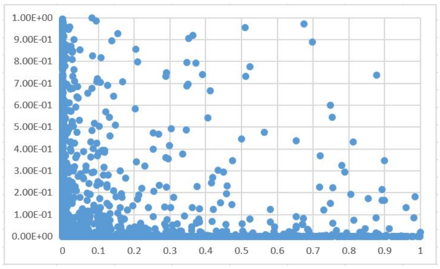


Figure 6.3: Zipfian distribution for input of size 10,000, scale = 1.001 and shape set as default.

6.1.2 The tested algorithms

Greedy algorithms (without delay). The first set of algorithms that we consider are all greedy, that is if a point p is revealed, it is matched with some existing unmatched point as long as there is at least one point q that p can be matched to (that is, the line segment between p and q does not cross the line segment between any pair of matched points). If there is only one point that p can be matched with, the two points are matched, and there is no decision to make. Otherwise, when the set Q of points that p can be matched to contains at least two points, the greedy algorithm has to break ties to select a point $q \in Q$ to match with p. We can define four greedy algorithms as follows:

- E-closest: the greedy algorithm that matches p with $q \in Q$, where q is the closest point in Q to p based on the Euclidean distance.
- X-closest: the greedy algorithm that matches p with $q \in Q$, where q is the point whose x-coordinate is closest to p among all points of Q.
- Random: the greedy algorithm that selects q to be the point in Q that appears earliest in the input sequence. Given that the input is generated randomly, selecting the first point is somehow a random choice. Therefore, we refer to this algorithm as Random.

- Furthest: the greedy algorithm that matches p with $q \in Q$, where q is the the furthest point in Q to p, based on the Euclidean distance.

Algorithms with delay. The second set of algorithms that we consider are “almost-online" in the sense that they know the length n of the input (but they do not have any other information about the input sequence). These algorithms “delay" making a decision for the first $x = c \cdot n$ points, that is, they do not match any pairs of points for the first x points. After the x'th point appears, the algorithms treat the remaining point in a greedy manner, using tie-breaking rules described in the previous section. We refer to $c \in [0, 1]$ as the “buffer" of the algorithm. Intuitively, the buffer helps the algorithm to find a better match for points, given that more information is available at the time of matching.

Note that the above algorithms can be defined in both monochromatic and bichromatic settings. The way the set Q of suitable points is generated is different under the two settings. However, the same tie-breaking rules can be applied for both settings.

6.1.3 Implementation details

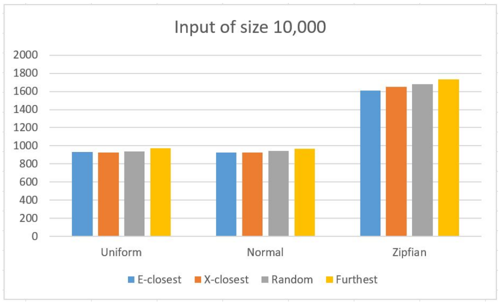
Here are some details about the set-up for our experiments:

- Experiments are run by Dell Latitude 7400 64-bit Operating System, equipped by Intel(R) Core(TM) i5-8265U CPU and 8.00 GB RAM.
- IDEs that are used for programming are IntelliJ IDEA Community Edition and Jupyter Lab 2.1.5.
- Java is the main programming language to implement algorithms and generate input. Zipf distributed input is generated by Python 3.0.

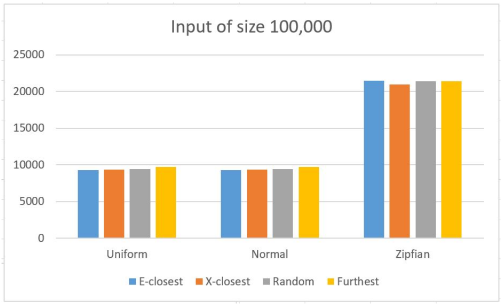
6.2 Results

6.2.1 Monochromatic algorithms (without delay)

We ran the greedy algorithms (without delay) as described in Section 6.1.2 on ten input sequences of size $n = 10,000$, and $n = 100,000$, where the $x-$ and $y-$coordinates of points are iid variables that follow distributions described in Section 6.1.1.



