Continuous Ply Minimization

by

Daniel Busto

B.Sc., The University of British Columbia, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

July 2015

Abstract

This thesis is on the Continuous Ply Minimization problem, which is an exercise in minimizing the effects of uncertainty. Many geometric problems have been studied under models where the exact location of the inputs are uncertain; this adds a layer of complexity that depends on the level of the uncertainty. The level of uncertainty is related to the regions in which a given entity may lie. At any given time the maximum number of these regions which overlap at a single point is called the "ply" of those entities at that time. These problems may be simplified by assuming the entities have low ply at specific times. Previous work has investigated the problem of obtaining low ply at a single targeted time, in a setting where only single entity can be probed each time step. It was shown that, given a long enough period of time, ply that was within a constant factor of the minimum ply that could be obtained at the target time. Continuous Ply Minimization works under a similar system, but we are interested in maintaining low ply over an entire interval of time. In order to prove results about this problem we introduce several new tools, which aid our examination. We then produce an algorithm that can maintain constant factor competitiveness with any algorithm's average ply, as long as the entities are not moving. This algorithm relies on maintaining constant competitiveness with a new notion, namely the "optimal blanket value" of a given set of entities. In the case where the entities are moving, if we are given the maximum optimal blanket value, then we can produce an algorithm that maintains ply that is no worse than a constant factor more than it.

Preface

The body of this thesis is based on original, unpublished research, which was conducted with my co-supervisors David Kirkpatrick and Will Evans.

Table of Contents

List of Figures

Chapter 1

Introduction

Motion related problems have long been of interest to the Computer Science community, and also to members of related fields. There are many different areas these problems might fall into. The area of motion planning deals with problems related to calculating routes for agents to follow, from an initial position to a goal position, without colliding with obstacles [13]. Kinetic Data Structures maintain information on certain properties of a point set, whose members are moving [11]. The simplest versions of these problems rely on knowing the true locations of certain objects at all times. In practice, this is often not feasible. Frequently we base our perception of a given environment on data that was collected in the past. There is no way to know if the situation has changed since the data was collected. However stale data are sufficient for many real life tasks. For example, when crossing the street we need to look both ways, but we do not continue to check once it has been established that the street is empty. This is because we know that cars have bounded speed, and thus if a certain stretch of road is empty we conclude our path will remain unobstructed for the near future. Another example, of using stale data to predict future events, is an Elo-type rating system, used to quantify a player's (or team's) strength in competitive games (e.g. chess). Stronger players generally have higher scores, and so one may reasonably expect that in a tournament setting the player with the highest rating will win. However such a rating system is based on past performance, and it doesn't take into account changes in a player's strength over time. Gains may be made due to more experience and practice, on the other hand a player's strength may drop due to time spent without play. There are also the uncertainties that are inherent in any such system, as a participant's play on any given day may be affected by a myriad of factors, and is not completely regular. However, there generally are bounds on how far off the rating can be, and the more games the rating is based on, the more accurate it will be. A complete beginner

has no hope of beating a master. Thus, as long as one has a way to measure the uncertainty associated with a given ranking, one can think of organizing a world championship as the problem of calculating which players could potentially be ranked number one (by looking at their ranking, and then accounting for the uncertainty). Inviting all of those players, and holding the tournament, then resolves the uncertainty, and gives you a precise ordering at that specific moment in time. This example illustrates that our methods may be more generally applied, even though we will mainly focus on the idea of physical entities moving in space.

As the number of autonomous devices being used commercially, personally, and recreationally increases, the interest in algorithms which deal with the interactions between such devices increases. Many of these devices are equipped with signalling equipment so that the device can transmit its precise location to some distant operator. As these entities move it may be infeasible to keep track of the exact location at all times, especially if your application requires several such devices. Thus, if one is doing computation using the location of these entities, there needs to be some sort of protocol to decide which entity's location should be updated at a given time step, to ensure the computation continues smoothly.

1.1 Problem Overview

In this paper we consider the problem of "Continuous Ply Minimization". It is an abstraction of the kind of problems discussed in the opening. The problem is based on the unpredictable movement of a group of entities. We are not instantaneously alerted to the movement of any given entity, and thus cannot keep accurate data on the locations of all entities. However we can probe any given entity to receive updated information on its location at any time step, but only for one entity per time step. We want to ensure that no group of entities becomes too clustered. To that end we want to probe the entities in such a way that no matter how they have moved since being probed, the maximum number of entities that could be at the same point, which we refer to as the ply, is small.

It may not always be possible to achieve query patterns which lead to "optimal" overlap. Thus we will often aim to be within a small factor of the best possible behaviour of any algorithm.

1.2 Paper Overview

In the next chapter we review some previous work. This includes online algorithms, data in motion, and previous work on the ply minimization problem. Online algorithms receive new data in the middle of their computation. Data in motion is a model for dealing with information which changes with time, and figuring out what is important to update and what is not. Previous work on the ply minimization problem has shown that good competitive ratios can be achieved, at pre-specified time steps.

In the third chapter we set up the model that we will use in our discussion of this problem. This model will overview how our entities can move, how we query their locations, and how we measure the effectiveness of our algorithm.

In the fourth chapter we examine a simplification of the problem where none of the entities move. We refer to this as the "Static Case". The lack of motion is known to the algorithm, but not to an observer, so we treat the uncertainty intervals the same way. With this restriction we create an algorithm that maintains ply that is no worse than a constant factor more than the worst ply of any algorithm, over the same time period. Additionally we show that this ply is within a logarithmic factor of the best possible ply that could be achieved at any specific time in the interval.

In the fifth chapter we discuss how the problem changes when we remove the static restriction. We show that a tweaked algorithm, which assigns entities to be probed in "windows" of time, rather than at specific steps, maintains a relatively good ply. Unfortunately this algorithm relies on having an oracle that gives it some information about the "crowdedness" of the entities.

We conclude with a review of our results, and ideas for future work. This includes ways to potentially improve on the result in chapter five to give a more general algorithm for continuous ply minimization.

Chapter 2

Related Work

There are many different ways uncertainty manifests itself in computational problems. We may be interested in making decisions which will have consequences for a long period of time, before all of the information about the behaviour of our system during that time is available, as is the case with online algorithms. It may be expensive to acquire data, consequently we want to acquire only the data that is crucial to getting an accurate result. The cost of acquiring data comes in many forms, it may require physically moving a device some distance, or procuring more expensive equiment, or the expenditure of some other resource. In the Ply Minimization problem the cost associated with acquiring of data is the time it takes to probe an entity. Whatever the cause of the uncertainty, we want to deal with it as gracefully as possible. The algorithms for these kinds of problems are often hard to analyze. Algorithms that know what data to acquire can perform better than "fair" algorithms that may make useless queries searching for the relevant data.

2.1 Performance Measures for Online Algorithms

There is a large amount of previous work on problems that deal with data that is updated in real time. Algorithms that have to deal with data as it is produced, rather than analyzing data after it has been generated, are called "online algorithms". A canonical example, from the area of computer systems, is paging. When software requests data that is stored on a hard drive, or some other storage medium, the operating system needs to decide what data should be kept in main memory and what should be removed to make room for the newly requested data. Reading data from a hard drive takes a (relatively) long time, and thus the operating system wants to keep data in memory if it will be accessed several times, in order to speed up the program. However the operating system does not know what data will be needed

again in the future, and thus it cannot make optimal choices about which data to keep. This is similar to our problem, since we do not know ahead of time which entities will stay close together, and which will spread apart. If we did know the entities trajectories calculating the optimal query pattern would be possible. Unlike the paging problem, in the continuous ply minimization problem the algorithm has control over what data it receives at each time step, through its probes.

There are many different ways to measure the performance of Online algorithms. Reza Dorrigiv and Alejandro López-Ortiz reviewed several of the most popular methods in their 2005 survey [7].

The most commonly used measure of performance for online algorithms is the "Competitive Ratio". This measure compares the performance of the algorithm to the performance of the "optimal algorithm". The "performance" measure depends on some concept of cost, which will depend on the specific application. For example in the case of paging the cost of a given algorithm is usually formulated as the number of cache misses, that is the number of times a program tries to access data not in main memory. Let $\mathscr{A}(\sigma)$ be the cost of the algorithm $\mathscr{A}$ on input σ, and $\mathrm{OPT}(\sigma)$ be the cost of the optimal algorithm. Then $\mathscr{A}$ is $C(n)$-competitive if and only if

$$C(n) = \max_{|\sigma|=n} \left\{ \frac{\mathscr{A}(\sigma)}{\mathrm{OPT}(\sigma)} \right\}.[7]$$

There can be several problems with using such a model of comparison. One specific problem is in deciding what the optimal algorithm is. Often, in order to make that decision, more power it given to the optimal algorithm. For example, the optimal algorithm may be one that is clairvoyant, and knows all the data it needs ahead of time. One has to balance giving power to the optimal algorithm to make it easy to describe and argue algorithmic costs for, with making sure it remains reasonable measuring stick for other algorithms' performance.

2.2 Data in Motion

In his 1991 PhD thesis Simon Kahan studied "Data in Motion" [12]. Data in Motion represents a broad class of problems, in which you are trying to answer queries about moving objects. The problem is exacerbated by the fact that the position of objects are not automatically updated, instead one has to request their current location. The main premise of the thesis is that as long as the queries you want to answer do not require the exact location of every object you do not have to update

all of their locations. Instead you can probe objects based on whether or not their true value will affect the answer to the query. For example if the query is "What is the maximum of a set of integer valued variables?", any variable whose range of possible values is bounded above by an integer that is smaller than the minimum bound on another variable could not possibly be the current maximum, and therefore can be ignored in responding to the query. The trade-off this encourages is sacrificing potentially longer query times in future, if more elements need to be probed, to expedite the query that is being done at the moment. Whether or not this trade-off is worth while depends on your model of computation, and how expensive probes are compared to the computation required to determine whether or not they are necessary.

2.3 The Black Box Model for Kinetic Algorithms

Many classic geometric structures on point sets, such as Convex Hulls or Delaunay Triangulations, are also interesting structures to study when the point set is in motion. The focus of Kinetic algorithms is the problem of how to maintain these structures as the entities move[11]. One can reduce the amount of computation necessary, by using insight on what changes can occur as time passes. These problems were first examined with the assumption that the motion of the objects was known ahead of time. In the "Black Box Model" the algorithm does not know the trajectories of the points ahead of time, but gets updates at regular intervals. Using the "Black Box Model" makes it harder to predict when changes will occur in the structure. The "Black Box Model" has been used in the study of many different kinetic problems, including: Convex Hulls and Delaunay Triangulations [5], Compressed Quadtrees [4], 2-Centres [6], and Spanners [10].

2.4 Calculations on Uncertain Point Sets

In "The Post Office Problem on Fuzzy Point Sets" Franz Aurrenhammer, Gerd Stöckl, and Emo Welzl study two different objectives for partitioning the plane, so that each partition represents a section of the plane which is closest to a certain entity. Their model is very similar to the one used for Voronoi Diagrams, except they look at uncertain point sets, where an entity's location may be anywhere within a given disc [2]. The first objective they consider is answering queries about the "Probably-closest Disc". The goal, in this case, is to partition the plane in to regions, one for each entity, so that any point in a given entity's region is most likely closest to that entity, rather than to other entities. In order to calculate which entity

a point is closest to Aurenhammer et al assume that the position of an entity is represented by a uniformly distributed random variable over the disc corresponding to that entity. Aurenhammer et al show how to calculate the probably-closest entity for a given point, and prove that the resulting regions are star-shaped, as long as the discs are disjoint. They then move on to "Expected-closest Disc". In this case, an entity's region contains all points whose minimal expected distance to an entity is to the entity e_i. They show that solving this problem is equivalent to finding a power diagram, for the expected minimal distances to the entities. Thus the solution consists of convex polygons, which can be constructed in $O(n \log n)$ time, and $O(n)$ space.

Maarten Löffler, and Jack Snoeyink showed that, with $O(n \log n)$ prepossessing on a set of uncertainty discs, when given the location of points within corresponding uncertainty discs one can compute Vornoi diagrams in linear time [14]. These results are originally presented under the assumption that all the discs have the same radius, and are non-overlapping. However they provide an extension showing that as long as the ply of the discs is bounded the algiorithm still works, although the constant in the time bound of the algorithm is increased by a constant.

Maarten Löffler, and Marc van Kreveld showed how to calculate the largest and smallest possible convex hulls of a set of entities that have uncertain locations [15]. They examine several different classes for the uncertainty regions, including line segments, circles, and squares, giving polynomial algorithms for calculating the largest and smallest convex hull in each case.

2.5 Ply Minimization

William Evans, David Kirkpatrick, Maarten Löffer, and Frank Staals wrote a paper on a ply minimization problem, in which the algorithm only cares about achieving the lowest possible ply at a single target time [8][9]. This is contrasted with our goal of maintaining consistently low ply over an entire time interval.

The adversary in the Evans et al paper is a clairvoyant algorithm, which uses the available time steps as effectively as possible. Hence it only probes during the n steps prior to the specified time step, because it has no need to probe any entity more than once. Evans et al showed that even if one knew the location of the entities, calculating the optimal query pattern for a single target time, is NP-Hard. Thus, their adversary is a relatively strong algorithm. The algorithm they constructed to solve their problem queries all entities, then recurses on the half of

the entities which will have regions that cover points of high ply at the targeted time. In this way Evans et al [8] focused their queries on high-priority entities. The recursion requires you have twice as much time to query as you have entities. If there is enough time to run the recursion, then their algorithm ensures a ply which is within a constant multiple of the optimal ply [8].

In previous work the algorithm could project forward to see what the ply would be at the target time, if no more probes were made. The algorithm would try to affect the final sizes of the entities' uncertainty regions to produce lower overlap. After each query it can see what the effect will be on the final situation, and update its plan. This allows it to ignore entities which it knows will not participate in a large overlap at the target time. The problem we examine in this paper is different because there is no one single time to project forward to. We are worried about the size of the entities' uncertainty regions at every time step. This means we must choose queries that will be most effective across all future time steps.

2.5.1 Recurrent Case

The recurrent case studied in Evans et al's paper [8] was a step in-between their "One-shot" problem, and our continuous one. Instead of trying to minimize the ply at one specific time, the recurrent case aims to minimize the ply every time a certain number of time steps have elapsed. The authors note that if the gap between times of interest are long enough (i.e. at least twice the number of entities) one can simply treat every target time as a separate instance of their one-shot problem. They conclude, in the case where the algorithm doesn't have enough time just to run the entirety of their algoirthm, by alternating between "Round Robin" and their method for minimizing ply they are able to get a $O(\sqrt{\frac{n}{\tau}})$ competitive ratio with the optimal ply (in one dimensional space, the general bound depends on the dimension of the space), where τ is the length of the gap between probes. Note that as τ approaches 1, the ratio approaches $\sqrt{n}$. In chapter four we show that Round Robin's ply is $O(\sqrt{n\Delta})$.

Chapter 3

Model

In this chapter we outline the model we use to investigate the Continuous Ply Minimization problem. We work with a set of entities that have real-valued locations, and these entities move unpredictably, but with bounded speed. To maintain a idea of how the entities are located, we probe a single entity at each time step. The aim of this process is to minimize the overlap between the potential locations of the set of entities; we measure this overlap as a quantity known as "ply". It is impossible to use worst case analysis to judge the effectiveness of a given algorithm, since if the entities are tightly packed high ply is unavoidable. Thus, we need different tools to deal with competitiveness. We discuss competing against other algorithms, and introduce the notion of "intrinsic ply" as another measure to compete against.

3.1 Movement Model

Let $\mathscr{E} = \{e_1, ..., e_n\}$ be a set of entities that we are maintaining information about, specifically their current location. In this paper we restrict our attention to a one dimensional universe, that is the entities are living on the real line, $\mathbb{R}$. Issues that arise from dealing with the continuous version of the ply minimization problem appear in this simple case. We believe the findings should extend to higher dimensions.

We need to introduce notation for discussing the position and movement of the entities in $\mathscr{E}$. First we describe the notation for a given entity's position at a specific time.

Definition 1. *$l_i(t)$ is the location of entity e_i at time t.*

Note that since we are dealing with a one dimensional universe $l_i(t) \in \mathbb{R}$.

Next we discuss how entities move. Each entity's speed is less than $\frac{1}{2}$ unit per time step. This model is universally applicable, since all objects have bounded upper speeds. Thus one can simply scale the unit of distance until the upper speed is $\frac{1}{2}$ per unit time. In the extreme case any object's speed is theoretically bounded by the speed of light. More practically, the maximum speed of a given device is usually well studied, and, unless acted on by outside agents, the device will stay within expected operating parameters.

The interactions between many different entities will be crucial in several of our arguments. It will be important to have ways to talk about the distribution of those entities. One notion we will use is the "diameter" of a set of entities, which is simply how far apart the extreme entities of the set are.

Definition 2. *The diameter of a set of points S is the furthest distance between two points in S. More formally*

$$diameter(S) = \max_{x,y \in S} d(x,y)$$

Where $d(x,y)$ is the Euclidean distance from x to y.

The diameter of a set of entities at a given time is the diameter of the set of those entities' locations at that time.

There are many ways this model might be varied to translate better to specific applications. Evans et al [9] study how their results change when there are several classes of entities, each of which have different bounds on their speed. There are many other variables that may be considered such as a bounds on acceleration, or turning radius. Many real life devices, such as cars, have bounds on their acceleration. If there was extra information about the direction and magnitude of the entities motion at a given time, knowing an acceleration bound would allow the algorithm to reduce its uncertainty about the entity's location at a future time. These are all interesting avenues for future work.

3.2 Probe Model

We assume that at time zero we have full knowledge of where all the entities are. At this point in time the entities start to move unpredictably, but with bounded speed, from their starting position. The interval that contains all of potential locations of an entity is referred to as the "uncertainty region" of e_i. The uncertainty

region grows $\frac{1}{2}$ unit in each direction at each time step, thus an entity's uncertainty region's volume is the difference between the current time and the last time the entity was probed.

