On the Online Weighted Non-Crossing Matching Problem

Joan Boyar
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Shahin Kamali
Department of Electrical Engineering and Computer Science, York University, Toronto, Canada

Kim S. Larsen
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Ali Mohammad Lavasani[1]
Department of CSSE, Concordia University, Montreal, Canada

Yaqiao Li
Department of CSSE, Concordia University, Montreal, Canada

Denis Pankratov
Department of CSSE, Concordia University, Montreal, Canada

Abstract

We introduce and study the weighted version of an online matching problem in the Euclidean plane with non-crossing constraints: $2n$ points with non-negative weights arrive online, and an algorithm can match an arriving point to one of the unmatched previously arrived points. In the vanilla model, the decision on how to match (if at all) a newly arriving point is irrevocable. The goal is to maximize the total weight of matched points under the constraint that straight-line segments corresponding to the edges of the matching do not intersect. The unweighted version of the problem was introduced in the offline setting by Atallah in 1985, and this problem became a subject of study in the online setting with and without advice in several recent papers.

We observe that deterministic online algorithms cannot guarantee a non-trivial competitive ratio for the weighted problem. We study various regimes of the problem which permit non-trivial online algorithms. In particular, when weights are restricted to the interval $[1, U]$ we give a deterministic algorithm achieving competitive ratio $\Omega\left(2^{-2\sqrt{\log U}}\right)$. We also prove that deterministic online algorithms cannot achieve competitive ratio better than $O\left(2^{-\sqrt{\log U}}\right)$. Interestingly, we establish that randomization alone suffices to achieve competitive ratio $1/3$ even when there are no restrictions on the weights. Additionally, if one allows an online algorithm to revoke acceptances, then one can achieve a competitive ratio ≈ 0.2862 deterministically for arbitrary weights. We also establish a lower bound on the competitive ratio of randomized algorithms in the unweighted setting, and improve the best-known bound on advice complexity to achieve a perfect matching.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms, weighted matching problem, Euclidean plane, non-crossing constraints, competitive analysis, randomized online algorithms, online algorithms with advice, online algorithms with revoking

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.16

Funding The first and third authors were supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B. Other authors were supported by NSERC Canada.

Acknowledgements The authors thank the anonymous referees for their valuable suggestions.

[1] Corresponding author

1 Introduction

We introduce and study the following problem, which we call Online Weighted Non-Crossing Matching (OWNM). Suppose $2n$ points $p_1, \ldots, p_{2n}$ in Euclidean plane arrive online one-by-one. When p_i arrives, its positive weight $w(p_i) \in \mathbb{R}_{>0}$ is revealed and an algorithm has an option of matching p_i to one of the unmatched previously revealed points, or leave p_i unmatched. In the vanilla online model, the decisions of the algorithm are irrevocable. There is a non-crossing constraint, which requires that the straight-line segments corresponding to the edges of the matching do not intersect. Assuming that the points are in general position, the goal is to design an algorithm that maximizes the weight of matched points.

The interest in geometric settings, particularly the Euclidean plane setting, for the matching problem stems from applications in image processing [14] and circuit board design [20]. In such applications, one is often required to construct a matching between various geometric shapes, such as rectangles or circles, representing vertices, using straight-line segments or, more generally, curves. Geometry enters the picture due to constraints on the edges, such as avoiding intersections among the edges, as well as avoiding edge-vertex intersections. These constraints can have a significant impact on the offline complexity of the problem, often resulting in variants of problem that are $\mathcal{NP}$-hard (see the survey by Kano and Urrutia [22]).

The unweighted version of the Non-Crossing Matching problem (i.e., when $w(p_i) = 1$ for all $i \in \{1, \ldots, 2n\}$) has been studied both in the offline setting ([8, 19]) and the online setting ([11, 31, 21, 25]). We go over the history of the problem in detail in Section 2. For now, it suffices to observe that an offline algorithm that knows the locations of all the points in advance can match all the points while satisfying the non-crossing constraint. Thus, the value of offline OPT is always $W := \sum_{i=1}^{2n} w(p_i)$. Performance of an online algorithm is measured by its competitive ratio, which for our problem corresponds to the fraction of W that the algorithm can guarantee to achieve in the worst-case.

It is relatively easy to see that when there are no restrictions on the weights of points, no deterministic online algorithm can guarantee a non-trivial competitive ratio bounded away from 0 (in particular, this is an immediate corollary of Theorem 1). We study different regimes under which the problem admits algorithms achieving non-trivial competitive ratios. Our results can be summarized as follows:

- In the Restricted OWNM, we assume that the weights of points are restricted to lie in the interval $[L, U]$ for some $L \le U \in \mathbb{R}_{>0}$ that are known to the algorithm at the beginning of the execution. Note that by scaling, we can assume that $L = 1$; thus, without loss of generality, we assume that all the weights are in the interval $[1, U]$ in Restricted OWNM. We show that the competitive ratio of any deterministic online algorithm is $O\left(2^{-\sqrt{\log U}}\right)$ (Theorem 1). We also present a deterministic online algorithm, Wait-and-Match (WAM), which has competitive ratio $\Omega\left(2^{-2\sqrt{\log U}}\right)$ (Theorem 5).
- We show, perhaps surprisingly, that randomization alone is enough to guarantee a constant competitive ratio for arbitrary weights. We present a simple randomized online algorithm, called Tree-Guided-Matching (TGM), and prove that it has competitive ratio $1/3$ (Theorem 7). We supplement this result by showing that no randomized online algorithm can achieve a competitive ratio better than $16/17$, even for the unweighted version of the problem (Theorem 6).[2]

[2] Sajadpour [31] gave a proof that no randomized algorithm can achieve a competitive ratio better than 0.9262, which is stronger than our result $16/17 \approx 0.9411$. However, their argument has not been peer-reviewed at the time of this paper. Moreover, our argument is much simpler and shorter.

- We show that allowing revocable acceptances (see beginning of Section 6 for the definition of the model) permits one to obtain competitive ratio ≈ 0.2862 by a deterministic algorithm even when the weights of points are unrestricted (Theorem 10). We supplement this result by showing that no deterministic algorithm with revoking can achieve competitive ratio better than $2/3$ (Theorem 8).
- Lastly, we present a new algorithm, called Split-And-Match (Sam), that uses $\lceil \log C_n \rceil < 2n$ bits of advice (see beginning of Section 7 for the definition of the model) to achieve optimality (Theorem 12), where C_n is the n^{th} Catalan number. This improves upon the previously known bound of $3n$ on the advice complexity of the problem [25]. Since Sam achieves a perfect matching, it does not matter whether the given points are weighted or not.

2 Related Work

Given $2n$ points in $\mathbb{R}^2$ in general position, the basic non-crossing matching (NM) problem is to find a non-crossing matching with the largest possible number of edges. Observe that the minimum-length Euclidean matching is non-crossing, hence a perfect NM always exists. The NM problem and its variants have been extensively studied in the offline setting. Hershberger and Suri [19] gave an algorithm that finds a perfect NM in time $\Theta(n \log n)$. Atallah [8], and Dumitrescu and Steiger [15] gave efficient algorithms for the bichromatic version of the problem, where the points are divided into two subsets, and matching edges can only be formed between the two subsets. Other versions of the problem considered in the research literature include requiring the NM to be stable [30, 18], requiring two NMs to be compatible (edges in two NMs are also non-crossing, only sharing endpoints) [2], and requiring compatible NMs to satisfy certain diversity constraint [24].

Several studies considered optimization problems over all NMs. The objective functions include maximizing the sum of the Euclidean length of matching edges [5], minimizing the length of the longest matching edge [1], and other similar combinations of min and max [23].

Another line of research is to relax the non-crossing constraint and allow certain crossings. An important problem is to understand the size of a crossing family, that is, matching edges that are pairwise crossing. A recent breakthrough by Pach et al. [26] showed that the largest crossing family has linear size. Aichholzer et al. [4] studied the counting problem of k-crossing matchings.