	Normal	Uniform	Zipfian
E-closest	929.6	924	1607.2
X-closest	926	926.2	1649.2
Random	939	942.9	1677.8
Furthest	974.6	967.4	1735.4



	Uniform	Normal	Zipfian
E-closest	9302	9253.6	21467.6
X-closest	9316.8	9330.8	20984
Random	9406.6	9423.6	21411.2
Furthest	9722	9682.4	21423.2

Figure 6.4: Comparison of various greedy algorithms for input sequences generated using normal, uniform and Zipfian distributions

Figure 6.4 compares results of various algorithms, where the average number of unmatched points is reported (the average is taken over the ten input sequences).

From the reported numbers, we can conclude the following:

- For inputs of size $n \in \{10,000, 100,000\}$, the number of unmatched points is approximately $n/10$ for uniform and normal distributions and around $n/5$ for the Zipfian distribution. As expected, the number of unmatched points in the average setting is less than the number of unmatched points in the worst-case setting, which is roughly $n/3$.

- The probability distribution used to generate input points can make a difference in the number of unmatched points. In particular, input sequences generated using the Zipfian distribution constitute “harder" instances where a larger number of points stay unmatched. This can be attributed to the “anti-uniform" nature of the Zipfian distribution, where there is an asymmetry in the input sequence, with heavy skew towards certain regions of the plane and sparsity in other regions (see Figure 6.3). Consider the partitioning of the plane into convex regions, by extending the line segments between matched pairs (as in Figure 4.1), at some early point during the execution of an algorithm. In the case of Zipfian distribution, it is possible that a point p appears in one of the convex partitions and no other point appear in the same region (which happens when the area around p is sparse as the distribution is skewed towards another area), which results in p staying unmatched. This justifies the larger number of unmatched points in the case of Zipfian distribution, when compared to other distributions. Intuitively speaking, in the case of uniform and normal distribution, a point p is likely to find another point q close-by (and they can match). The skew present in the Zipfian distribution, on the other hand, results in situations where p is isolated and likely to stay unmatched.

- In almost all cases, the E-closest algorithm results in a smaller number of unmatched points compared to other algorithms. Intuitively speaking, it is best to match each point to its closest suitable point. This results in a shorter line segment between the matched pair and hence a smaller chance that such line segment crosses the line segment between any potential future matched pair of points.

6.2.2 Monochromatic algorithm with delay

We ran the greedy algorithms with delay, as described in Section 6.1.2, on ten input sequences of size $n = 10,000$, and $n = 100,000$, where the $x-$ and $y-$coordinates of points are iid variables that follow distributions described in Section 6.1.1. We have tested different values of buffer for the algorithms.

Figure 6.5 compares results of various algorithms for inputs of size $n \in \{10,000, 100,000\}$ and buffer size $c \in \{0.005, 0.01, 0.1, 0.25, 0.5, 0.75\}$, where the average number of unmatched points is reported (the average is taken over the 10 input sequences). Figures 6.6 and 6.7 report the same numbers for the normal and Zipfian distributions, respectively.

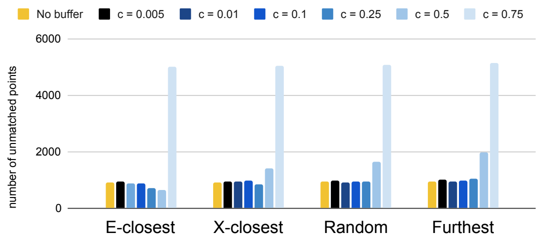
From the reported numbers, we can conclude the following:

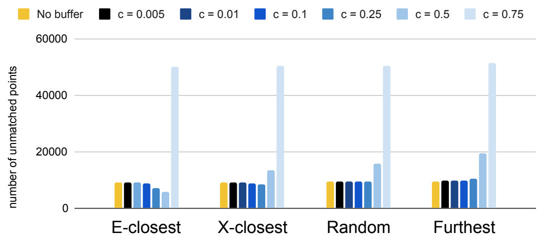
- For the E-closest algorithm, using the buffer size always helps in reducing the number of unmatched points (when compared to the algorithm with no buffer). In particular, buffers of size as large as 0.25 to 0.5 result in the best performance. Given that the E-closest generally produces the best results, for practical purposes, it is best to use this algorithm with the buffer of size $c \in [0.25, 0.5]$.