In order to keep the ply of the uncertainty regions low, we need to be able to reduce the size of the uncertainty region of a given entity. In our model that tool is a "probe". Probes are instantaneous queries of a given entity's current location. This means if we probe an entity, e_i, at time t, we receive $l_i(t)$. It is important to note for any given time we consider the uncertainty region of an entity at that time to be the uncertainty region before we make the probe, rather than after. Hence all entities have uncertainty regions of length at least 1 at all times (since it has been at least a time step since they've been probed).

Definition 3. *For any given entity e_i and any time t, $p_i(t)$ is the last time prior to t that e_i was probed.*

Definition 4. *The uncertainty region associated with entity e_i at time t, $r_i(t)$, is the open ball centered at the location of e_i when it was last probed, with diameter $t - p_i(t)$. More formally:*

$$r_i(t) = (l_i(p_i(t)) - \frac{1}{2}(t - p_i(t)), l_i(p_i(t)) + \frac{1}{2}(t - p_i(t)))$$

Ignoring the delay between choosing the entity to probe and receiving the current location of the entity is reasonable as long as the time it takes to acquire the entity's location is short when compared to a single time unit. If it is not, such as when working with entities which are spatially distant from the computation device, it may be necessary to compensate for the delay between choosing an entity to query and receiving the requested information. It may also be interesting to consider cases were some entities can have their location updated more rapidly than others. It is important to note we ignore the time it takes for the computation to decide which entity the algorithm wants to probe. Again, unless the unit of time is very small, this delay is negligible.

3.3 Ply

The notion of ply is at the heart of our discussion. It is the measure we use to determine how well we are maintaining a clear picture of the current distribution of the entities. Briefly, the ply of a set of entities at a given time is the maximal overlap between those entities' uncertainty regions at that time.

Definition 5. *Given a set of intervals $\mathscr{I}$ the depth of a point p at time t, $\delta_{\mathscr{I}}(p,t)$, is the number of intervals of $\mathscr{I}$ which overlap that point. That is $\delta_{\mathscr{I}}(p,t) = |\{I : p \in I, I \in \mathscr{I}\}|$.*

Definition 6. *Given a set of intervals $\mathscr{I}$ the ply of that set of intervals at time t, $\rho_{\mathscr{I}}(t)$, is the maximum depth of a point, over all points $p \in \mathbb{R}$, at time t. That is $\rho_{\mathscr{I}}(t) = \max_p(\delta_{\mathscr{I}}(p,t))$.*

Previous work by Evans et al introduced this notion of ply [8]. We will often refer to the ply of a set of entities, by which we mean the ply of the associated uncertainty regions. When the set of intervals is obvious from context it is omitted from the notation.

3.3.1 Competitiveness

There are many different ways we could approach competitiveness. The simplest is to use worst case analysis. As was mentioned at the beginning of the chapter worst case analysis does not work well for this problem. This is because, in the case where all n entities are at the same location, the ply will be n regardless of the sequence of probes. Instead we rely on looking at the competitive ratio between our algorithm's ply and the minimum ply any other algorithm could achieve. The first quantity we consider is the minimum ply an algorithm can achieve at a specific point in time. This quantity is unfairly low, since it is achieved once, and not maintained over the entire time interval. Constructing a bound on this quantity relies on the notion of "uncertainty realizations". An uncertainty realization is a set of intervals, which theoretically could be the uncertainty regions for the entities, given only their current positions. Note that the set of intervals which could actually be the uncertainty regions for a given set of entities is much smaller than the set of uncertainty realizations, as the actual uncertainty regions depend on the trajectories of the entities.

Definition 7. *For a set of entities $\mathscr{E} = \{e_1, ..., e_n\}$ a set of open intervals $A = \{A_1, ..., A_n\}$ is an uncertainty realization of $\mathscr{E}$ at time t if the following three properties hold:*

1. $l_i(t) \in A_i$
2. $|A_i| \in \mathbb{Z}^+$
3. $|A_i| \neq |A_j|$ *if* $i \neq j$

Note that, as discussed above, $|r_i(t)|$ is never zero, so it makes sense for uncertainty realizations to be restricted to intervals which have positive volume.

Definition 8. *For a set of entities $\mathscr{E} = \{e_1, ..., e_n\}$, $\mathscr{A}(\mathscr{E})$ is the set of all possible uncertainty realizations.*

When the set of entities is clear from context it is omitted.//

Our surrogate for the lowest ply that can be achieved at a specific time is called the "intrinsic ply" at that time. Each uncertainty realizations has a ply. The intrinsic ply is the minimum ply that could be achieved by an uncertainty realizations.

Definition 9. *For a set of entities $\mathscr{E} = \{e_1, ..., e_n\}$ the intrinsic ply at time t, $\Delta(t) = \min_A(\rho_A(t))$, where A is an uncertainty realization in $\mathscr{A}(\mathscr{E})$.*

One can restrict the size of the uncertainty regions to those of size at most n, since reducing the size of a region can never increase ply. This is useful because it reduces the number of realizations one has to consider when calculating $\Delta(t)$. This follows fairly directly from the fact that larger uncertainty regions can only increase the ply.

Theorem 1. *When calculating $\Delta(t)$ we only need to consider uncertainty realizations which use intervals whose sizes are at most n. That is $\min_A(\rho_A(t)) = \min_{\tilde{A}}(\rho_{\tilde{A}}(t))$, where A is in in $\mathscr{A}(\mathscr{E})$, and $\tilde{A} = \{A : A \in \mathscr{A}\ \forall A_i \in A\ |A_i| \leq n\}$.*

Proof. Assume to the contrary there is no uncertainty realization for which all the intervals have size at most n whose ply is the intrinsic ply. Let A be an uncertainty realization which does achieve intrinsic ply. Note that A contains at least one interval A_i for which $|A_i| > n$.

We can transform A into a set which contains no intervals of length greater than n with the following process. First let $A^{'} = A$. For any A_i in A such that $|A_i| > n$ there exists an interval $A_i^{'}$, such that $|A_i^{'}| = j < n$, $A_i^{'} \subset A_i$, $l_i(t) \in A_i^{'}$ and no other A_k in $A^{'}$ has length j. Set $A^{'} = (A \cup \{A_i^{'}\}) \setminus \{A_i\}$, and repeat until there are no intervals of length greater than n.

For any point p, the depth at that point for $A^{'}$ is at most as large as in A, that is $\delta_A(p,t) \geq \delta_{A'}(p,t)$, since the only possible change is that some intervals which used to cover p no longer do. Hence $A^{'}$ realizes the intrinsic ply, and contains intervals whose volume is at most n. This is a contradiction, and thus $\Delta(t)$ is always realized by an uncertainty realization which contains no interval whose volume is greater than n. □

Note that $\Delta(t)$ is a measure which is oblivious to the past or future behaviour of the entities. That is the measure only depends on the current location of the entities. In this way it has a lot fewer restrictions on it than an algorithm does. An algorithm has to make decisions based on the present situation, but its worst ply over a time interval depends heavily on what has happened before that time step, and what happens after that time step, over the course of the interval.

In the discussion of how well an algorithm can do with respect to the intrinsic ply, it is important to have an example of a point set for which it is hard to maintain low ply. Our canonical example is "arithmetically spaced" points. The intuition behind this example is to force all of the entities as close together as possible while still having low intrinsic ply. This makes sense, since if the entities are close to one another it will be harder to probe the entities so that their uncertainty regions remain pairwise independent. Our example is constructed in such a way that the intrinsic ply is one, regardless of the number of entities used. Consider building an example in terms of the uncertainty realization we assign to the entities in the example. Each entity is thought of as corresponding to a "brick" of length equal to the uncertainty region it is assigned in the optimal uncertainty realization. Since we are trying to force the entities as close together as possible, without causing overlap between the regions in the uncertainty realization (as the intrinsic ply will be one) we should have the uncertainty regions all have volume n or less, and have no gaps between the entities' uncertainty regions. However it is not obvious what order the uncertainty regions should appear in. We decided to place the entities in order of increasing uncertainty region volume, the smallest is placed at the origin, and the larger regions are placed to its right. The example is described formally below.

Definition 10. *For any positive integer n, let $A(n) = \{A_i\}$ be the set of n open intervals such that the following conditions hold for all positive integers i less than n:*

1. *A_1 is the interval $(0,1)$.*
2. *A_i has volume i, that is $|A_i| = i$.*
3. *The right endpoint of A_i is the left endpoint of A_{i+1}, that is $\sup A_i = \inf A_{i+1}$.*

A set of n entities are "arithmetically spaced" if the entity e_i is at the midpoint of the interval A_i.

It is now straightforward to confirm that our construction leads to the ply we intended.

Lemma 2. *A set of arithmetically spaced entities has intrinsic ply one.*

Proof. By definition the set of intervals $A(n)$ described in the definition of arithmetically spaced entities is an uncertainty realization of said entities. By definition each region shares its endpoints with its immediate neighbours, thus no regions overlap. Thus $A(n)$ has ply one, and hence $\Delta = 1$. □

We are also interested in situations where the entities have larger intrinsic ply values than one. We have an analogous concept which represents how tightly packed entities can be if the intrinsic ply is higher than one. We again think about the entities as "bricks", which are the length of the uncertainty region assigned to it. Consider the smallest 2Δ entities. Each of these entities can be paired with another entity, whose assigned uncertainty region has volume at most 2Δ, so that the combined volume of their uncertainty regions is $2\Delta + 1$. Thus all of the first 2Δ entities can be placed in the interval $(0, 2\Delta + 1)$. We repeat this process for each subsequent group of 2Δ entities, creating "blocks" of length $(4i-2)\Delta + 1$. Thus the size of the blocks follows an arithmetic sequence. We refer to these sets of entities as Δ-arithmetically spaced entities.

Definition 11. *For any positive integer Δ define the set of intervals $A^1(\Delta)$ to be:*

$$A^1(\Delta) = \{(0, j) : 1 \leq j \leq \Delta\} \cup \{(j, 2\Delta + 1) : 1 \leq j \leq \Delta\}$$

We will define the rest of the $A^i(\Delta)$'s recursively. Let m_i be the supremum of intervals in $A^i(\Delta)$. Then m_i is also the infimum of intervals in $A^{i+1}(\Delta)$. Note $m_0 = 0$. Thus, since each $A^i(\Delta)$ has diameter $(4i-2)\Delta + 1$, the value m_i is $2i^2\Delta + i$. Let $A^i_j = (m_{i-1}, m_{i-1} + 2i\Delta + j)$ and $A^i_{j+\Delta} = (m_{i-1} + 2i\Delta + j, m_i)$, for $j = 1, ..., \Delta$. $A^i(\Delta)$ is the set of all of these intervals. That is

$$A^i(\Delta) = \cup_{j=1}^{2\Delta} A^i_j$$

Let

$$A(n, \Delta) = \bigcup_{i \leq \frac{n}{\Delta}} A^i(\Delta)$$

A set of n entities are "Δ-arithmetically spaced" if the entity $e_{2\Delta i + j}$ is located at the midpoint of A^i_j.

Note that our definition for $A(n)$ corresponds to the definition of $A(n, 1)$.

Next we argue that the name Δ-arithmetically spaced entities makes sense, since their intrinsic ply is in fact Δ. This result follows directly from our definition.

Lemma 3. *A set of Δ-arithmetically spaced entities has intrinsic ply Δ.*

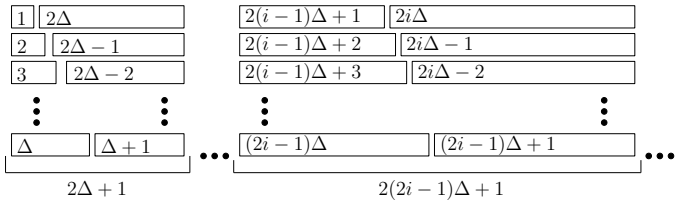


Figure 3.1: This diagram illustrates how the regions are layed out for Δ-arithmetically separated points.

Proof. By definition the set of intervals $A(n,\Delta)$ described in the definition of Δ-arithmetically spaced is an uncertainty realization of the given set of uncertainty regions. In order to show the ply of $A(n,\Delta)$ is Δ we will argue that the intervals in any given $A^i(\Delta)$ do not overlap with intervals outside of $A^i(\Delta)$, and that they overlap at most $\Delta-1$ of the other entities in $A^i(\Delta)$.

Note that, by construction, the union of the intervals A^i_j and $A^i_{\Delta+j}$ is the interval (m_i, m_{i+1}), minus the point where they meet, since the point at which they meet is in neither interval. Thus no intervals in $A^i(\Delta)$ overlap with entities from $A^k(\Delta)$, for any $k \neq i$. Also note the intervals in the pair are non-overlapping. There are Δ pairs of intervals which do not overlap in $A^i(\Delta)$ (since there are 2Δ entities total). Each point in the interval is overlapped by at most one of the two intervals in each pair, thus any point is overlapped by at most Δ intervals, and the ply is at most Δ. The point $m_i + \frac{1}{2}$ is in all of the first Δ intervals, hence the ply is at least Δ. Since we have matching upper and lower bound, the ply of $A^i(\Delta)$ is Δ.

Since none of the intervals overlap with intervals from other $A^i(\Delta)$'s, and the ply in any given $A^i(\Delta)$ is Δ the set of all the intervals, $A(n,\Delta)$, has ply Δ. □

However, we will find that no algorithm can compete favourably with the intrinsic ply, because it is a short sighted measure, which ignores the side effects of minimizing ply at one specific point in time. Δ-arithmetically spaced entities will provide an example in which no algorithm can maintain ply that is a constant factor of $\Delta(t)$ at all time steps (proof is provided in chapter four).

We will also compare our algorithm's performance directly to that of other algorithms. When comparing against other algorithms it no longer makes sense to compare their plies at each time step. An algorithm may have behaviour that makes their ply very low at a certain time step, while it is much higher at another time step. One does not want to conclude an algorithm is bad simply because one

evaluates its behaviour at a time when the other is doing much better than it would at any other given time step. To that end we will argue about the maximum ply algorithms produce over some time period.

Chapter 4

Static case

The ply minimization problem can be hard to approach in its full generality. There are two main sources of difficulty in constructing a competitive algorithm for this problem. The first snag is the uncertainty associated with the movement of the entities. This uncertainty obscures which entity is best to probe, as a probe's effectiveness depends on how close the entity we are probing is to the neighbouring points. The second problem is the fact that we are restricted to probing at most once per time step. In order to gain insight into the techniques and tools we use in the general case, we first consider a simplification of the problem which focuses our attention on this second difficulty. That simplification is the "static case" in which no entities are moving. More formally, in the static case entity e_i's position $l_i(t)$ is a constant, and does not depend on the value of t. Because of this fact we drop the time from the expression and use l_i to denote the position of e_i at any time.

In the static case there is no longer any uncertainty associated with the movement of the entities. This constraint means we have a clear picture of each entities' location, and hence of a probe's effect. Specifically, we can calculate what an entities uncertainty region will be at each time step, based on our probe sequence. Thus, an adversary that knows the trajectories of the entities no longer has a competitive advantage. From the known location of the entities one could construct an algorithm which follows the optimal probe sequence. However, it is not clear how one would determine the optimal probe sequence. It seems reasonable to surmise that determining the optimal probe sequence would be hard, as in the one-shot case calculating the optimal probe sequence is NP-hard [8].

The main problem is how to choose an entity to probe based on the known locations of the entities and their current uncertainty regions. To approach this

problem we first analyze how well an algorithm can maintain low ply if it has no information about the position of the entities, other than that they are static. Round Robin is a prime example of such an algorithm. We will argue that Round Robin's performance is the best one can get from an "oblivious" algorithm. We show that Round Robin manages to maintain a sub-linear competitive ratio, an interesting result for such a simple algorithm.

We then consider how well algorithms in general can perform as a function of the intrinsic ply, Δ. Although a constant competitive ratio can be achieved at specific points in time, as discussed in the previous work [8], trying to maintain low ply at all times is impossible. Any algorithm that runs for a sufficiently long period of time will have average ply that is at least a $\log\left(\frac{n}{\Delta}\right)$ factor greater than Δ, for certain sets of entities. Finally, we construct an algorithm which probes entities with frequency proportional to a measure of their "closeness" to other entities, which we call their "x-blanket". The algorithm, which we call the "Rounded Blankets algorithm", achieves ply $O\left(\Delta \log\left(\frac{n}{\Delta}\right)\right)$. More importantly, it achieves constant factor competitiveness with any other algorithm. This shows the "optimal blanket value" of a given set of entities is indicative that set of entities' optimal ply, in the static case. We use the Rounded Blankets algorithm as a guide for how to build an algorithm in the dynamic case.

4.1 Oblivious Algorithms

We begin our investigation by assuming our algorithm stores no information. Crucially this means it does not remember an entity's location after it probes that entity. We refer to such algorithms as "oblivious" algorithms. All an obvious algorithm can do is partition the time steps, and then assign a set of time steps from the partition to each entity. This assignment corresponds to the decision to probe the given entity at each time step in its assigned set. We call an assignment of entities to these partitions a "labelling" of the entities. The only thing that affects this partition is the number of entities, so any particular algorithm has the same behaviour on every set of entities that have the same cardinality. However, the resulting ply will be vastly different depending on the locations of the entities. We cannot get constant factor competitiveness with an algorithm which knows where the entities are. This can be seen by considering a situation where certain entities are close together, and other entities are spread out, then the ones that are close together require more probes to minimize ply. The question we aim to answer is how good a competitive ratio can we achieve.

Observation 1. *Each oblivious algorithm can be described by its partition of the time steps $P = \{P_1, ...P_n\}$. At any time t the entity corresponding to the partition P_i has some gap since the last time it was probed. More formally this gap is $g_i(t) = \min\{t - \tau : \tau < t,\ \tau \in P_i\}$. The volume of the uncertainty region of the entity associated with P_i will be $g_i(t)$, at time t. Thus finding the worst case labelling is equivalent to placing uncertainty regions that correspond to these gaps such that the resulting ply is maximized. Note that unlike the intrinsic ply these uncertainty regions must be centered on their associated entity's location.*

Round Robin is a simple algorithm, and hence is a nice place to start. Round Robin is an oblivious algorithm that partitions the time steps evenly into the sets $P_i = \{z : z \equiv i \pmod{n}\}$. This means each entity is probed periodically, with a period of n time steps. We show that Round Robin gives the best worst case performance for an oblivious algorithm. We then move on to arguing that it always manages to maintain ply which is $O(\sqrt{n\Delta})$, for any set of n static entities. We then show that, for any given Δ, a set of Δ-arithmetically spaced entities will cause Round Robin to have ply $\Omega(\sqrt{n\Delta})$ at least half the time in the worst case, showing the above bound is tight.