At least two works [10, 32] considered weighted NM on n points, where every point has weight in $\{1, 2, \ldots, n\}$. Balogh et al. [10] considered the weight of an edge to be the sum of the weights of the two endpoints modulo n, and studied the typical size of NM with distinct edge weights (this is called non-crossing harmonic matching). Sakai and Urrutia [32] considered the weight of an edge to be the minimum weight of the two endpoints, and they studied the lower and upper bounds of the maximum weighted NM.

In pure mathematics, NM has been studied as a tool to understand the representation theory of groups [7, 27]. A tuple of NMs (so-called a necklace) satisfying a specific property is used to study the topology of harmonic algebraic curves associated with a polynomial over $\mathbb{C}$ [33]. Extremal graph problems where NM of size k plays the role of a forbidden subgraph are studied in [3, 17].

Besides the application in image processing and circuit design, as mentioned in the Introduction, NM has also found other applications. One major application is in computational biology. A restricted version of NM (e.g., points all on a circle), and k-non-crossing matching (no k edges pairwise intersecting, which reduces to the standard NM when $k = 2$) have been studied to understand RNA structures [9, 13, 34]. In applications that are related to visibility problems (such as in robotics) and geometric shape matching, one replaces all or a subset of points in question by geometric objects [28, 6]. For example, when the question is to match objects to objects, then an edge (p, q) between two objects A and B can be formed by choosing arbitrary points p from A and q from B, conditioned on that the edge (p, q) does not cross other objects.

The online NM has only been studied very recently. Bose et al. [11] initiated the study of online (unweighted) NM and showed that the competitive ratio of deterministic algorithms is 2/3, while Kamali et al. [21] gave a randomized algorithm that matches in expectation about 0.6695 fraction of all points. The online bichromatic NM has also been studied in [11, 31]. Finally, the advice complexity was studied in [11, 25]. In particular, Lavasani and Pankratov [25] resolved the advice complexity of solving online bichromatic NM optimally on a circle and gave a lower bound of $n/3 - 1$ and an upper bound of $3n$ on the advice complexity of online NM on a plane.

3 Preliminaries

The input to the matching problems considered in this work is an online sequence $I = (p_1, \ldots, p_{2n})$ of points in general position, where p_i has a positive real-valued weight $w(p_i) \in \mathbb{R}_{>0}$. We use W to denote the total weight of all the points, i.e., $W = \sum_{i=1}^{2n} w(p_i)$. For the Restricted OWNM, the weights are assumed to lie in the interval $[1, U]$ for some known value of U, which is considered to be a hyper-parameter, and not part of the input. Upon the arrival of p_i, an online algorithm must either leave it unmatched or match it with an unmatched point p_j $(j < i)$, in which case the line segment between p_i and p_j, denoted by $\overline{p_i p_j}$, must not cross the line segments between previously matched pairs of points. The objective is to maximize the total weight of matched points. For an online algorithm ALG (respectively, offline optimal algorithm OPT), we use ALG(I) (respectively, OPT(I)) to denote the total weight of points matched by the algorithm on input I. By abuse of notation, the symbol $\overline{pq}$ is also used to denote the full line passing through the two points p and q, dividing a convex region into two sub-regions.

We say that a deterministic online algorithm ALG is ρ-competitive if there exists a constant c such that for every input sequence I we have

$$\mathrm{ALG}(I) \geq \rho \cdot \mathrm{OPT}(I) - c.$$

For a randomized ALG the above inequality is replaced by the following

$$\mathbb{E}(\mathrm{ALG}(I)) \geq \rho \cdot \mathrm{OPT}(I) - c.$$

If $c = 0$ then we call the competitive ratio ρ strict, and we say that ALG is strictly ρ-competitive. If $c \neq 0$ then, for emphasis, we shall sometimes say that the competitive ratio is asymptotic. Note that for the Restricted OWNM, we allow c to depend on the hyper-parameter U when considering asymptotic competitiveness. Thus, an algorithm achieving asymptotic competitive ratio ρ is allowed to leave a constant number of points unmatched (regardless of their weights) beyond the $(1 - \rho)$-fraction of W.

4 Deterministic Algorithms for Restricted OWNM

4.1 Point Classification

In both lower and upper-bound arguments, we use a point classification, based on parameters, $k \in \mathbb{N}$ and $U \in \mathbb{R}$, which we explain here. Let $k = \lceil \sqrt{\log U} \rceil$, and define values of $a_0, a_1, \ldots, a_k$ so that

$$a_0 = 1, a_k = U, \quad r = a_1/a_0 = a_2/a_1 = \ldots = a_k/a_{k-1},$$

which implies that $r = U^{1/k}$ and $a_i = r^i$. For a given value $w \in [1, U]$, define $\llbracket w \rrbracket$ as the largest a_i such that $a_i \leq w$. In what follows, a point with weight w is said to have type i if $\llbracket w \rrbracket = a_i$. Thus, there are $k + 1$ distinct types, with type k containing only the value U. The type of a line segment between two matched points x and y is defined by the type of the end-point with larger weight, that is, $\overline{xy}$ has type i if one of its endpoints has type i and the other endpoint has type at most i.

4.2 Negative Result

▶ **Theorem 1.** *For a sufficiently large value of U, the asymptotic competitive ratio of any deterministic online algorithm for the Restricted OWNM problem is $O\left(2^{-\sqrt{\log U}}\right)$.*

Proof. Let ALG be any online deterministic algorithm. We use an adversarial argument. The adversary sends all points on a circle C, so any match the algorithm makes creates a chord in the circle, dividing a previous region into two. At any point in time, the adversary sends a point in an *active region* of C, which is formed by one or two *arcs*, the segments of the circle bounded by two consecutive points, in the boundary of C. Initially, the entire circle forms the active region. The adversary's strategy is to maintain a mapping from unmatched points to matched points to ensure the ratio between the total weight of matched points and unmatched points is $O\left(2^{-\sqrt{\log U}}\right)$. Note that this implies the ratio between the total weight of matched points and all points is also $O\left(2^{-\sqrt{\log U}}\right)$.

The adversary starts the input with an arbitrarily large number, m (this is required to guarantee that our bound is asymptotic). The adversary puts points of weight 1 in arbitrary positions on the circle until either the algorithm matches m pairs of points or it reaches $m2^k$ points on the circle. In the latter case, the competitive ratio is at most $O(2^{-k}) = O(2^{-\sqrt{\log U}})$.

Therefore, we may assume that ALG eventually matches m pairs of points, creating non-intersecting chords, and $m + 1$ regions. Now, make each matched pair responsible for a distinct region created, though with the first matched pair being responsible for two regions, initially the first two regions. Suppose a new chord $\overline{xy}$ divides region R into two. Let $\{x_R, y_R\}$ be the responsible pair for R, R_1 be the side of R that has $\overline{x_R y_R}$ on its boundary and R_2 be the other side. Leave $\{x_R, y_R\}$ responsible for R_1 and make $\{x, y\}$ responsible for R_2. This ensures that each matched pair is responsible for at least one region.

For each region R, the adversary makes R the active region, runs the following procedure and continues with the next region until it covers all the regions. Let $\{x_R, y_R\}$ be the responsible pair of points for R. Consider the following two cases, depending on the number of unmatched points in R:

Case 1. If the number of unmatched points in R is $\geq 2^k - 1$, the adversary does not send any point in R and continues to the next region. In this case, we map the unmatched points in R to the matched pair $\{x_R, y_R\}$. Note that $2^k - 1$ points of weight 1 are mapped to a segment of total weight 2. The ratio between the weight of matched points to the unmatched points will be $\leq 2/(2^k - 1) \in O\left(2^{-\sqrt{\log U}}\right)$.

