The Point Placement Problem on a Line - Improved Bounds for Pairwise Distance Queries*

Francis Y.L. Chin[1] Henry C.M. Leung[1] W.K. Sung[2] S.M. Yiu[1]

[1] Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong
{chin, cmleung2, smyiu}@cs.hku.hk
[2] Department of Computer Science, National University of Singapore, Singapore
ksung@comp.nus.edu.sg

Abstract. In this paper, we study the adaptive version of the point placement problem on a line, which is motivated by a DNA mapping problem. To identify the relative positions of n distinct points on a straight line, we are allowed to ask queries of pairwise distances of the points in rounds. The problem is to find the number of queries required to determine a unique solution for the positions of the points up to translation and reflection. We improved the bounds for several cases. We show that $4n/3 + O(\sqrt{n})$ queries are sufficient for the case of two rounds while the best known result was $3n/2$ queries. For unlimited number of rounds, the best result was $4n/3$ queries. We obtain a much better result of using $5n/4 + O(\sqrt{n})$ queries with three rounds only. We also improved the lower bound of $30n/29$ to $17n/16$ for the case of two rounds.

1 Introduction

The point placement problem on a line is defined as follows. Suppose that there are n points located at n distinct positions on a straight line. We are allowed to ask queries of pairwise distances of the points. The problem is to find the number of queries required to determine a unique solution for the positions of the points up to translation and reflection. The points are distinguishable (i.e., each point has a unique label). In this paper, we study the adaptive version of the problem. We submit our queries in rounds. In each round, the queries to be asked can be based on the answers to the queries of the previous rounds. For the non-adaptive version of the problem, only one round of queries is allowed.

The problem is motivated by a biological problem, called DNA mapping. In general, we are interested to recover the whole DNA sequence for an organism. One approach is to make use of some known short substrings, called *markers* or *restriction sites*. The first step in this approach is to find the relative positions of these markers in the DNA sequence. Unfortunately, there is no direct method to identify these positions. A common technique for handling it is the *flourescent in situ hybridization* (FISH) [5] Given any two markers, a FISH experiment can

* This research is supported by the RGC grant HKU7119/05E.

measure the distance between the markers in the sequence. The locations of these markers can be deduced if enough information about the pairwise distances of the markers are known. For more details about this problem, one can refer to [6]. Note that doing experiments incurs resources such as time and money. To reduce the cost, one would like to perform as few experiments as possible. We can design the experiments adaptively depending on the results of all previous experiments. However, this is too time-consuming. A more reasonable approach is to perform the experiments in a few rounds. In each round, we design a set of FISH experiments based on the measurements obtained from the previous rounds and perform the designed FISH experiments in parallel. This motivates the study of the adaptive versions of the point placement problem on a line.

The problem is easy to describe and solve optimally if the point positions are not distinct; it can be shown in [1] that $C(n, 2)$ nonadaptive and $2n-3$ adaptive queries are necessary and sufficient to solve the problem. However when the point positions are distinct, a more realistic assumption for the DNA mapping problem in which each maker should be distinct, this point placement problem becomes surprisingly difficult. It was shown that $8n/5$ queries are sufficient while the lower bound is $4n/3$ except for some small n cases [1]. For the case of two rounds, it was shown that $3n/2$ queries are sufficient [1] while the lower bound could only be shown to be $30n/29$ [2]. For unlimited number of rounds, the upper bound result of $3n/2$ in [1] was improved to $4n/3$ in [2] (to be more precise, they derived a strategy with $[4/3 + O(1/t)]n$ queries for $O(t)$ rounds, note that this result only applies to the case when the number of rounds is sufficiently large). So, the best upper bound for two rounds was still $3n/2$ while for unlimited number of rounds, the best upper bound was $4n/3$. In this paper, we manage to improve these results and our contribution is summarized in the following table. For two rounds, we obtained an upper bound of $4n/3 + O(\sqrt{n})$ queries and improved the lower bound to $17n/16$. For unlimited number of rounds, we are able to reduce the number of queries to $5n/4 + O(\sqrt{n})$ and we only need three rounds.