First we show that Round Robin is the "best" oblivious algorithm. By best we mean its worst case ply for a given set of n static entities and time t is no worse than any other algorithm's worst case ply for the same set of entities. The worst case behaviour is understood to be taken over all labellings of the entities. Note that Round Robin probes all entities with the same period, thus at any given time the entities' uncertainty regions have volume one through n. Therefore, unlike with a generic algorithm, the worst case assignment of uncertainty regions to entities may be the same at each time step. However, these uncertainty regions will be produced by a different labelling at each time step.

Theorem 4. *For any set $\mathscr{E}$ of static entities, and any oblivious algorithm $\mathscr{A}$, at any time t there is a labelling of the entities such that $\mathscr{A}$ has ply greater than or equal Round Robin's worst possible ply at t over all potential labellings of the entities.*

Proof. Let ρ_{RR} be Round Robin's worst ply at t, over all possible labellings of $\mathscr{E}$. Note that as Round Robin's worst ply is the same at all times it doesn't depend on t, and thus using just ρ_{RR} to represent its worst ply makes sense. Let L_{RR} be a labelling of the entities that causes Round Robin to have ply ρ_{RR} at t. Since we are interested in the worst case, we may assume that Round Robin is being run with labelling L_{RR}. For clarity I will refer to the uncertainty region generated by Round

Robin for entity e_i at time t as "$r_i^{RR}(t)$". Similarly those produced by $\mathscr{A}$ will be denoted "$r_i^{\mathscr{A}}(t)$". For simplicity assume $t > n$, there will be a discussion about how to deal with smaller time steps at the end of the proof.

Let $\mathscr{E}_\rho(t) = \{e_i, ..., e_{\rho_{RR}}\}$ be a subset $\mathscr{E}$ which is a minimal set that realizes Round Robin's worst ply at t. That is $|\mathscr{E}_\rho(t)| = \rho_{RR}$, and all the $r_i^{RR}(t)$ for e_i in $\mathscr{E}_\rho(t)$ contain a common point. Assume the entities are numbered in such a way that $|r_1^{RR}(t)| > |r_2^{RR}(t)| > ... > |r_{\rho_{RR}}^{RR}(t)|$. Recall Round Robin probes entities once every n time steps. Thus we know that the earliest the entity e_i was last probed is the time step $t - n + i$, since all the entities have been probed in the last n time steps, and the entities $e_1, ..., e_{i-1}$ were all last probed before it. Thus $|r_i^{RR}(t)| \leq n - i$.

We will now discuss how to label the entities so that $\mathscr{A}$ gets ply at least ρ_{RR} at t. By the definition of oblivious algorithms we have n sets of probe times. Let these sets of time steps be called $\mathscr{P}_1, ..., \mathscr{P}_n$. Without loss of generality we can assume that the partitions are ordered by the length of the gap since their last time step prior to t. That is $\mathscr{P}_1$ has the largest gap, and $\mathscr{P}_n$ has the smallest. Note that $\mathscr{P}_1$ cannot contain any of the time steps $\{t - n + 1, ..., t - 1\}$, since there are $n - 1$ sets which all must contain times between $\mathscr{P}_1$'s last time before t and t itself. By induction $\mathscr{P}_i$ cannot contain any of the time steps $\{t - n + i + 1, ..., t - 1\}$. Assign $\mathscr{P}_i$ to e_i. Then $|r_i^{\mathscr{A}}(t)| \geq n - i$. Label the remaining entities arbitrarily. At time t, the probe sequence of $\mathscr{A}$ leads to uncertainty regions such that $|r_i^{\mathscr{A}}(t)| \geq |r_i^{RR}(t)|$ for all $i \leq \rho_{RR}$. Since for these entities the associated uncertainty regions $r_i^{RR}(t)$ all overlap at t, the uncertainty regions $\{r_i^{\mathscr{A}}(t) : i \leq \rho_{RR}\}$ will all overlap at t, and thus $\mathscr{A}$ will have ply at time t which is at least ρ_{RR}.

If $t \leq n$ then there are some entities which haven't yet been probed by either algorithm, since not enough time has passed since $t = 0$. The argument is the same as before, except the first $t - n$ entities are assigned sets from the partition which do not include times less than t. □

Now to move to how well Round Robin competes with the intrinsic ply. The maximum possible ply is achieved if all of the entities' uncertainty regions overlap. Thus the worst ratio would be if our algorithm had a ply of n but the optimal query pattern led to a ply of 1. This leads to a ratio of n between our ply, ρ, and the intrinsic ply, Δ. We will show that even the simple algorithm of Round Robin does much better than a linear competitive factor, and manages to maintain ply $O(\sqrt{n\Delta})$ in all cases. We also give a specific example where average ply $\Omega(\sqrt{n\Delta})$ is unavoidable for Round Robin, showing that this bound is optimal.

Theorem 5. *For all times t and all sets of static entities $\mathscr{E}$ with cardinality n, Round Robin has ply $O(\sqrt{n\Delta})$.*

Proof. Consider an arbitrary group of n entities at some arbitrary time t. Consider a subgroup $\mathscr{E}_\rho$ of $\mathscr{E}$, such that the cardinality of $\mathscr{E}_\rho$ is ρ, and the uncertainty regions of entities in $\mathscr{E}_\rho$ all overlap at some point p at time t. That is this set of entities is one of the minimal sets (in terms of number of entities) which has ply ρ at time t. Assume without loss of generality $\mathscr{E}_\rho = \{e_1, ..., e_\rho\}$.

Round Robin queries these entities every n times steps. Thus each of their uncertainty regions is at most n units wide, and so for each entity e_i in $\mathscr{E}_\rho$ the position of e_i, l_i, must be in the interval $(p-n, p+n)$. Let $A = \{A_1, ..., A_n\}$ be an uncertainty realization for $\mathscr{E}_\rho$ which achieves ply Δ. By Theorem 1 we can assume that each A_i in A has volume at most n. Note that since l_i is in the interval $(p-n, p+n)$, and $|A_i|$ is at most n, we know A_i is a subset of $(p-2n, p+2n)$. Since the length of this interval is $4n$, and the regions $\{A_i, ..., A_\rho\}$ have ply at most Δ the total volume of these regions is at most $4n\Delta$. Thus

$$\begin{aligned} 4n\Delta &\geq \sum_{i=0}^{\rho} |A_i| \\ &\geq \sum_{i=0}^{\rho} i \\ &\geq \frac{1}{2}\rho^2 \\ \rightarrow \rho &\leq \sqrt{8n\Delta} \end{aligned}$$

Thus Round Robin's ply is $O(\sqrt{n\Delta})$

□

This result is very interesting, because it provides a glimpse into the underlying relationship between how densely packed the entities are and the intrinsic ply. It shows that we can get a non-trivial bound on the competitive ratio between any intelligent algorithm and the intrinsic ply just because of the amount of space a set of entities with ply Δ must take up.

We now move on to showing that this bound is tight for Round Robin. We will do this by showing for a set of Δ-arithmetically spaced points Round Robin produces ply $\Omega(\sqrt{n\Delta})$.

Theorem 6. *If n static entities are* Δ*-arithmetically spaced then Round Robin produces ply* $\Omega(\sqrt{n\Delta})$ *at least half of the time.*

Proof. Let $\mathscr{F}$ be the first $\frac{1}{2}\sqrt{n\Delta}$ of the n entities, which are Δ-arithmetically spaced. Recall one can divide Δ-arithmetically spaced entities into groups of 2Δ entities each that have diameter $(4i-2)\Delta+1$. $\mathscr{F}$ covers $\frac{1}{4}\sqrt{\frac{n}{\Delta}}$ of these groups.

Consider a point at time t when less than half of the elements of $\mathscr{F}$ have been probed in the past $\frac{n}{2}$ time steps, that is in the interval $[t-\frac{n}{2},t]$. Then at t half the entities in $\mathscr{F}$ have uncertainty regions which are wider than $\frac{n}{2}$, thus they each have a radius greater than $\frac{n}{4}$. The distance from the e_1 to $e_{\frac{1}{2}\sqrt{n\Delta}}$ is at most the diameter of the union of the first $\frac{1}{4}\sqrt{\frac{n}{\Delta}}$ groups. This is

$$\begin{aligned}
|l_1 - l_{\frac{1}{2}\sqrt{n\Delta}}| &\leq \sum_{i=1}^{\frac{1}{4}\sqrt{\frac{n}{\Delta}}} ((4i-2)\Delta+1) \\
&\leq 4\Delta\left(\frac{1}{4}\sqrt{\frac{n}{\Delta}}\right)^2 - (2\Delta-1)\left(\frac{1}{4}\sqrt{\frac{n}{\Delta}}\right) \\
&\leq \frac{n}{4}
\end{aligned}$$

Thus all of the entities in $\mathscr{F}$ that have been probed more than $\frac{n}{2}$ time steps ago have uncertainty regions that cover all of the entities in $\mathscr{F}$. Hence ply is $\Omega(\sqrt{n\Delta})$.

Consider any arbitrary time step t. Some number of entities in $\mathscr{F}$ are probed in the interval $[t-\frac{n}{2},t]$ and the others are probed during $[t+1,t+\frac{n}{2}+1]$, since the union of these two intervals is n time steps long. If more than half are probed during the first, less than half will be probed in the second and vice versa. Thus one of either t or $t+\frac{n}{2}+1$ must be after a period of $\frac{n}{2}$ time steps when less than half of the elements of $\mathscr{F}$ have been probed. For any pair of time steps which are separated by $\frac{n}{2}$ time steps Round Robin produces a ply which is $\Omega(\sqrt{n\Delta})$ at one. This means at half of all time steps Round Robin has ply $\Omega(\sqrt{n\Delta})$.

□

This shows that there are cases where the ply Round Robin gets can be as bad as $\sqrt{n\Delta}$, which we know from Theorem 5 is also an upper bound on Round Robin's ply. Since we know no oblivious algorithm can do better than Round Robin, if we want to guarantee ply less than $\sqrt{n\Delta}$ we have to use some information about the location of the entities. However it isn't clear yet that we should expect to get ply better $\sqrt{n\Delta}$. In the next section we will argue the best ply one can get relative to

the intrinsic ply is $O\left(\Delta \log\left(\frac{n}{\Delta}\right)\right)$.

4.2 Intrinsic Ply Ratio

In this section we will show that no algorithm can maintain a constant competitive ratio to the intrinsic ply, even in the static case. The tool we utilize to demonstrate this fact is the trade-off between probing an entity frequently enough that its uncertainty region remains manageable, and leaving enough probes for the other entities. We will conclude that we do not have enough probes to maintain small uncertainty regions for all the entities.

Our example is again a set of Δ-arithmetically spaced entities, as it was in the section on oblivious algorithms. The intuition behind using this example, as discussed in the chapter on the model, is that it packs the entities in as tightly as possible. As a preview we will informally argue that any algorithm will suffer $\log n$ ply for a set of 1-arithmetically spaced entities. We will make simplifying assumptions to make this argument easier. Assume that each element is probed with a fixed frequency, that is the period between probes for a given entity is unchanging. Note that how often we can probe entities is bounded by the fact we can only probe one entity per time step. Hence the sum of these frequencies needs to be less than one. An easy way to ensure this is to use Round Robin. Unfortunately, as shown in the previous section, we can't avoid ply $O(\sqrt{n\Delta})$ by using Round Robin. So instead we want to distribute the probes so that the entities that are closely packed have smaller average uncertainty regions. Since the gap between entities increases by one for each entity going from e_1 to e_n for 1-arithmetically spaced entities, it would make sense for the probe frequency of the entities to be linear in the entities index as well (recall that e_1 is the leftmost entity, and e_n is the rightmost). So the total probe frequency should be $\sum_{i=1}^{n} \frac{1}{ki}$ for some k. We want this sum to be less than one so:

$$\begin{aligned} 1 &> \sum_{i=1}^{n} \frac{1}{ki} \\ &> \frac{1}{k} \log n \\ \rightarrow k &> \log n \end{aligned}$$

So setting each entity's probe period to a constant factor of $\log n$ times its index will give us a set of probe frequencies, which do not require more than one probe

per time step. This means that e_i's uncertainty region grows to have length on the order of $i \log n$. The region up to $i \log n$ units away from e_i contains about $\log n$ other entities. This means that we can expect a worst ply of approximately $\log n$.

Now we will build the tools we need to formally argue that ply proportional to $\Delta \log\left(\frac{n}{\Delta}\right)$ is sometimes required to guarantee a usable probe schedule. First, we introduce a concept that will be important in several upcoming proofs. We will be interested in averaging the area that an entity's uncertainty region covers over a certain time period. However we may be only interested in a subset of the total uncertainty region that lies within some interval.

Definition 12. *$\sigma(i, I, T)$ is the total overlap between the interval I and e_i's uncertainty region over the time period T. That is $\sigma(i, I, T) = \sum_{t \in T} |r_i(t) \cap I|$.*

Using this newly defined quantity we will describe the relationship between how often an entity is probed and the area covered by its uncertainty region, over a given time period. This will be useful in arguing how many probes a group of entities needs to receive to avoid having large amounts of area covered by their uncertainty regions, which would lead to high ply. Note that this theorem's hypothesis is stated in terms of the closure of a given interval I. This will be important in our application, because we will be looking at open intervals, but want to argue about entities at the boundary of these intervals.

Lemma 7. *For any interval I, and entity e_i such that l_i is in the closure of I, if e_i is probed fewer than $p_i \geq 1$ times over the time interval T then $\sigma(i, I, T) > \frac{|T| \min\left(\frac{|T|}{2p_i}, |I|\right)}{2}$.*

Proof. There are three variables which affect the size of the overlap of e_i's uncertainty region and the interval I. These variables are: how many times the entity is probed, when it is probed, and where it is positioned.

Note that the area can only be increased by having fewer probes, since a probe added at time t decreases the area of e_i's uncertainty region at that time, and all times after it up to the next probe. We know e_i is probed less than p_i times, thus we can bound $\sigma(i, I, T)$ with the minimal area when e_i is probed $p_i - 1$ times (note since $p_i \geq 1$ this is non-negative).

Consider an arbitrary probe sequence for entity e_i, such that the j^{th} probe occurs at time t_j. Such a sequence breaks T into a number of "gaps", that is time intervals between probes. More formally for all $k \leq p_i$ the gap g_k is $[t_{k-1}, t_k]$, where $t_0 = \min(T)$ and $t_{p_i} = \max(T)$. Note that over the gap g_k the uncertainty region

for e_i, $r_i(t)$, goes from being the ball $B(l_i,1)$ to being the ball $B(l_i,|g_k|)$ (with the possible exception of the first gap, since e_i may not have been probed at t_0). Let $G = \{g_1,...,g_{p_i}\}$. Then

$$\sigma(i,I,T) \geq \sum_{g\in G}\sum_{j=1}^{g} |I \cap B(l_i,j)|$$

Now we will argue what position of e_i leads to the minimal $\sigma(i,I,T)$ value. The position of e_i affects the length of the intersection in the sum. The length of this intersection is minimized by placing e_i at one of the extremes of I, since then only half of the ball is within the interval. Thus

$$\sigma(i,I,T) \geq \sum_{g\in G}\sum_{j=1}^{g} \min\left(\frac{j}{2},|I|\right)$$

We will apply Jensen's Inequality using $f(g) = \sum_{j=1}^{g}\min(\frac{j}{2},|I|)$. In order to use Jensen's Inequality we need to show that $f(g)$ is convex. To see that this is true, note the derivative of $f(g)$ when g is less than $2|I|$ is $\frac{g}{2}$, and the derivative is $|I|$ when g is greater than $2|I|$. Thus the derivative is non-decreasing, and $f(g)$ is convex. Since $|G| = p_i$ by Jensen's Inequality we have :

$$\begin{aligned}
\frac{\sum_{g\in G} f(g)}{p_i} &\geq f\left(\frac{\sum_{g\in G} g}{p_i}\right) \\
&= f\left(\frac{|T|}{p_i}\right) \\
\leftrightarrow \frac{\sum_{g\in G}\sum_{j=1}^{g}\min\left(\frac{j}{2},|I|\right)}{p_i} &\geq \sum_{j=1}^{\frac{|T|}{p_i}} \min\left(\frac{j}{2},|I|\right) \\
\leftrightarrow \sum_{g\in G}\sum_{j=1}^{g}\min\left(\frac{j}{2},|I|\right) &\geq p_i \sum_{j=1}^{\frac{|T|}{p_i}} \min\left(\frac{j}{2},|I|\right)
\end{aligned}$$

Thus

$$\sigma(i,I,T) \geq p_i \sum_{j=1}^{\frac{|T|}{p_i}} \min\left(\frac{j}{2},|I|\right)$$

Consider the plane $\mathbb{R}^2$, let the x-axis be position, and the y-axis be time. Then we can consider σ to be the area of the union of horizontal bars $(r_i(t)\cap I)\times[t,t+1]$. We can lower bound this area by the area of right triangles, which each have height

$\frac{|T|}{p_i}$ and base either $\frac{|T|}{2p_i}$ or $|I|$, whichever is less. The total area of all triangles is half the area of the rectangle with base length $\min\left(\frac{|T|}{2p_i}, |I|\right)$, and height T.

Thus we get

$$p_i \sum_{j=1}^{\frac{|T|}{p_i}} \min\left(\frac{j}{2}, |I|\right) > \frac{|T| \min\left(\frac{|T|}{2p_i}, |I|\right)}{2}$$

Which implies

$$\sigma(i,I,T) > \frac{|T| \min\left(\frac{|T|}{2p_i}, |I|\right)}{2}$$

□

We will probe entities at time intervals which are a constant factor of the length of an interval they are contained in (this interval, the x-blanket, will be discussed shortly). The following corollary deals with the case where the number of times an entity is probed is less than a constant times the ratio between the length of the time interval and diameter of the spatial interval of interest.