Case 2. If the number of unmatched points in R is $< 2^k - 1$, the adversary plans to send a sequence of points, $P = (p_1, p_2, \ldots, p_k)$, with weights $a_1, a_2, \ldots, a_k$ (respectively), one point from each weight, in the ascending order of their weights, in the following manner, (see Fig. 1). The point p_1 of weight a_1 appears in an arbitrary position in R (on the circle). Upon

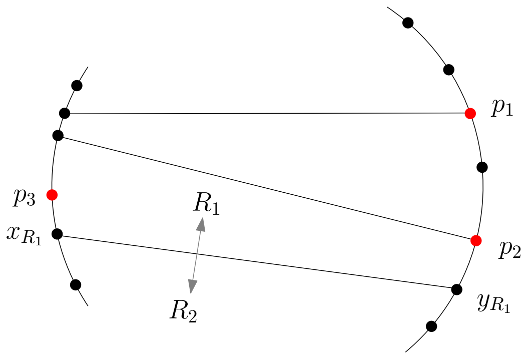

Figure 1 An illustration of the adversary's strategy for $k = 3$. The two arcs form the active region. Black points have weight 1. Suppose in the first phase ALG matched x_{R_1} and y_{R_1}, which became responsible for R_1 region. Note that the number of unmatched points (of weight 1) in R_1 is 6, which is less than $2^k - 1 = 7$. Thus, in the second phase, the adversary plans to send points p_1, p_2, p_3 of weights a_1, a_2, a_3 in R_1. Suppose ALG matches the point of weight a_1; then the adversary sends p_2, p_3 below the line segment between the matched pair (there are fewer unmatched points there). Similarly, after the point p_2 of weight a_2 is matched, the adversary sends p_3 to the side of the resulting segment with no unmatched points. This ensures that some point of weight a_i (here a_3) stays unmatched and is mapped to the matched pairs.

the arrival of a point p_i with weight a_i ($i \in \{1, \dots, k\}$), either ALG matches it with a point of weight 1 or leaves it unmatched. In the latter case, the adversary does not send more points in R and continues with the next region.

In the former case, when ALG matches a point p_i of weight a_i with a point q of weight 1, make the side of $\overline{p_i q}$ that contains at most half of the unmatched points, the active region. The adversary continues putting the remaining points of P in the active region. Thus the unmatched points on the opposite side of $\overline{p_i q}$ stay unmatched, since $\overline{p_i q}$ is between the new point and those unmatched points.

Therefore, after matching p_i and q, the number of unmatched points of weight 1 that can match with future points in P decreases by a factor of at least 2. Let p_j be the first point in P that the algorithm leaves unmatched. Given that the adversary can send up to k points, and there are initially less than $2^k - 1$ unmatched points in R, there exists such p_j of weight a_j. At this point, the adversary ends the procedure for R and continues with the next region.

The total weight of points in matched pairs in R before the arrival of p_j is:

$$M = \underbrace{2}_{\text{for}(x_R, y_R)} + \underbrace{j-1}_{\text{endpoints of weight 1}} + \underbrace{a_1 + a_2 + \ldots + a_{j-1}}_{\text{endpoints with weight } a_i} \leq 2\frac{a_j - 1}{r - 1}.$$

Given that the unmatched point p_j is of weight a_j, the ratio between the weight of matched points and unmatched points is at most $M/a_j \in O(1/r) = O\left(2^{-\sqrt{\log U}}\right)$.

Given that each matched pair is responsible for at least one region, the above procedure creates a mapping of matched points to unmatched points with a weight ratio of $O(2^{-\sqrt{\log U}})$ in all cases, as desired. This finishes the proof. ◀

4.3 Positive Result: The Wait-and-Match Algorithm

We propose an algorithm called "Wait-and-Match" (WAM). Assume the points appear in a bounding box B. Throughout its execution, WAM maintains a "convex partitioning" of B. Initially, there is only one region formed by the entire B. As we will describe, the algorithm

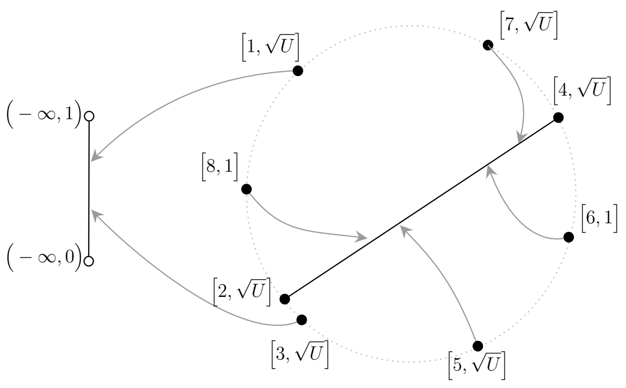

Figure 2 An illustration of the mapping used to analyze WAM. In this example, we have $k = 2$ and 8 points with weights in $\{1, \sqrt{U}, U\}$. Here, $[t, w]$ indicates the t^{th} point in the input sequence having weight w. Note that points 1 and 3 are mapped to the segment corresponding to the imaginary points $(-\infty, 0)$ and $(-\infty, 1)$ of weight U.

matches two points only if they appear in the same convex region. Whenever two points in a convex region R are matched, the line segment between them is extended until it hits the boundary of R, which results in partitioning R into two smaller convex regions. We use the same point classification as defined in Section 4.1.

Suppose a new point p appears, and let R denote the convex region of p. In deciding which point to match p to (if any), the algorithm considers all unmatched points in R in the non-increasing order of their weights. Let q be the next point being considered, and let i be the maximum of the type of p and the type of q. The algorithm matches p with q if there are at least $2^{k-i} - 1$ unmatched points on each side of $\overline{pq}$. If all points in R are examined, and no suitable q exists, p is left unmatched.

Example. Suppose $k = 2$. Then $a_0 = 1, a_1 = \sqrt{U}$, and $a_2 = U$. Let p be a point with weight 1. Upon the arrival of a point p, the algorithm matches p with any point q of weight U when there are at least $2^{2-2} - 1 = 0$ points on each side of $\overline{pq}$. That is, if there is an unmatched point of weight U in the region, the algorithm would match p to it unconditionally. Similarly, if there are no unmatched points of weight U in the region, the algorithm tries to match p with any point q of weight $[a_1 = \sqrt{U}, a_2 = U)$ provided there is at least $2^{2-1} - 1 = 1$ point on each side of $\overline{pq}$. Finally, if previous scenarios do not occur, the algorithm tries to match p with any point q of weight $[a_0 = 1, a_1 = \sqrt{U})$ provided there are at least $2^{2-0} - 1 = 3$ unmatched points on each side of $\overline{pq}$. This will happen if there were at least 7 unmatched points in the region.

To analyze the algorithm, we match each unmatched point into a matched pair. For the sake of analysis, we introduce two "imaginary" points $(-\infty, 0)$ and $(-\infty, 1)$ of weight U and treat them as if they were matched before the input sequence is revealed. Suppose a new point, p, arrives in a region R that is not matched. In this case, we map p to the most recent segment that forms a boundary of the region R. See Figure 2 for an illustration of this mapping.

▶ **Lemma 2.** *Every point of type i is mapped to a segment of type $j \geq i$.*

Proof. For the sake of contradiction, suppose a point p with type i arrives in the region R and gets mapped to $\overline{p_R, q_R}$ of type $\leq i - 1$.

