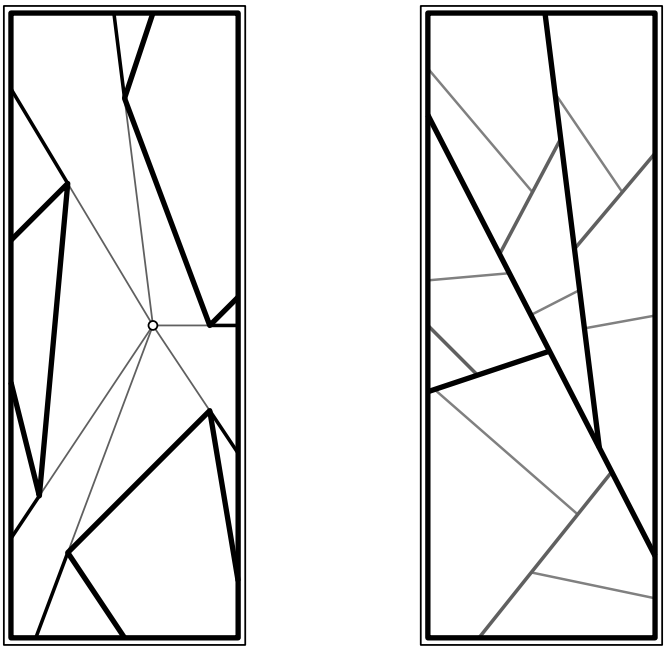
Guarding Problems and Geometric Split Trees

James Alexander King



A thesis submitted to McGill University
in partial fulfillment of the requirements of the
degree of Doctor of Philosophy

Abstract

Many geometric problems are intrinsically linked to the issue of splitting or classifying points. We investigate two such families of problems in two separate branches of research.

Guarding problems are motivated by the issue of guarding a region with security cameras or illuminating it with lights. Such problems have been studied for decades, but there are two significant guarding problems whose complexity is not completely understood. First, we investigate the problem of guarding simple polygons; this problem is known to be NP-complete but its approximability is not known. Second, we investigate the complexity of guarding monotone chains, also known as 1.5-dimensional terrains. Understanding the interaction of 'visibility polygons' and how they separate point sets is crucial for the investigation of such problems. We resolve a significant open problem by proving strong NP-completeness for terrain guarding. We also present an approximation algorithm for guarding simple polygons with perimeter guards; this new algorithm improves the state of the art.

A geometric split tree is a data structure for storing point sets that recursively splits the space, and in turn the data, in some random way. Understanding the distribution of such a tree's structure is a matter of understanding the distribution of the splits. We investigate the distributions associated with several natural splitting methods. We make new connections between an important problem in discrete geometry and natural probability distributions. With the goal of analyzing geometric split trees based on their splits, we introduce a random tree model that is general while still allowing powerful comparisons with random trees from more restricted models.

Résumé

Beaucoup de problèmes géométriques sont intrinsèquement liés à la question de la division ou classification des points. Nous étudions deux familles de problèmes dans deux branches distinctes de recherche.

Les problèmes de surveillance sont motivés par la question de la surveillance d'une région avec des caméras de sécurité ou d'éclairage avec des feux. Ces problèmes ont été étudiés depuis des décennies, mais il y a deux problèmes importants dont la complexité n'est pas complètement élucidée. Tout d'abord, nous étudions le problème de surveiller des polygones simples. Ce problème est connu pour être NP-complet, mais son approximabilité n'est pas connue. Deuxièmement, nous étudions la complexité de surveiller des chaînes monotones, aussi connues comme terrains en dimension 1,5. Comprendre l'interaction des polygones de visibilité et la façon dont ils divisent les ensembles de points est crucial pour l'étude de ces problèmes. Nous résoudrons un problème important ouvert en prouvant que surveiller les terrains est fortement NP-complet. Nous présentons également un algorithme d'approximation pour la surveillance des polygones simples avec des gardes sur le périmètre. Ce nouvel algorithme améliore l'état de l'art.

Un arbre de division géométrique est une structure de données pour stocker des ensembles de points qui divise de manière récursive l'espace, et aussi les données, d'une certaine manière aléatoire. Le compréhension de la répartition de la structure d'un tel arbre est une question de compréhension des répartitions des divisions. Nous étudions les répartitions associées à plusieurs méthodes de division naturelles. Nous faisons de nouvelles connexions entre un problème important en géométrie discrète et des distributions de probabilité naturelles. Dans le but d'analyser les arbres de division géométriques en fonction de leurs divisions, nous introduisons un modèle d'arbre aléatoire qui est général tout en permettant des comparaisons puissants avec les arbres aléatoires dans des modèles plus restreints.