Corollary 8. *If e_i is probed fewer than $p_i = \frac{k|T|}{|I|} \geq 1$ times over the time period T, then $\sigma(i,I,T) > \frac{|T||I|}{4k}$, for $k \geq 1$. Note that since we require $p_i \geq 1$, we must have $|T| \geq \frac{|I|}{k}$.*

Proof. By Lemma 7 we know

$$\sigma(i,I,T) > \frac{|T| \min\left(\frac{|T|}{2p_i}, |I|\right)}{2}$$

Substituting in our value for p_i we get

$$\sigma(i,I,T) > \frac{|T| \min\left(\frac{|I|}{2k}, |I|\right)}{2}$$

Since $k \geq 1$ the value of the min will be its first element, and hence $\sigma(i,I,T) > \frac{|T||I|}{4k}$. □

Now we use these new tools to show for Δ-arithmetically spaced entities a ply of $\Delta \log\left(\frac{n}{\Delta}\right)$ is unavoidable.

Theorem 9. *For every set of Δ-arithmetically spaced entities any algorithm will have average ply $\Omega(\Delta \log\left(\frac{n}{\Delta}\right))$ over any time period T, where $|T| > n \log\left(\frac{n}{\Delta}\right)$.*

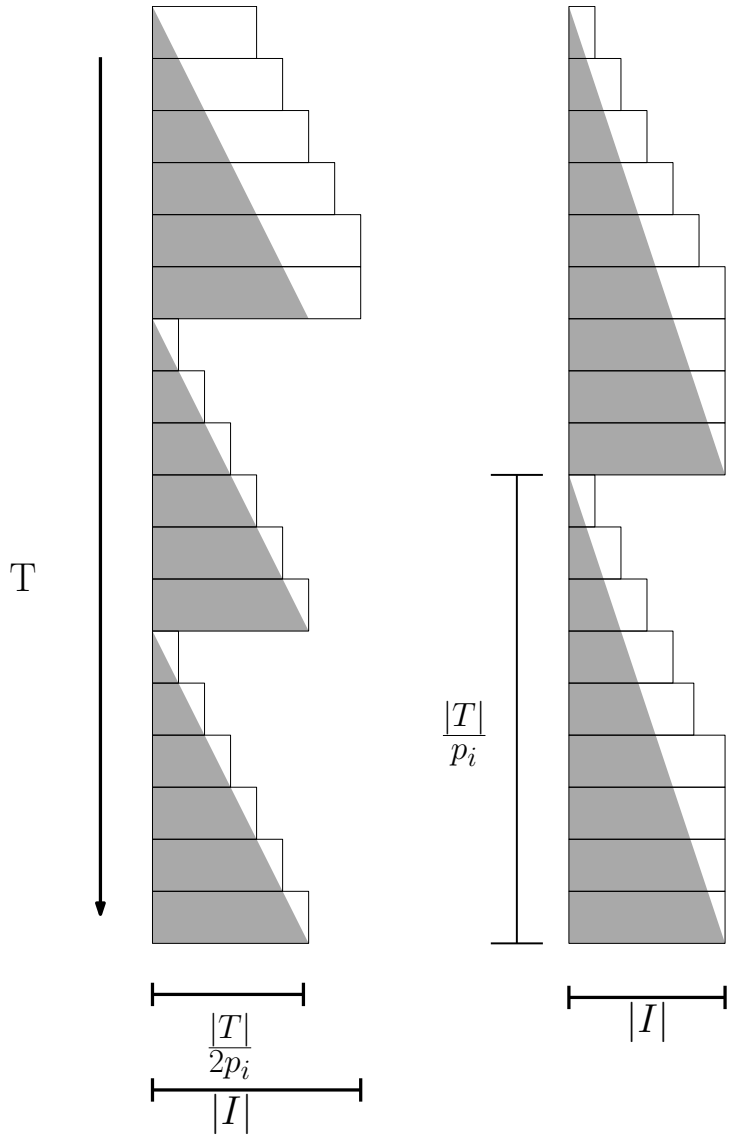


Figure 4.1: Lower bounding the area of the uncertainty regions for entity e_i over time interval T and space interval I. The white bars represent the true uncertainty regions, and the gray triangles represent the area we use for a lower bound. Notice the triangle may underestimate the uncertainty regions from the first gap by a large margin. In the figure on the left p_i is such that $\frac{|T|}{2p_i}$ is less than $|I|$, in the figure on the right p_i is much smaller, so $|I|$ is larger than $\frac{|T|}{2p_i}$.

Proof. Let n be the size of the set of Δ-arithmetically spaced entities. To start we will subdivide the entities into groups of size $\Delta\log\left(\frac{n}{\Delta}\right)$. The first $\Delta\log\left(\frac{n}{\Delta}\right)$ entities will constitute the first group, G_1, the next $\Delta\log\left(\frac{n}{\Delta}\right)$ the second, G_2, and so on. This means that each group covers $\frac{\log\left(\frac{n}{\Delta}\right)}{2}$ of the sets of 2Δ intervals we used in the definition of Δ-arithmetically spaced entities. Recall that diameter(G_i) is the distance between the entities in G_i that are furthest apart.

We now consider the ply an algorithm will get for each group of entities, ignoring overlap that may occur with uncertainty regions of entities from other groups. This is a lower bound on the actual ply the algorithm achieves. In order to maintain a low ply within a group's entities, a certain number of probes are required. Below we will prove that over a time interval T, to maintain ply which is $O\left(\Delta\log\left(\frac{n}{\Delta}\right)\right)$ requires at least $\frac{2\Delta|T|\log\left(\frac{n}{\Delta}\right)}{\text{diameter}(G_i)}$ probes. Thus we call any group which receives more than $\frac{2\Delta|T|\log\left(\frac{n}{\Delta}\right)}{\text{diameter}(G_i)}$ probes during the time period T "good", and all other groups "bad". We will show that there exists a bad group, and that bad groups must have high ply.

First we argue there is a bad group. Assume to the contrary there is not. Then each group uses at least $\frac{2\Delta|T|\log\left(\frac{n}{\Delta}\right)}{\text{diameter}(G_i)}$ probes. To bound the total number of probes used we first need a bound on diameter(G_i). Recall the j^{th} set from our construction of Δ-arithmetically spaced entities has supremum $m_k = 2k^2\Delta + k$. Since each group covers $\frac{\log\left(\frac{n}{\Delta}\right)}{2}$ of these sets, the largest such set in G_i has diameter

$$\left(4\frac{i\log\left(\frac{n}{\Delta}\right)}{2} - 2\right)\Delta \leq 2i\log\left(\frac{n}{\Delta}\right)\Delta$$

Thus we get the following bound on the diameter of G_i

$$\begin{aligned}\text{diameter}(G_i) &\leq \left(\frac{\log\left(\frac{n}{\Delta}\right)}{2}\right)\left(2i\log\left(\frac{n}{\Delta}\right)\Delta\right)\\ &\leq \Delta\log^2\left(\frac{n}{\Delta}\right)i\end{aligned}$$

One important inequality for our next derivation is $2\log\left(\frac{n}{\Delta\log\left(\frac{n}{\Delta}\right)}\right) > \log\left(\frac{n}{\Delta}\right)$. The justification for that inequality is as follows. Note that $\frac{n}{\Delta} > \log^2\left(\frac{n}{\Delta}\right)$, since $n > \Delta$ by definition. Thus:

$$\frac{n}{\Delta} > \log^2\left(\frac{n}{\Delta}\right)$$

$$\leftrightarrow \frac{n}{\Delta \log^2\left(\frac{n}{\Delta}\right)} > 1$$
$$\leftrightarrow \left(\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}\right)^2 > \frac{n}{\Delta}$$
$$\leftrightarrow 2\log\left(\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}\right) > \log\left(\frac{n}{\Delta}\right)$$

Using these bounds to sum over all the groups we calculate that the total number of probes used is:

$$\begin{aligned}
\sum_{i=1}^{\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}} \frac{2\Delta|T|\log\left(\frac{n}{\Delta}\right)}{\text{diameter}(G_i)} &> 2\Delta|T|\log\left(\frac{n}{\Delta}\right) \sum_{i=1}^{\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}} \frac{1}{\Delta \log^2\left(\frac{n}{\Delta}\right) i} \\
&= \frac{2|T|}{\log\left(\frac{n}{\Delta}\right)} \sum_{i=1}^{\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}} \frac{1}{i} \\
&> \frac{2|T|}{\log\left(\frac{n}{\Delta}\right)} \log\left(\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}\right) \\
&> |T|
\end{aligned}$$

The last line follows from the above inequality.

This is more than $|T|$ probes over T. This is a contradiction since we only probe once for each time step. Thus, there must be some group G_k which uses fewer than $\frac{2\Delta|T|\log\left(\frac{n}{\Delta}\right)}{\text{diameter}(G_k)}$ probes. That means that in G_k at least $\frac{\Delta \log\left(\frac{n}{\Delta}\right)}{2}$ entities are probed fewer than $\frac{4|T|}{\text{diameter}(G_k)}$ times.

Let $\mathscr{G}_k$ be the smallest interval containing all the entities in G_k, that is $\mathscr{G}_k = \arg\min_I\{|I| : \forall e_j \in G_k\ l_j \in I\}$. Note that $|\mathscr{G}_k| = \text{diameter}(G_k)$. Also, note that the diameter of G_k is less than the diameter of $G_{n/\Delta \log\left(\frac{n}{\Delta}\right)}$, which is less than $\frac{n}{\Delta \log\left(\frac{n}{\Delta}\right)}\Delta(\log^2\left(\frac{n}{\Delta}\right)) = n\log\left(\frac{n}{\Delta}\right)$, by our above bound on the diameter of G_i. By Corollary 8, since $|T| > \text{diameter}(G_k)$, for each entity e_j that is probed fewer than $\frac{4|T|}{\text{diameter}(G_k)}$ times $\sigma(j,\mathscr{G}_k,T) > \frac{|T|\text{diameter}(G_k)}{16}$. Since $\sigma(j,\mathscr{G}_k,T)$ is the sum over all times in T of the overlap between e_j's uncertainty region, and $\mathscr{G}_k$, in order to calculate the average ply we divide the sum of the $\sigma(j,\mathscr{G}_k,T)$ values for a given group by the length of the time interval, and the diameter of G_k. Since at least half

of the entities in G_i were probed fewer than $\frac{4|T|}{\text{diameter}(G_k)}$ the average ply is greater than

$$\frac{1}{|T|\text{diameter}(G_k)} \frac{\Delta|T|\log\left(\frac{n}{\Delta}\right) \text{ diameter}(G_k)}{64} = \frac{\Delta\log\left(\frac{n}{\Delta}\right)}{32}$$

So any algorithm must have average ply at least $\frac{\Delta\log\left(\frac{n}{\Delta}\right)}{32}$, which is $\Omega(\Delta\log\left(\frac{n}{\Delta}\right))$. □

This shows that when expressed as a function of the intrinsic ply the best ply we can hope for is $\Omega(\Delta\log\left(\frac{n}{\Delta}\right))$.

4.3 Our Algorithm

We saw in the previous section that, for a set of n static entities, no algorithm can maintain ply less than a constant times $\Delta\log\left(\frac{n}{\Delta}\right)$, where Δ is the intrinsic ply, over a sufficiently long interval. This shows that intrinsic ply is an unfair measuring stick for the effectiveness of our algorithm. We define a measure of entity crowdedness, called the "optimal blanket value", in order to produce a more manageable comparison for our algorithm. Any algorithm must have average ply which is at least a constant factor of the optimal blanket value, over a sufficiently long interval. We also describe an algorithm, the Static Rounded Blankets algorithm, that maintains ply less than twice the optimal blanket value, at all time steps.

In the previous section breaking up the set of entities into smaller groups, and then arguing about the number of probes each of those groups requires to maintain low ply, was shown to be an effective way to argue a certain ply is unavoidable. It also suggests a simple formula for probing entities in the given example. The ones which were in the more tightly packed groups require more probes, while the ones that are more spread out are probed less frequently. It makes sense to prioritize entities that are closely packed together since they lead to high ply more quickly. What is not clear from the example is how to decide how large a group one should be considering in general. To this end, we introduce a measure of local clustering, which we call an "x-blanket". The x-blanket is a simple measure of how close one specific entity is to its neighbours. Our algorithm, Static Rounded Blankets, is a simple periodic schedule, where each entity's period is relative to its x-blanket volume.

4.3.1 x-Blankets

In this subsection we introduce the concept of an "x-blanket", and related notions. x-blankets form a core part of the way we formulate our algorithm. They are a measure of how tightly crowded entities are. Before we define the x-blanket, let us look at an associated set of entities, $\Gamma_i^x(t)$. Informally $\Gamma_i^x(t)$ is the set of the x entities closest to e_i at time t. These will be the basis of e_i's x-blanket and are referred to as the entities "covered" by e_i's x-blanket.

Definition 13. *For any given $x \in \mathbb{Z}$, index $i \in \{1,...,n\}$, and time t, the set $\Gamma_i^x(t)$ contains e_i and $x-1$ other entities, such that for any entity $e_j \in \Gamma_i^x(t)$ and $e_n \notin \Gamma_i^x(t)$, $|l_i(t) - l_j(t)| \leq |l_i(t) - l_n(t)|$. That is at time t all other entities are at least as far from e_i as any given entity in $\Gamma_i^x(t)$.*

Note that if there are several entities which are the same distance from e_i, there may be multiple different sets which satisfy the requirements of the above definition for $\Gamma_i^x(t)$. In such a case we choose the set which contains the entities with the lowest indices to be $\Gamma_i^x(t)$, arbitrarily.

Now we turn our attention to the main construct, the x-blanket. Informally an entity e_i's x-blanket is the largest interval centered at e_i's location, which contains at most x entities.

Definition 14. *For any given $x \in \mathbb{Z}$, index $i \in \{1,...,n\}$, and time t, e_i's x-blanket at time t, $b_i^x(t)$, is the largest open ball centered at $l_i(t)$ that only contains entities in $\Gamma_i^x(t)$. This ball is the "(true) x-blanket", of e_i at time t.*

Note that this interval does not contain all of $\Gamma_i^x(t)$ when an entity in $\Gamma_i^x(t)$ is the same distance from $l_i(t)$ as some entity that is not in $\Gamma_i^x(t)$. Thus e_i's x-blanket wouldn't change for a different choice of $\Gamma_i^x(t)$, since any entity that is the same distance from $l_i(t)$ as an entity in $\Gamma_i^x(t)$ are on the boundary of $b_i^x(t)$; their inclusion in $\Gamma_i^x(t)$ would lead to the same $b_i^x(t)$. Also note the entity which is x^{th} furthest from $l_i(t)$ (discounting e_i) is on the boundary of $b_i^x(t)$. Hence the volume of $b_i^x(t)$ is simply twice the distance from $l_i(t)$ to the entity which is x^{th} furthest from e_i.

The above definitions were given with a dependence on time because this generality will be required for the dynamic case. In the static case the x-blanket is invariant with respect to time, since entities don't move. Thus we will drop the dependence on t in our notation for the static case. We use $|b_i^x|$ to refer to the volume of the blanket b_i^x.

Definition 15. *If x is the smallest integer such that $\sum_i \frac{1}{|b_i^x|} < \delta$ then x is called the "optimal blanket value for probe density δ".*

The value δ is called the "probe density" because it is related to the fraction of time steps in which we need to probe entities in order to keep their uncertainty regions contained within their blankets. The optimal blanket value can be thought of the smallest value that can be used to generate blankets for a given set of entities, without needing more than the appropriate probe density to keep the uncertainty regions inside those blankets. The concept of optimal blanket value is important for determining what blanket value we should use in our algorithm. Considering a probe density that is less than one can be helpful to allow flexibility in the timing of the probes, so that we can avoid cases where multiple entities require probes simultaneously.

4.3.2 Upper Bound on Blanket Size

We show that the optimal blanket value for any set of n static entities is related to the intrinsic ply of the entities. In particular, we show that the optimal blanket value for a given probe density δ is $O(\frac{\Delta}{\delta}\log(\frac{n}{\Delta}))$.

Theorem 10. *For any given set of n static entities the optimal blanket value for probe density δ is $O(\frac{\Delta}{\delta}\log(\frac{n}{\Delta}))$.*

Proof. Let $\mathscr{E}$ be an arbitrary set of static entities with intrinsic ply Δ. Let $A = \{A_1, ..., A_n\}$ be an uncertainty realization for $\mathscr{E}$ with ply Δ. We want to show the optimal blanket value is $O(\frac{\Delta}{\delta}\log(\frac{n}{\Delta}))$. This will be achieved by showing $\frac{\Delta}{\delta}\log(\frac{n}{\Delta})$ is greater than the largest value of x for which the sum $\sum_{i=1}^{n} \frac{1}{|b_i^x|}$ is less than the probe density δ. We can assume that $x > 8\Delta$ without loss of generality, since if $x \leq 8\Delta$, then it is in $O(\frac{\Delta}{\delta}\log(\frac{n}{\Delta}))$

In order to show the optimal blanket value is less than $\frac{\Delta}{\delta}\log(\frac{n}{\Delta})$, we will provide an upper bound on the sum $\sum_{i=1}^{n} \frac{1}{|b_i^x|}$. To do this we need a lower bound on the volume of the b_i^x's. For any given e_i the volume of b_i^x is determined by the location of the x^{th} closest entity to e_i (excluding e_i itself). Naïvely the way to minimize the volume of b_i^x would be to simply place x entities right on top of e_i. However we also know that the uncertainty realization A has ply Δ, hence at most Δ entities can be at the same location. As shown belov, the minimum width for the b_i^x will be achieved by creating Δ "layers" of entities, each with ply 1, which are as close to the same width as possible.

