Approximations for Covering in a Streaming Setting, Covering with Orthants

Berwin Gan
Computer Science, NYUAD
wqg203@nyu.edu

Advised by: Saurabh Ray

ABSTRACT

In this paper, we looked at two problems of covering in different settings. Firstly,we consider the problem of the unit disc covering and its variations for a streaming setting, in which, given a set of streaming points, the goal is to approximate the minimum number of unit discs needed to cover all incoming points without storing information on all points. We first give a 2-approximations algorithm for the thin annulus variation on the UDC problem. After that, we give an algorithm with a factor of 1.5 for the interval covering variation when the optimal number of covering is exactly two. Following that, we give a construction to show a lower bound of 1.5 for the square variation of the UDC problem. Lastly, we give a two approximation algorithm for the half-space variation when there exist a hitting set of size 1.

We then consider the problem of covering with orthants. We first show that any case can be reduced to the general case of a monotonic subset then perform further analysis on the general case. Our first construction show a way to create arbitrarily long single connections by using two intermediary points. We then show that clique creation using a single type of orthant is equal to the planar graph and is impossible past 4 points. Lastly, we show that restricting the number of intermediary points to 0 or 1 to create cliques of arbitrarily large size is impossible.

KEYWORDS

single pass, streaming algorithm, orthant, approximation algorithm, half-spaces, computational geometry

This report is submitted to NYUAD's capstone repository in fulfillment of NYUAD's Computer Science major graduation requirements.

جامعة نيويورك أبوظبي
NYU ABU DHABI

Capstone Project in Computer Science 2, Spring 2021, Abu Dhabi, UAE

Reference Format:
Berwin Gan. 2021. Approximations for Covering in a Streaming Setting, Covering with Orthants. In *NYUAD Capstone Project in Computer Science 2 Reports, Spring 2021, Abu Dhabi, UAE.* 12 pages.

1 INTRODUCTION

Approximation schemes are algorithms that approximate an answers when finding the exact solution would be undesirable. Computation and memory limitation are the main reason approximation schemes are favour over an exact algorithm. These approximation algorithms are used to approximate solutions to optimizations problem. In practical terms, these optimization problems could translate into operation research such as finding the optimal area to place critical infrastructure such as hospitals in a city. The static view aside, the place where approximation schemes truly shine is in a streaming setting. The idea behind this combination is to find an approximate solution without storing all incoming information which in a streaming setting will be dynamic and random.

Before applications can be made using approximation algorithms, we must first understand the bound of these algorithms, namely where the limits of these algorithms are. As approximations schemes do not give an exact solution by nature, the most important thing to take away from any approximation schemes is how much bigger the solution given is compared to the actual solution. The two terms of importance here are the upper bound and the lower bound. The upper bound of an approximation scheme tell us how much bigger the approximated solution is when compared to the actual solution. For example, if the upper bound for a particular algorithm is 4, it means that the ratio of the approximated solution to the actual solution will be 4 in the worst case. Research working on the upper bound of an approximation scheme would try to come up algorithms that approximate more accurately in the worst case scenario. The lower bound on the other hand tells us the lowest ratio of approximation to actual solution we can theoretically achieve. What the lower bound tells us is that no mater what clever

schemes are made, no approximation to actual solution ratio can be less the the lower bound if some information is loss.

The general question posed would be "What is the minimum number of unit disc it would take to cover all the points?" given a stream of points onto a plane. Hence, this paper tackles the upper and lower bound of this question for the UDC in 2D and its variations.

For the second problem, instead of using closed shapes such as discs and squares, we looked at using orthants to cover points in a classical non-streaming setting. We work towards showing that creating cliques of arbitrary large size using orthants is impossible. If a proof for showing that a clique of arbitrarily large size can be found, then the Local Search algorithm by [6] would be considered as optimal due to the absence of a large minor.

2 RELATED WORK

The three papers which most work was built on for the problem of covering in a streaming setting are *Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI* [3] by Dorit S. Hochbaum and Wolfgang Maass, *Approximation Schemes for Covering and Packing in the Streaming Model* [4] by Christopher Liaw, Paul Liu and Robert Reiss and *Interval Selection in the Streaming Model* [1] by Sergio Cabello and Pablo Pérez-Lantero.

The first paper deals with square packing, covering with discs and covering with squares or rectangles. For the purpose of this paper, the result in focus would be the shifting strategy introduced to produce a polynomial approximation scheme for covering. The general problem tackled in the paper can be summarised as such: "Given some points on a plane, what is the minimum number of disc required to cover all the points?". Deceptively simple. The greedy brute force approach to this problem would be to simply test every configuration of discs on the points. However, practically, this breaks down as the number of operations required in order to enumerate through all possible configurations increases exponentially as the number of points increases. Hence the solution being presented in this paper is a way to cut down the exponential pie into a polynomial one. The shifting strategy uses a divide-and-conquer approach by first dividing the plane into strips of width $\mathcal{D}$ that is bigger than 1. After that, a certain shifting size l that is smaller that $\mathcal{D}$ is chosen. For any particular set of strips, a shift would be defined as the shifting of the boundaries of the strips to the right by l amount. By definition it would only take $\mathcal{D}/l$ shifts for the original configuration to match the starting configuration as we are set on an infinite plane. Now withe these strips, we can apply a greedy algorithm to each of these strips separately finding the minimum number of unit discs required to cover all points in the strip. By adding up the output of all strips we would get the minimum number of unit discs to cover all points in this particular configuration. We then shift the strips to the right and repeat. After each shift, we compare and keep the minimum number of unit discs. By going through each shift we will get the minimum number of unit discs required. An important point to note about this algorithm is that the accuracy of how close it gets to the actual solution is dependent on how small we make each l shift to be. With an approximation of $(1 + \frac{1}{l})$, as we make the steps smaller and smaller, the algorithm would become more and more accurate but conversely it would take more and more time.