Acknowledgements

First and foremost I would like to thank my supervisor Luc Devroye, who has taught me so much, supported me in every way, and without whom I could not imagine my time at McGill. I would also like to thank David Kirkpatrick, Erik Krohn, and Colin McDiarmid, all of whom were coauthors on papers related to this thesis. Collaborating with them has been a pleasure as much as it has been a learning experience.

I would like to thank Bill Steiger for his helpful comments and more generally for his service as the external examiner of this thesis. Similarly, I owe a debt of gratitude to the other members of my committee, particularly Odile Marcotte, Mohit Singh, and Adrian Vetta, all of whom prepared for my defence on short notice during a busy time of year.

The last five years have been spent in the company of office mates and colleagues — grad students, postdocs, and profs — who have made this place feel like home. The coffee breaks, intramural sports, and tropical workshops have been unforgettable.

Outside of McGill, my life in Montreal has been filled with amazing friends. It is hard to believe that I knew so few of them five years ago, and I hope that my life brings me back to Montreal before long.

Finally, I would like to thank my family. My parents have given me nothing but love and support, and it has been a blessing to live near my brother again after living for so many years on opposite sides of the continent.

Contents

Chapter 1

Introduction

This thesis investigates problems from two areas of computational geometry and is therefore written in two parts. In the first part we investigate guarding problems — combinatorial optimization problems concerning lines of sight. In the second part we investigate random geometric split trees. These are random space partition trees whose analysis is closely related to problems in combinatorial geometry.

1.1 Guarding Problems

Given a number of objects it is natural to ask how to guard them with the minimum number of security cameras or how to illuminate them with the minimum number of lights. We may want to guard the interior of a building or we may want to illuminate a highway. These problems concern lines of sight and in the field of computational geometry they are known as guarding problems.

Most of the interesting problems take place in $\mathbb{R}^2$. Problems in $\mathbb{R}$ are usually too easy to be interesting and problems in $\mathbb{R}^3$ are usually too hard to be interesting. In $\mathbb{R}^2$ our 'guards' (*i.e.*, cameras or lights) are typically modeled as points and obstacles are typically modeled as line segments. The objects to be guarded are typically modeled as the finite union of points, line segments, and/or polygons. We say that one point *sees* another if the line segment between them does not pass through an obstacle. A guarding set is a

set of guards that sees all the objects that need to be guarded.

The study of guarding problems began in 1973 with a question raised by Victor Klee (83). He asked the following:

> For a simple polygon P with n vertices, what is the maximum number of vertex guards needed to guard the interior of P?

The answer, given as Chvátal's *Art Gallery Theorem* (21), is $\lfloor n/3 \rfloor$. Many variations of this problem have been studied; for a survey of these problems and results we refer the reader to O'Rourke's book (83).

This problem and others like it are existential in nature. The problem can be stated as, "What is the maximum over all simple n-gons P of the size of the minimum guarding set for P?" It is perhaps more interesting to ask, "For a *given* simple n-gon P, what is the size of the minimum guarding set for P?" The problem then becomes an optimization problem of algorithmic interest.

Range Spaces and Approximation

In the first chapter on guarding problems, we discuss *range spaces*. Also known as set systems or hypergraphs, range spaces represent systems of constraints. The constraints involved in guarding problems are naturally modeled as range spaces.

We focus on the hitting set problem on range spaces under certain restrictions. Many general techniques developed for this problem, especially approximation algorithms, are applicable to guarding problems. We introduce ε-nets and discuss the general framework for turning an algorithm that finds small ε-nets into an approximation algorithm for hitting set. We then discuss the concept of a range space's VC-dimension, particularly as it applies to the construction of small ε-nets.

Guarding Terrains

The first guarding problem we investigate is the terrain guarding problem. In this case a terrain is modeled as a monotone polygonal chain; as such, a terrain is a natural way to model a topologically linear area of interest that has varying

altitude over a linear range. We determine the maximum VC-dimension of the range spaces associated with terrain guarding problems. We then discuss the state of the art regarding approximation algorithms. Finally, we give a proof that the terrain guarding problem is NP-complete, resolving a long-standing open problem.