Without loss of generality, assume p_R arrived after q_R. By the definition of the algorithm, at the time p_R appeared, there were at least $2^{k-i+1}-1$ unmatched points in R (otherwise, p_R would not have been matched with q_R). These unmatched points are still unmatched when p appeared (otherwise, R should have been partitioned, and p should have been mapped to some other segment). Thus, when p appeared, the algorithm could match it with the point that bisects these unmatched points, and there would be at least $(2^{k-i+1}-2)/2 = 2^{k-i}-1$ points on each side of the resulting line segment. This contradicts the fact that the algorithm left p unmatched. ◀

▶ **Lemma 3.** *Let s be any line segment between two matched points. For any i, at most $2^{k-i+2}-2$ unmatched points of type i are mapped to s.*

Proof. For the sake of contradiction, assume at least $2^{k-i+2}-1$ points of type i are mapped to s. Then, there must be at least $\lceil (2^{k-i+2}-1)/2 \rceil = 2^{k-i+1}$ points of type i in a convex region R formed by extending s. At the time the last of these points, say p, arrives, it could be matched to the point q that bisects the other points; there will be at least $(2^{k-i+1}-2)/2 = 2^{k-i}-1$ points on each side of $\overline{pq}$. Since $\overline{pq}$ is of type i, the algorithm must have matched p with q, which contradicts the fact that p and q are unmatched and mapped to s. ◀

▶ **Lemma 4.** *Assuming U is sufficiently large, the total weight of unmatched points mapped to a segment of type j is at most $a_{j+1}2^{k-j+3}$.*

Proof. Note that a point of type i has weight at most $a_{i+1} = r^{i+1}$. Hence, by Lemma 2 and Lemma 3, the total weight of unmatched points mapped to a segment of type j is at most

$$\sum_{i=0}^{j} r^{i+1}2^{k-i+2} = 2^{k+2} r \sum_{i=0}^{j} \left(\frac{r}{2}\right)^i = 2^{k+2} r \frac{(r/2)^{j+1}-1}{r/2-1} \le a_{j+1}2^{k-j+3}. \quad ◀$$

Here, we assumed that $r \ge 4$, which holds for a sufficiently large U.

▶ **Theorem 5.** *The competitive ratio of the deterministic online algorithm* Wam *for the Restricted OWNM problem is $\Omega\left(2^{-2\sqrt{\log U}}\right)$.*

Proof. For every matched pair $\overline{pq}$ by Wam consider the set of points formed by p, q, and the unmatched points mapped to them. By Lemma 4, if $\overline{pq}$ has type j, the ratio of the weight of the matched pair over all the points in this set is at least $\frac{a_j}{2a_j + a_{j+1}2^{k-j+3}} \ge \frac{1}{r2^{k+4}}$.

Since the algorithm Wam guarantees that every unmatched point is mapped to some matched pair, the competitive ratio of Wam is at least $2^{-(2k+4)}$, where we used $k = \lceil \sqrt{\log U} \rceil$ and $r = U^{1/k} = 2^{(\log U)/k} \le 2^k$. ◀

5 Randomized Algorithms

5.1 Negative Result

To bound the competitive ratio of randomized algorithms, we will use Yao's minimax principle. We create a randomized unweighted input similar to what Lavasani and Pankratov [25] used for the advice model. We prove an upper bound on the competitive ratio of every deterministic algorithm on this input, and this gives us an upper bound for randomized algorithms in the adversarial setting. We consider a circle and generate points on the circumference of this circle. For a point p, let the left and the right arcs of p be the clockwise and counter-clockwise arcs that are bounded by p.

Put p_1 and p_2 on two arbitrary antipodals of the circle, creating two arcs. Make p_2 the current *active* point. At each step, we choose one of the arcs of the current active point randomly and then we put the next active point on that arc. To deceive the algorithm, sometimes we generate a *fake* point on one of the arcs of the active and then put the next active point on the other arc.

Consider two sequences $L_1, \ldots, L_{2n}$ and $F_1, \ldots, F_{2n}$ of Bernoulli i.i.d. random variables with parameter $1/2$. Iterate the following procedure to make $2n$ points. Let p_i be the current active point, L_i determines the position of p_{i+1}. If L_i is 1, put p_{i+1} in the middle of the left arc of p_i, and if L_i is 0, put it in the middle of the right arc of p_i.

Given p_i is an active point, if F_{i+1} is 1, the point p_{i+1} becomes fake point. Make p_{i+2} the next active point and put it in the middle of the other arc of p_i (e.g. if p_{i+1} is on the left arc of p_i, put p_{i+1} on the right arc of p_i). If F_{i+1} is 0, make p_{i+1} the new active point. Continue the procedure with the new active point.

▶ **Theorem 6.** *No randomized online algorithm can achieve a competitive ratio better than* $16/17$ *in expectation.*

Proof. We aim to bound the competitive ratio of any deterministic algorithm on the described input sequence. Fix a deterministic algorithm ALG. Segments of matched points by ALG divide the circle into convex regions. If an unmatched point is in a region that no new points arrive in, it cannot be matched anymore and we call it an *isolated* point. Given that ALG matches p_i upon its arrival, let X_i be the indicator random variable that p_i is an active point, and matching it causes at least one point to become isolated. Let A_i be the indicator random variable that p_i becomes an active point. For $i \geq 3$, p_i is a fake point if and only if p_{i-1} was an active point and F_i is 1. Thus we can write A_i as $1 - A_{i-1}F_i$.

Suppose p_i is an active point that arrives in a convex region R, that ALG matches upon its arrival, splitting R into R_L and R_R, which contain the left and right arcs of p_i, respectively. If R_L and R_R are both empty, meaning they do not contain any unmatched point, p_{i+1} becomes isolated if it is a fake point. If R_L and R_R are both non-empty and the point p_{i+1} becomes an active point, then the unmatched points of the opposite side of p_{i+1} become isolated. Now suppose R_L is empty and R_R is not empty and p_{i+1} arrives on the left arc of p_i. If p_{i+1} is a fake point, it becomes isolated, and if it is the new active point, unmatched points in R_R become isolated. Similarly, if R_R is empty and R_L is not empty and p_{i+1} arrives in the right arc of p_i, the segment $\overline{p_i p_j}$ creates isolated points. If $i = 2n$, there is no p_{i+1}, and matching p_i makes points isolated if R_L or R_R are not empty. Since we are interested in the asymptotic competitive ratio we can ignore this case. Therefore, given ALG matches p_i we can write X_i as follows.

$$X_i = \begin{cases} A_i F_{i+1} & \text{if } R_L \text{ and } R_R \text{ are empty} \\ A_i L_i & \text{if } R_L \text{ is empty and } R_R \text{ is not} \\ A_i(1 - L_i) & \text{if } R_R \text{ is empty and } R_L \text{ is not} \\ A_i(1 - F_{i+1}) & \text{if } R_L \text{ and } R_R \text{ are not empty} \end{cases}$$

Let the random variable M be the size of the matching made by ALG, and for each $1 \leq i \leq M$, let T_i be the step number in which ALG makes the i^{th} match. Thus, the algorithm is guaranteed to have at least $\sum_{i=1}^{M} X_{T_i}$ unmatched points at the end of the execution. In order to bound the expectation of M, it may be beneficial to view it in the context of the following game. Suppose that ALG has a budget of $2n$ points. The game proceeds in rounds. In round j the algorithm pays 2 points from the budget to make a guess (this corresponds to a pair of points getting matched) of a Bernoulli random variable outcome (which corresponds to ALG's match either resulting in an isolated point or not). If the guess is correct (this corresponds to $X_{T_j} = 0$, no isolated points are guaranteed to be created), then the algorithm does not pay any more points for this round. If the guess is incorrect (this corresponds to $X_{T_j} = 1$), then the algorithm pays one more point from the budget. ALG tries to maximize the total number of rounds before the budget is exhausted. Thus, in round j, the algorithm uses $X_{T_j} + 2$ points from the budget. Overall, M is the largest integer such that $\sum_{j=1}^{M}(X_{T_j} + 2) \leq 2n$. If X_{T_j} were i.i.d., we could use the renewal theorem to bound $\mathbb{E}(M)$. The issue is that X_{T_j} are not i.i.d., because X_i depends on A_i and F_{i+1}; thus there are correlations between X_i and X_{i+1}. The idea is to lower bound the expression $\sum_{j=1}^{M}(X_{T_j} + 2)$ by the sum of some i.i.d. random variables Z_i, compute the corresponding value of M' for the Z_i, and then relate it back to the value of M.