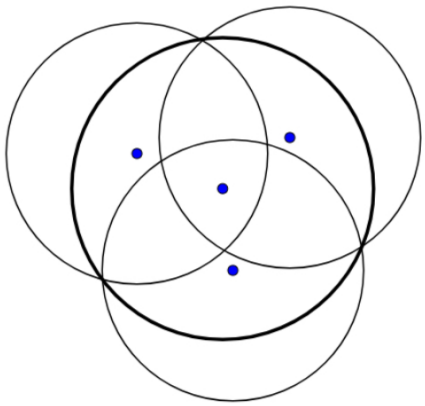

Figure 1: A Covering Of A Radius $2/\sqrt{3}$ by 3 Discs Of Radius 1

The second paper built upon the first using the shifting strategy as a based and showed an an approximation algorithm for a streaming setting on a plane that have have an upper bound of 3 and way of proving a lower bound of 2. The approximation algorithm is as follows, divide a plane up into squares with length $\delta < \frac{2/\sqrt{3}-1}{\sqrt{2}}$. For each incoming point onto the plane, only mark the first point for all points falling in the same square. At the end, use the points recorded and the shifting strategy [3] to generate the least number of unit disc required to cover all recorded point. Because the generated disc only covered the recorded points and not the entire square the point sits in, it could have missed some points because the algorithm only recorded the first point in any square. To remedy for the situation. Three copy of the unit disc can be used to expand the coverage to cover all possible points. Doing so give an upper bound of 3. This is because it could very well be possible that that the optimal solution be the exact solution given when the unit disc were generated by the first recorded points.

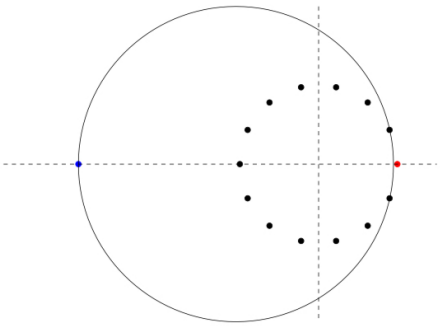

Figure 2: The Lower Bound Construction For UDC In 2D.

Liaw, Liu and Reiss used communication theory to show a lower bound of 2. By streaming points in the shape of a circle, queries can be make at $1+\epsilon$ distance away from any point. The algorithm is forced to keep every point because if a particular point is not kept, the query would return 1 instead of 2. This proof gives us a lower bound of 2, meaning that there could not exist an algorithm in which the approximation ratio to the optimal solution is less than 2.

The third paper on the interval selection looks at several questions. But the focus for this paper is the result of 2-approximation for a streaming algorithm on a line. Which gives an upper bound of 2 for a streaming setting on a line.

The two papers that was the backdrop for the second problem of covering with orthants are *Improved Results on Geometric Hitting Set Problems* [5] by Nabil H. Mustafa and Saurabh Ray and *Packing and Covering with Non-Piercing Regions* [6] by Sathish Govindarajan, Rajiv Raman, Saurabh Ray and Aniket Basu Roy. The general question the two papers looked at was given a bunch of orthants and points in 3D, how can we obtain the minimum subset of orthant that covers all points. The Local Search algorithm presented works as follows:

(1) We first start off with all given orthants
(2) We then perform local searches and discard any orthants whose points are already covered by another orthant.
(3) Then we perform a search to see if it is possible to swap out 3 orthants for 2 and do so if it is possible for all orthants remaining.
(4) The remaining orthants is the optimal minimum subset of orthants needed to cover all points.

The ambiguity on the local search algorithms is weather or not it gives the optimal solution. The lack of a proof of weather or not there exist a sub-linear size balanced separator can be solved by finding out if there exist a large minor within the graph form by the Local Search algorithm.

A view of the problem is to consider the set of points and orthants as a graph with the points as vertices and that there is an edge between two points if and only if there is an orthant that contains only those two points. With this graph, by showing that no large minor [2] exists, we can show that the Local Search algorithms gives the optimal solution.

3 STREAMING SETTING

Our attempts on the general case with any number of covering discs in the solutions was unfruitful but an interesting insight can be seen when we were playing with the dual problem where instead of covering streamed points, we wanted to hit streamed discs with points. No matter what we tried, there does not seem to be a way to store geometric information without storing information on every streamed discs. For example, when looking at the intersection between discs, the number of edges of the intersection increases linearly to the number of discs, saving no memory. It seems to be a unique property to discs which have only one edges.

Likewise, even when we limit our plane to a $1+\epsilon$ discs, where only 3 unit discs are required to cover the entire $1+\epsilon$, we are still unable to differentiate between when 1 or 2 or 3 discs are needed without storing all points.

3.1 Thin Annulus: 2-Approximation Algorithm

A variation on the local problem we looked at is through the thin annulus variation. The inspiration for this came from the fact that in the local problem, points falling near the middle of the $1+\epsilon$ circle play less of a role in where the solution would be than compared to points falling near the edge. In the case where only one unit disc is needed, the points falling on the edge would inform any algorithm on where the placement of the unit disc would be. Hence the annulus variation is as such, given a unit disc within a $1+\epsilon$ disc with both centered at the same point, the streaming points only fall within the spaces of the two disc. Would it be possible to create a 2-approximation algorithm to cover the points with unit discs ? Note that ϵ needs to be small enough so that we are able to cover the $1+\epsilon$ disc with 3 unit discs.

The statement can be re-framed into two smaller statement: 1) Show that the points cannot be covered by 1 unit disc or 2) Show that the points can be covered by 2 unit disc. In this variation, we prove the 2-approximation using the first statement.

The algorithm to check if 1 unit disc is not enough:

(1) Keep the first two points that fall in the annulus and designate the left-end point and right-end point from the perspective of the middle of the circle.