- Using a buffer helps to improve the performance of X-closest as well, while no improvement is observed for Random and Furthest (the performance of these algorithms is generally better when no buffer is used). This can be justified by the fact that Random and Furthest are not the best tie-breaking rules as they potentially create long segments between the matched pairs. As such, observing a larger number of points before matching pairs of points results in longer segments between the matched pairs and hence more chance for the remaining points to stay unmatched.

- When the buffer is too large, e.g., $c = 0.75$, the performance of all algorithms degrades. This is expected as when $c = 0.75$, the algorithm has waited too long before matching points: even if all remaining $0.25n$ points are matched, at least $n/2$ points stay unmatched.

- The distribution used for generating the input points can make a difference in the number of unmatched points. In particular, a larger number of points stay unmatched when the input is generated using the Zipfian distribution. This is consistent with the observation we made about the harder nature of the Zipfian distribution in the previous section.





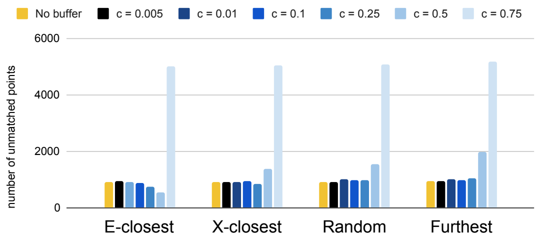
Buffer size	No buffer	%0.5	%1	%10	%25	%50	%75
E-closest	916	936	892	872	718	643	5000
X-closest	922	964	932	976	850	1400	5056
Random	943	987	917	939	937	1649	5061
Furthest	960	1030	936	986	1052	1970	5150

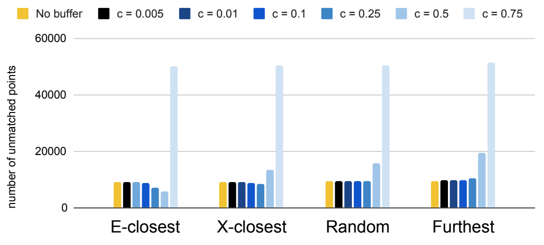
(a) inputs of size $n = 10,000$

Buffer size	No buffer	%0.5	%1	%10	%25	%50	%75
E-closest	9302	9240	9266	8796	7188	5876	50000
X-closest	9296	9266	9306	8916	8414	13622	50476
Random	9358	9443	9527	9477	9579	15697	50581
Furthest	9620	9662	9648	9790	10472	19474	51538

(b) inputs of size $n = 100,000$

Figure 6.5: Monochromatic algorithm with delay on uniform distributed input of size 10,000 and 100,000. The best result of each algorithm is highlighted by a different color.





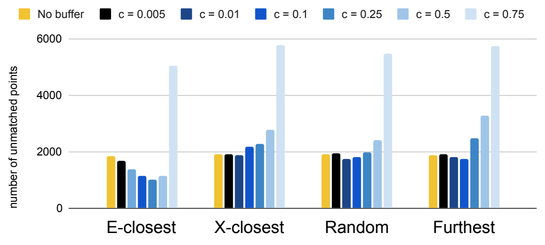
Buffer size	No buffer	%0.5	%1	%10	%25	%50	%75
E-closest	902	932	914	880	759	556	5000
X-closest	928	926	910	950	836	1368	5042
Random	925	919	1001	987	983	1557	5067
Furthest	942	950	1028	982	1038	1978	5164

(a) inputs of size $n = 10,000$

Buffer size	No buffer	%0.5	%1	%10	%25	%50	%75
E-closest	9302	9240	9266	8796	7188	5876	50000
X-closest	9296	9266	9306	8916	8414	13622	50476
Random	9358	9443	9527	9477	9579	15697	50581
Furthest	9620	9662	9648	9790	10472	19474	51538

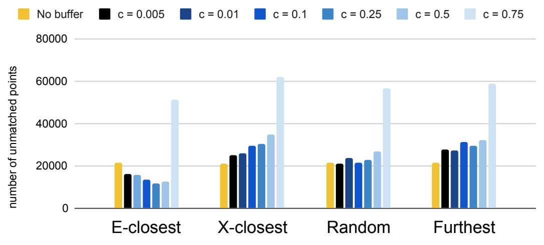
(b) inputs of size $n = 100,000$

Figure 6.6: Monochromatic algorithm with delay on normal distributed input of size 10,000 (a) and 100,000 (b). The best result of each algorithm is highlighted by a different colour.