Guarding Polygons

After our treatment of the terrain guarding problem, we turn our attention to the problem of guarding polygons. This is the optimization version of the *art gallery problem* — in this context the input polygons are often referred to as *galleries*. We introduce the problem and define the associated range spaces. With the goal of applying general results for finite range spaces, we discuss issue of discretization. Discretization is far more complicated for polygons than for terrains. We then present an approximation algorithm for certain variants of the problem (specifically, guarding simple polygons with perimeter guards) that improves upon the state of the art.

1.2 Geometric Split Trees

Binary search trees are based on the idea of recursively splitting a data set. Informally, if the internal nodes in the tree split the data evenly, the height of the tree and the average depth of a node in the tree will be smaller. These properties of a tree are of particular interest because they correspond to the worst-case search time and the average-case search time respectively.

We begin our discussion of random tree data structures with a simple example, the *random binary search tree*. Given a fully ordered set $S = \{x_1, \ldots, x_n\}$ of n distinct elements, this random tree model can be defined several different ways. Perhaps the most natural and intuitive is to consider constructing a binary search tree from scratch by inserting the n elements one at a time in random order. This description also motivates the model since real-world data often behave similarly to random data.

Random Split Trees

Random binary search trees are a canonical but extremely restricted model of random tree data structures. Many generalizations have been studied and analyzed; perhaps the most notable model being the *random split tree* model. For trees in this model, asymptotically tight bounds can be obtained for the height of a tree and the average depth of a node in a tree (31).

The key to the asymptotic behaviour of a random split tree is the distribution with which a node divides the elements in its subtree among the subtrees rooted at its children. An unfortunate restriction of the random split tree model is that, for every node in the tree, the random split must be identically distributed. We present a model that avoids this restriction, yet allows the structural distribution of a tree to be bounded using a tree from the random split tree model.

Splitting with Random Hyperplanes

Random binary search trees store elements with one-dimensional keys which, in geometric terms, we can consider to be points in $\mathbb{R}$. There are natural ways to generalize this one-dimensional splitting to higher dimensional data sets, *e.g.*, points in $\mathbb{R}^d$ for $d \geq 2$. Arguably the most natural generalization is to sample d points at random to define a hyperplane, then split the remaining points based on membership in the two associated halfspaces.

Before discussing data structures based on such a splitting strategy, we focus on the analysis of such random hyperplane splits. As it turns out, analyzing random hyperplane splits is equivalent to a well-studied problem in combinatorial geometry — the problem of counting *k-facets*. A k-facet of a set of n points is essentially a subset of d points that define a hyperplane that splits the remaining points into subsets of size k and $n - d - k$. In Chapter 7 we discuss bounds involving k-facets that imply bounds on the distributions of random hyperplane splits.

Search Trees from Random Hyperplanes

In Chapter 8 we analyze two random search tree data structures that are based on splitting with random hyperplanes. These are both space partition trees. The first data structure, the *random hyperplane search tree*, is a binary space partition tree. It acts by storing d randomly chosen points in the root node, partitioning the remaining points based on the hyperplane defined by the points in the root, and continuing recursively in the two subtrees. The second data structure, the *random hyperplane arrangement search tree*, or simply *random arrangement tree*, generalizes the random hyperplane search tree. A set of $k \geq d$ points defines an arrangement of $\binom{k}{d}$ hyperplanes. Each internal tree node stores k randomly chosen points and splits the remaining points in its subtree according to the corresponding arrangement of hyperplanes.

Our analysis of these random trees brings together our work from previous chapters. We use results from Chapter 7 to bound the distribution of a split at a particular node. Results from Chapter 6 then allow us to bound the structural distribution of an entire tree.

1.3 Thesis Contributions

At the end of each chapter we summarize the novel contributions. Here we mention the most significant progress put forward in this thesis.

Part I Our research on guarding problems has resulted in two significant contributions: one for terrains and one for polygons. For guarding terrains, in joint work with Erik Krohn we have proved that the decision problem is strongly NP-complete. This resolves a problem of significant interest that was open for the past 15 years. This result is given in Section 3.3. For the problem of guarding polygons, in joint work with David Kirkpatrick we have developed a new polynomial-time approximation algorithm for guarding simple polygons with guards on the perimeter. The approximation guarantee is $\mathcal{O}(\log \log \textsc{opt})$. This is the first algorithm for guarding polygons to beat

the $\mathcal{O}(\log \text{OPT})$ guarantee obtained from general methods for range spaces of bounded VC-dimension. Our algorithm is given in Section 4.2.