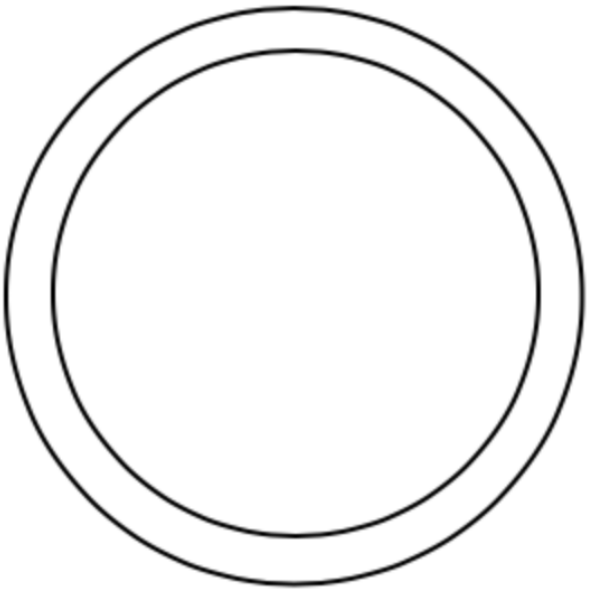

Figure 3: Unit Disc Within A Unit 1+ϵ Disc

(2) For each incoming point after that, check if it is within the bound of the left-end and right-end points.
(3) If the incoming points are within the bounds, the points can be discarded. If the points are outside the bounds, the points will replace either the left-end or right-end point depending on which is closer.
(4) If the angle between the left-end point, the mid point and the right-end point exceeds $180 - \delta$ degrees in any moment of the stream, 1 unit disc would not be enough to cover the points.

We can check this logic by taking ϵ to be 0 and let the points stream onto the edge. In this case, 1 unit disc would be enough to cover all possible points. But the moment we increase ϵ by a small amount, we are able to cover almost half of the ring by moving a unit disc centered at the same point as the unit disc a small amount in any direction. As this is for a small ϵ, as we increase the ϵ, the δ needed for step 4 of the algorithm would also increase for the check of one unit disc. That said, when ϵ tends towards 0, this exact algorithm becomes more and more true.

Using the algorithm we can tell exactly when one disc is not enough cover all disc in which we can return a solution of size 3, else we can return a solution of size 2. When the angle is less than $180 - \delta$, the only two possible solution is 1 or 2 disc. If we return a solution of 2 when the angle is less than $180 - \delta$, we give an exact approximation if the actual solution is 2 discs; however, if the actual solution is 1 disc, we give an approximation of factor 2. Similarly, if the angle is greater than $180 - \delta$ and we return a solution of size 3, we would give an exact solution if the actual solution is 3 discs and a approximation of factor 1.5 if the actual solution is 2 discs. Hence, the algorithm give us a 2-approximation in the worst case.

As this algorithm only keep the information on two points at anytime, the memory needed is constant.

3.2 Interval Covering: 1.5-Approximation Algorithm

The question is, given a stream of points, what is the minimum number of unit intervals required to cover all points? This is the UDC problem but in 1D. Cabello and Pérez-Lantero showed in their paper a 2-approximation upper bound for this problem while simultaneously showing a 1.5 lower bound for the UDC problem in all dimensions. Hence in the case for 1D, an approximation algorithm factor could lie somewhere between 1.5 and 2. Although we did not give an approximation algorithm for the general case, we observe that the greedy algorithm gives us an approximation with a factor of 1.5 when there exist a covering set of size 2.

The greedy algorithm uses the dual problem of the interval covering problem where instead of streaming points, we stream in unit intervals with the goal of hitting them with points. Hence, the greedy algorithm obtain a hitting set of at most size 3 if there exist a hitting set of size 2.

The greedy algorithm:

(1) Keep the first interval streamed in.
(2) For all incoming intervals after that, check the incoming intervals against the recorded intervals for any intersections. If there exist intersections, record only the intersections. Else, record the entire incoming interval.
(3) Place a point in all recorded intervals.

The points placed in all recorded intervals will hit all streamed intervals. The greedy algorithm keep a record of at most 3 intervals and hence uses constant memory.

For all possible variation where the hitting set is 2 points, this algorithm would output 3 intervals or less where if a point is placed in each interval would cover all intervals that were streamed in. This gives us a 1.5-approximation.

3.3 Lower Bound for Square Variation

One of the variations we looked at was the square variation where given streamed points, the goal was to find the minimum number of unit squares to cover all points.

We give a construction to show a lower bound of 1.5 when covering with unit squares in a streaming setting via reduction to the INDEX problem in communication complexity.

We first stream the test points onto a small diagonal line. In order to test any points, we can stream in two points, one at $1 + \epsilon$ to the left of the target point and one at $1 + \epsilon$ bellow

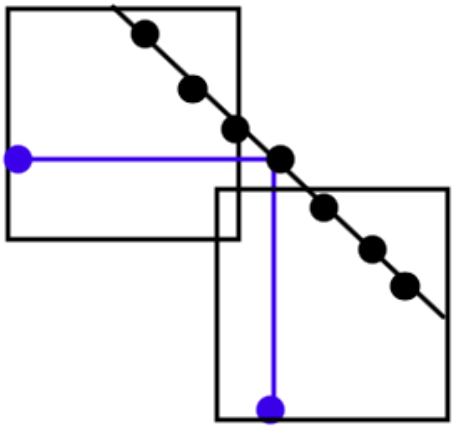

Figure 4: The Lower Bound Construction For The Square Variation Of UDC In 2D.

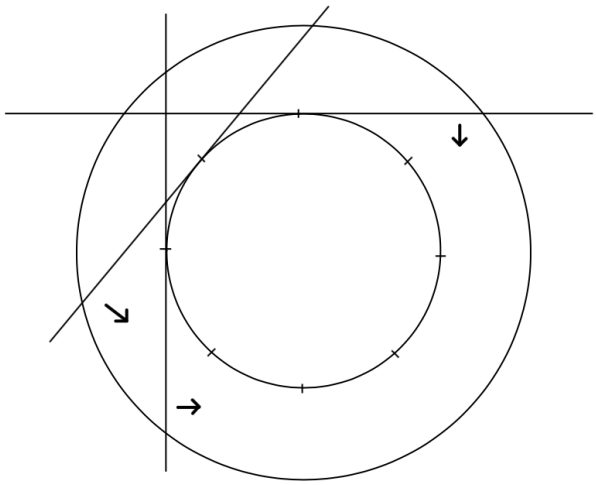

Figure 5: Lower Bound Construction With 3 Test Half-Spaces

the target point. After streaming the two points, we then query to find the number of unit squares needed to cover all points. If the target point is remembered, three squares would be needed, else only two is needed. This gives us a lower bound of 1.5.

We note that the test points can be streamed onto an arbitrarily small line which allows the rest of the test points which are not the target points to fit in two squares if the target point is not remembered.