	Two Rounds	Unlimited number of Rounds
Upper Bound	$3n/2 \rightarrow 4n/3 + O(\sqrt{n})$	$4n/3 \rightarrow 5n/4 + O(\sqrt{n})$ in 3 rounds
Lower Bound	$30n/29 \rightarrow 17n/16$	—

The improvement we obtained is based on an observation on the induced graph (we call it the point placement graph) of the given points and the pairwise distances being queried. We give a characterization to these graphs for which the corresponding locations of the points can be determined uniquely (up to translation and reflection). The definitions of the point placement graph and the characterization will be given in Section 2. Section 3 shows the algorithm for solving the two-round case. The solution for the three-round case is presented in Section 4. The lower bound for the two-round case is discussed in Section 5.

As a remark, there are some other related works in this area. For example, [4] studied the problem of finding the best estimation of the locations of points from a partial set of pairwise distances of the points. In their study, they assumed that there may be errors in measuring the pairwise distances and provided heuristics algorithms to solve the problem. In [3], a similar problem has been studied by assuming that the pairwise distances are given in terms of distance intervals. They proposed a randomized algorithm to solve the problem. They also proved that the problem is NP-hard even for those point alignment graphs which have only chordless cycles.

2 Preliminaries

Given a set S of n distinct points, p_1, p_2, ..., p_n, on a line and a set D of pairwise distances of some of these points, we can construct a point placement graph, $G(S, D)$, as follows. Each point is represented as a node and if the distance between p_i and p_j is given, there is an edge between the nodes representing p_i and p_j with weight equal to the given distance. The distance is denoted as $|p_i p_j|$. Not every point placement graph corresponds to a unique set of n distinct points (up to translation and reflection) on a line. For a given point placement graph, any set of n distinct points on a line for which the pairwise distances are consistent with the graph is called a linear layout of the graph. The following lemma shows a simple observation on a point placement graph.

Lemma 1. *There are at most 2 edges with the same length l adjacent to any node in the graph G. [2]*

Proof. Consider the linear layout of G, there are at most two distinct points whose distance from a node is l.

Definition 1. *A point placement graph G is* line rigid *if there is only one unique set of n distinct points (up to translation and reflection) on a line that is consistent with G, i.e. there exists a unique linear layout of G.*

The following shows a characterization of a line rigid point placement graph. We first define what a layer graph is.

Definition 2. *A point placement graph G is called a layer graph if G can be plotted in a xy-plane (represented by two unit vectors $\boldsymbol{x}$ and $\boldsymbol{y}$) with the following properties.*

1. *All edges uv are parallel to $\boldsymbol{x}$ or $\boldsymbol{y}$, i.e. $|\boldsymbol{u} - \boldsymbol{v}| = (\boldsymbol{u} - \boldsymbol{v}) \cdot \boldsymbol{x}$ or $(\boldsymbol{u} - \boldsymbol{v}) \cdot \boldsymbol{y}$*
2. *Length of edge uv, $|\boldsymbol{u} - \boldsymbol{v}|$, is the same as the weight of the edge.*
3. *There are two nodes p and q with different x-coordinates and y-coordinates. i.e. $(\boldsymbol{p} - \boldsymbol{q}) \cdot \boldsymbol{x} \neq |\boldsymbol{p} - \boldsymbol{q}|$ and $(\boldsymbol{p} - \boldsymbol{q}) \cdot \boldsymbol{y} \neq |\boldsymbol{p} - \boldsymbol{q}|$.*
4. *When the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ tends to 0 or π, no nodes overlap.*

Figure 1 contains three examples of layer graphs with $\boldsymbol{x}$ and $\boldsymbol{y}$ as two perpendicular vectors.