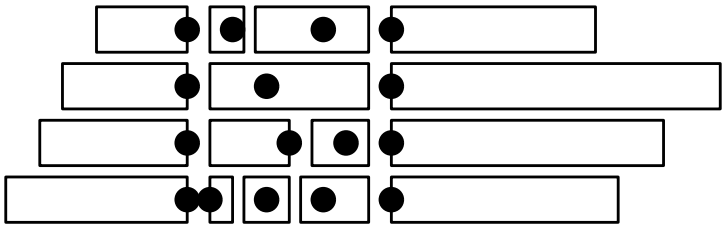


Figure 4.2: A graphic displaying the ideas behind our bound on $|b_i^x|$. Here Δ is four. The black circles represent the static locations of the entities, and the white bars represent their associate intervals in A. The entities have been split into four layers to make it easier to see. The eight extremal entities have large intervals, which do not contribute to the size of $|b_i^x|$. The volume of b_i^x can be bounded by summing over the inverse volumes of the white regions of the eight interior entities.

Consider partitioning the Γ_i^x into Δ subsets, $L_1, ..., L_\Delta$, such that the ply the uncertainty realization A gets for L_i is 1. We will refer to these subsets as layers. More specifically for all e_j, e_k in L_i the associated intervals of the uncertainty realization do not overlap, that is $A_j \cap A_k = \emptyset$. The diameter of L_i is the distance from $l_{min}(i) = \min_{L_i}\{l_j : e_j \in L_i\}$ to the similarly defined $l_{max}(i)$. This is at least the sum $\sum_{e_j \in L_i} |A_j|$, minus any of the volume of the A_j's which lies outside of $[l_{min}(i), l_{max}(i)]$. That sum is minimized when the two entities in L_i with the largest $|A_j|$ values are placed at $l_{min}(i)$ and $l_{max}(i)$, so the entirety of their A_j's can reside outside of $[l_{min}(i), l_{max}(i)]$. Thus in each layer we can ignore the A_j values of the two extremal entities. Let $\mathscr{X}_{i,j}$ be the two extremal entities of layer L_j. Then let $\mathscr{X}_i$ be the set of extremal entities, that is $\mathscr{X}_i = \bigcup_j \mathscr{X}_i, j$. For a helpful visualization see Figure 4.2.

The layer which has the longest width is the only one that affects the length of b_i^x. The length of the longest layer is longer than the average layer length. Hence

$$|b_i^x| > \frac{1}{\Delta} \sum_{e_j \in \Gamma_i^x \setminus \mathscr{X}_i} |A_j|$$

Now we move to looking at the sum of the inverse values as a whole. From above we can derive

$$\sum_{i=1}^{n} \frac{1}{|b_i^x|} < \Delta \sum_{i=1}^{n} \frac{1}{\sum_{e_j \in \Gamma_i^x \setminus \mathscr{X}_i} |A_j|}$$

Note that each e_j can be covered by at most $2x$ entities' x-blankets. This means e_j is a member of Γ_i^x for at most $2x$ different entities. So each $|A_j|$ value shows up in the sum

$$\sum_{e_j \in \Gamma_i^x \setminus \mathscr{X}_i} |A_j|$$

at most $2x$ times. Note the sum

$$\sum_{i=1}^{n} \frac{1}{\sum_{e_j \in \Gamma_i^x} |A_j|}$$

is maximized if the smallest $\frac{n}{2x}$ A_j's are used $2x$ times each. If you don't use the smallest $\frac{n}{2x}$ A_j's $2x$ times each you simply replace one of the instances of a larger one with one for which $|A_j| \leq \frac{n}{2x}$ that has not been used $\frac{n}{2x}$ and the sum increases.

Each entity needs x entities assigned to its Γ_i^x. Note that as the volume of the uncertainty regions of entities in $\mathscr{X}_i$ don't contribute to the sum, their volume is unimportant, so only the volume of $x-2\Delta$ entity's A_j's need to be considered in our bound on b_i^x. We will assign one of the groups $(1,...,x-2\Delta),(x-2\Delta+1,...,2x-4\Delta),...,((\frac{n}{2x}-1)(x-2\Delta)+1,...,\frac{n}{2x}(x-2\Delta))$ to each entity, each group will be assigned $2x$ times.

In order to see this allocation leads to the largest possible sum, we consider an abstraction of the problem. There are n different variables z_i that are the sum of γ distinct positive integers, that is z_i is equal to the sum $y_{i_1}+...+y_{i_\gamma}$ where each y_{i_j} is an integer. We will refer to the set of integers in z_i's sum as Y_i, that is $Y_i = \{y_{i_1},...,y_{i_\gamma}\}$. We can choose which integers are in z_i's sum, but each integer can only be used for a limited number of sums. Our goal is to maximize the sum $\sum_{i=1}^{n} \frac{1}{z_i}$. Now we will show that assigning the smallest y's to the smallest z's will lead to the optimal solution. Assume that we have a system that satisfies the constraints as described above. We can assume without loss of generality that $z_1 \leq z_2 \leq ... \leq z_n$. Assume that for some i, j such that $i < j$ there is an integer α in Y_j that is not in Y_i that is less than an integer β which is in Y_i but not Y_j. Then by swapping α and β so that α is in Y_i and β is in Y_j we increase the inverse sum of the z_i's by

$$\frac{1}{z_i+(\alpha-\beta)} - \frac{1}{z_j+(\beta-\alpha)} = \frac{z_j - z_i}{[z_i+(\alpha-\beta)][z_j+(\beta-\alpha)]}$$

Since we know $z_j > z_i$, $z_j > \alpha$ and $z_i > \beta$ this quantity is positive, and hence the swap increases the inverse sum. Hence since we are looking for the optimal blanket value for probe frequency δ we can assume the values are distributed in

groups as discussed above. From this we produce the following bound, in which we use the quatity $\tilde{x} = x - 2\Delta$. Note that $\tilde{x} > \frac{3x}{4}$ since $x > 8\Delta$:

$$
\begin{aligned}
\delta &< \sum_{i=1}^{n} \frac{\Delta}{\sum_{e_j \in \Gamma_i^x \setminus \mathscr{X}_i} |A_j|} \\
&< 2x\Delta \sum_{i=1}^{\frac{n}{2x}} \frac{1}{\sum_{j=(i-1)\tilde{x}}^{i\tilde{x}} j} \\
&< 4x\Delta \sum_{i=1}^{\frac{n}{2x}} \frac{1}{(i\tilde{x})^2 + i\tilde{x} - [(i-1)\tilde{x}]^2 - [(i-1)\tilde{x}]} \\
&= 4x\Delta \sum_{i=1}^{\frac{n}{2x}} \frac{1}{(i\tilde{x})^2 + i\tilde{x} - (i\tilde{x})^2 + 2i\tilde{x}^2 - \tilde{x}^2 - i\tilde{x} + \tilde{x}} \\
&= 4x\Delta \sum_{i=1}^{\frac{n}{2x}} \frac{1}{(2i-1)\tilde{x}^2 + \tilde{x}} \\
&< 4x\Delta \sum_{i=1}^{\frac{n}{2x}} \frac{1}{(2i-1)\tilde{x}^2} \\
&= \frac{4x\Delta}{\tilde{x}^2} \sum_{i=1}^{\frac{n}{2x}} \frac{1}{2i-1} \\
&< \frac{4x\Delta}{(\frac{3x}{4})^2} \sum_{i=1}^{\frac{n}{2x}} \frac{1}{2i-1} \\
&< \frac{8\Delta}{x} \sum_{i=1}^{\frac{n}{2x}} \frac{1}{2i-1} \\
&= \frac{8\Delta}{x} \left(\sum_{i=1}^{\frac{n}{x}} \frac{1}{i} - \sum_{j=1}^{\frac{n}{2x}} \frac{1}{2j} \right) \\
&< \frac{8\Delta}{x} \sum_{i=1}^{\frac{n}{x}} \frac{1}{i} \\
\leftrightarrow x &< \frac{8\Delta}{\delta} \sum_{i=1}^{\frac{n}{x}} \frac{1}{i} \\
&\in O\left(\frac{\Delta}{\delta} \log\left(\frac{n}{x} \right) \right) \\
&\in O\left(\frac{\Delta}{\delta} \log\left(\frac{n}{\Delta} \right) \right)
\end{aligned}
$$

Thus the optimal blanket value for probe density δ is $O(\frac{\Delta}{\delta}\log(\frac{n}{\Delta}))$. □

Although our main motivation to work with the optimal blanket value is to maintain ply close to any other algorithm's average ply, this result implies that if we have an algorithm which maintains ply equal to the optimal blanket value, we can maintain ply which is $\Delta\log(\frac{n}{\Delta})$.

4.3.3 Rounded Blankets Algorithm

Previously we introduced the notion of "x-blankets", now we want to show how we can use that tool to create an algorithm which maintains low ply. The simple idea behind the algorithm is to query entities when their uncertainty region is the same as their x-blanket. First we need to ensure that the x-blankets are large enough that probing with this frequency doesn't require more probes than there are time units. Probing an entity every time its uncertainty region is the same as its x-blanket means probing it once every $|b_i^x|$ time steps, since the uncertainty region grows by 1 each time step. So as a simple first step we ensure that the sum $\sum_{i=1}^{n}\frac{1}{|b_i^x|}$ is less than 1.

However, even if $\sum_{i=1}^{n}\frac{1}{|b_i^x|}$ is less than 1, it might be impossible to query the entities at the appropriate times. For example, two entities might have their uncertainty regions grow to be the same as their x-blankets at the same time. Thus we will consider rounding the values of the blanket volume in order to make sure all of the periods are in sync. But, if we use the blankets designated by the optimal blanket value for probe density 1, then we cannot probe with a frequency greater than the inverse of the blanket volume, since then the sum of the frequencies will be greater than one. On the other hand, we cannot probe with a frequency less than the inverse of the blanket volumes, since then the ply could be higher than we expect. Instead, we will use the optimal blankets of probe density $\frac{1}{2}$, since this will give us room to round the blanket volumes to the nearest power of two. This will make scheduling probes for the entities simple.

The algorithm, which is referred to as the "Rounded Blankets Algorithm", is an algorithm which produces a recursive round robin schedule. A recursive round robin schedule is a schedule that splits its probes evenly among subsets of entities. These subsets partition the set of entities, that is no entity is in more than a single subset. Each subset splits its probes in an identical way to the main set, that is evenly amongst some partition of its elements. In this way round robin is applied

to each set of entities recursively. It is important to note that even though the probes are split evenly among the subsets, the subsets may not contain the same number of entities. Having an unbalanced partition ensures some entities will be probed more often than others. The notion of recursive round robin is discussed in more depth in "Scheduling Techniques for Media-on-Demand" by Amotz Bar-Noy, Richard E. Ladner, and Tami Tamir [3].

The algorithm has two main steps. The first is producing new rounded frequencies, which will be powers of two, and the second is constructing a schedule for these new frequencies. The algorithm for schedule construction is presented in "The Scheduling of Maintenance Service" by Shoshana Anily, Ceilia A. Glass, and Rafael Hassin [1].

The Rounded Blankets algorithm is as follows:

1. For a given set of n entities calculate $x =$ the optimal blanket value for probe density $\frac{1}{2}$ (i.e. the smallest integer such that $\sum_{i=1}^{n} \frac{1}{|b_i^x|} < \frac{1}{2}$).
2. Set $q_i = 2^{\lfloor \lg |b_i^x| \rfloor}$.
3. Query entity e_i periodically, with period q_i

Now that we have an algorithm for probing entities we need to produce a bound on its performance, that is we want a bound on the ply the Rounded Blankets algorithm achieves. We will show the ply that the Rounded Blankets algorithm produces is always bounded by the blanket value we use.

Now to show that our algorithm will never have ply greater than some constant times the optimal blanket value. This follows fairly directly from the fact that our blankets never cover more than x entities.

Theorem 11. *For all sets of n static entities, and all time steps t the Rounded Blankets algorithm maintains ply $O(x)$, where x is the optimal blanket value for probe density $\frac{1}{2}$.*

Proof. Note that the q_i values are at least half the volumes of the associated x-blanket, $|b_i^x|$, since they are the largest power of two less than or equal to$|b_i^x|$. Thus the inverse sum of the q_i values will be less than one. Because of this fact, and the fact they are powers of two, our q_i satisfy the conditions on the τ_i in the hypothesis of lemma 6.2 in [1]. Thus we can build a schedule in which each e_i is probed once every q_i time steps by following their algorithm.

Since the algorithm probes e_i once every q_i time steps, it probes entities before their uncertainty region exceeds their blanket. Thus the depth of any given point is bounded by the number of blankets that contain that point. Hence the ply is bounded by the point which is contained in the most blankets.

The maximum number of blankets a point can be contained within is $2x$. We will prove this via a contradiction. Assume to the contrary that $2x+1$ x-blankets cover the point p. Let $\mathscr{E}$ be the set of all entities whose x-blankets cover p. Let $\mathscr{E}_L$ be the subset of $\mathscr{E}$ such that for any entity, e_L, in $\mathscr{E}_L$ its position l_L is at most p. Similarly let $\mathscr{E}_R$ be the subset whose locations are at least p. Assume, without loss of generality, $|\mathscr{E}_L| \geq |\mathscr{E}_R|$. Then $\mathscr{E}_L$ must contain at least $x+1$ entities. Note that the entity in $\mathscr{E}_L$ which has the minimal location has a blanket which covers all of the entities in $\mathscr{E}_L$, since it covers p. Thus it covers $x+1$ entities, which is a contradiction with the definition of x-blanket. Hence the ply at any given point can be at most $2x$. And the Rounded Blanket algorithm's ply is $O(x)$.

□

So the Rounded Blankets Algorithm never has ply worse than a constant factor of the optimal blanket value. Since we have a bound on the intrinsic ply with respect to the optimal blanket value, we can bound our algorithm's performance with respect to the intrinsic ply.

Corollary 12. *For any set of static entities the Rounded Blankets algorithm has ply which is* $O(\Delta \log(\frac{n}{\Delta}))$.

Proof. Let x be the optimal blanket value for probe density $\frac{1}{2}$. From Theorem 10 we know that x is $O(\frac{\Delta}{\delta}\log(\frac{n}{\Delta}))$, since δ is constant $O(\Delta\log(\frac{n}{\Delta}))$. From Theorem 11 we know the Rounded Blanket algorithm's ply is $O(x)$. Therefore the Rounded Blanket algorithm ply is $O(\Delta\log(\frac{n}{\Delta}))$. □

This shows that our algorithm can maintain ply which is $O(\Delta\log(\frac{n}{\Delta}))$ at all time steps. The intrinsic ply can be thought of as the minimal ply any algorithm can achieve at any time. Thus, the ply obtained by the Rounded Blanket's algorithm at every time step is at most $O(\Delta\log(\frac{n}{\Delta}))$ times the smallest ply any algorithm can obtain at any time step. The fact that Rounded Blanket's ply is $O(\Delta\log(\frac{n}{\Delta}))$, and not the $\Omega(\sqrt{n\Delta})$ bound achieved by Round Robin shows that there is an advantage to looking at how close entities are to each other. Note that, since $\log(\frac{n}{\Delta}) < \sqrt{\frac{n}{\Delta}}$, $\Delta\log(\frac{n}{\Delta}) < \sqrt{n\Delta}$, and thus the Rounded Blankets Algorithm performs better than Round Robin, in their respective worse cases.

Next we will show that no algorithm, which runs for a sufficiently long time, can achieve average ply better than $\Omega(\delta x)$, where x is the optimal blanket value for probe density δ. Hence the maximum ply achieved by the Rounded Blankets algorithm is within a constant factor of the average ply achieved by any other algorithm, over a sufficiently large interval of time. For the purpose of arguing results about the Rounded Blankets algorithm we are only interested in the case where δ is $\frac{1}{2}$, but the generalization to other blanket values will be useful in our discussion of the dynamic case. This theorem is a stronger version of Theorem 9.

Theorem 13. *For all sets of static entities any algorithm has average ply $\Omega(\frac{x}{\delta})$, where x is the optimal blanket value for probe density δ, over any time interval T, such that $|T| > \delta \max_i |b_i^x|$.*

Proof. To prove this theorem we will consider two cases. Either entities are being probed too frequently, leading us to probe more times than we are allowed, or they are being probed too infrequently, thus leading to large uncertainty regions and high ply.

Let x be the optimal blanket value for probe density δ, and consider a time interval T. Let p_i be the number of times e_i is probed over this interval. The first case is that for all e_i, the majority of the entities in $\Gamma_i^{\frac{x}{2}}$ are probed at least $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times. Our second case is that there is some entity e_i such that the majority of the entities in $\Gamma_i^{\frac{x}{2}}$ are probed fewer than $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times. Note that the choice of constant is irrelevant in the second case, since probing fewer than any constant factor of $|b_i^{\frac{x}{2}}|$ will give ply on the order of x, as will be discussed later in the proof, however it is vital for the first argument that the constant is large enough to cause the probe density to be too high.

In the first case, for all e_i, the majority of the entities in $\Gamma_i^{\frac{x}{2}}$ are probed more than $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times. Note the total number of probes used on entities in a given $\Gamma_i^{\frac{x}{2}}$ will be at least the number of probes used on the $\frac{x}{4}$ who are probed more than $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times, that is

$$\begin{aligned}\sum_{e_j \in \Gamma_i^{\frac{x}{2}}} p_j &> \frac{x}{4} \frac{4|T|}{\delta|b_i^{\frac{x}{2}}|} \\ &= \frac{x|T|}{\delta|b_i^{\frac{x}{2}}|}\end{aligned}$$

$$\rightarrow \sum_{i=1}^{n} \frac{x|T|}{\delta|b_i^{\frac{x}{2}}|} < \sum_{i=1}^{n} \sum_{e_j \in \Gamma_i^{\frac{x}{2}}} p_j$$

Also note that each entity can be included in at most x of the $\Gamma_i^{\frac{x}{2}}$'s, since there are at most x entities which it is within $\frac{x}{2}$ entities of. Thus

$$\begin{aligned} \sum_{i=1}^{n} \sum_{e_j \in \Gamma_i^{\frac{x}{2}}} p_j &< x \sum_{k=1}^{n} p_k \\ &= x|T| \end{aligned}$$

Combining the above equations we derive

$$\begin{aligned} \sum_{i=1}^{n} \frac{x|T|}{\delta|b_i^{\frac{x}{2}}|} &< x|T| \\ \leftrightarrow x|T| &> \frac{x|T|}{\delta} \sum_{i=1}^{n} \frac{1}{|b_i^{\frac{x}{2}}|} \\ &> x|T| \end{aligned}$$

The last line follows since x is the optimal blanket value for probe density δ, which means that for any smaller value y $\sum_{i=1}^{n} \frac{1}{|b_i^y|} > \delta$. $x|T| > x|T|$ is a contradiction, thus we can conclude that we are always in the second case.