Part II Our research on random hyperplane splits and random geometric trees has led to new bounds for the structural distributions of several random trees. These new bounds, obtained for random hyperplane search trees and random arrangement trees in joint work with Luc Devroye and Colin McDiarmid, are given in Chapter 8. Most of these bounds are a consequence of domination results comparing the depths of average elements in a range of random trees. In addition to these bounds, Part II describes a useful connection between random hyperplane splits, or equivalently the problem of counting k-facets, and well-known distributions generated from uniform random variables. For example, random hyperplane splits for a certain family of d-dimensional point sets (the vertices of cyclic polytopes) are distributed like splits based on the median of d uniforms. This sheds new light on the problem of counting k-facets and may help resolve a major conjecture that these splits are as uneven as random hyperplane splits can get.

Some of the results herein have already been published or submitted for publication. These include VC-dimension bounds for terrain guarding (sole author, 2008 (61)), NP-hardness of terrain guarding (with Erik Krohn, 2010 (63)), an improved approximation algorithm for guarding polygons from the perimeter (with David Kirkpatrick, 2010 (62)), and analysis of random hyperplane search trees (with Luc Devroye and Colin McDiarmid, 2009 (35)).

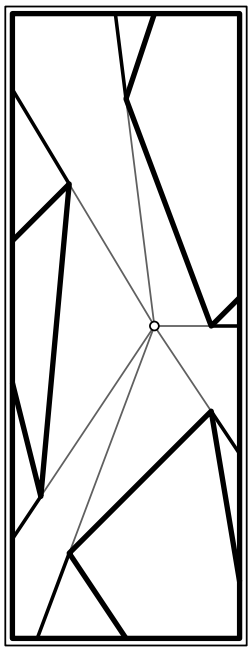
With the exception of Figure 8.1, all figures are original and were created independently by the author. Figure 8.1 was created by Luc Devroye.

1.4 Future Directions

Along with the novel contributions, we note open problems and future research directions at the end of each chapter. For guarding polygons, approximation algorithms still do not match inapproximability bounds. This is perhaps the

clearest direction for future work on guarding problems. Our work connecting k-facets and natural probability distributions leaves unresolved issues that are more exciting. The distributions of random hyperplane splits become more tightly concentrated around perfect splits as d increases. This blessing of dimensionality is somewhat surprising. With Luc Devroye we are currently giving the issue the rigorous treatment that did not make its way into this thesis. Lower bounds for $(\leq k)$-facets are another area that we are working to improve.

Part I
Guarding Problems



Chapter 2

Range Spaces and Approximation

In this chapter we introduce range spaces. This is general material relevant to the subsequent chapters on guarding problems. We discuss approximation of hitting set and set cover for general range spaces. We then introduce the powerful framework of Brönnimann and Goodrich for obtaining approximation algorithms using ε-nets. We follow this with the definition of VC-dimension and its implications on the problem of finding small ε-nets.

Contents

2.1 Introducing Range Spaces

Range spaces concern subsets of a universe $\mathcal{U}$ of elements.

Definition 2.1 (**Range Space**)**.** A range space $\mathcal{S} = (X, \mathcal{R})$ is a ground set $X \subseteq \mathcal{U}$ of elements along with a collection $\mathcal{R}$ of subsets of X.

Thus $\mathcal{R}$ is a subset of the power set 2^X of X. The elements of $\mathcal{U}$ and X are sometimes called *points*. Each subset $R \in \mathcal{R}$ is called a *range*. Outside of computational geometry range spaces are often called *set systems* or *hypergraphs*.

We say the range space is finite if X is finite (this implies that $\mathcal{R}$ is finite). For a finite range space we use n to denote $|X|$ and m to denote $|\mathcal{R}|$, so that $X = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$. It is sometimes useful to consider a finite range space as represented by an incidence matrix, defined as the $n \times m$ 0/1 matrix $A = (a_{ij})$ where $a_{ij} = 1$ if and only if $x_i \in R_j$. Incidence matrices facilitate the following definition.