Now we define an auxiliary random variable sequence $Y_1, \dots, Y_M$ as follows:

$$Y_i = \begin{cases} (1 - F_{T_i})F_{T_i+1} & \text{if } R_L \text{ and } R_R \text{ are empty} \\ (1 - F_{T_i})L_{T_i} & \text{if } R_L \text{ is empty and } R_R \text{ is not} \\ (1 - F_{T_i})(1 - L_{T_i}) & \text{if } R_R \text{ is empty and } R_L \text{ is not} \\ (1 - F_{T_i})(1 - F_{T_i+1}) & \text{if } R_L \text{ and } R_R \text{ are not empty} \end{cases}$$

By replacing A_i with $1 - A_{i-1}F_i$, we can see $Y_i \leq X_{T_i}$. Note that $Y_2, Y_4, \dots, Y_{2\lfloor \frac{M}{2} \rfloor}$ are i.i.d. Bernoulli random variables with parameter 1/4. Thus for every $m \leq M$, we can bound $\sum_{j=1}^{m}(X_{T_j} + 2)$ as follows:

$$\sum_{j=1}^{m}(X_{T_j} + 2) \geq \sum_{j=1}^{\lfloor m/2 \rfloor}(X_{T_{2j-1}} + X_{T_{2j}} + 4) \geq \sum_{j=1}^{\lfloor m/2 \rfloor}(Y_{2j-1} + Y_{2j} + 4) \geq \sum_{j=1}^{\lfloor m/2 \rfloor}(Y_{2j} + 4)$$

Let us define yet another auxiliary random variable sequence $Z_1, Z_2, \dots$ as follows. For $1 \leq i \leq \lfloor \frac{M}{2} \rfloor$, let $Z_i = 4 + Y_{2i}$ and for $i > \lfloor \frac{M}{2} \rfloor$ let $Z_i = 4 + Y_i'$ such that Y_i's are i.i.d. Bernoulli random variables with parameter 1/4. This makes the Z_i i.i.d. random variables that take on values of either 4 or 5 with probability 1/4 and 3/4, respectively.

Let the random variable M' be the maximum m such that $\sum_{i=1}^{m} Z_i < 2n$. Note that $\sum_{i=1}^{\lfloor M/2 \rfloor} Z_i = \sum_{i=1}^{\lfloor M/2 \rfloor}(Y_{T_j} + 4) \leq \sum_{i=1}^{M}(X_{T_j} + 2) \leq 2n$. Therefore $M' \geq \lfloor M/2 \rfloor$. Since the Z_i's are i.i.d. and $\mathbb{E}(Z_i) = 17/4$, by the renewal theorem $\mathbb{E}(M') = 8n/17$ and therefore $\mathbb{E}(M)$ is at most $16n/17$. By Yao's minimax principle, this shows an upper bound of 16/17 on the competitive ratio of randomized algorithms in the adversarial model. ◀

5.2 Positive Result: Tree-Guided-Matching Algorithm

We propose a randomized algorithm called "Tree-Guided-Matching" (TGM) that has the following uniform guarantee, regardless of the weights of the points: each point appears in a matching with probability at least 1/3.

The algorithm TGM uses a binary tree to guide its matching decisions. The binary tree is created online, with each node of the tree corresponding to an online point. Intuitively, the binary tree, as it grows, gives an online refining of the partition of the plane into convex regions, such that for each region there is some online point *responsible* for it. Initially, set p_1 as the root of the tree and p_2 the child of p_1. By an abuse of notation, we also use $\overline{pq}$ to denote the straight line determined by points p and q. Let R_1 and R_2 denote the two regions corresponding to the half-spaces created by $\overline{p_1p_2}$. Let p_2 be responsible for both R_1 and R_2. In general, when p_i arrives into a region R for which p_j is responsible (of course, $j < i$), make p_i a child of p_j in the binary tree. The line $\overline{p_ip_j}$ divides the region R into two sub-regions R'

and R'', let p_i be responsible for both of them, and at this point the responsibility of p_j on R is lost as region R has been refined to R' and R''. Note that this implies every node of the tree has at most two children. Next, we describe how TGM chooses to match points. At the beginning, TGM matches p_2 with p_1 with probability $1/3$. After that, upon the arrival of p_i, let p_j be its parent in the tree. If p_j is unmatched and p_i is its first child, match p_i to p_j with probability $1/2$. If p_j is unmatched and p_i is its second child, match p_i to p_j deterministically. Note that TGM only tries to match an online point with its parent in the tree.

▶ **Theorem 7.** *Every point, regardless of its weight, is chosen into a matching by the randomized algorithm TGM with probability at least $1/3$. Hence, TGM achieves a strict competitive ratio at least $1/3$.*

Proof. Note that since TGM only matches a child to its parent in the binary tree, the matching is non-crossing. Indeed, by our construction of the tree, every child is a point *inside*[3] a convex region for which its parent is responsible, and its parent lies on the boundary of that region. Hence, the line segment formed by them does not cross any existing line segment.

Next, we show the claimed performance of TGM. By the definition of TGM, p_1 is matched (by p_2) with probability $1/3$. We will show that every p_i, $i \geq 2$, *upon its arrival* gets matched to its parent with probability exactly $1/3$, which implies the claim. To see this, proceed inductively. The base case is true for p_2. Let p be the currently arrived point and q be its parent. We consider two cases.

- If p is the first child of q, then by the induction hypothesis q at this moment is unmatched with probability $2/3$, hence according to TGM, p is matched (to q) with probability $(2/3) \cdot (1/2) = 1/3$.
- If p is the second child of q, then q at this moment is unmatched with probability $1 - 1/3 - 1/3 = 1/3$. By TGM, p is matched (to q) with probability $(1/3) \cdot 1 = 1/3$. ◀

6 Revocable Acceptances

In this section, we consider the revocable setting. When a new point p arrives, an algorithm has an option of removing one of the existing edges from the matching prior to deciding on how to match p. The decision to remove an existing edge is irrevocable. The benefit of making this decision is that the end-points of the removed edge, along with possible points on the other side of the edge (though our positive result does not use this possibility), become available candidates to be matched with p, provided the non-crossing constraint is respected.

6.1 Negative Result

Bose et al. [11] showed that a deterministic greedy algorithm without revoking can achieve $2/3$ competitive ratio in the unweighted version. In this section, we prove that in the unweighted version, no deterministic algorithm with revoking can beat the ratio $2/3$.

▶ **Theorem 8.** *No deterministic algorithm with revoking can achieve a competitive ratio better than $2/3$ even in the unweighted version.*

[3] Recall that online points are in general position.

Proof. Fix a deterministic algorithm ALG, an arbitrary large n, and a circle in the plane. The adversary adds at least $2n$ points, all of weight 1, on the circle, one by one, and let ALG match them into pairs. We maintain the invariants that there is always one active region of the circle, and that for each matched pair, there is always at least one unmatched point.

Initially, the entire circle is the active region. A phase consists of the adversary presenting points on the circle, in the active region, until ALG either matches a pair or revokes a matching, or until $2n$ points have been given. The adversary stops if there are $2n$ points and the last point is unmatched. Otherwise, if a match has just occurred, there are two cases.

In Case 1, the current point, p, is simply matched to a point, q, on the circle. The chord $\overline{pq}$ divides the active region into two sub-regions, R_1 and R_2. If neither region has any points, add a point, p', to R_1. Without loss of generality, assume that R_1 contains at least as many unmatched points as R_2. If p' is matched, ALG has revoked a matching; and we get the extra point from Case 2. Otherwise, R_2 becomes the active region, some unmatched point in R_1 is associated with the matched pair, and the phase ends.

In Case 2, ALG revokes a matching and either matches the current point, p, or leaves p unmatched. Removing the one match, removes a chord of the circle, joining two regions into a new convex region. This region is the active region if p is not matched. In either case, the number of matched points is not increased. However, the number of unmatched points is increased by at least 1, since at least one of the points, q, from the revoked match is now unmatched and p is only matched to one point. If there is a new match for p, the sub-region created by the match that does not contain q becomes the active region, and q is the unmatched point associated with the new matched pair. The current phase ends.

Inductively, the invariants hold after each phase, and the unmatched point associated with each matched pair ensures that no more than 2/3 of the points are matched. Although the number of points may be odd, this gives an asymptotic lower bound of 2/3 on the competitive ratio. ◀

Note that ignoring the revoking option, the above proof is a simpler alternative to bound the competitive ratio of the deterministic algorithm which was given by Bose et al. [11].