In the second case there is some entity e_i such that the majority of the entities in $\Gamma_i^{\frac{x}{2}}$ are probed fewer than $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times. By Corollary 8, for each entity e_i which is probed fewer than $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times over a time period that is longer than $\frac{\delta|b_i^{\frac{x}{2}}|}{4}$, which $|T|$ is by definition, $\sigma(i, b_i^x, T) > \frac{\delta|T||b_i^{\frac{x}{2}}|}{16}$. Then the total area for all entities in Γ_i^x is at least the sum of the areas of the entities which were probed fewer than $\frac{4|T|}{\delta|b_i^{\frac{x}{2}}|}$ times. Thus

$$\sum_{\{e_j \in \Gamma_i^x\}} \sigma(j, b_i^{\frac{x}{2}}, T) \geq \frac{x}{4} \frac{\delta|T||b_i^{\frac{x}{2}}|}{16}$$

Averaging over the blanket and the time interval, the average ply is greater than $\frac{\delta x|T||b_i^{\frac{x}{2}}|}{64|T||b_i^{\frac{x}{2}}|} = \frac{\delta x}{64}$. This shows the average ply is $\Omega(\delta x)$.

Since we are always in the second case all algorithms have average ply which is $\Omega(\delta x)$, over a time interval longer than $\delta \max_i |b_i^x|$. □

Now that we know any algorithm has average ply that is within a constant factor of x, we can argue the Rounded Blankets algorithm performs well against other algorithms.

Theorem 14. *Given any set of entities and any time interval T, such that $|T| > \frac{\max_i |b_i^x|}{2}$ the Rounded Blankets algorithm's maximum ply over T is $O(x)$ and any other algorithm's average ply over T is $\Omega(x)$ where x is the optimal blanket value for probe density $\frac{1}{2}$.*

Proof. The Rounded Blankets algorithm always has ply $O(x)$ by Theorem 11. Because $|T| > \delta \max_i |b_i^x|$ any algorithm has average ply $\Omega(\delta x)$ by Theorem 13, since $\delta = \frac{1}{2}$ this is $\Omega(x)$. □

This last theorem is the most important result in this chapter. It shows that our algorithm, the Rounded Blankets algorithm, has maximum ply that is within a constant factor of any other algorithm's average ply. It relies on the fact that the optimal blanket value is representative of the best behaviour an algorithm can exhibit in the static case. Note that average ply is a very good measure of an algorithm's performance for the continuous ply minimization problem, since ply cannot change rapidly. At any given time step ply can be decreased by at most one, since only a single entity may be probed, and may at most double, if two large groups of uncertainty regions go from non-overlapping to overlapping (this bound on increase only holds in the one dimensional case). Thus the variance of any algorithm is bounded. Any algorithm whose ply does not change by more than a constant factor of the optimal blanket value will be within a constant factor of the Rounded Blanket algorithm's ply at all time steps. Thus in certain cases it may be possible to make stronger statements about the competitiveness of the Rounded Blankets algorithm.

Note that, even if an algorithm is designed to genrate the lowest ply at a specific point in time, an algorithm's ply can never be less than Δ. Thus the Rounded Blankets algorithm is always within a log factor of any other algorithm's ply at all time steps. It is unclear in which cases another algorithm might do better than a constant factor of x for a significant portion of the time.

We now turn our attention to the case where entities are moving, which we refer to as the "dynamic case". In this setting our algorithm no longer works, as

it assumes that the entities are stationary. However, by lowering the probe density we will give ourselves time to react to a changing environment, and probe entities frequently enough so that their uncertainty regions are never bigger than their blankets.

Chapter 5

Dynamic Case

The static case was a good launching point for our exploration of the continuous ply minimization problem. It allowed us to focus on a few of the factors that cause high ply, mainly how closely packed the entities are, without worrying about the uncertainty of our knowledge. In the dynamic case, where the entities are moving, there are many more factors that contribute to the success of our algorithm. We will show that, as long as entities are probed frequently enough, our view of how crowded the entities are is not too far off the true situation. This means that the tools we developed while exploring the static case will be applicable in the dynamic case, as long as we compensate for our lack of current information.

As we move into the dynamic case our model changes slightly, since entities are allowed to move, but many aspects of the problem remain the same. Our goal is still to maintain consistently low ply. As we saw in the static case, comparing against the intrinsic ply, Δ, at each time step is inherently unfair. In certain situations the minimum average ply any algorithm can maintain is $\Omega(\Delta \log(\frac{n}{\Delta}))$. The examples from the static case can be used to prove the result holds in the dynamic case, since the dynamic case does not prohibit static trajectories. However, it may be possible to set up a situation, in the dynamic case, for which no algorithm can maintain ply $O(\Delta_{min} \log(\frac{n}{\Delta_{max}}))$, where $\Delta_{min} = \min_t(\Delta(t))$ and $\Delta_{max} = \max_t(\Delta(t))$. In the static case we had a bound on the intrinsic ply with respect to the optimal blanket value. This bound is still applicable in the dynamic case, but the value of the intrinsic ply now depends on the time step. Thus we know the optimal blanket value at time t is at most $O(\Delta(t) \log(\frac{n}{\Delta}))$. A dynamic algorithm that is similar to our Rounded Blankets algorithm will have worst case performance that is tied to the maximum optimal blanket value over a time interval, because that value characterizes the maximum number of blankets which may overlap at any one time

over the interval. The obvious comparison with intrinsic ply would be with its minimum value, which could much less than the intrinsic ply at the time step when we have the maximum optimal blanket value. This means that intrinsic ply is an even worse measuring stick in the dynamic case. The movement of the entities will also make it harder for us to compare against other algorithms. In the static case we were able to bound all algorithm's ply with respect to the maximum optimal blanket value (which is the optimal blanket value at all time steps, since all the entities were static), but it is no longer clear in the dynamic case that any other algorithm's ply should be bounded by the maximum optimal blanket value in the dynamic case. We cannot even make the weaker claim that an arbitrary algorithm's ply is bounded by the average, or even minimum, optimal blanket value.

We would like to be able to argue about the competitiveness between our algorithm and any other algorithm. However our inability to argue a bound on other algorithm's ply means we have to aim for an easier target. So our goal will be based on the target in the static case, that is to maintain ply within a constant factor of the worst optimal blanket value over a given time period. This means our algorithm does not need to adjust to the changing blanket value during the running of the algorithm, which simplifies the process. Regrettably, it is impossible to know the worst optimal blanket value over a given time period without knowing the trajectories of the entities. So, to make the problem manageable, we assume there is an oracle that tells us which blanket value to use. Will we refer to this blanket value as χ.

With the aid of these simplifications we craft an algorithm for the dynamic case that has the same core ideas as the Rounded Blankets algorithm. However the movement of the entities, and our uncertainty of their current location, means it is impossible for an algorithm to know when a given entity's uncertainty region will expand beyond its associated x-blanket. We show that, as long as we probe frequently enough, our perception of an entity's x-blanket's volume cannot be too much larger than its true volume. Since the entities' rate of movement is bounded, x-blankets don't change too quickly over time. Thus we can calculate, by deducing the minimal possible volume of e_i's x-blanket based on the volume of its perceived x-blanket when it is probed, the first time e_i's uncertainty region could extend beyond its x-blanket. By reprobing e_i before this time we can ensure that e_i's uncertainty region does not cover more than x other entities. As we saw in the static case as long as any entity's uncertainty region never covers more than x other entities, our ply is less than $2x$.

5.1 The Bucket Algorithm

Now we will describe the algorithm that we will use to probe entities in the dynamic case. In many ways it is similar to the algorithm we used in the static case. However, we can no longer have a "static" probe sequence, because the demands of each entity will change as it moves. So we restrict our attention to the next time an entity will be probed. Our goal is to probe the entity before its uncertainty region expands beyond its x-blanket. We use the perceived x-blanket volume of the entity we are currently probing to calculate the first time such an event could occur. Since we don't have accurate data on the locations of entities we need to determine the first time at which the uncertainty region could expand beyond its x-blanket. By looking at the x-blanket sizes for small probe densities we are able to ensure there will be enough time to probe all the entities we want to probe.

Since we cannot rely on a static probe sequence we need a way to schedule the next probe time. The algorithm will use the notion of "buckets". In our application a "bucket" is a queue of entities which is associated with a given time interval. All the entities in a given bucket must be probed within the interval. We will split the time line into a set of intervals for each power of two, in the natural way. Each of these intervals is associated with a bucket.

Definition 16. *$\beta_{i,j}$ is the i^{th} bucket of length 2^j. It is associated with the interval $[i2^j+1,(i+1)2^j]$, and contains a queue of entities.*

All the buckets are aligned such that, given a pair of buckets that overlap, the smaller is wholly contained in the larger. This is described more formally in the following observation, note we use the notation $\beta_{i,j} \subset \beta_{k,l}$ to indicate the interval associated with $\beta_{i,j}$ is a subset of the interval associated with $\beta_{k,l}$

Observation 2. *For each pair of buckets $\beta_{i,j}$, $\beta_{k,l}$, such that $j < l$, if $\beta_{i,j} \cap \beta_{k,l} \neq \emptyset$, then $\beta_{i,j} \subset \beta_{k,l}$.*

For convience, we will use the notation $|\beta_{i,j}|$ to refer to the length of the associated interval, that is $|\beta_{i,j}| = 2^j$. When talking about a general bucket we drop the indicies i and j.

The Bucket algorithm is as follows:

Assume that we are given a value χ, which we use as our blanket value.

1. At time t probe an entity from the smallest bucket that overlaps t and contains unprobed entities.

2. Let j be the largest power of two which is less than $\frac{1}{22}|\hat{b}_i^{\chi}(t)|$, that is, $j = 2^{\left\lfloor \lg(\frac{1}{22}|\hat{b}_i^{\chi}(t)|) \right\rfloor}$. Place e_i in the queue of the first bucket of length j that contains times after t, that is $\beta_{\lceil t/j \rceil, j}$.

3. Increment t, and return to step 1.

We will present an example situation to illustrate how the algorithm works. A visual representation of the process is shown in Figure 5.1.

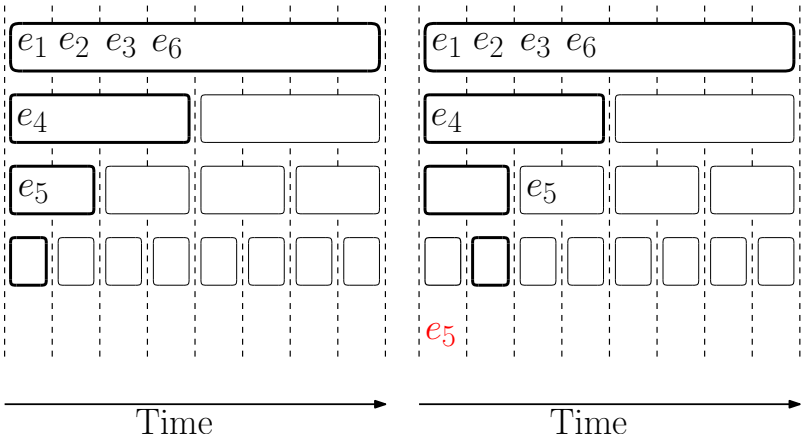
We focus on buckets of volume 8 or less which overlap a given time interval, this narrowing of perspective is necessary since there are infinitely many buckets. Our starting state is that there are no entities in the smallest bucket which overlaps the first time step, one in the bucket of volume two, one in the bucket with volume four, and four entities in the bucket of volume eight. Thus we choose to probe the entity which is in the bucket of volume two, and it is put in the bucket which overlaps time steps three and four.

After a few more probes we reach the situation in the third diagram, only one of the entities in the largest bucket has been probed, and both of the entities which were in smaller buckets are in the bucket of volume four which overlaps the fifth through eighth time steps. Since e_5 has been placed in a larger bucket its x-blanket must have grown. In the last figure, neither of these entities is probed in the last two time steps, so their x-blankets must have grown to the point where they were placed in the next bucket of volume eight (not shown in figure). Also, we can see in the last figure that there is one entity left in the bucket of size eight after the eight time steps have elapsed. This means our algorithm has failed in this case.

We want to argue that, in situations where such a failure occurs, the entities must have been located in a distribution that had a larger optimal blanket value than the χ we used. In order to argue about the cases where our algorithm fails, we need to argue our perceptions are close to the true situation, as we do in the next section.

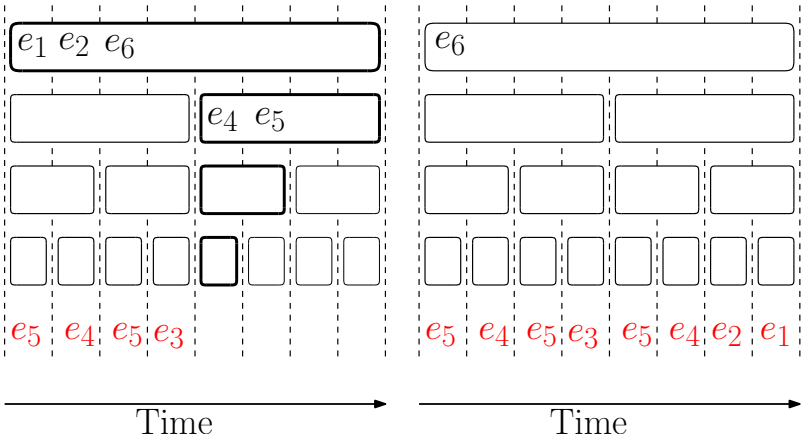
5.1.1 Perception Versus Reality

It will be important for us to distinguish between the true location of an entitiy and the algorithm's perception of that entity's location. Our algorithm will often need to know the volume of an entity's x-blanket, but it doesn't know the current location of the entities. Instead, the algorithm will use the last known location of an entity as a surrogate for its current location. To that end, we need notation to represent



(a) At the first time step the algorithm will probe e_5 as it is in the smallest non-empty "active" bucket.

(b) At the second time step e_4 is in the smallest non-empty "active" bucket.



(c) The situation after the algorithm has run for half of the time steps in the interval.

(d) The algorithm has run for the entire interval interval. Note there is still an entity in the top buckets queue, thus it has overflowed.

Figure 5.1: An illustration of the state of the bucket algorithm at four different time steps. The rectangles represent buckets, which contain a queue of entities inside them, and the dotted lines separate the interval into eight distinct time steps. Each bucket is associated with the time steps it overlaps. The bolded buckets are the buckets the algorithm is considering probing an entity from at that time step. The entities in red, along the bottom of the diagrams, indicate what the algorithm probed at the associated time step.

this situation. First recall that $p_i(t)$ is the last time e_i was probed. We refer to the location that the entity was at the last time it was probed, as its "perceived location" at the current time.

Definition 17. *For any entity e_i, its perceived location at time t, $\hat{l}_i(t)$, is the location of that entity when it was last queried. That is $\hat{l}_i(t) = l_i(p_i(t))$.*

The definitions of the perceived covered set and the perceived x-blanket are similar to their true counterparts, with the true locations of the points replaced by their perceived locations.

Definition 18. *For any given $x \in \mathbb{Z}$ and $i \in \{1,...,n\}$, $\hat{\Gamma}_i^x(t)$ is the set that contains the perceived locations of e_i and x other entities, such that for any entity $e_a \in \hat{\Gamma}_i^x(t)$ and $e_b \notin \hat{\Gamma}_i^x(t)$ $|\hat{l}_i(t) - \hat{l}_a(t)| \leq |\hat{l}_i(t) - \hat{l}_b(t)|$.*

Definition 19. *For any given $x \in \mathbb{Z}$ and $i \in \{1,...,n\}$, $\hat{b}_i^x(t)$ is the largest open ball centered at $\hat{l}_i(t)$ that contains $\hat{\Gamma}_i^x(t)$. This ball is the "perceived x-blanket" of e_i at time t.*

All of our uncertainty comes from movement over time. There are two distinct ways in which this manifests itself in our analysis. First the width of an entity's x-blanket may change over time due to the motion of all of the entities. Secondly, since we are looking at old location information, there is a difference between our perception of what the entity's x-blanket is at any time, and what it truly is. We need to argue that, even though we have flawed information about the location of entities, we still can deduce how crowded an entity is. Equivalently, we argue that our perception of an entity's x-blanket is always close to its true x-blanket. The argument will rely on us probing the entities with an appropriate frequency.

One fact that will help with this argument is the relationship between an entity's x-blanket volume, and the volume of the x-blanket of the entities in $\Gamma_i^x(t)$.

Observation 3. *For any entity e_j that is in $\Gamma_i^{x+1}(t)$ at time t, $|b_j^x(t)| \leq 2|b_i^x(t)|$, since e_j is in the closure of $b_i^x(t)$ and there are at least x other entities in the closure of $b_i^x(t)$*

Note that if we want a non-trivial lower bound for e_i's x-blanket volume at time t based on the uncertainty regions of the entities, then there needs to be a gap between e_i's uncertainty region and the uncertainty region which is x^{th} furthest from e_i, otherwise the x closest entities to e_i might all be on top of one another. In that situation $b_i^x(t)$'s volume would be zero, so we would not be able to discount the trivial bound. We know the x^{th} closest entity to e_i is a distance of $\frac{1}{2}|b_i^x(t)|$ away from e_i. Thus, if we want these entities to have no overlapping uncertainty regions

they must have been probed in the last $\frac{1}{2}|b_i^x(t)|$ time steps, since no entity could have moved further than $\frac{1}{4}|b_i^x(t)|$ in that time. From this we can conclude that any entity must be probed within a number of time steps less than half the volume of the smallest blanket it is in. As we see from Observation 3, the entities in $\Gamma_i^x(t)$ may have x-blankets which have twice the volume of $b_i^x(t)$. Thus, we can ensure that an entity has been probed within a number of time steps less than half the volume of any x-blanket it may be in, by probing it within a number of time steps equal to a quarter of its own x-blanket's volume.