Theorem 1. *A point placement graph G is line rigid iff it is not a layer graph.*

Proof. Without loss of generality, we assume that G is connected, otherwise, Theorem 1 can be proved by considering each connected component of G. If G is a layer graph, we can find two linear layouts of the graph by considering angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ tends to 0 and π (Property 4). Assume that G is not line rigid. Consider a spanning tree S of G and two linear layouts P and P' of G. Pick any node of S as the root. We plot the root of S at the origin and the remaining edges uv with length l in S as follows:

1. Node v is on the right of node u in both P and P', $\boldsymbol{v} = \boldsymbol{u} + l\boldsymbol{x}$.
2. Node v is on the left of node u in both P and P', $\boldsymbol{v} = \boldsymbol{u} - l\boldsymbol{x}$.
3. Node v is on the right of node u in P and on the left of node u in P', $\boldsymbol{v} = \boldsymbol{u} + l\boldsymbol{y}$.
4. Node v is on the left of node u in P and on the right of node u in P', $\boldsymbol{v} = \boldsymbol{u} - l\boldsymbol{y}$.

It is easy to see that the spanning tree S constructed by the above procedure satisfies Property 1, 2, and 3 of a layer graph. When the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ tends to 0 and π, S is degenerated to a straight line on the x-axis with the positions of all vertices the same as the linear layouts P and P' respectively, therefore S satisfy property 4 and is a layer graph.

Consider the edges of G not in S. Let uv be a length-l edge between vertices $u(x_1, y_1)$ and $v(x_2, y_2)$. When the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ tends to 0, i.e. the positions of the vertices are the same as the linear layout P, $v(x_2, y_2)$ is on the right of $u(x_1, y_1)$ with distance $(x_2 - x_1) + (y_2 - y_1)$. Note that when $(x_2 - x_1) + (y_2 - y_1)$ is negative, $v(x_2, y_2)$ is on the left of $u(x_1, y_1)$. When angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ tends to π, i.e. the positions of the vertices are the same as the linear layout P', $v(x_2, y_2)$ is on the right of $u(x_1, y_1)$ with distance $(x_2 - x_1) - (y_2 - y_1)$. Note that when $(x_2 - x_1) - (y_2 - y_1)$ is negative, $v(x_2, y_2)$ is on the left of $u(x_1, y_1)$. Since $|(x_2 - x_1) + (y_2 - y_1)| = |(x_2 - x_1) - (y_2 - y_1)| = l$, we have two possible solutions: i) $(x_2 - x_1) = l$ and $(y_2 - y_1) = 0$, ii) $(x_2 - x_1) = 0$ and $(y_2 - y_1) = l$. Therefore, uv is either parallel to $\boldsymbol{x}$ or $\boldsymbol{y}$ and the distance between $u(x_1, y_1)$ and $v(x_2, y_2)$ is l. G is a layer graph.

Lemma 2. *A 4-cycle C with nodes p, q, r, s and edges pq, qr, rs, sp is not line rigid if $|pq| = |rs|$ and $|qr| = |sp|$. [1] (proof omitted)*

Based on the above characterization, we identify the following property for 5-cycle and 6-cycle point placement graph to be line rigid. Note that since we only consider point placement graphs, we will simply refer them as graphs.

Lemma 3. *A 5-cycle C with nodes p, q, r, s, t and edges pq, qr, rs, st, tp is line rigid if $|pq| \notin \{|rs|, |st|, ||rs| \pm |st||\}$ and $|st| \notin \{|qr|, ||pq| \pm |qr||\}$.*

Proof. A 5-cycle layer graph with nodes a, b, c, d, e must be plotted in one of the three ways in Figure 1 with $(a, b, c, d, e) = (p, q, r, s, t)$ or (t, p, q, r, s) or (s, t, p, q, r) or (r, s, t, p, q) or (q, r, s, t, p). In all cases, we show that C cannot be mapped to any of these three layer graphs because (a, b, c, d, e) cannot be equal to