3.4 Half-spaces Variation

A variation of the local problem we looked at was that of streaming half-spaces onto a unit disc. The question is, given a streaming number of half-spaces onto a unit disc, what is the minimum number of points to hit all streamed half-spaces. We first use communication complexity to show a lower bound of 2 for this problem before giving an algorithm to obtain a hitting set of size 2 if there exist a hitting set of size 1 while storing a sub-linear amount of information. Hence, this is a algorithm for the specific case where the actual solution consists of only one point.

A half-space is a line with a direction. Each unique line can have only two direction. A half-space is hit by a point if there is a point in the direction of the line within the disc. A test to check if the point hits the line is to move the line in its direction until the line is out of bound of the disc. If at any point of time the line touches the point, the line is hit by the point.

To show a lower bound, we will reduce from the INDEX problem in communication complexity. We first stream the test half-spaces onto a $1 - \epsilon$ disc with all half-spaces tangent to the $1 - \epsilon$ disc and facing the center point of the disc. In order to test any half-space, we can stream another half-space onto the target half-space with an opposite direction. If the target half-space is remembered, the query would return that 2 points is needed to cover all half-spaces. If the target half-space is forgotten, the query would return that 1 point is needed. This gives us a lower bound of 2. It is important to note that we need to either disallow placing points directly on half-spaces or place the query half-space a small distance behind the target half-space in order for the construction to hold.

Since the lower bound for this variation is 2, any algorithm that does not keep the information of all streamed half-spaces cannot give an approximation more accurate than a factor of 2. Below we give an algorithm that give a 2-approximation for a hitting set of size 1.

The claim we make is that any situation where the streams of half-spaces can be hit by one point, there are two diametric points on the edge of the disc that can hit all half-spaces. We can find these two diametric points given a point that hits all half-spaces by first creating a line connecting the point to the center point. We then extend the line both ways to intersect with the unit disc. By placing two points where the line and unit disc intersect, we create two diametric points that will hit all the half-spaces that are hit by the one point.

The first thing to consider is that there are two types of half-spaces, half-spaces that hit the center point of the unit

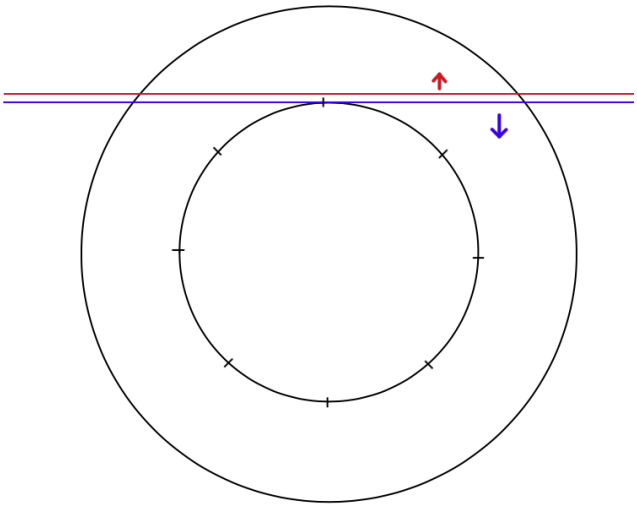

Figure 6: Target Half-Space(blue), Query Half-Space(red)

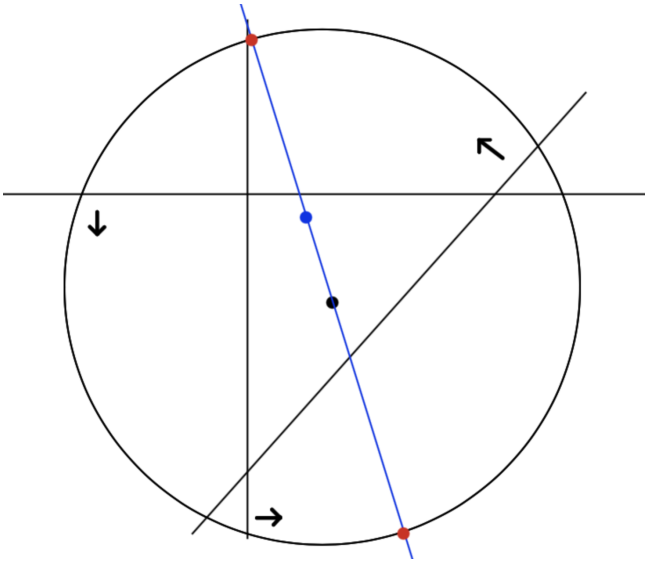

Figure 7: Center Point(black), Point That Hits Every Half-Space(blue), Two Diametric Points That Hits Every Half-Space(red)

disc and half-spaces that do not hit the center point of the unit disc.

We can observe that any half-space that hits the center point of the unit disc will cover at least half the area of the unit disc. As a result, no matter how we place our two diametric points along the edge of the disc, the half-space will hit at least one of the two diametric points. Hence, when

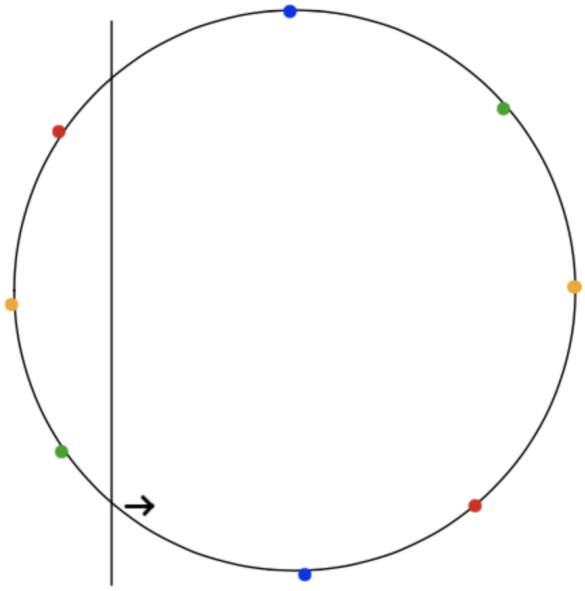

Figure 8: A Half-Space That Hits the Center Point

dealing with these type of half-spaces, we can ignore them as we intend to use two diametric point in the final solution.