We will ensure that we have probed all entities within a number of time steps equal to a quarter of their true x-blanket volume. We will show this is enough to prove the entities' true x-blanket volumes have not changed much since the last time they were probed.

Lemma 15. *If* $t - t' \leq \frac{1}{4}|b_i^x(t)|$ *then* $\frac{2}{3}|b_i^x(t')| \leq |b_i^x(t)| \leq 2|b_i^x(t')|$.

Proof. At time t entity e_i could be at most $d = \frac{1}{8}|b_i^x(t)|$ units away from where it was at t', since $t - t' \leq \frac{1}{4}|b_i^x(t)|$ and all entities have a maximum speed of $\frac{1}{2}$ units per time step. Similarly, any other entity e_j is at most d units from where it was at t'.

First we consider the lower bound on the blanket volume at time t. Recall that the volume of e_i's x-blanket is determined by the entity which is $x+1^{st}$ closest to e_i's location, since it is the largest open ball that contains at most x entities. At least the $x+1$ closest entities to e_i at time t, that is $\Gamma_i^{x+1}(t)$, are in the closure of $b_i^x(t)$. Thus they are all in the interval $[\min(b_i^x(t)) - d, \max(b_i^x(t)) + d]$ at time t'. e_i could have been anywhere in the interval $[l_i(t) - d, l_i(t) + d]$ at time t'. Since $l_i(t) - \min(b_i^x(t)) = \max(b_i^x(t)) - l_i(t) = \frac{|b_i^x(t)|}{2}$, this means that $|l_i(t') - \min(b_i^x(t'))| \leq \frac{|b_i^x(t)|}{2} + 2d$. The x-blanket volume is twice this length, and hence

$$|b_i^x(t')| \leq |b_i^x(t)| + 4d = \frac{3}{2}|b_i^x(t)|$$

Thus

$$\frac{2}{3}|b_i^x(t')| \leq |b_i^x(t)|$$

See Figure 5.2 for an illustration of the bounds on the location of points in $b_i^x(t)$ at time t'.

The proof of the upper bound is similar, except that we argue at most x entities (including e_i) could be in the interval $(\min(b_i^x(t)) + d, \max(b_i^x(t)) - d)$ at t'. This

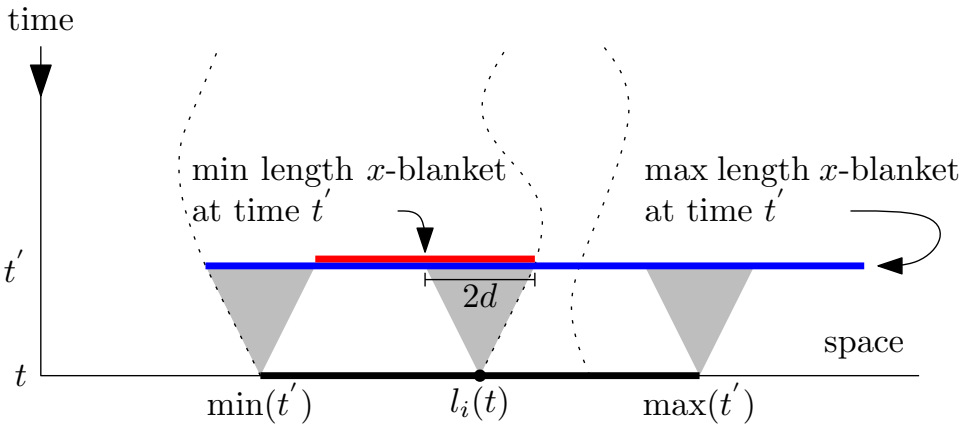


Figure 5.2: Bounds on the possible true x-blanket of e_i at time $t^{'}$ given the true x-blanket of e_i at time t. The outer grey cones illustrate the bound on the positions of the extremal points of $\Gamma_i^x(t)$ from the time $t - \frac{1}{4}|b_i^x(t)|$ up to t. The middle grey cone shows the potential locations of e_i.

is because any entity in this interval at $t^{'}$ will be in $b_i^x(t)$, which we know contains at most x entities. Hence

$$\begin{aligned} |b_i^x(t^{'})| &\geq |b_i^x(t)| - 4d \\ &= \frac{1}{2}|b_i^x(t)| \end{aligned}$$

Thus

$$|b_i^x(t)| \leq 2|b_i^x(t^{'})|$$

Combining these bounds we produce $\frac{2}{3}|b_i^x(t^{'})| \leq |b_i^x(t)| \leq 2|b_i^x(t^{'})|$. □

Next we will show that if we have probed all entities within a number of time steps equal to a quarter of their true x-blanket volume then their true x-blanket volume is relatively close to the perceived x-blanket volume for that entity at that time. The proof of this lemma is very similar to the proof of the previous lemma, so our discussion will be more terse in this proof.

Lemma 16. *For any time t, if for all entities e_j, $t - p_j(t) \leq \frac{1}{4}|b_j^x(t)|$ then for all entities e_i, $\frac{4}{7}|\hat{b}_i^x(t)| \leq |b_i^x(t)| \leq 4|\hat{b}_i^x(t)|$.*

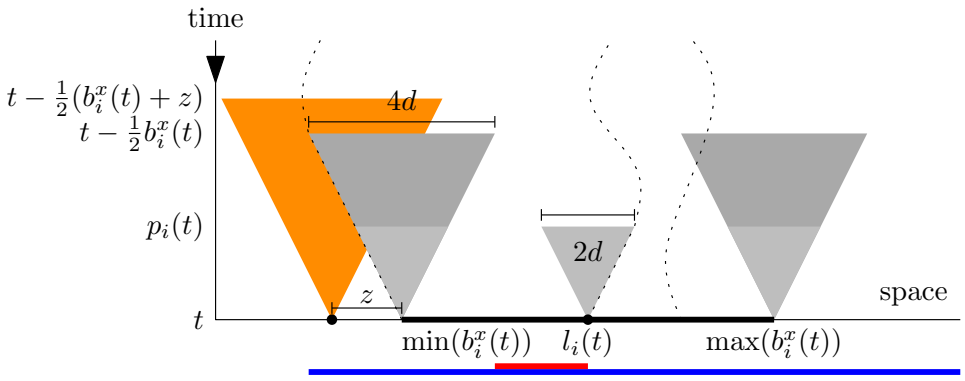


Figure 5.3: Bounds on the possible perceived x-blanket of e_i at time t given the true x-blanket of e_i at time t. Note that the perceived location of any point in $\hat{\Gamma}_i^x(t)$ may be based on its location up to $\frac{1}{2}b_i^x(t)$ time steps ago. The orange cone represents a bound on the location of an entity outside of $\hat{\Gamma}_i^x(t)$. Note that its right endpoint is necessarily further from e_i than the right endpoint of the leftmost grey triangle.

Proof. For any entity e_j that is in $\Gamma_i^{x+1}(t)$ at time t, $|b_j^x(t)| \leq 2|b_i^x(t)|$, by Observation 3. Thus, all of these entities have been probed in the past $\frac{1}{4}|b_j^x(t)| \leq \frac{1}{2}|b_i^x(t)|$ time steps. Hence at time t all of the entities in $\Gamma_i^{x+1}(t)$ could have moved a distance of at most $2d = \frac{1}{4}|b_i^x(t)|$ since they were last probed.

To prove the lower bound we will consider the largest $\hat{b}_i^x(t)$ could be. Note that all of the entities in $\Gamma_i^{x+1}(t)$ must have been in the interval $[\min(b_i^x(t)) - 2d, \max(b_i^x(t)) + 2d]$ the last time they were probed. Also note that e_i must have been within the interval $[l_i(t) - d, l_i(t) + d]$, since it was probed in the last $\frac{1}{4}|b_i^x(t)|$ time steps. Thus the maximum possible distance between $\hat{l}_i(t)$ and the perceived location of an entity in $\Gamma_i^{x+1}(t)$ is $\frac{|b_i^x(t)|}{2} + 3d = \frac{7}{8}|b_i^x(t)|$. Hence $\frac{4}{7}|\hat{b}_i^x(t)| \leq |b_i^x(t)|$.

The argument for the upper bound is similar. It is clear that the entity in $\Gamma_i^x(t)$ which is furthest from e_i at time t, e_x cannot be in the interval $(\min(b_i^x(t)) + 2d, \max(b_i^x(t)) - 2d)$ at $p_x(t)$. This would be enough to argue our result, except that we also must consider entities not in $\Gamma_i^x(t)$. We need to argue all of the entities that are not in $\Gamma_i^x(t)$ have perceived locations that are outside of $[\min(b_i^x(t)) + 2d, \max(b_i^x(t)) - 2d]$ at $p_i(t)$, since any entity inside that interval would shrink the volume of e_i's perceived x-blanket. We know for any entity e_o not in $\Gamma_i^x(t)$, e_o's x-blanket can at most be large enough to contain all of $b_i^x(t)$, since there are x entities in $b_i^x(t)$. Thus

$$
\begin{aligned}
|b_o^x(t)| &< 2|l_i(t) - l_o(t)| + |b_i^x(t)| \\
&= 2(z + b_i^x(t)) \\
\text{where } z &= \min\{|\min(b_i^x(t)) - l_o(t)|, |\max(b_i^x(t)) - l_o(t)|\}
\end{aligned}
$$

Informally z is the distance from e_o to $b_i^x(t)$. Hence $t - p_o(t) < \frac{z+b_i^x(t)}{2}$, which means that $|\hat{l}_o(t) - l_o(t)| < \frac{z+b_i^x(t)}{4} = \frac{z}{4} + 2d$. And so no entity which is outside of $b_i^x(t)$ at time t could have a perceived location at time t that is inside $[\min(b_i^x(t)) + 2d, \max(b_i^x(t)) - 2d]$, since the distance from e_o to that interval is $z + 2d$. This means the minimal distance between $\hat{l}_i(t)$ and entities in $\hat{\Gamma}_i^x(t)$ is at least $\frac{1}{4}|b_i^x(t)|$. Thus $|b_i^x(t)| \leq 4|\hat{b}_i^x(t)|$.

Combining the two bounds we generate $\frac{4}{7}|\hat{b}_i^x(t)| \leq |b_i^x(t)| \leq 4|\hat{b}_i^x(t)|$. □

The above results all rely on knowing that we have probed each entity recently with respect to its true x-blanket. It will not be possible for our algorithm to ensure this directly as our algorithm will be scheduling probes into the future based on the current perceived volume of a given entity's x-blanket. However we will show as long as an entity is probed within a number of time steps that is a small fraction of our current perception of an entity's x-blanket volume then it will never have been left unprobed for more than a quarter of its true x-blanket volume. Note that this is a bound on the volume of e_i's uncertainty region, as it grows by one each time step. This result will be important in showing we can maintain a reasonable ply.

Theorem 17. *If for all times t and all entities e_j, $t - p_j(t) \leq \frac{1}{11}|\hat{b}_j^x(p_j(t))|$ then for all times t and any entity e_i, $t - p_i(t) < \frac{1}{4}|b_i^x(t)|$.*

Proof. Assume the hypothesis of the theorem holds. We will prove the result by induction. Our base case is at time zero. The result is trivially true, since at time 0 for each entity e_i, $t - p_i(t) = 0$.

Assume that at all time steps $\tau < t$, all entities e_j have been probed in the last $\frac{1}{4}|b_j^x(\tau)|$ time steps. Take e_i some arbitrary entity. Our goal to bound the time that has elapsed since e_i was last probed with respect to the volume of its true x-blanket at time t. To do so we need to bound this volume by the volume of the perceived x-blanket at $p_i(t)$, since that is what we base our probe schedule on. We will do this in two steps, first we bound the true volume of the x-blanket at $p_i(t)$ by the perceived volume at $p_i(t)$, and then we bound the true volume at t, by the true volume at $p_i(t)$.

To bound the volume of the perceived x-blanket at time $p_i(t)$ by the true x-blanket volume at that time, consider the set of entities $\Gamma_i^x(p_i(t))$. For every e_j in $\Gamma_i^x(p_i(t))$ the last time it was probed, $p_j(p_i(t))$, is less than $p_i(t) - \frac{1}{4}|b_j^x(p_i(t))|$ time steps by our inductive hypothesis. Thus, by Lemma 16, $\frac{4}{7}|\hat{b}_i^x(p_i(t))| \leq |b_i^x(p_i(t))|$.

Now we want to bound the volume of $b_i^x(t)$. By the hypothesis of the theorem,

$$\begin{aligned} t - p_i(t) &< \frac{1}{11}|\hat{b}_i^x(p_i(t))| \\ &< \frac{7}{44}|b_i^x(p_i(t))| \\ &< \frac{1}{6}|b_i^x(p_i(t))| \end{aligned}$$

Thus at most the entities in Γ_i^x were in the interval $[\min(b_i^x(p_i(t))) + \frac{1}{12}b_i^x(p_i(t)), \max(b_i^x(p_i(t))) - \frac{1}{12}b_i^x(p_i(t))]$, and e_i itself has moved at most $\frac{1}{12}b_i^x(p_i(t))$ units. Thus,

$$|b_i^x(t)| \geq |b_i^x(p_i(t))| - \frac{1}{3}|b_i^x(p_i(t))| = \frac{2}{3}|b_i^x(p_i(t))|$$

This implies,

$$\frac{1}{6}|b_i^x(p_i(t))| \leq \frac{1}{6}\left(\frac{3}{2}|b_i^x(t)|\right) = \frac{1}{4}|b_i^x(t)|$$

Combing this with our previous result we find that $t - p_i(t) < \frac{1}{4}|b_i^x(t)|$.

Thus by induction our result holds for all time steps. □

Corollary 18. *For all pairs of times t,t', such that $t - t' < \frac{1}{11}|\hat{b}_j^x(p_j(t))|$ for all entities e_i, $\frac{1}{2}|b_i^x(t')| < |b_i^x(t)| < \frac{3}{2}|b_i^x(t')|$.*

Proof. The result follows from Lemma 15 and Theorem 17. □

Corollary 19. *If for all t and all entities e_j, $t - p_j(t) < \frac{1}{11}|\hat{b}_j^x(p_j(t))|$ then for all times t and any entity e_i, $\frac{4}{7}|\hat{b}_i^x(t)| \leq |b_i^x(t)| \leq 4|\hat{b}_i^x(t)|$.*

Proof. The result follows from Lemma 16 and Theorem 17. □

The final result for this section combines the above two results to argue that the volume of our perceived x-blanket at a given time is a good approximation for the true x blanket, for an interval of time stretching into the future.

Lemma 20. *If for all t and all entities e_j, $t - p_j(t) < \frac{1}{11}|\hat{b}_j^x(p_j(t))|$ then for all times t and any entity e_i, $\frac{8}{21}\hat{b}_i^x(t) \leq b_i^x(t^*) \leq 8\hat{b}_i^x(t)$ for all t^* in $[t, t_f]$ where $t_f = t + \frac{\hat{b}_i^x(t)}{11}$*

Proof. We will prove the upper and lower bounds separately. First we prove the lower bound.

From Corollary 19 we know that

$$\frac{4}{7}|\hat{b}_i^x(t)| \leq |b_i^x(t)|$$

From Corollary 18 we get that

$$|b_i^x(t)| \leq \frac{3}{2}|b_i^x(t^*)|$$

Therefore

$$\frac{4}{7}|\hat{b}_i^x(t)| \leq \frac{3}{2}|b_i^x(t^*)|$$
$$\leftrightarrow \frac{8}{21}|\hat{b}_i^x(t)| \leq |b_i^x(t^*)|$$

For the upper bound, from Corollary 19 we get

$$|b_i^x(t)| \leq 4|\hat{b}_i^x(t)|$$

From Corollary 18 we know that

$$\frac{1}{2}|b_i^x(t^*)| \leq |b_i^x(t)|$$

This means,

$$\frac{1}{2}|b_i^x(t^*)| \leq 4|\hat{b}_i^x(t)|$$
$$\leftrightarrow |b_i^x(t^*)| \leq 8|\hat{b}_i^x(t)|$$

By combining our upper and lower bounds we get the statement of the lemma:

$$\frac{8}{21}|\hat{b}_i^x(t)| \leq |b_i^x(t^*)| \leq 8|\hat{b}_i^x(t)|$$

□

In this section we have shown that, by probing frequently enough based on the perceived x-blanket volume, our perception of the entities is not too far off their true locations. Knowing this we can show the bucket algorithm will be able to do fairly well, assuming we have an idea of what the optimal blanket value is.

5.1.2 Bucket Algorithm Analysis

Using our results from the previous section we will show that the Bucket Algorithm performs well, if we are given a reasonable value for χ.

Definition 20. *We say a bucket has "overflowed" if there is still an element in its queue at the end of its associated time interval.*

As long as no bucket overflows, we have managed to probe every entity before its uncertainty region exceeds its χ-blanket. By an argument similar to the one used in the static case, this leads us to conclude we have low ply.

Theorem 21. *If no bucket overflows the Bucket Algorithm maintains ply $O(\chi)$, for the value χ we are given.*

Proof. Assume that for a given χ value the Bucket Algorithm runs without a bucket ever overflowing.