6.2 Positive Result: Big-Improvement-Match

We present a deterministic algorithm with revoking, called "Big-Improvement-Match" (Bim). This algorithm has a strict competitive ratio of ≈ 0.2862 even when weights of points are unrestricted. This shows that while revoking does not improve the competitive ratio in the unweighted version, it provides us with an algorithm with a constant competitive ratio, which is unattainable for a deterministic algorithm without revoking.

Bim maintains a partitioning of the Euclidean space into regions. Each region in the partition is assigned an edge from the current matching to be responsible for that region. Each edge can be responsible for up to two regions. Bim starts out by matching the first two points, p_1 and p_2 regardless of their weights, dividing the plane into two half-planes by $\overline{p_1p_2}$. Bim then assigns $\overline{p_1p_2}$ to be responsible for the two half-plane regions. Next, consider a new point p_i (for $i \geq 3$) that arrives in an existing region R. Suppose that $\overline{p_jp_{j'}}$ is the responsible edge for R. If there is at least one unmatched point in R, Bim matches p_i with an unmatched point p_k in R with the highest weight. Then $\overline{p_jp_{j'}}$ is no longer responsible for R, and the region R is divided into two new regions by $\overline{p_ip_k}$. The responsibility for both new regions is assigned to $\overline{p_ip_k}$. If p_i is the only point in R, then Bim decides to revoke the matching $(p_j, p_{j'})$ or not as follows. Without loss of generality, assume $w(p_j) \leq w(p_{j'})$. If $w(p_i) < rw(p_{j'})$, then Bim leaves p_i unmatched. Otherwise, Bim removes the matching $(p_j, p_{j'})$ and matches

p_i with $p_{j'}$. We note that r is a parameter that is going to be chosen later so as to optimize the competitive ratio. If R is the only region that $\overline{p_j p_{j'}}$ was responsible for when p_i arrived, then R is divided into two regions by $\overline{p_i p_{j'}}$, and $\overline{p_i p_{j'}}$ becomes responsible for the two new regions. (The regions on the other side of $\overline{p_j p_{j'}}$ from p_i keep their boundaries, even though $(p_j, p_{j'})$ is no longer in the matching.) Otherwise $\overline{p_j p_{j'}}$ was responsible for R' in addition to R when p_i arrived. In this case, after removal of the match $(p_j, p_{j'})$, regions R and R' are merged to give region $R'' = R \cup R'$, and R'' is divided by $\overline{p_i p_{j'}}$ into two regions, and Bim makes $\overline{p_i p_{j'}}$ responsible for both new regions.

▶ **Proposition 9.** *The following observations concerning* Bim *hold:*
1. *All responsible edges are defined by two currently matched points.*
2. *Each edge is responsible for at most two regions.*
3. *All regions are convex.*
4. *When a matched edge $(p_j, p_{j'})$ is replaced due to the arrival of a point p_i in region R, then edge $(p_i, p_{j'})$ is contained in R.*

Proof. (1) follows since an edge only becomes responsible when its endpoints become matched. When another edge becomes responsible for a region, the original edge is no longer responsible. (2) follows since the only two regions an edge is made responsible for are the two regions created when the endpoints of the edge were matched. When two points in one of the regions an edge is responsible for are matched, the edge is no longer responsible for that region, but will still be responsible for one region if it had been responsible for two up until that point. (3) follows inductively, since separating two convex regions by a line segment creates two convex regions. In addition, when Bim removes an edge, that edge was the last matching created in either of the two regions it was responsible for. (4) follows by (3). ◀

▶ **Theorem 10.** Bim *with $r \in (1, \sqrt{2}]$ has strict competitive ratio at least $\min\left(\frac{r^2-1}{r^3}, \frac{1}{1+2r}\right)$ for the OWNM with arbitrary weights.*

Proof. We consider for each region an edge is responsible for, the total weight of unmatched points in that region. These points come in two flavours: those that were matched at some point during the execution, but due to revoking became unmatched, and those that were never matched during the entire execution of the algorithm.

Consider any subsequence of all created edges, $\langle e_1, \ldots, e_k \rangle$, where e_1 was created when a second unmatched point arrived in some region, and the possible remaining edges were created via revokings, i.e., e_i caused e_{i-1} to be revoked for $2 \le i \le k$, and e_k is in Bim's final matching. Let $e_j = (p_{i_j}, p_{i_{j+1}})$ and $w(p_{i_j}) \le w(p_{i_{j+1}})$, so $e_{j+1} = (p_{i_{j+1}}, p_{i_{j+2}})$. Thus, for $3 \le j \le k+1$, p_{i_j} arrived after $p_{i_{j-1}}$. Every pair ever matched by Bim is included in some such sequence of edges. The points, $p_{i_1}, \ldots, p_{i_{k-1}}$ could be unmatched points in a region for which e_k is responsible.

Let $\alpha = w(p_{i_k})$, so for every $2 \le j \le k$, $w(p_{i_j})$ is at most $\alpha r^{-(k-j)}$ and $w(p_{i_1}) \le w(p_{i_2}) \le \alpha r^{-(k-2)}$. Let $\beta = w(p_{i_{k+1}})$. The total weight of points in this sequence is

$$\begin{aligned}\sum_{j=1}^{k+1} w(p_{i_j}) = w(p_{i_1}) + \sum_{j=2}^{k} w(p_{i_j}) + w(p_{i_{k+1}}) &\le \alpha r^{-(k-2)} + \alpha\left(\frac{r}{r-1}\right)(1 - r^{-(k-1)}) + \beta \\ &= \alpha\left(r^{-(k-1)}\frac{r(r-2)}{r-1} + \frac{r}{r-1}\right) + \beta.\end{aligned}$$

Now, we consider other points that were never matched, but were at some time in a region for which one of the e_j was responsible. After e_1 is created and before e_2, a first point q_1 could arrive in one of the regions for which e_1 is responsible. Note that q_1 is not matched if $w(q_1) < rw(p_{i_2})$. (Note that a second point arriving in that region will then be matched to q_1, dividing the region, and the sub-regions will not be considered part of the region for which e_k eventually becomes responsible.) Now, suppose that another point, q_2, arrives between when e_j and e_{j+1} are created for some $2 \leq j < k$, remaining unmatched in one of the regions for which e_k is responsible. Then, neither q_2 nor $p_{i_{j+2}}$ is in the same region as $p_{i_{j-1}}$ or one of them would have been matched to $p_{i_{j-1}}$ (or $p_{i_{j-1}}$ was already matched and the region divided). By Proposition 9.2, e_i is responsible for at most two regions, so $p_{i_{j+2}}$ arrives in the same region as q_2, while unmatched. This is a contradiction, since BIM would match them. Thus, other than q_1, the only never-matched point, q_2, in a region for which e_k is responsible, arrives after e_k and $w(q_2) < rw(p_{i_{k+1}})$.

Then, for $k \geq 2$, the total weight of unmatched points for which e_k is responsible is at most $\left(r^{-(k-3)} + r^{-(k-1)}\frac{r(r-2)}{r-1} + \frac{r}{r-1}\right)\alpha + (1+r)\beta$. If $r \leq \sqrt{2}$, then $r^{-(k-3)} + r^{-(k-1)}\frac{r(r-2)}{r-1}$ is at most zero and we can bound the total weight when $k \geq 2$ by: $(\frac{r}{r-1})\alpha + (1+r)\beta$. Thus, the ratio between the weight of matched points in sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_{k+1}}$ and the total weight of all points associated with this sequence for $k \geq 2$ is at least $\frac{\alpha+\beta}{(\frac{r}{r-1})\alpha+(1+r)\beta}$. Since $\frac{r}{r-1} > 1 + r$, for $1 < r \leq \sqrt{2}$, this ratio is minimized when β is minimized, which happens at $\beta = r\alpha$. Thus, the competitive ratio for $k \geq 2$ is at least $(1+r)/(\frac{r}{r-1} + r(1+r)) = \frac{r^2-1}{r^3}$.