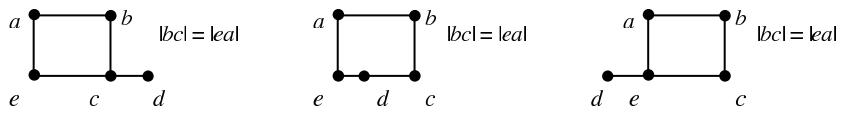

Fig. 1. Layer graphs for 5-cycle

i. (p, q, r, s, t) because $|pq| \neq ||rs| \pm |st||$
ii. (t, p, q, r, s) because $|pq| \neq |st|$
iii. (s, t, p, q, r) because $|st| \neq ||pq| \pm |qr||$
iv. (r, s, t, p, q) because $|st| \neq |qr|$
v. (q, r, s, t, p) because $|pq| \neq |rs|$

Since C cannot be mapped to these three graphs, C is not a layer graph. Therefore, it is line rigid (Theorem 1).

Lemma 4. *A 6-cycle C with nodes o, p, q, r, s, t and edges op, pq, qr, rs, st, to is line rigid if*

1. $|op| \notin \{|qr|, |rs|, |st|, ||qr| \pm |rs||, ||rs| \pm |st||, ||qr| \pm |st||, ||qr| \pm |rs| \pm |st||\}$
2. $|pq| \notin \{|rs|, |st|, ||rs| \pm |st||\}$
3. $|qr| \notin \{|st|, ||op| \pm |st||\}$
4. $|rs| \neq ||op| \pm |pq||$
5. $|st| \notin \{||op| \pm |pq||, ||pq| \pm |qr||, ||op| \pm |qr||, ||op| \pm |pq| \pm |qr||\}$
6. $||op| \pm |pq|| \neq ||rs| \pm |st||$

Proof. Similar to the proofs of Lemma 3, Lemma 4 can be proved by considering all possible mapping function from (op, pq, qr, rs, st, to) to 15 ways of plotting a 6-cycle layer graph in the xy-plane.

3 Upper Bound for Two Rounds

In this section, we show that $4/3n$ (more precisely, $4n/3 + O(\sqrt{n})$) pairwise distance queries are sufficient to determine the positions of n distinct points on a line using two rounds. The main idea is based on Lemma 1 and 3. From Lemma 3, we can always have a line rigid 5-cycle (p, q, r, s, t) no matter what the distance between node p and node t is. Since at each point, there are at most two edges with the same length, we can always find many links of 4 edges pq, qr, rs, st satisfying the conditions stated in Lemma 3 and form 5-cycles by making the last query on the distance bewteen node p and node t. Since each 5-cycle can determine the positions of 3 points (2 points of the 5-cycle should be fixed) with 4 queries, we might achieve the ratio $4/3$ when the number of points is large.

Algorithm 1. Let $n = 3b^2 + 17b + 31$ for some positive integer b. In the first round, we choose $3b^2 + 17b + 30$ queries represented by the tree in Figure 2.

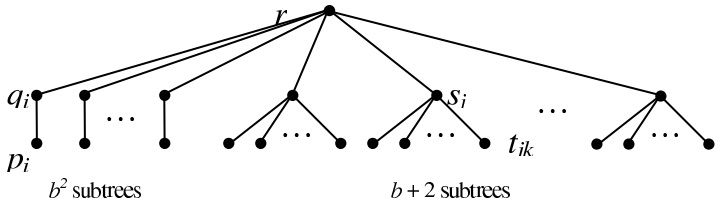

Fig. 2. First round queries for 2-round algorithm.

Let vertex r be the root of the tree. There are b^2 subtrees (2-links) with exactly one child $((p_iq_i, q_ir), i = 1, 2, \ldots, b^2)$ and and $b+2$ subtrees (b-trees) with roots $s_j, j = 1, 2, dots, b+2$, each with $b+14$ children $t_{jk}, k = 1, 2, \ldots, b+14$.