Let e_i be an arbitrary entity, and t be an arbitrary time step. Recall that when the Bucket Algorithm probed entity e_i at time $p_i(t)$ it put e_i into a bucket which has length at most $\frac{1}{22}|\hat{b}_i^x(p_i(t))|$, and note that the time after $p_i(t)$ before the start of that bucket is at most $\frac{1}{22}|\hat{b}_i^x(p_i(t)))|$, since it is the next bucket of its length which does not overlap the current time step. Since the algorithm runs without a bucket overflowing, this means that any e_i will have been probed before $\frac{1}{11}|\hat{b}_i^x(p_i(t))|$ at any time t. Then by Theorem 17 the entity e_i was probed in the last $\frac{1}{4}|b_i^\chi(t)|$ time steps. Thus $|r_i(t)| < \frac{1}{4}|b_i^\chi(t)|$ (recall $r_i(t)$ is the uncertainty region of e_i at t). Since $l_i(t) \in r_i(t)$

$$r_i(t) \subseteq \left[l_i(t) - \frac{1}{4}|b_i^\chi(t)|, l_i(t) + \frac{1}{4}|b_i^\chi(t)| \right]$$

Thus $r_i(t) \subset b_i^\chi(t) = \left[l_i(t) - \frac{1}{2}|b_i^\chi(t)|, l_i(t) + \frac{1}{2}|b_i^\chi(t)| \right]$ for all times t.

Note that, similar to the static case, no more than 2χ of the χ-blankets overlap on any one given point. This is because for any given point the χ-blankets of at most the χ entities to the left, and χ entities to the right overlap that point (this is more formally discussed in Theorem 11). Since an entities uncertainty region is contained in its χ-blanket, our ply is always less than 2χ, which is $O(\chi)$ □

Now we will show that a bucket overflows only if there is a time step in its associated time interval when the inverse sum of the χ-blanket volumes is greater than $\frac{1}{800}$.

Theorem 22. *If a bucket overflows then there is a time τ, in that bucket's associated time interval, such that $\sum_{i=1}^{n} \frac{1}{|b_i^x(\tau)|} > \frac{1}{800}$.*

Proof. Assume that some bucket, β^*, overflows, and let T^* be its associated time interval.

Let $\mathscr{B}(t) = \{\beta_{i,j} : t \in [i2^j + 1, (i+1)2^j]\}$, the set of buckets which include time t in their intervals. Note that there is one bucket in $\mathscr{B}(t)$ for each length j.

We define $\phi(t)$ as follows (recall $|\beta|$ is the length of the associated interval, not the number of entities in bucket β):

$$\phi(t) = \sum_{\beta \in \mathscr{B}(t)} \sum_{e_i \in \beta} \frac{1}{|\beta|}$$

That is, we sum over all the placements of entities in buckets which overlap time t, taking the sum of the inverse of the size of the bucket the entity is placed in. We can think of $\phi(t)$ as a surrogate for the inverse sum of the x-blanket volumes, since the volume of the bucket an entity is placed into is closely related to the volume of its x-blanket. There are many factors which will cause the sum of the inverse bucket volumes to be much larger than the sum of the inverse blanket volumes, these necessitate the $\frac{1}{800}$ constant. Since we never remove elements from buckets, just mark them as probed, this sum includes entities that were chosen to be probed at a time step earlier than t. Note that if an entity e_i that is in bucket β is probed and placed in a bucket that is shorter than β, the new bucket it is placed in may overlap β. Thus, it will contribute multiple times to $\phi(t)$, for all the times t when both of the buckets overlap. Note that $\phi(t)$ also includes contributions from entities which are never probed if a bucket which overlaps t overflows.

There are now two steps to our proof:

1. Prove there exists a time τ such that $\phi(\tau) \geq 1$.
2. Show that this implies at that time τ the sum $\sum_{i=1}^{n} \frac{1}{|b_i^x(\tau)|}$ is greater than $\frac{1}{800}$

For the first step of the proof we will consider $\sum_{t \in T^*} \phi(t)$. Recall that T^* is the interval associated with β^*, the bucket which overflowed. The sum over all the buckets is larger than the sum over just those buckets whose intervals are contained within β's. By Observation 2 we know that each of these buckets are wholly contained within T^*. Thus we sum over the entities in the bucket β, $|\beta|$ times. Hence

$$\begin{aligned}
\sum_{t\in T^*}\phi(t) &= \sum_{t\in T^*}\sum_{\beta\in\mathscr{B}(t)}\sum_{e\in\beta}\frac{1}{|\beta|}\\
&\geq \sum_{t\in T^*}\sum_{\substack{\beta\in\mathscr{B}(t)\\ \beta\subseteq\beta^*}}\sum_{e\in\beta}\frac{1}{|\beta|}\\
&= \sum_{\beta\subseteq\beta^*}|\beta|\sum_{e\in\beta}\frac{1}{|\beta|}\\
&= \sum_{\beta\subseteq\beta^*}\sum_{e\in\beta}1\\
&> |T^*|
\end{aligned}$$

The last inequality is a consequence of β^* overflowing and thus there are more than T^* entities that are scheduled to be probed over the interval T^*, each of which are in buckets β such that $\beta\subseteq\beta^*$. This implies

$$\bar{\phi} = \frac{\sum_{t\in T^*}\phi(t)}{|T^*|} > 1$$

That is the average of the $\phi(t)$ values over the interval T^*, $\bar{\phi}$, is greater than 1. This means there is some time in T^* where the $\phi(t)$ is greater than one. Let τ be such a time.

Having established there exists a time τ when $\phi(\tau)\geq 1$ we will show that $\sum_i \frac{1}{|b_i^\chi(\tau)|} > \frac{1}{800}$, that is the inverse sum of the blanket volumes is only a factor of $\frac{1}{800}$ less than $\phi(\tau)$.

Our goal is to describe the relationship between the volume of the bucket an entity is probed in that overlaps time τ, and the volume of the x-blanket of that same entity at time τ. Consider some arbitrary entity e_i, and some bucket β that e_i is placed in that overlaps time τ. Let $p_i(\beta)$ be the time at which e_i was probed and placed in bucket β. The distinction between $p_i(\beta)$ and $p_i(\tau)$ is important, since the time step $p_i(\tau)$ may be in β. We will be interested in the probe time at which e_i was placed in β. We aim to probe an entity before time steps equal to $\frac{1}{11}^{th}$ of its perceived x-blanket volume have past. Our algorithm does this by placing the entity in the first bucket whose size is less than $\frac{1}{22}^{nd}$ of the entity's perceived x-blanket volume. Note the factor of two is sufficient to insure that the time between

when the entity was probed and the end of the bucket it is placed in is less than $\frac{1}{11}$. Thus we know that

$$\tau - p_i(\beta) < \frac{\hat{b}_i^x(p_i(\beta))}{11}$$

Hence from Lemma 20

$$\frac{8}{21}\hat{b}_i^x(p_i(\beta)) \leq b_i^x(\tau) \leq 8\hat{b}_i^x(p_i(\beta))$$

We know from the definition of our algorithm that β's volume is at least $\frac{1}{44}^{th}$ the volume of $\hat{b}_i^x(p_i(\beta))$. This means,

$$\begin{aligned} |b_i^x(\tau)| &\leq 8|\hat{b}_i^x(p_i(\beta))| \\ \rightarrow |b_i^x(\tau)| &\leq 352|\beta| \end{aligned}$$

It is important to note that an entity may show up in several different buckets in the calculations of $\phi(\tau)$. Observe the sum $\sum_{\tau\in\beta}\frac{1}{|\beta|}$ is a geometric sum. It follows that the entire contribution from a single entity is not more than twice its largest contribution, since the bucket sizes are powers of two. That is if ω_i is the length of the smallest bucket that e_i is in that intersects τ then

$$\phi(\tau) < 2\sum_{i=1}^{n}\frac{1}{\omega_i}$$

Since $|b_i^x(\tau)| \leq 352|\beta|$,

$$\begin{aligned} \phi(\tau) &< \sum_{i=1}^{n}\frac{704}{b_i^x(\tau)} \\ \rightarrow \sum_{i=1}^{n}\frac{1}{b_i^x(\tau)} &> \frac{1}{800} \end{aligned}$$

Thus if β^* overflows, then $\sum_i \frac{1}{|b_i^x(\tau)|} > \frac{1}{800}$ for some time step τ in T^*. □

Now we know that if a bucket overflows, it means there is a time when the inverse sum of the blanket sizes is relatively large, we argue this means the distribution of the entities at that time is "bad". Specifically if the entities were frozen in those positions ply $\Omega(\chi)$ would be unavoidable.

Theorem 23. *If a bucket overflows, then there is a time at which the entities are located in such a way that any algorithm would produce average ply $\Omega(\chi)$ if the entities did not move.*

Proof. By Theorem 22 we know if a bucket overflows, there is a time when $\sum_i \frac{1}{|b_i^{\chi}(t)|} > \frac{1}{800}$, let τ be such at time. This means that at τ the optimal blanket value for probe density $\frac{1}{800}$ is greater than χ. By Theorem 13 we know that any algorithm will have average ply which is at most a constant factor times the optimal blanket value, for any given probe density, given T is sufficently long. Since the entities do not move T can be arbitrarily long, eventually any algorithm's average ply is $\Omega(\chi)$ for the distribution of entities at τ. □

For the following it is important to note the distinction between the value of χ we are given, and the true optimal blanket value for probe density $\frac{1}{800}$ at time t, which I will call $x(t)$.

To elaborate on the above theorem, if a bucket overflows then there is some time τ where the $x(\tau)$ was greater than the value χ we were given, by Theorem 22. That means that the value of χ was smaller than the optimal blanket value at τ. As in the static case, we know the high optimal blanket value is indicative of a configuration of entities such that high ply is unavoidable if the entities stay in the "bad" configuration for an extended period of time. However, it is not clear in this case whether a smart algorithm could avoid having high ply by "preparing" for this time step when the entities are too closely crowded anticipating that the entities will become uncrowded quickly.

It is also not clear that our algorithm will fail simply because there is some time step where the inverse sum of the x-blankets is relatively large. It is entirely plausible that despite their being a time when $x(\tau) > \chi$ that our algorithm still runs successfully. In fact, in our proof of Theorem 22 one can see that although we only consider a single time step, the sum $\sum_{t \in T} \phi(t)$ is relatively high on average. It is unclear from our analysis how quickly this sum value can change, and whether a single identifiable violation actually implies a more lengthy interval of poor positioning of the entities. A study of how sensitive our algorithm actually is to these violations would be an interesting avenue for future work.

The algorithm presented in this chapter provided a good first step towards solving the continuous ply minimization problem in its full generality. The algorithm did depend on a oracle that provided with an χ value to shoot for, that future work would aim to remove. It was also unfortunate that our algorithm does not adjust to an ever changing environment. We can get a good estimate of what the optimal blanket value is, based on the perceived location of the points. It would be interesting to design an algorithm that is more adaptive, and adjusts the blanket value it uses to generate probes, based on how this estimate of the optimal blanket value

changes. This is not possible with our current tools, as our estimate of the blanket value is not tight enough, and it changes to rapidly to be used effectively. Although there is still work to be done until the question of how to efficiently continuously probe entities is answered in full, this work has provided a strong base to develop on.

Chapter 6

Concluding Remarks

Variants of the ply minimization problem has been explored in a few different papers [8][9]. The main focus, in these papers, is the "single-shot" case, that is the case where one wishes to minimize the ply at a single point in time. The current work built on those investigations, by considering the continuous version of the Ply Minimization problem. This continuity required the introduction of new tools to deal with the challenges of being competitive across an entire interval of time.

To start, we simplified the problem by assuming that the entities were not moving. This simplification decreased the complexity of the problem, while still yielding interesting results. It allowed us to focus on issues that arise from the repetitive nature of the problem, and ignore those that are caused by the uncertainty of our information. Looking at the static case lead to the conclusion that being competitive over an interval of time should not be interpreted as being competitive with the best ply that could be achieved at each time step, because it is impossible for an algorithm to maintain optimally low ply at each time step. Specifically, the optimal ply at a given time step may be a factor of $\log n$ smaller than the minimum maximum ply any algorithm could achieve over the interval in question. Thus, the focus of our work was on the competitive ratio one could get when comparing their own algorithm's maximum minimum ply to that of any other algorithm's average ply, over a given time interval. We produced an algorithm, the Rounded Blankets algorithm, which has worst case ply which is a constant factor more than any other algorithm's average ply, over any sufficiently long interval of time. It operates by calculating the optimal x-blanket for each entity, and then probing that entity before its uncertainty region is as large as the blanket. This maintains ply which is within a constant factor of the optimal blanket value. Any other algorithm's maximum minimum ply is within a constant factor of the optimal blanket value. Thus

we get the aforementioned bound on the competitive ratio of the Rounded blankets algorithm.

We then adapted the Rounded Blankets algorithm to work in the dynamic case. It is impossible to calculate a single, optimal, x-blanket for an entity, as we did in the static case, since the movement of the points means it will change over time. The reliability of the unchanging locations of the entities in the static case was important for building a schedule where only one entity is probed at each time step. In the dynamic case we schedule entities into "windows" of time, instead of at specific time steps. We call these windows buckets. They are based on the current blanket volumes, and we require the algorithm to probe an entity before its window ends. As long as we work with a blanket value which no less than the true maximum optimal blanket value over the interval, our algorithm will probe all the entities in the allotted time. We also show that we can conclude our ply will be close the blanket value we use. Unfortunately the fact the point locations are changing in the dynamic case means that the optimal blanket value is no longer a tight bound on the behaviour of all algorithms.

6.1 Future Work

One potential avenue for future work is producing a more adaptive algorithm, and discovering a bound on the ply algorithms produce, in the dynamic case. An idea for how such algorithms might improve on our results is to maintain a "working blanket value", which is our approximation of the true current blanket value, and updating it when our perceived optimal blanket value crosses certain thresholds, such as being double or half the working blanket value. Another idea is increasing the blanket value when a bucket overflows. Currently we view overflows as a "critical error", and they cause the algorithm to halt. By increasing the blanket value when a bucket overflows we can work around this error state. In such a situation, entities would be moved into larger buckets, based on their new perceived blanket volume. Whatever adjustments are made, one will have to argue about the competitiveness of the new algorithm. Arguing about the minimum maximum ply any algorithm can get in the dynamic case, and how it is related to the optimal blanket values of the time steps in the time interval of interest, would be useful results.

Future work might also consider different models for the Continuous Ply Minimization problem. Some differing models been explored for single shot case, including having a variety of speed bounds, and extending to higher dimensional spaces [8][9]. The easiest of these would appear to be extending to a higher di-

mensional space, which should follow fairly directly from the arguments in this work. One might also consider bounding the acceleration of the entities, or using probabilistic models to represent the uncertainty regions of the entities. Another modification to consider is changing the probe model. It would be interesting to examine how the results might change if the algorithm had more than one probe available per time step, or if the algorithm had a finite number of probes, but was allowed to use more of them at times when the optimal blanket value was high, if it used fewer when the optimal blanket value was low. The last model change one might consider is changing the way ply is calculated. Currently we consider ply to be the maximal overlap between uncertainty regions. However, the usefulness of this measure will depend on the application. It may be better to consider the total number of uncertainty regions which are pairwise overlapping. Or alternatively, to consider some function of the number of uncertainty regions which overlap a point, and then integrate over the real line. For example, take $f(x) = \delta(x)^2$, where $\delta(x)$ is the number of entities which overlap x. Then the value we are interested in would be $\int_{-\infty}^{\infty} f(x)dx$.

One subject, which was only briefly mentioned in this document, is the hardness of calculating the optimal query pattern. In the one shot case, Evans et al show that calculating the optimal probe sequence is NP-hard, when one knows the true trajectories of the entities [8]. It is hard to fathom that the problem gets easier in the continuous case. However the problem is distinct enough to admit the possibility, due to the difference between optimizing at a single time, versus optimizing over an entire interval. Perhaps the wider target area makes the correct probe at any one given step more tractable. Another way to try to gain insight into the hardness of continuous probing, or indeed optimal probes in general, would be to relax the restriction on probing once per time step. Consider instead the situation if we simply required the entities be probed periodically, that is every P_i time steps, and that the inverse sum of these periods is less than one, that is $\sum_i \frac{1}{P_i} < 1$. The problem then transforms into calculating optimal periods for each entity, rather than the optimal query at each time step. It would be interesting to see if this makes the calculation simpler.

As we have discussed, there are many interesting directions for future research to take. Many other variations exist which have not been touched on in this section. The answers to these questions will lead to a better understanding of the ply minimization problem in general.

Bibliography

[1] S. Anily, C. A. Glass, and R. Hassin. The scheduling of maintenance service. *Discrete Applied Mathematics*, 82(13):27 – 42, 1998.

[2] F. Aurenhammer, G. Stöckl, and E. Welzl. The post office problem for fuzzy point sets. In *Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications*, CG '91, pages 1–11, London, UK, UK, 1991. Springer-Verlag.

[3] A. Bar-Noy, A. Nisgav, and B. Patt-Shamir. Nearly optimal perfectly periodic schedules. *Distributed Computing*, 15(4):207–220, 2002.

[4] M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic compressed quadtrees in the black-box model with applications to collision detection for low-density scenes. In L. Epstein and P. Ferragina, editors, *Algorithms ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 383–394. Springer Berlin Heidelberg, 2012.

[5] M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic convex hulls, delaunay triangulations and connectivity structures in the black-box model. *Journal of Computational Geometry*, 3(1):223 – 249, 2012.

[6] M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic 2-centers in the black-box model. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 145–154, New York, NY, USA, 2013. ACM.

[7] R. Dorrigiv and A. López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.

[8] W. Evans, D. Kirkpatrick, M. Löffler, and F. Staals. Competitive query strategies for minimising the ply of the potential locations of moving points. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 155–164, New York, NY, USA, 2013. ACM.

[9] W. Evans, D. Kirkpatrick, M. Löffler, and F. Staals. Query strategies for minimizing the ply of the potential locations of entities moving with different speeds. *Abstr. 30th European Workshop on Computational Geometry (EuroCG)*, 2014.

[10] J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. *Computational Geometry*, 35(12):2 – 19, 2006. Special Issue on the 20th {ACM} Symposium on Computational Geometry 20th {ACM} Symposium on Computational Geometry.

[11] L. J. Guibas. Kinetic data structures – a state of the art report, 1998.

[12] S. Kahan. *Real-Time Processing of Moving Data*. PhD thesis, University of Washington, 10 1991.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[14] M. Löffler and J. Snoeyink. Delaunay triangulations of imprecise pointsin linear time after preprocessing. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 298–304. ACM, 2008.

[15] M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, Feb. 2010.