For half-spaces that do not hit the center point of the unit disc, we observe that all such half-spaces must share an intersection arc. If we divide the unit disc into a major and minor arc using the half-space, in order for a half-space to not hit the center point, it must be facing the minor arc. Furthermore, for there to exist a hitting set of size 1, the minor arcs of all half-spaces that do not hit the center point must overlap and have an intersection. If there exist two half-spaces that does not hit the center point and have minor arcs that do not intersect, there cannot exist a hitting set of size 1 that will hit both half-spaces.

With that in mind, we present our algorithms for 2-approximation when the solutions only requires 1 point:

(1) If the half-space streamed in hits the center point, discard the half-space. Else, divide the unit disc into a minor and major arc using the intersection of the half-space and the circle as the end-points.
(2) If there are no previously stored arc, repeat from step 1. Else, record the intersection the the minor arc with previously stored arc as a new arc and keep this new arc.
(3) Repeat step 1 and 2 until all half-spaces are streamed in.
(4) If there is an intersection arc, reflect the intersection arc across the center point and place two points opposing to each other with one on each arc. Else, arbitrarily choose two opposing points on the disc edge.

Note that this 2-approximation algorithm even works when all streamed half-spaces hits the center point in which case any two diametric points on the disc edge would hit all half-spaces.

4 COVERING WITH ORTHANT

An orthant in n-dimensions is the intersection of n mutually orthogonal half-spaces. In the case for 2D, orthants can be thought of four different type of L's that divide a plane into four different quadrant. For the purpose of the 2D plane that we are working on, we will name the orthant by the directions their half-spaces points.

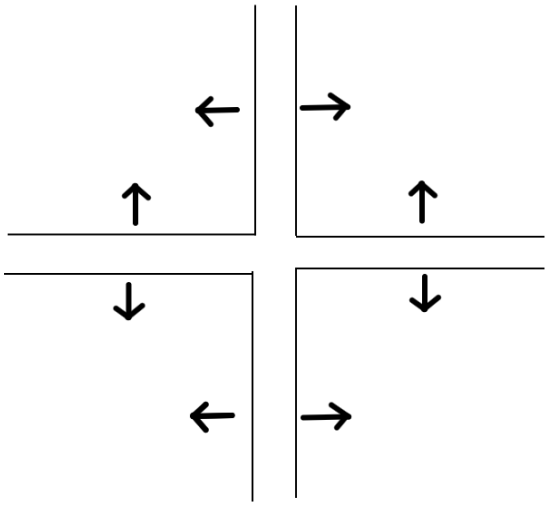

Figure 9: 4 Types Of Orthants For 2D

The connection rule is as such, there are points on the plane, each with their unique x-coordinate, y-coordinate and priority, when an orthant is placed, the two points covered by the orthant with the highest priority will form a connection. A point is covered by an orthant if its hit by either of its half-spaces.

For example, given a down right orthant, a point is covered if it is to the right of the vertical leg of the orthant and below the horizontal leg of the orthant. To simplify explanations, any points which are on the legs of an orthant is considered to be covered by the orthant. An orthant can be defined by its type and its location which is the coordinates its two half-spaces meet.

In the case for Figure 10, when the orthant was placed, the two points with the highest priority was the point with priority 10 and the point with priority 422, hence a connection is formed between them.

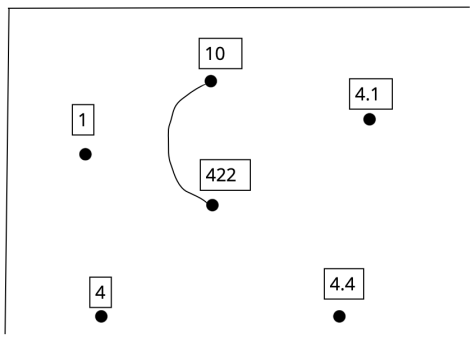

Figure 10: Example Use Of An Orthant

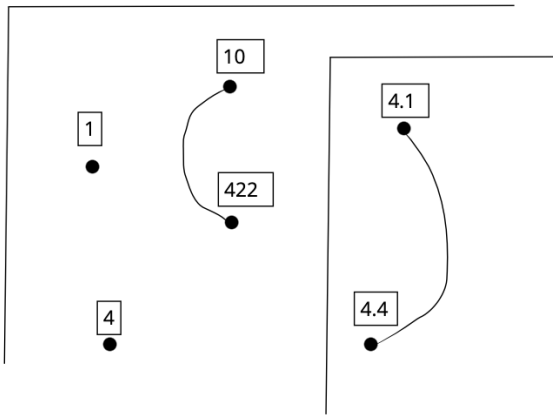

Figure 11: Example Use Of Two Orthants

However, if we would like to connect the point with priority 4.1 to point with priority 4.4, we could place another orthant of the same type to the right of the points with priority 10 and 442. In which case, the only points contained in the new orthant is the points with priority 4.2 and 4.4 respectively. This by defecto makes them the two points with the largest priority under the new orthant.

Two points are considered connected if there exist a path between the two points be it directly or through another point. Given three points in Figure 12, we can connect the point with priority 1 to the point with priority 3 without going through the point with priority 2 by creating a new point (in blue) with priority 1.1, connecting the point with priority 1 to it with an orthant and connecting it to the point with priority 3 with another separate orthant.

It is noted that when creating new points, the rule of forming a connection between two points with the highest

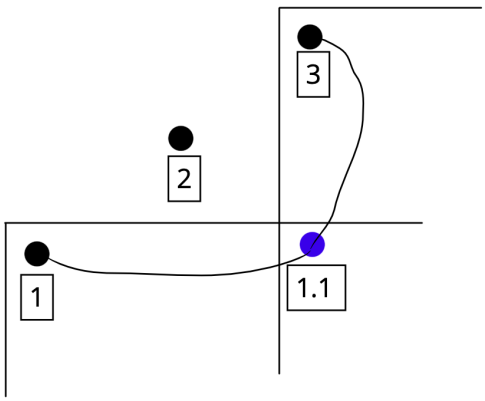

Figure 12: Adding New Points

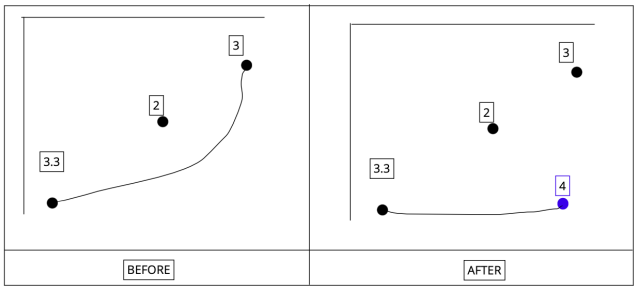

Figure 13: When Adding A New Point

priority still holds. Hence, if an orthant already connects two points and a new point is introduced and is contained by the orthant and have the highest or second highest priority among the points contained, the connections would change to include the new point.