In the second round, for each 2-link (p_iq_i, q_ir), we find a distinct path (rs_j, s_jt_{jk}) in the $b+2$ b-trees $((b+2)(b+14) = b^2+16b+28$ possible paths) to form a 5-cycle which satisfies the requirements stated in Lemma 3 and query the length of $t_{jk}p_i$. Note that there are at most $16b+28$ paths that do not satisfy Lemma 3 because there are at most a) 2 b-trees with $|p_iq_i| = |rs_j|$ (Lemma 1) and b) for each tree rooted at s_j there are at most 2 edges s_jt_{jk} for each case

i. $|p_iq_i| = |s_jt_{jk}|$
ii. $|p_iq_i| = |rs_j| + |s_jt_{jk}|$
iii. $|p_iq_i| = |rs_j| - |s_jt_{jk}|$
iv. $|p_iq_i| = |s_jt_{jk}| - |rs_j|$
v. $|s_jt_{jk}| = |q_ir|$
vi. $|s_jt_{jk}| = |p_iq_i| + |q_ir|$
vii. $|s_jt_{jk}| = |p_iq_i| - |q_ir|$ or $|q_ir| - |p_iq_i|$

So we can always find a distinct path (rs_j, s_jt_{jk}) by matching with (p_iq_i, q_ir). For each of the unused $16b+28$ leaves t_{jk} in the b-trees, we query the distance between node t_{jk} and the root r (rt_{jk}). Last, we query the length of $s_js_{j+1}, j = 1, 2, \ldots, b+1$.

Theorem 2. *The graph constructed by Algorithm 1 is line rigid.*

Proof. The relative positions of the $b+2$ b-trees and the root r can be determined because they form $b+1$ connected triangles $(rs_j, rs_{j+1}, s_js_{j+1})$ with the root r [1]. The relative positions of $16b+28$ leaves t_{jk} are also fixed because each forms a triangle $(s_jt_{jk}, rs_j, rt_{jk})$ with the root r and its parent [1]. Each 5-cycle formed in the second round is line rigid (Lemma 3). Therefore the graph is line rigid.

Theorem 3. *Algorithm 1 uses* $4n/3 + O(\sqrt{n})$ *queries.*

Proof. In the first round, we have chosen $3b^2+17b+30$ queries. In the second round, we have chosen $b^2+(16b+28)+(b+1) = b^2+17b+29$ queries. Therefore,

the total number of queries is

$$\begin{aligned}
&(3b^2 + 17b + 30) + (b^2 + 17b + 29) \\
&= \frac{4}{3}(3b^2 + 17b + 30) + \left(\frac{34}{3}b + \frac{53}{3}\right) \\
&\leq \frac{4}{3}n + \frac{34}{3}\sqrt{n} \\
&= \frac{4}{3}n + O(\sqrt{n})
\end{aligned}$$

4 Upper Bound for Three Rounds

In this section, we show that $5n/4$ (more precisely, $5n/4 + O(\sqrt{n})$) pairwise distance queries are sufficient to determine the positions of n distinct points on a line using only three rounds. The idea is based on the same idea for the 2-round case. Instead of building 2-links, we try to build 3-links. However, there is a problem of using 3-link. Let (o_ip_i, p_iq_i, q_ir) be a particular 3-link (note that we assume r is the root in Algorithm 1). If $|o_ip_i| = |q_ir|$, then we may not be able to get a unique solution (Lemma 4). However, according to Lemma 1, for any edge with weight $|o_ip_i|$, there are at most two edges $q_\beta r, \beta = 1, 2, \ldots, b^2$, such that $|q_\beta r| = |o_ip_i|$. Instead of constructing b^2 3-links in a single round, we first construct $b^2 + 2$ 1-links and b^2 edges $o_ip_i, i = 1, 2, \ldots, b^2$. For each edge o_ip_i, we can always find a distinct 1-link $q_\beta r$ such that $|o_ip_i| \neq |q_\beta r|$. By querying the length of p_iq_β, we will get a 3-link $(o_ip_i, p_iq_\beta, q_\beta r)$ with $|o_ip_i| \neq |q_\beta r|$.