Now, consider the case of $k = 1$, and let α and β have the same meaning as above. Then the sequence $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ consists of a single edge. Thus, the weight of the matched points is $\alpha + \beta$, and there could be two unmatched points q_1 and q_2 at the end of the execution of the algorithm charged to this edge. We have $w(q_i) < r\beta$, so the ratio between the weight of matched points and the total weight of all points associated with the sequence in case of $k = 1$ is at least $\frac{\alpha+\beta}{\alpha+(1+2r)\beta}$. Observe that this ratio is minimized when β goes to infinity and becomes $1/(1 + 2r)$.

Taking the worse ratio between the above two scenarios proves the statement of the theorem. ◀

▶ **Corollary 11.** *With the choice of parameter for* BIM, r^*, *defined as the positive solution to the equation* $\frac{1}{1+2r} = \frac{r^2-1}{r^3}$, *approximately* 1.2470, *we get a competitive ratio of* $\frac{1}{1+2r^*}$, *at least* 0.2862.

Proof. The value r^* is obtained by setting the two terms in the minimum in Theorem 10 equal to each other and solving for r, giving the lower bound on the competitive ratio.

To show that this result is tight, consider the following input: p_1 of weight α arrives at the north pole of the unit sphere, followed by p_2 of weight $\beta \geq \alpha$ at the south pole of the unit sphere, followed by p_3 of weight $r^*\beta - \epsilon$ at the west pole of the unit sphere, and followed by p_4 of weight $r^*\beta - \epsilon$ at the east pole of the unit sphere. The algorithm would end up matching p_1 with p_2, leaving p_3 and p_4 unmatched. Thus, in such an instance, the competitive ratio of the algorithm is $(\alpha + \beta)/(\alpha + \beta + 2r^*\beta - 2\epsilon)$. Taking β to ∞ and ϵ to 0 shows that the algorithm does not guarantee a competitive ratio better than $1/(1 + 2r^*)$ in the strict sense. ◀

7 Algorithms with Advice

In this section, we consider the OWNM problem in the tape advice setting. In the advice setting, a trustworthy oracle cooperates with an online algorithm according to a pre-agreed protocol. The oracle has access to the entire input sequence in advance. The oracle communicates with an online algorithm by writing bits on the advice tape. When an input item arrives, an algorithm reads some number of advice bits from the tape, and makes a decision for the new item based on the advice it read from the tape so far, and the items that have arrived so far. The question of interest is to bound the number of bits that need to be communicated between the oracle and the algorithm on the worst-case input to allow an online algorithm to solve the problem optimally. For an introduction to online algorithms with advice, an interested reader is referred to the survey [12] and references therein.

We propose an online algorithm with advice, which we call "Split-And-Match" (Sam), and show that it achieves optimality. For input sequences of size $2n$, Sam uses a family of C_n advice strings, where C_n is the n^{th} Catalan number. The oracle encodes each advice string using Elias delta coding scheme [16], which requires $\lceil \log C_n \rceil + \log n + O(\log\log(C_n))$ bits. We ignore the $O(\log\log(C_n))$ term for simplicity.

The Sam oracle and algorithm jointly maintain a partitioning of the plane into convex regions, and a responsibility relation, where a point can be assigned to be responsible for at most one region, and each region can have at most one point responsible for it. Each region defined will eventually receive an even number of points in total. No region which has not yet been divided into sub-regions contains more than one unmatched point. When a new point p arrives in a region R, if R does not have a responsible point, then p is assigned to be the responsible point for R, and p is left unmatched at this time. Otherwise, suppose that q is the responsible point for R at the time p arrived. In this case, the responsibility of q is removed, and the plane partition is refined by subdividing R into R_1 and R_2 – the sub-regions of R formed by $\overline{pq}$. If the total number of points (including future points, but excluding p and q) in R_i is even for each $i \in \{1, 2\}$, then p and q are matched (we refer to this event as a "safe match"), and R_1 and R_2 do not have any responsible points assigned to them. Otherwise, p and q are not matched, and q is made responsible for R_1, and p is made responsible for R_2. Note that when a region has a responsible point, that point is assumed to lie in the region by convention, though it can lie on the boundary.

To implement the above procedure in the advice model, the Sam oracle creates a binary string D of length $2n$, where the i^{th} bit indicates whether p_i arrives in a region which has some responsible point p_j assigned to it, and p_i and p_j form a safe match. The string D is encoded on the tape and is passed to Sam. The Sam algorithm reads the encoding of D from the tape (prior to the arrival of online points), recovers D from the encoding, and then uses the information in D to run the above procedure creating safe matches.

Observe that we aim to show the bound $\log C_n + \log n \sim 2n - \frac{1}{2}\log n$ on the advice complexity. The reason for the additive savings of $\frac{3}{2}\log n$ in the $\log C_n$, as compared to $2n$, is that not all binary strings of length $2n$ can be generated as a valid D. Thus, the oracle and the algorithm can agree beforehand on the ordering of the universe of possible strings D, which we call the advice family. Then the oracle writes on the tape the index of a string in this ordering that corresponds to D for the given input. The following theorem establishes the correctness of this algorithm, as well as the claimed bound on the advice complexity.

▶ **Theorem 12.** *Sam achieves a perfect matching with the advice family of size C_n.*

Proof. Since the request sequence contains $2n$ points, an even number of points eventually arrive. Inductively, a region that is divided always has an even number of points in both sub-regions, and each of these sub-regions is convex. Thus, any point arriving in a region can be matched to the point that is already there and responsible for the region. There are only two kinds of regions that occur during the execution of Sam, as a new point arrives:

- type I: this region does not have a responsible point, it is empty at the time of creation, and there are an even number of points arriving in this region in the future, and
- type II: this region has a responsible point, which is the only point in the region at the time of its creation, and there are an odd number of points arriving in this region in the future.

We argue inductively (on the number of future points arriving in a region) that the algorithm ends up matching all points inside a region, regardless of their type. The base case for a type I region is trivial: the number of future points is 0, and there is nothing to prove. The base case for type II region is easy: one point arrives in the region, then according to the algorithm it will be matched to the responsible point (since R_1 and R_2 are empty).

For the inductive step, consider a type I region R, and suppose that $2k$ points will arrive inside the region. The first point that arrives in the region becomes responsible for this region, changing its type to II. There are $2k-1$ future points arriving in this region, and the claim follows by the inductive assumption applied to the type II region. Now, consider a type II region R, and suppose that $2k-1$ points arrive inside the region. Let q be the responsible point for R, and let p be the first point arriving inside R. Note that $\overline{pq}$ partitions R into R_1 and R_2. There are two possible cases. Case 1: If R_1 and R_2 are both of type I, then p is matched with q and the inductive step is established for R by invoking induction on R_1 and R_2. If Case 2: R_1 and R_2 are both of type II, then inductive step is established for R by invoking induction on R_1 and R_2.

Observe that the entire plane is a region of type I at the beginning of the execution of the algorithm (prior to arrival of any points). Thus, correctness of the algorithm follows by applying the above claim to this region.

To establish the bound on advice complexity, observe that by the definition of the algorithm, SAM matches the most recent point whenever $D[i]$ is 1 and does not match otherwise. Thus, D has an equal number of zeros and ones and no prefix of D has more ones than zeros. This makes D a Dyck word and it is known that there are C_n Dyck words of size $2n$ [29]. ◀

8 Conclusion

We introduced the weighted version of the Online Weighted Non-Crossing Matching problem. We established that no deterministic algorithm can guarantee a constant competitive ratio for this problem. Then, we explored several ways of overcoming this limitation and presented new algorithms and bounds for each of the considered regimes. In particular, we presented the results for deterministic algorithms when weights of the points are restricted to lie in the range $[1, U]$, randomized algorithms without restrictions on weights, deterministic algorithms with revoking, and deterministic algorithms with advice. Many open problems remain. In particular, our bounds are not tight, and closing the gap in any of the settings would be of interest. It is also interesting to study the online setting of other versions of the problem that were considered in the offline literature. For example, one could allow an online algorithm to create some number of crossings up to a given budget, or one could consider the k-non-crossing constraint as inspired from understanding RNA structures. In this paper, we considered the vertex-weighted version, but one could also consider an edge-weighted version of the problem, where edge weights could be either abstract, or related to geometry.