Lastly, although there are four types of orthants in 2D, the construction available to these four orthants are equivalent to just using just two types, either the down right and up left orthants or up right and down left orthants. For future observations, we would work with the down right and up left orthants for simplicity. What we would like to show is that given a large number of points, it is not possible to create a clique, where every point is connected to every other point.

4.1 General Case

We first looked into the properties of a general case where the points are lined up in a diagonal fashion that stretches out to infinity where all points are above by a distance of 1 and to the right at a distance of 1 from the previous points and have increasing priorities.

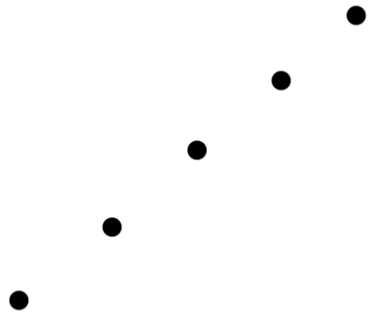

Figure 14: General Case Without Priorities

The general case can be obtained from any random set of points and priorities by noting that given n points with random but unique priorities, there exist a monotonic subset of size $\sqrt[3]{n}$. Hence, the general case is a generalization of an arbitrary set of points.

4.2 Orthant Steps

A local problem we first tackled was asking if there is a limit when performing a uniform single connection with one type of orthant. This means that given the general case, can we always connect a point to its d^{th} neighbour on the right for all points given some d. For the sake of clarity, the orthant we used points down and right but this will work for any of the four types. Using the general case as a backdrop, let the priorities of the points as they ascend the diagonal increase linearly with the lowest and left most point having a priority of 1.

The question is then, using one type of orthant is it possible to connect points with priority n to points with priority $n + d$ for an arbitrary d.

The solution is trivial when $d = 1$, as we can connect all points directly to its neighbour without any intermediary points as seen in Figure 16.

However, the construction becomes less clear as d increases in size. To the best of our knowledge, there was not a uniform way of showing such types of constructions and weather or not there is a limit to d by which after the construction would not be possible. Hence, we show a general purpose construction method for connecting any point with priority n on the diagonal to another point $n+d$ for any $d > 1$.

The constructions dubbed the 'orthant steps' is as follows:

(1) Create a new intermediary point at a distance of $2n-1$ below the point with priority n with a priority of $1 + 0.1^n$.

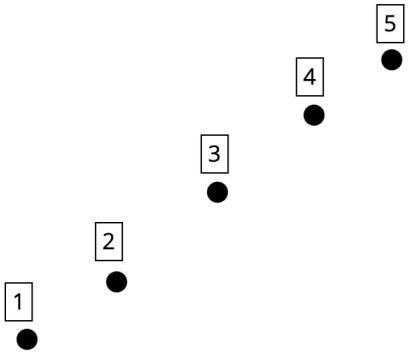

Figure 15: General Case With Priority

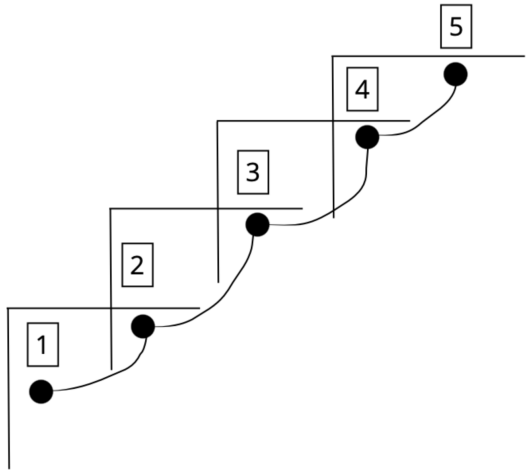

Figure 16: $d = 1$

(2) Connect the point with priority n and this new point with an orthant placed on the point with priority n.
(3) Create another point at a distance of $2n - 1$ to the left of the point with priority $1 + 0.1^n$ with a priority of $n + d + 0.1$.
(4) Connect the point with priority $n + d + 0.1$ to the point with priority $1 + 0.1^n$ with an orthant placed on the point with priority $n + d + 1$
(5) Place an orthant $2n - 1 + d$ distance above the point with priority $n + d + 0.1$ to connect it to the point with priority $n + d$

Using this construction for all points, it is clear that there is no limit when performing a uniform single connection

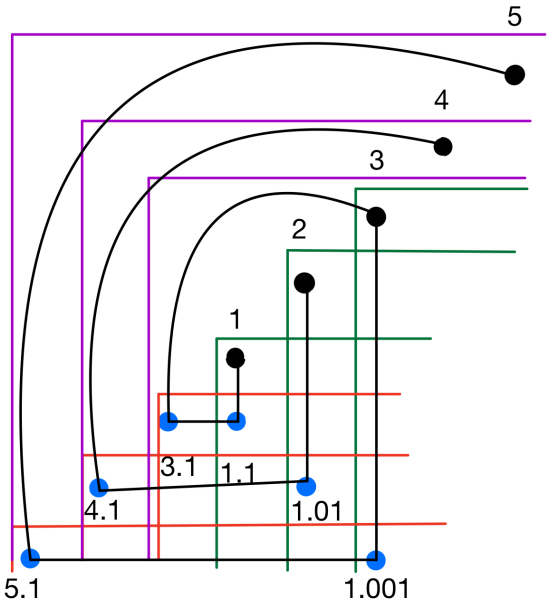

Figure 17: Orthant Steps With $d = 2$ For The First Three Points

for all points on the general case. There will never be a clash of priorities as the first intermediary points will have descending priorities while the second intermediary points will have ascending priorities resulting in the intermediary points having strictly increasing priorities going from left to right.

4.3 Limits of Single Type Orthant

We show that using a single type of orthant to form the connections is a equal to a planar graph.