Algorithm 2. Let $n = 4b^2 + 87b + 773$ for some positive integer b. In the first round, we choose $2b^2 + 87b + 772$ queries represented by the tree in Figure 3. We also pair up the rest $2b^2$ nodes and query their pairwise distances $|o_ip_i|, i = 1, 2, \ldots, b^2$

Let vertex r be the root of the tree. There are $b^2 + 2$ children (1-links) are leave $(q_\beta r, \beta = 1, 2, \ldots, b^2 + 2)$ and $b + 10$ subtrees (b-trees) with roots $s_j, j = 1, 2, \ldots, b + 10$, each with $b + 76$ children $t_{jk}, k = 1, 2, \ldots, b + 76$.

In the second round, for each edge o_ip_i, we find a distinct 1-links $q_\beta r$ such that $|o_ip_i| \neq |q_\beta r|$ and query the length of p_iq_β to form a 3-link $(o_ip_i, p_iq_\beta, q_\beta r)$. For the rest two 1-link $q_\beta r$, we query the length of $q_\beta s_1$.

In the third round, for each 3-link $(o_ip_i, p_iq_\beta, q_\beta r)$, we find a distinct path (rs_j, s_jt_{jk}) in the $b+10$ b-trees ($(b+10)(b+76) = b^2+86b+760$ possible paths) to form a 6-cycle which satisfies the requirements stated in Lemma 4 and query the length of $t_{jk}o_i$. Note that there are at most $86b + 760$ paths that do not satisfy Lemma 4 (Lemma 1). So we can always find a distinct path (rs_j, s_jt_{jk}) for $(o_ip_i, p_iq_\beta, q_\beta r)$. For each of the unused $86b+760$ leaves t_{jk} in the b-trees, we query the distance between node t_{jk} and the root r (rt_{jk}). Last, we query the length of $s_js_{j+1}, j = 1, 2, \ldots, b + 1$.

Theorem 4. *The graph constructed by Algorithm 2 is line rigid.*

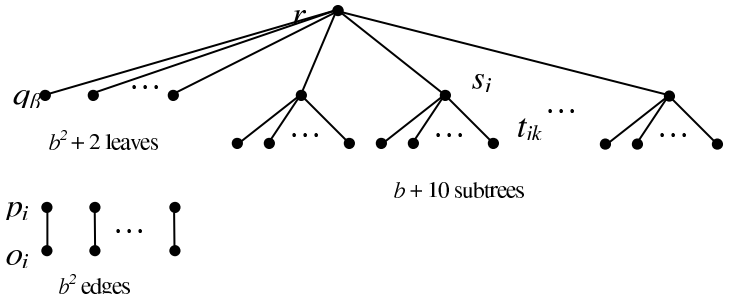

Fig. 3. First round queries for three round algorithm.

Proof. The relative positions of the $b+10$ b-trees and the root r can be determined because they form $b+1$ connected triangles $(rs_j, rs_{j+1}, s_js_{j+1})$ with the root r [1]. The two unused 1-links rq_β are line rigid because they form two triangles $(rs_1, rq_\beta, q_\beta s_1)$ with the root r and node s_1. The relative positions of $10b+76$ leaves t_{jk} are also fixed because each forms a triangle $(s_jt_{jk}, rs_j, rt_{jk})$ with the root r and its parent [1]. Each 6-cycle formed in the third round is line rigid (Lemma 4). Therefore the graph is line rigid.

Theorem 5. *Algorithm 2 uses* $5n/4 + O(\sqrt{n})$ *queries.*

Proof. In the first round, we have chosen $(2b^2+87b+772)+b^2 = 3b^2+87b+772$ queries. In the second round, we have chosen b^2+2 queries. In the third round, we have chosen $b^2+86b+760+(b+9) = b^2+87b+760$. Therefore, the total number of queries is

$$\begin{aligned}
&(3b^2+87b+772)+(b^2+2)+(b^2+87b+760)\\
&= \frac{5}{4}(4b^2+87b+773)+\left(\frac{261}{4}b+\frac{2271}{4}\right)\\
&\le \frac{5}{4}n+\frac{261}{8}\sqrt{n}\\
&= \frac{5}{4}n+O(\sqrt{n})
\end{aligned}$$

5 Lower Bound for Two Rounds

In this section, we show that any 2-round adaptive algorithm for solving the 1-dimensional point placement problem requires at least $17n/16$ queries.