Definition 2.2. For a range space $\mathcal{S}$, the *dual range space*, denoted $\widehat{\mathcal{S}}$, is the unique range space satisfying the following. If $\mathcal{S}$ is finite then the respective incidence matrices of $\mathcal{S}$ and $\widehat{\mathcal{S}}$ are transposes of each other. Generalizing this for a possibly infinite range space $\mathcal{S} = (X, \mathcal{R})$, we have $\widehat{\mathcal{S}} = (\mathcal{R}, \{r(x) : x \in X\})$ where $r(x) = \{R : R \in \mathcal{R}, x \in R\}$.

We now define *set covers* and *hitting sets* for range spaces.

Definition 2.3 (**Set Cover**)**.** A set cover for a given range space $\mathcal{S} = (X, \mathcal{R})$ is a subset $\mathcal{C} \subseteq \mathcal{R}$ of ranges that covers X, *i.e.*

$$\forall_{x \in X} \; \exists_{R \in \mathcal{C}} : x \in R \; .$$

For a given cost function $c : \mathcal{R} \to \mathbb{Q}$, a minimum set cover $\mathcal{C}$ is a set cover that minimizes $c(\mathcal{C}) = \sum_{R \in \mathcal{C}} c(R)$.

If a cost function is not given we assume it is uniform. The decision version of the minimum set cover problem asks, given a range space $\mathcal{S}$ and cost k,

whether $\mathcal{S}$ has a set cover of cost at most k. This problem is NP-complete (59).

Definition 2.4 (Hitting Set). A hitting set for a given range space $\mathcal{S} = (X, \mathcal{R})$ is a subset $H \subseteq X$ of points that hits every range $R \in \mathcal{R}$, *i.e.*

$$\forall_{R \in \mathcal{R}} \; \exists_{x \in H} : x \in R \; .$$

For a given cost function $c : \mathcal{U} \to \mathbb{Q}$, a minimum hitting set H is a hitting set that minimizes $c(H) = \sum_{x \in H} c(x)$.

Hitting set and set cover are dual problems, not in the sense of LP duality but in the sense of dual range spaces. A hitting set for $\mathcal{S}$ corresponds to a set cover for $\widehat{\mathcal{S}}$ and vice versa. This is clear if we consider incidence matrices. A set cover is a set of columns that hits every row (*i.e.*, in each row, at least one of the columns has a 1) and a hitting set is a set of rows that hits every column. So there is a linear-time reduction from minimum set cover to minimum hitting set. Minimum set cover and minimum hitting set are, in fact, two of Karp's original 21 NP-complete problems (59).

Approximating Set Cover

Inapproximability

The optimization version of the problem is not only hard to solve exactly, but is also hard to approximate in general. For some constant c, unless P = NP no polynomial time approximation algorithm can have a guaranted approximation factor as good as $c \ln n$ (88; 8). Under the stronger assumption that NP is not contained in DTIME($n^{\log \log n}$), the lower bound on approximability is improved to $(1 - o(1)) \ln n$ (46). We now present two simple and classical approximation algorithms for set cover. See, *e.g.*, Vazirani's book (99, pp. 16–19) for more details including approximation analysis.

The Greedy Algorithm

There is a classical approximation algorithm for set cover with a guaranteed approximation factor of $H_n \leq \ln n + 1$. Consider the following greedy algorithm. Start with an empty cover $\mathcal{C}$. While there are elements not covered by $\mathcal{C}$, choose a range R that covers the most uncovered elements and add R to $\mathcal{C}$.

Lemma 2.1 (Johnson (57), Lovász (72), Chvátal (21))**.** *The greedy set cover algorithm is an H_n-approximation.*

Proof. To bound the approximation factor, consider the per-element cost to add a set to a partial cover. This is defined as the cost of the set, divided by the number of uncovered elements it would cover. Index the elements in the order in which they are covered by the greedy algorithm. Let c_i be the per-element cost of the set that first covers the i^{th} element. Note that the total cost of the set cover is $\sum_i c_i$.

Just before the greedy algorithm covers the i^{th} element there must be at least $n - i + 1$ uncovered elements. Since all of these elements can be covered by sets of total cost OPT, there must be a set with a per-element cost of at most $\text{OPT}/(n - i + 1)$. The greedy algorithm always picks the set with the lowest per-element cost. We therefore have $c_i \leq \text{OPT}/(n - i + 1)$ and the total cost of the cover is $\sum_i c_i \leq \text{OPT} \cdot H_n$. □

Low Frequency Systems

The *frequency* of an element is the number of ranges containing it. If every element has frequency at most k, there is a simple k-approximation algorithm. Start with an empty cover $\mathcal{C}$. While there are elements not covered by $\mathcal{C}$, choose an uncovered element x and add every range containing x to $\mathcal{C}$. At each step we add at most k ranges to $\mathcal{C}$, at least one of which must be in any optimal cover. In the next section we discuss more sophisticated approximation methods for the hitting set problem that can be used in special cases.