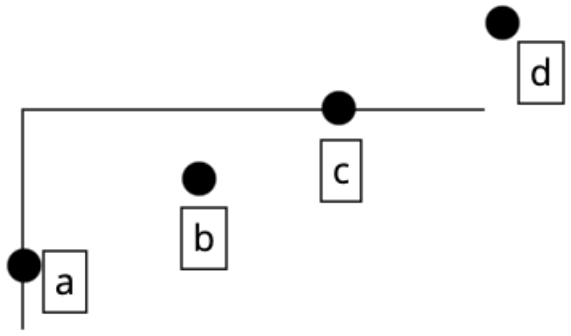

Figure 18: Connections With Single Type Orthant

Let there be four points with priority a, b, c and d. Assume that there is already an orthant connecting the points with priority a and c. Also assume that the connections follow the shape of the orthant, that it first go straight up then to the right. The only way to for the connection between the points b and d to cross with the connection between the points a and c is for the next orthant to be on the same x-coordinate as the point with priority b and the same y-coordinate as the point with priority d. This however requires that $b > c$ which prevents the first orthant from connecting the points with priority a and c as they are no longer the two highest priorities.

The key idea is that there is no way to create connections that crosses over each other while being valid when using only one orthant type.

Therefore, by substituting the points as vertexes and the connections as edges, using a single type of orthant for all connections is equal to creating a planar graph. Accordingly, we know that planar graph cannot be a complete graph [2] when the number of vertexes is 5 or more. Hence, the limit to creating cliques with one type of orthant 4 points. It would not be possible to create cliques with 5 points or more with one type of orthant.

4.4 Zero Hop

While creating cliques of arbitrary large size is not possible using one type of orthant, it might be possible with two types. This subsection and the next aims to show that this too is not possible.

The question we would like to ask is, would it be possible to create arbitrarily large cliques without any intermediary points and with any orthant types. For this local problem, we only need to gather three points from the general case.

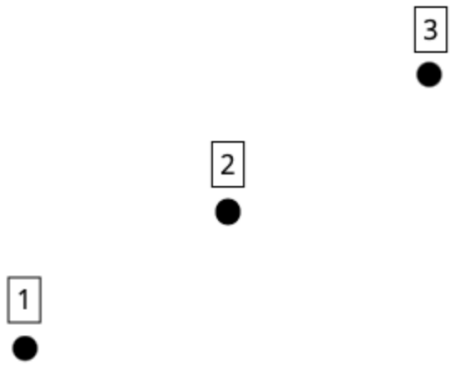

Figure 19: Three Points From The General Case

From Figure 19, we can observe that without the use of an intermediary point, we are unable to connect the point with priority 1 and the point with priority 3 with the use of any type of orthant. Hence, we can conclude that restricting the connections between all points to zero intermediary points make creating cliques of size greater than 2 impossible.

4.5 One Hop

Moving from the zero hop, the next logical step would be to check if it is possible to create arbitrary large size clique with exactly one intermediary point. Using the general case as a backdrop, we draw a line through the points, separating the plane into two.

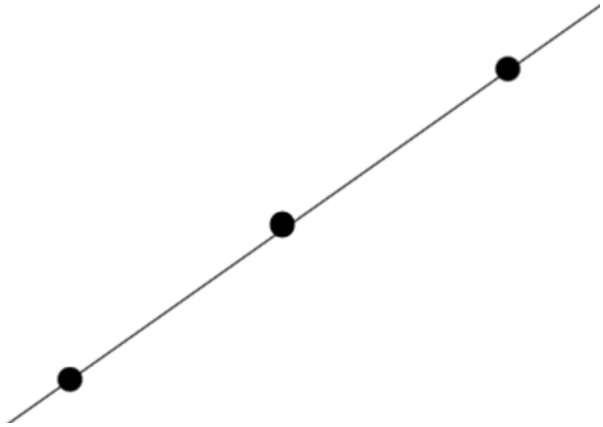

Figure 20: Split Plane In Half

Then we use Ramsey's Theorem [2] to restrict the intermediary points to one side of the line for 5 points. Ramsay's Theorem tells us that given a large enough clique where two types of connection is used to form the clique, there will be a smaller clique where only one type of connection is used. The two types of connections we assume are when the intermediary points are on the left and when they are on the right.

We use this idea to set all intermediary points to one side of the line for a clique of 5 points. In order to use the results from before, we use Ramsey's Theorem once again to create cliques where all connections are of the same combinations of two orthants types. We then show that this construction is impossible by showing that the four combination of orthants used can be reduced to an impossible state.

Firstly, connecting all 5 points to each other with only one type of orthant is shown to be impossible from subsection 4.3. This immediately rules out the two combinations that uses the same kind of orthants to create the clique for 5 points. This leaves us with the combinations that uses two different orthant type.

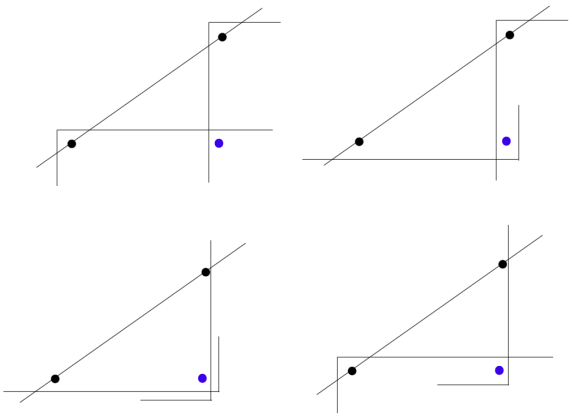

Figure 21: Four Combinations Of Two Orthants Types With One Intermediary Point

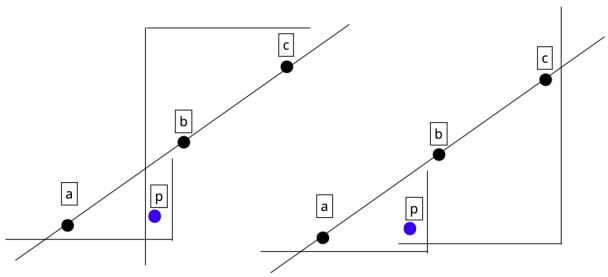

Figure 22: Up Left Then Down Right Combination(left)

With the combination of using up left orthant to connect the starting point to the intermediary point and using the down right orthant to connect the intermediary point to the end point, we can create a scenario where we can show that this combination is equivalent to a combination that uses similar types of orthants. In Figure 22, we have a scenario where the points have priorities a,b and c where $a < b < c$ with the goal of connecting the point with priority a to the point with priority c using the point with priority p. We can observe that the intermediary point with priority p must be placed strictly to the left of the point with priority b. If placed to the right of the point with priority b, there would be no way to connect the two points together using a up left orthant as $p > b > a$ or $b > p > a$ or $b > a > p$in which case the orthant will connect the point with priority b with the point with p or a. Hence, with the point with priority p to the left of the point with priority b, the only way to connect the point with priority p to the point with priority c is by having $p > b$ so that either $p > c > b$ or $c > p > b$.