Let V be the set of points. Suppose that a particular 2-round algorithm can uniquely determine the relative positions of a set V of n nodes in two rounds. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_1 \cup E_2)$ be the graphs of queried node pairs (or edge), where $E_i (i = 1, 2)$ contains all edges whose lengths have been measured in round i.

Let us consider round 1 first. The algorithm has queried the edges in G_1. The adversary will report the length of the edges based on the following strategy:

1. For nodes of degree at least 3, the adversary fixed the layout of all these nodes and answers the queries of E_1 related to these nodes accordingly.
2. For nodes of degree 2, we consider the maximal paths $(p_1, p_2, p_3, \ldots, p_k)$ formed by these nodes such that the number of degree 2 nodes is at least 3. Let p_0 and p_{k+1} be the nodes of degree not equal to 2 which are adjacent to p_1 and p_k, respectively. The adversary fixed the layout of nodes p_i if $i = 0 (\text{mod } 3)$.

Denote (p_i, p_{i+1}) as special node pair if $i = 1 (\text{mod } 3)$. For each special node pair, say (p_i, p_{i+1}), suppose p_i is adjacent to p_{i-1} and p_{i+1} is adjacent to p_{i+2}. Note that the positions of p_{i-1} and p_{i+1} are fixed. The adversary sets $|p_{i-1}p_i| = |p_{i+1}p_{i+2}|$ and $|p_ip_{i+1}| = |p_{i-1}p_{i+1}|$. By setting the length in this way, the positions of p_i and p_{i+1} are ambiguous. (More precisely, p_i and p_{i+1} have two possible layouts: (1) p_i to the left of p_{i-1} and p_{i+1} to the left of p_{i+2} or (2) p_i to the right of p_{i-1} and p_{i+1} to the right of p_{i+2}.) Then, we consider G_2 (that is, round 2). We have the following two properties:

Lemma 5. *In G_2, for each special node pair (p_i, p_{i+1}), there exists at least one edge in E_2 connecting to either p_i or p_{i+1}.*

Proof. Suppose that both p_i and p_{i+1} do not attach to any edges in E_2. Then, the ambiguity we introduced in round 1 cannot be resolved.

Lemma 6. *For any maximal path P of degree 2 in G_2, the number of nodes in the maximal path is at most 4.*

Proof. By construction, no three consecutive edges in the maximal path can be in E_1. If there exist three consecutive edges in E_1, two of the nodes will form a special node pair.

Suppose that the number of degree-2 nodes is 5, let the nodes be p_0, p_1, p_2, p_3, p_4, p_5, p_6 where p_0 and p_6 are heavy nodes adjacent to the length-5 maximal path. Suppose that $|p_0p_1| = w_1$, $|p_1p_2| = w_2$, $|p_2p_3| = w_3$, $|p_3p_4| = w_4$, $|p_4p_5| = w_5$, and $|p_5p_6| = w_6$. There are the following combinations of E_1 and E_2 edges for $(w_1, w_2, w_3, w_4, w_5, w_6)$.

1. $E_2, E_1, E_1, E_2, E_1, E_1$
2. $E_1, E_2, E_1, E_2, E_1, E_1$
3. $E_1, E_2, E_1, E_1, E_2, E_1$
4. $E_1, E_1, E_2, E_2, E_1, E_1$
5. $E_1, E_1, E_2, E_1, E_2, E_1$
6. $E_1, E_1, E_2, E_1, E_1, E_2$

For each combination, we can set the length of the edge of E_2 to make it ambiguous. For example, for combination 1, we can set $w_1 + w_2 + w_3 = w_5 + w_6$ and $w_4 = |p_6p_0|$; then $(p_0, p_1, p_2, p_3, p_4, p_5)$ can be degenerated to a 4-cycle which is not line rigid (Lemma 2) and becomes ambiguous.