Buffer size	No buffer	%0.5	%1	%10	%25	%50	%75
E-closest	1860	1692	1394	1142	998	1156	5058
X-closest	1926	1914	1876	2170	2268	2794	5780
Random	1923	1939	1741	1801	1981	2427	5473
Furthest	1868	1922	1814	1746	2466	3262	5734

(a) inputs of size $n = 10,000$



Buffer size	No buffer	%0.5	%1	%10	%25	%50	%75
E-closest	21350	16326	15742	13392	11668	12654	51420
X-closest	21102	24912	26116	29620	30468	35022	61990
Random	21512	21271	23625	21317	22901	26845	56543
Furthest	21542	27786	27146	31342	29566	32044	58620

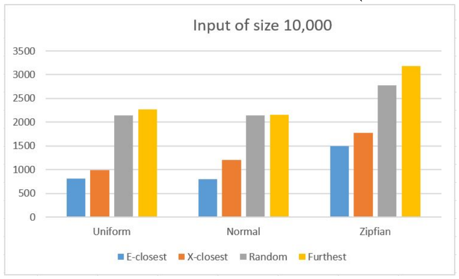
(b) inputs of size $n = 100,000$

Figure 6.7: Monochromatic algorithm with delay on the Zipfian distributed input of size 10,000 (a) and 100,000 (b). The best result of each algorithm is highlighted by a different colour.

6.2.3 Bichromatic algorithms

We ran the greedy algorithms (without delay), as described in Section 6.1.2, on ten input sequences of size $n = 10,000$, where the $x-$ and $y-$coordinates of points are iid variables that follow distributions described in Section 6.1.1. Given that the colour of each point is red or blue with equal chances, it is expected that the number of points from each colour to be $n/2$.

Figure 6.8 compares results of various algorithms, where the average number of unmatched points is reported (the average is taken over the five input sequences).



	Uniform	Normal	Zipfian
E-closest	818	800	1496.4
X-closest	987.2	1206.4	1772
Random	2144.8	2144.4	2779.6
Furthest	2272.8	2159.2	3186.8

Figure 6.8: Comparison of various greedy algorithms for input sequences generated using normal, uniform and Zipfian distributions

From the reported numbers, we can conclude the following:

- There is a big disparity between the average number of unmatched points (reported in Figure 6.8), and the number of unmatched points in the worst-case scenarios. In all cases reported in Figure 6.8, at least two-third of points are matched when the input is generated randomly, while Theorem 4 implies that only $\log n - o(\log n) \approx 14$ points are matched, and the remaining 9986 points stay unmatched in the worst-case.

- In contrast to the monochromatic case, where the number of unmatched points is almost equal for different algorithms, there is a noticeable difference between the number of unmatched points by different algorithms in the bichromatic case. In particular, E-closest has a clear advantage, and Furthest is by far the worst algorithm. This can be justified by the observations that we made earlier. In particular, maintaining shorter segments between the matched pairs helps in reducing the number of unmatched points.

6.3 Summary of results

The results in this section highlight the importance of studying the average-case scenarios. In particular, the worst-case results from the theoretical analysis of the online non-crossing matching algorithms are not always aligned with the typical performance of such algorithms. Studying inputs that are generated randomly from a known distribution is one way to understand the typical performance of different algorithms. From our results in this section, we can make the following conclusions:

- Among the greedy algorithms, it is best to use the tie-breaking rule that selects the closest point, that is E-closest. In the monochromatic setting, E-closest leaves less than $n/10$ points unmatched (for all distributions that we tried), which is less than the $n/3$ unmatched points in the worst-case scenarios. For the bichromatic setting, the algorithm leaves less than $n/6$ points unmatched, which is considerably less than $n - \log(n)$ points unmatched in the worst-case scenarios.