Geometric Hitting Set

Many problems in computational geometry have natural interpretations as instances of hitting set and/or set cover. We introduce an example here that we revisit later.

Example 2.1 (Unit disk cover)**.** Let X and Y be subsets of $\mathbb{R}^2$. Consider the problem of covering all points in X with the minimum set of unit disks whose centres belong to Y. This problem can be formulated as set cover for a range space $\mathcal{S} = (X, \mathcal{R})$, where the ranges in $\mathcal{R}$ are given by $\{x : (x, y) \in X \times Y, d(x, y) \leq 1\}$.

We use this somewhat clumsy formulation to point out a special property that this family of range spaces has — it is closed under duality. For a range space defined by point sets (X, Y), the dual range space is defined by (Y, X). This is a consequence of the symmetry of Euclidean distance used to define the ranges.

2.2 Approximation from ε-Nets

An Introduction to ε-Nets

Informally, if we wish to relax the hitting set problem, we can ask for a subset of X that hits all *large* ranges in $\mathcal{R}$. This is the idea behind ε-nets. The general definition requires a non-negative weight function $w : X \to \mathbb{Q}^{\geq 0}$ on the elements of X with $w(R) = \sum_{x \in R} w(x)$.

Definition 2.5 (ε**-Net).** Given a range space $\mathcal{S} = (X, \mathcal{R})$, a weight function $w : X \to \mathbb{Q}^{\geq 0}$, and a parameter $\varepsilon \in [0, 1]$, an ε-net is a subset of X that hits every range $R \in \mathcal{R}$ having $w(R) \geq \varepsilon w(X)$.

An ε-net is a kind of approximate hitting set. When $\varepsilon = 1$, any non-empty set is an ε-net. When ε is close to 1, minimum ε-nets are small. When ε is close to 0, ε-nets closely approximate hitting sets. When $\varepsilon = 0$, every ε-net is a hitting set. The task of finding small ε-nets for a given range space, weight function, and ε is a crucial one, but we reserve its discussion for Section 2.3.

Haussler and Welzl (54) introduced ε-nets and originally used them for the purpose of halfspace range queries. However, ε-nets have since risen to prominence because of their use in approximation algorithms for restricted instances of hitting set, particularly those that arise in geometric settings.

Brönnimann and Goodrich (17) were the first to exploit ε-nets for this purpose. They developed a general approximation algorithm for hitting set that requires two oracles: a *net finder* and a *verifier*. The net finder takes as input a range space $\mathcal{S} = (X, \mathcal{R})$, a weight function w, and a parameter $\varepsilon > 0$; as output it returns an ε-net. The verifier is for checking if an ε-net is a hitting set; it either states correctly that it is, or returns a range $R \in \mathcal{R}$ that is not hit by the ε-net.

Iterative Reweighting

The algorithm of Brönnimann and Goodrich is primarily concerned with finding the right parameters for the net finder. More precisely, it learns a weight function and an ε that ensure the net finder returns a small hitting set. This is done using iterative reweighting.

The algorithm starts with a uniform weight function and a constant c' which the algorithm guesses to be close to OPT. The net finder is used to find an ε-net for $\varepsilon = 1/2c'$. If there is a range in $\mathcal{R}$ not hit by the ε-net, the algorithm picks such a range and doubles the weight of every element in it. It then repeats, finding a new ε-net given the new weight function. This continues until the algorithm finds an ε-net that is a hitting set. The approximation factor of the algorithm depends on the size of the ε-nets returned by the net finder. If the net finder constructs ε-nets of size $f(1/\varepsilon)$, their main algorithm finds a hitting set of size $f(4 \cdot \text{OPT})$.

Example 2.2 (Unit disks revisited)**.** Matoušek *et al.* (77) developed a net finder for range spaces obtained from points and unit disks as in Example 2.1. The net finder builds ε-nets of size $\mathcal{O}(1/\varepsilon)$. The technique of Brönnimann and Goodrich therefore yields a constant-factor approximation algorithm for the problem of covering points with unit disks.