References

1 A Karim Abu-Affash, Paz Carmi, Matthew J Katz, and Yohai Trabelsi. Bottleneck non-crossing matching in the plane. *Computational Geometry*, 47(3):447–457, 2014.

2 Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport, Shakhar Smorodinsky, et al. Compatible geometric matchings. *Computational Geometry*, 42(6-7):617–626, 2009.

3 Oswin Aichholzer, Sergio Cabello, Ruy Fabila-Monroy, David Flores-Penaloza, Thomas Hackl, Clemens Huemer, Ferran Hurtado, and David R Wood. Edge-removal and non-crossing configurations in geometric graphs. *Discrete Mathematics & Theoretical Computer Science*, 12(Graph and Algorithms), 2010.

4 Oswin Aichholzer, Ruy Fabila-Monroy, Philipp Kindermann, Irene Parada, Rosna Paul, Daniel Perz, Patrick Schnider, and Birgit Vogtenhuber. Perfect matchings with crossings. *Algorithmica*, pages 1–20, 2023.

5 Noga Alon, Sridhar Rajagopalan, and Subhash Suri. Long non-crossing configurations in the plane. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 257–263, 1993.

6 Greg Aloupis, Esther M Arkin, David Bremner, Erik D Demaine, Sándor P Fekete, Bahram Kouhestani, and Joseph SB Mitchell. Matching regions in the plane using non-crossing segments. *XVI Spanish Meeting on Computational Geometry*, 2015.

7 Drew Armstrong, Christian Stump, and Hugh Thomas. A uniform bijection between nonnesting and noncrossing partitions. *Transactions of the American Mathematical Society*, 365(8):4121–4151, 2013.

8 Mikhail J Atallah. A matching problem in the plane. *Journal of Computer and System Sciences*, 31(1):63–70, 1985.

9 Vineet Bafna, S Muthukrishnan, and R Ravi. Computing similarity between rna strings. In *Combinatorial Pattern Matching: 6th Annual Symposium, CPM 95 Espoo, Finland, July 5–7, 1995 Proceedings 6*, pages 1–16. Springer, 1995.

10 József Balogh, Boris Pittel, and Gelasio Salazar. Near-perfect non-crossing harmonic matchings in randomly labeled points on a circle. *Discrete Mathematics & Theoretical Computer Science*, Proceedings, 2005.

11 Prosenjit Bose, Paz Carmi, Stephane Durocher, Shahin Kamali, and Arezoo Sajadpour. Non-crossing matching of online points. In *32 Canadian Conference on Computational Geometry*, 2020.

12 Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: a survey. *ACM Computing Surveys*, 50(32):1–34, 2017. Article No. 19.

13 William YC Chen, Hillary SW Han, and Christian M Reidys. Random k-noncrossing rna structures. *Proceedings of the National Academy of Sciences*, 106(52):22061–22066, 2009.

14 Scott Cohen. *Finding color and shape patterns in images.* PhD dissertation, Stanford University, 1999.

15 Adrian Dumitrescu and William Steiger. On a matching problem in the plane. *Discrete Mathematics*, 211(1-3):183–195, 2000.

16 Peter Elias. Universal codeword sets and representations of the integers. *IEEE Trans. Inf. Theory*, 21(2):194–203, 1975.

17 András Gyárfás. Ramsey and turán-type problems for non-crossing subgraphs of bipartite geometric graphs. In *Annales Univ. Sci. Budapest*, volume 54, pages 47–56, 2011.

18 Koki Hamada, Shuichi Miyazaki, and Kazuya Okamoto. Strongly stable and maximum weakly stable noncrossing matchings. *Algorithmica*, 83(9):2678–2696, 2021.

19 John Hershberger and Subhash Suri. Applications of a semi-dynamic convex hull algorithm. *BIT Numerical Mathematics*, 32(2):249–267, 1992.

20 John Hershberger and Subhash Suri. Efficient breakout routing in printed circuit boards. In *Workshop on Algorithms and Data Structures*, pages 462–471. Springer, 1997.

21 Shahin Kamali, Pooya Nikbakht, and Arezoo Sajadpour. A randomized algorithm for non-crossing matching of online points. In *34th Canadian Conference on Computational Geometry*, pages 198–204, 2022.

22 Mikio Kano and Jorge Urrutia. Discrete geometry on colored point sets in the plane—a survey. *Graphs and Combinatorics*, 37(1):1–53, 2021.

23 Ioannis Mantas, Marko Savić, and Hendrik Schrezenmaier. New variants of perfect non-crossing matchings. *Discrete Applied Mathematics*, 343:1–14, 2024.

24 Neeldhara Misra, Harshil Mittal, and Saraswati Nanoti. Diverse non crossing matchings. In *34th Canadian Conference on Computational Geometry*, pages 249–256, 2022.

25 Ali Mohammad Lavasani and Denis Pankratov. Advice complexity of online non-crossing matching. *Computational Geometry*, 110:101943, 2023. doi:10.1016/j.comgeo.2022.101943.

26 János Pach, Natan Rubin, and Gábor Tardos. Planar point sets determine many pairwise crossing segments. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1158–1166, 2019.

27 Rebecca Patrias, Oliver Pechenik, and Jessica Striker. A web basis of invariant polynomials from noncrossing partitions. *Advances in Mathematics*, 408:108603, 2022.

28 David Rappaport. Tight bounds for visibility matching of f-equal width objects. In *Japanese Conference on Discrete and Computational Geometry*, pages 246–250. Springer, 2002.

29 Steven Roman. *An Introduction to Catalan Numbers*. Compact Textbooks in Mathematics. Brikhäuser, 2015.

30 Suthee Ruangwises and Toshiya Itoh. Stable noncrossing matchings. In *International Workshop on Combinatorial Algorithms*, pages 405–416. Springer, 2019.

31 Arezoo Sajadpour. Non-crossing matching of online points. Master's thesis, University of Manitoba, 2021.

32 Toshinori Sakai and Jorge Urrutia. Heavy non-crossing increasing paths and matchings on point sets. In *XIV Spanish Meeting on Computational Geometry*, pages 209–212, 2011.

33 David Savitt. Polynomials, meanders, and paths in the lattice of noncrossing partitions. *Transactions of the American Mathematical Society*, 361(6):3083–3107, 2009.

34 Alexander A Vladimirov. Non-crossing matchings. *Problems of Information Transmission*, 49(1):54–57, 2013.

A Omitted Pseudocode

Algorithm 1 *Split-And-Match* Oracle.

```
procedure Split-And-Match-Oracle
    $D \leftarrow [0]$
    make $p_1$ responsible for the plane
    for $i = 2$ to $2n$ do
        let $R$ be the region that $p_i$ arrives in
        if $R$ has a responsible point $p_j$ then
            revoke the responsibility of $p_j$
            divide $R$ into $R_1$ and $R_2$ by $\overline{p_i p_j}$
            if $R_L$ (and $R_R$) is going to contain an even number of points in total then
                $D.append(1)$
            else
                make $p_j$ and $p_i$ responsible for $R_1$ and $R_2$ respectively
                $D.append(0)$
        else
            make $p_i$ responsible for $R$
            $D.append(0)$
    pass $D$ to the algorithm
```

Algorithm 2 *Split-And-Match* Algorithm.

```
procedure $Split\text{-}And\text{-}Match(D)$
    while receive a new point $p_i$ do
        let $R$ be the region that $p_i$ arrives in
        if $R$ has a responsible point $p_j$ then
            revoke the responsibility of $p_j$
            divide $R$ into $R_1$ and $R_2$ by $\overline{p_i p_j}$
            if $D[i] == 1$ then
                match $p_i$ with $p_j$
            else
                make $p_j$ and $p_i$ responsible for $R_1$ and $R_2$ respectively
                leave $p_i$ unmatched
        else
            make $p_i$ responsible for $R$
            leave $p_i$ unmatched
```