- Using a buffer helps in improving the performance of E-closest. In particular, if the algorithm waits to observe $c \cdot n$ points before matching the first point, for some $c \in [0, 25, 0.5]$, its performance is further improved. In particular, the number of unmatched points will become less than $n/14$.

- The distribution used to generate the input can make a difference in the number of unmatched points by E-closest. In particular, if the input points are uniformly distributed, the number of unmatched points is less when compared to the Zipfian distribution.

Chapter 7

Conclusions

We studied the online non-crossing matching problem under different settings and assumptions. Our main goal is to understand the performance of online algorithms in the worst-case settings, where an adversary generates the input points. Our results can be summarized as follows:

- For the monochromatic setting of the problem, We provided a tight upper bound (Theorem 1) as well as a tight lower bound (Theorem 2) for the number of unmatched points by deterministic algorithms. Similarly, we proved tight upper and lower bounds (Theorems 3 and 4) for the number of unmatched points under the bichromatic setting. In particular, we proved any deterministic algorithm with greedy property matches at least $\lceil 2(n-1)/3 \rceil$ points, and this is the best worst-case performance among all deterministic algorithms. Meanwhile, under the bichromatic, any algorithm is forced to leaves $n - o(n)$ points unmatched in the worst-case scenario. This shows the disparity between the monochromatic and bichromatic settings of the problem. In the bichromatic case, the adversary is more powerful, and hence an online algorithm is forced to leave almost all points unmatched.

- We showed the online algorithm with advice of size $O(n)$ and $\Omega(\log n)$ are respectively sufficient (Theorem 5) and necessary (Theorem 6) to match all points in the monochromatic case. For the bichromatic variant, advice of size $\Theta(n \log n)$ is both sufficient and necessary to match all points (Theorem 7). The main takeaway is that more bits are necessary to achieve an optimal solution for the bichromatic setting. This is yet another evidence for the harder nature of the bichromatic matching when compared to the monochromatic matching.

- We studied the power of randomization for the non-crossing matching of online points problem. We proved that using randomization can help improve the performance under the monochromatic setting. In particular, we presented a randomized algorithm that is expected to leave at most $0.3304n + 0.5754$ unmatched points (Theorem 8), which improves over $n/3$ unmatched points by any deterministic algorithm. (Theorem 4). We also showed a limitation for what randomized algorithms can achieve by showing that any randomized algorithm is expected to leave at least 0.0738 points unmatched (Theorem 9). We also showed that, unlike the monochromatic setting, randomization does not help for the bichromatic case (Theorem 10).

- In addition to the worst-case scenarios, we considered the average-case performance of online algorithms. We experimentally studied the performance of greedy algorithms with different tie-breaking rules on inputs generated independently at random from an identical distribution. Our results revealed a big difference between the worst-case and average-case performance attainable by online algorithms. For example, while any online algorithm is forced to leave $n - o(n)$ points unmatched under the bichromatic setting under the worst-case scenarios, an algorithm is expected to leave at most $n/3$ points unmatched for random inputs.

There are several open problems that can be considered as topics for future research:

- Tightening the gap between the upper and lower bounds of the number of advice bits sufficient and required to achieve a perfect matching.

- Improving the randomized algorithm of Chapter 5 to tighten the gap between the existing upper and lower bounds for the best competitive ratio attainable by randomized algorithms.

- Providing a theoretical analysis to find the number of unmatched points when the point coordinates are uniform iid variables.

- Studying the power of “deferral" for the online non-crossing matching problem. Under this setting, the algorithm can change some of its previous decisions for improved performance. This model is studied for other online problems (e.g., for the Steiner tree problem [20]) and seems very relevant in the context of online matching, particularly for the bichromatic setting. For example, one might for the number of matched pairs of points that should be “unmatched" in order to ultimately match a fraction of points in the bichromatic setting.