Bounding the number of iterations.

For now assume we know the value of OPT and we set $\varepsilon = \frac{1}{2 \cdot \text{OPT}}$. We give an upper bound for the number of doubling iterations the algorithm can perform. Each iteration increases the total weight $w(X)$ by no more than a multiplicative factor of $(1+\varepsilon)$ (since the range whose weight we double has at most an ε proportion of the total weight). Therefore, if each element in X starts with weight 1, after k iterations the weight has increased to at most

$$|X| \cdot (1+\varepsilon)^k \;\leq\; |X| \cdot \exp\left(\frac{k}{2 \cdot \text{OPT}}\right) \;\leq\; |X| \cdot 2^{\left(\frac{3k}{4 \cdot \text{OPT}}\right)} \; .$$

Let $\mathcal{H} \subseteq G$ be an optimal hitting set of size OPT. For an element $h \in \mathcal{H}$ define z_h as the number of times the weight of h has been doubled. Since $\mathcal{H}$ is a hitting set, in each iteration some element in $\mathcal{H}$ has its weight doubled, so we have

$$\sum_{h \in \mathcal{H}} z_h \geq k$$

and

$$\begin{aligned} w(\mathcal{H}) &= \sum_{h \in \mathcal{H}} 2^{z_h} \\ &\geq \text{OPT} \cdot 2^{\left(\frac{k}{\text{OPT}}\right)} \qquad \text{(since } 2^x \text{ is a convex function).} \end{aligned}$$

We now have

$$\text{OPT} \cdot 2^{\left(\frac{k}{\text{OPT}}\right)} \;\leq\; w(\mathcal{H}) \;\leq\; w(X) \;\leq\; |X| \cdot 2^{\left(\frac{3k}{4 \cdot \text{OPT}}\right)} \; ,$$

which gives us

$$k \leq 4 \cdot \text{OPT} \cdot \log\left(\frac{|X|}{\text{OPT}}\right) \; .$$

This bound also tells us that the total weight $w(X)$ never exceeds $\frac{|X|^4}{\text{OPT}^3}$.

We must now address the fact that the value of OPT is unknown. We maintain a variable c' which is our guess at the value of OPT, starting with $c' = 1$. If the algorithm runs for more than $4 \cdot c' \cdot \log\left(\frac{|X|}{c'}\right)$ iterations without

obtaining a hitting set, this implies that there is no hitting set of size c' so we double our guess. When our algorithm eventually obtains a hitting set, we have $\text{OPT} \leq c' \leq 2 \cdot \text{OPT}$. The hitting set obtained is a $\left(\frac{1}{2c'}\right)$-net built by our net finder.

Considering the LP

Let $\mathcal{S} = (X, \mathcal{R})$ be a fixed finite range space. For any non-negative weight function w there is some maximum $\varepsilon_{max} = \varepsilon_{max}(w) \geq 0$ for which every ε_{max}-net is a hitting set. To get a small hitting set from a net finder we must find a weight function w with a large ε_{max}. Even *et al.* (45) observed that this has a very natural LP formulation:

$$\begin{array}{lrcll} \text{Maximize} & \varepsilon & & & \\ \text{subject to} & w(R) & \geq & \varepsilon & \forall R \in \mathcal{R} \\ & w(X) & = & 1 & \\ & w(x) & \geq & 0 & \forall x \in X\ . \end{array} \tag{2.1}$$

Note that we have normalized the weight function. Even *et al.* solve the LP approximately, then make a single call to the net finder. Using, *e.g.*, the oblivious rounding technique of Young (104), solving the LP can be avoided altogether and this method becomes faster than that of Brönnimann and Goodrich, especially when the net finder or verifier is slow.

2.3 VC-Dimension

Origins and Definition

VC-dimension is named after Vapnik and Chervonenkis, who originally defined and used it (98). Consider a range space $\mathcal{S} = (X, \mathcal{R})$ and a subset $Y \subseteq X$. We have $\mathcal{R} \cap Y \subseteq 2^Y$, *i.e.*, the intersection $\mathcal{R} \cap Y = \{R \cap Y : R \in \mathcal{R}\}$ is a collection of subsets of Y contained in the power set 2^Y.

Definition 2.6 (**Shattered set).** We say that $Y \subseteq X$ is *shattered* by $\mathcal{R}$ if $\mathcal{R} \cap Y = 2^Y$. In other words, every possible subset of Y can be obtained from the intersection of Y and some $R \in \mathcal{R}$.