If these statements are true, then we can change the down right orthant to a up left orthant and the construction would remain the same connecting the point with priority a to the point with priority c through the intermediary point with priority p. If we perform this change for all 5 points and their connections, we get the scenario where all the connections are of one type which is impossible.

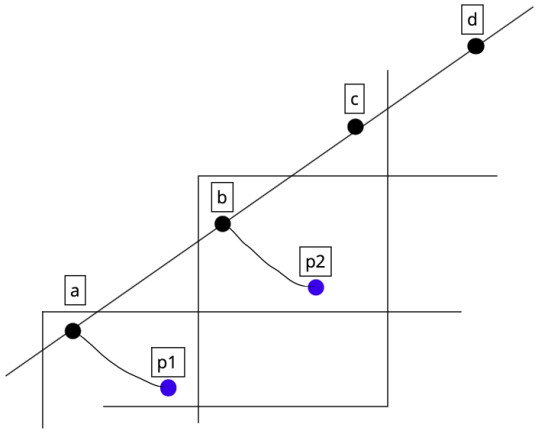

Figure 23: Down Right Then Up Left Midway Construction

Lastly, we show that connecting all 5 points with a down right orthant followed by a up left orthant is impossible. Let the scenario be that there are 4 out of the 5 points with priorities a, b, c and d where $a < b < c < d$. The goal would be to connect the point with priority a to the point with priority c and the point with priority b to the point with priority d. We observe that the intermediary point with priority $p1$ connecting the points with priority a and c needs to be below the point with priority b and have to have a priority $p1 > b$. The point with priority $p1$ needs to be below the point with priority b to ensure that the down right orthant does not include the point with priority b, further more it needs a priority of $p1 > b$ to allow it to connect to the point with priority c with a down left orthant. That said, at the same time, this scenario is the same for the point with priority $p2$ connecting the points with priority b and d. The point with priority $p2$ needs to be lower than the point with priority c and need to have a priority of $p2 > c$. As $p2 > c$, there is no way to use a up left orthant to connect the point with priority $p1$ and the point with priority c without including the point with priority $p2$ which because $p2 > c$ will either connect to the point with priority c or the point with priority $p1$ rendering this combination to be impossible.

Hence, by showing that all four combinations is impossible, we have shown that restricting the construction to

exactly one intermediary point makes it impossible to create cliques of arbitrarily large size.

5 CONCLUSION

For the UDC problem in a streaming setting, most of our results for the upper bound were in the realm of specific cases. By restricting the problem to the thin annulus variation where there are only three possible solution, we were able to use ideas such as the angle of the points to help differentiate when 1 disc is not enough. This simple technique allows us to check using only two points when 1 disc is no longer enough.

After that, our 1.5-approximation on the interval covering came from a brute-force method of checking the different variations of streamed intervals with a hitting set of size 2. This method for working out how an algorithm would play out hits its limits as soon as the size of hitting set grows. It is not a feasible way to confirm approximations for larger hitting sets.

Right after, out 1.5 lower bound for the square variation was a textbook application of the communication complexity to a new problem. The same can be said about the 2 lower bound for the half-space variation. The only improvement that could be made would be to increase the specifics of where the half-spaces and points are streamed in an algorithmic fashion.

Lastly, our algorithm for giving a hitting set of 2 points when there exist a hitting set of 1 point for the half-space variation took advantage of the properties of half-spaces that hit the center point in order to create a hitting set of 2 points that will hit all half-spaces. However, the limiting factor when trying to generalize would be how it deals with the half-spaces that does not hit the center point. The amount of information needed to be stored would increase linearly with the size of the hitting set which may not be ideal.

Following from the results for the half-space variation, it would not be too far off to see that by using the same observations, an algorithm could be made to give a hitting set of size $n + 1$ when there exist a hitting set of size n by keeping information on n intersections of minor arc and randomly reflecting one of the intersection to ensure all half-spaces that hit the center points are hit.

For the orthant problem, the orthant steps construction gave us a uniform way of thinking about using single types of orthants. However, this insight did not seem to transfer over when trying to solve the one hop problem. The problem with how we approached the one hop problem was that we brute-forces our way through by showing for each of the combinations that they are an impossible construction. This approach grows quickly unfeasible as the number of intermediary points grow. In order to improve our results, we need to either find a way to recursively use the results from zero hop and one hop or find a brand new method to prove that cliques of arbitrarily large size is not possible.

REFERENCES

[1] Sergio Cabello and Pablo Pérez-Lantero. 2017. Interval selection in the streaming model. *Theoretical Computer Science* 702 (2017), 77–96.

[2] Reinhard Diestel. 2018. *Graph theory.* Springer.

[3] Dorit S. Hochbaum and Wolfgang Maass. 1985. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM* 32, 130–136.

[4] Christopher Liaw, Paul Liu, and Robert Reiss. 2017. Approximation schemes for covering and packing in the streaming model. *arXiv preprint arXiv:1706.09533* (2017).

[5] Nabil H Mustafa and Saurabh Ray. 2010. Improved results on geometric hitting set problems. *Discrete & Computational Geometry* 44, 4 (2010), 883–895.

[6] Aniket Basu Roy, Sathish Govindarajan, Rajiv Raman, and Saurabh Ray. 2018. Packing and covering with non-piercing regions. *Discrete & Computational Geometry* 60, 2 (2018), 471–492.