Below, we try to analyze the average number of edges per node. Nodes of degree at least 3 or belong to special node pair are denoted as heavy nodes; otherwise, they are light nodes. We split every edge into two fractional edges owned by the two incident nodes. For an edge joining two heavy nodes or two light nodes, each incident node owns $1/2$ edge count. For an edge joining a light and a heavy node, the light node owns $1/2 + g$ edge count and the heavy node owns $1/2 - g$ edge count. (We will specify g below.) The nodes in V can be divided into three types:

a. Special node pairs
b. Degree 3 normal nodes
c. Maximal path formed by degree-2 normal nodes connecting using edges in E_1 and E_2.

For type (1), by Lemma 5, each special node pair (p_1, p_2) has at least one edge in E_2 connecting to either p_1 or p_2 (says p_1). Then, the edge count of p_1 is at least $(1/2 + 2(1/2 - g) = 1.5 - 2g)$ while the edge count of p_2 is at least $(1/2 + 1/2 - g = 1 - g)$. So, the average edge count of p_1 and $p_2 = (2.5 - 3g)/2 = (5 - 6g)/4$.

For type (2), each node of degree at least 3, the edge count of the node is at least $3(1/2 - g) = 3/2 - 3g$.

For type (c), by Lemma 6, each maximal path is of length $k (k \leq 4)$, the total edge count of all nodes in the path $= 2(1/2 + g) + (k - 1) = k + 2g$. Hence, the average edge count is $1 + 2g/k$. (For the worst case, we set $k = 4$ and the average degree is $1 + g/2$.)

In total, the average edge count of each node in G_2 is at least $max\{(5 - 6g)/4, 3/2 - 3g, 1 + g/2\}$. By setting $g = 1/8$, the total edge count of all n nodes in G_2 is at least $17n/16$. Hence we have the following theorem.

Theorem 6. *Any deterministic 2-round adaptive algorithm for solving the 1-dimensional point placement problem requires at least* $17n/16$ *queries.*

6 Conclusions

There are quite a number of related open problems. For examples, whether it is possible to have a strategy that matches the lower bound or whether a better lower bound exists for the case of two rounds. For unlimited number of rounds, the only lower bound we have is the trivial bound of n. It is challenging to obtain non-trivial lower bounds for the case of r rounds. Of course, it is also challenging to obtain better strategies for all cases including the non-adaptive version in which the upper bound $(8n/5)$ and the lower bound $(4n/3)$ are not closed yet. Another direction is to design randomized algorithms to solve the problem (e.g. [2]).

References

1. Damaschke, P.: Point Placement on the Line by Distance Data. Discrete Applied Mathematics. **127** (2003) 53–62
2. Damaschke, P.: Randomized vs. Deterministic Distance Query Strategies for Point Location on the Line. Discrete Applied Mathematics. **154** (2006) 478–484
3. Mumey, B.: Probe Location in the Presence of Errors: A Problem from DNA Mapping. Discrete Applied Mathematics. **104** (2000) 187–201
4. Redstone, J., Ruzzo, W.L.: Algorithms for a Simple Point Placement Problem. Proceedings of the Fourth CIAC, Lecture Notes in Computer Science, **1767** (2000), Springer, Berlin, 32–43
5. Trask, B.J., Allen, S., Massa, H., Fertitta, A., Sachs, R., Engh, G., and Wu, M.: Studies of Metaphase and Interphase Chromosomes using Fluorescence in situ Hybridication. Cold Spring Harbor Symposia on Quantitative Biology, **LVIII** (1993) 767–775
6. Waterman, M.S.: Introduction to Computational Biology. Maps, Sequences and Genomes. Chapman and Hall, London (1995)