Bibliography

[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. 9, 10

[2] S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, and M. Renault. Online computation with untrusted advice. *arXiv preprint arXiv:1905.05655*, 2019. 11

[3] M. J. Atallah. A matching problem in the plane. *J. Comput. Syst. Sci.*, 31(1):63–70, 1985. 1, 9

[4] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994. 2

[5] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994. 12

[6] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In *International Symposium on Algorithms and Computation*, pages 331–340. Springer, 2009. 11

[7] P. Bose, P. Carmi, S. Durocher, S. Kamali, and A. Sajadpour. Non-crossing matching of online points. In J. M. Keil and D. Mondal, editors, *Proceedings of the 32nd Canadian Conference on Computational Geometry, CCCG 2020, August 5-7, 2020, University of Saskatchewan, Saskatoon, Saskatchewan, Canada*, pages 233–239, 2020. 13

[8] 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. 15

[9] 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. 11

[10] G. Casella. Statistical inference/by george casella, rober l. berger. *Duxbury Advanced Series.*, 2002. 51

[11] S. Cohen. *Finding color and shape patterns in images.* Stanford University, Department of Computer Science, 1999. 1, 2, 6, 11

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009. 9

[13] C. Dürr, C. Konrad, and M. Renault. On the power of advice and randomization for online bipartite matching. *arXiv preprint arXiv:1602.07154*, 2016. 12

[14] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001. 1, 11

[15] A. Formella. Approximate point set match for partial protein structure alignment. *Proceedings of Bioinformatics: Knowledge Discovery in Biology (BKDB2005). Facultade Ciencias Lisboa da Universidade de Lisboa*, pages 53–57, 2005. 1, 2, 3, 6, 11

[16] B. Fuchs, W. Hochstättler, and W. Kern. Online matching on a line. *Theoretical Computer Science*, 332(1-3):251–264, 2005. 11

[17] N. R. Goodman. Statistical analysis based on a certain multivariate complex gaussian distribution (an introduction). *The Annals of mathematical statistics*, 34(1):152–177, 1963. 52

[18] K. Grauman and T. Darrell. Fast contour matching using approximate earth mover's distance. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004. 2, 6

[19] 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. 20

[20] A. Gu, A. Gupta, and A. Kumar. The power of deferral: maintaining a constant-competitive steiner tree online. *SIAM Journal on Computing*, 45(1):1–28, 2016. 63

[21] S. Gu, C. Lindsay, M. A. Gennert, and M. A. King. A quick 3d-to-2d points matching based on the perspective projection. In *International Symposium on Visual Computing*, pages 634–645. Springer, 2008. 2

[22] 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. 2

[23] M. Hoffmann, B. Speckmann, and C. D. Tóth. Pointed binary encompassing trees: Simple and optimal. *Comput. Geom.*, 43(1):35–41, 2010. 15

[24] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane—a survey—. In *Discrete and computational geometry*, pages 551–570. Springer, 2003. 6

[25] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990. 10, 11, 12

[26] S. Khuller, S. G. Mitchell, and V. V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994. 11

[27] E. Koutsoupias and A. Nanavati. The online matching problem on a line. In *International Workshop on Approximation and Online Algorithms*, pages 179–191. Springer, 2003. 11

[28] E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995. 11

[29] L. C. Larson. Intermediate real analysis. In *Problem-Solving Through Problems*, pages 192–240. Springer, 1983. 6

[30] A. Levin and A. Shashua. Principal component analysis over continuous subspaces and intersection of half-spaces. In *European Conference on Computer Vision*, pages 635–650. Springer, 2002. 2

[31] C. Lo, J. Matousek, and W. L. Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11:433–452, 1994. 7

[32] J. W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. *arXiv preprint arXiv:1511.05886*, 2015. 12

[33] S. Miyazaki. On the advice complexity of online bipartite matching and online stable marriage. *Inf. Process. Lett.*, 114(12):714–717, 2014. 11

[34] P. Neamatollahi, M. Hadi, and M. Naghibzadeh. Simple and efficient pattern matching algorithms for biological sequences. *IEEE Access*, 8:23838–23846, 2020. 2

[35] D. M. Powers. Applications and explanations of zipf's law. In *New methods in language processing and computational natural language learning*, 1998. 52

[36] P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18(6):1201–1225, 1989. 10

[37] P. B. Van Wamelen, Z. Li, and S. Iyengar. A fast expected time algorithm for the 2-d point pattern matching problem. *Pattern Recognition*, 37(8):1699–1711, 2004. 11

[38] A. C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proc. the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 222–227. IEEE Computer Society, 1977. 26