Definition 2.7 (**VC-dimension).** For a range space $\mathcal{S} = (X, \mathcal{R})$, let Y be a maximum cardinality subset of X that is shattered by $\mathcal{R}$. Then the *VC-dimension* of $\mathcal{S}$, or simply the *dimension* of $\mathcal{S}$, is equal to $|Y|$.

We illustrate this with a classical geometric example.

Example 2.3. Consider a range space $\mathcal{S} = (X, \mathcal{R})$ where X is a set of points in $\mathbb{R}^d$ and $\mathcal{R}$ is a collection of closed half-spaces. The VC-dimension of this range space is at most $d + 1$.

Proof. We prove this by induction on d. In $\mathbb{R}^1$, a set of 3 points cannot be shattered since no half-line can contain the median without containing another point. In general d, assume for the sake of contradiction that there is a subset $Y \subseteq X$ of $d + 2$ points that can be shattered. If Y is not in general position then there is a $(d - 1)$-flat containing $d + 1$ points of Y, so by induction Y cannot be shattered. If Y contains a point strictly in the interior of its convex hull, any half-space containing that point must also contain a point on the convex hull. If all $d + 2$ points are on the convex hull, then the convex hull is not a simplex. In this case there must be two vertices that are non-adjacent on the convex hull; no half-space can separate these two points from the other d points. □

Claim 2.1. *If $\mathcal{S} = (X, \mathcal{R})$ is a range space of dimension d, then for $X' \subseteq X$ and $\mathcal{R}' \subseteq \mathcal{R}$, the range space $\mathcal{S}' = (X', \mathcal{R}')$ has dimension $d' \leq d$.*

Proof. Any subset of X' shattered by $\mathcal{R}'$ is also shattered by $\mathcal{R}$, so $d' \leq d$. □

Claim 2.2. *If $\mathcal{S} = (X, \mathcal{R})$ is a range space of dimension d, the dimension $\widehat{d}$ of the dual range space $\widehat{\mathcal{S}}$ of $\mathcal{S}$ is bounded by $\lfloor \log_2 d \rfloor \leq \widehat{d} \leq 2^{d+1} - 1$.*

Proof. We prove the lower bound for $\widehat{d}$, with the upper bound following from the dual lower bound $\lfloor \log_2 \widehat{d} \rfloor \leq d$. Let $d' = 2^{\lfloor \log_2 d \rfloor}$ be the greatest power of 2 that does not exceed d. Let $Y \subseteq X$ be a set of d' points shattered by $\mathcal{R}$ and let

$\mathcal{R}_Y \subseteq \mathcal{R}$ be a set of $2^{d'}$ ranges that shatters Y. The incidence matrix induced by Y and $\mathcal{R}_Y$ has d' rows and its columns are exactly the binary strings of length d'. It is not hard to see that we can choose $\log_2 d'$ of these columns to form a matrix whose d' rows are exactly the binary strings of length $\log_2 d'$; further, these $\log_2 d'$ columns correspond to a shattered point set in $\widehat{\mathcal{S}}$. □

One of the key points to take away from Example 2.3 is that the definition holds even for infinite sets. Loosely speaking, VC-dimension measures how different the ranges in a range space can be from each other. Before discussing the origin and significance of VC-theory we give a bound for the number of ranges in a finite range space with VC-dimension d.

Definition 2.8. The function $\Phi_d(n)$ is defined as

$$\Phi_d(n) = \begin{cases} \sum_{i=0}^{d} \binom{n}{i} & , \quad n \geq d \\ 2^n & , \quad n \leq d \ . \end{cases}$$

The following was proved independently by Sauer (92) and Vapnik and Chervonenkis (97).

Lemma 2.2. *In a finite range space $(X, \mathcal{R})$ with VC-dimension d, $|\mathcal{R}| \leq \Phi_d(|X|)$.*

Proof. The proof of this lemma depends heavily on the recurrence

$$\Phi_d(n) = \Phi_d(n-1) + \Phi_{d-1}(n-1) \tag{2.2}$$

and its natural interpretation. Given n items, $\Phi_d(n)$ is the number of ways you can choose up to d of them. Let us consider the recurrence with the n^{th} item in mind. Either we choose it, in which case we choose up to $d-1$ of the $n-1$ other items, or we don't choose it, in which case we choose up to d of the $n-1$ other items.

The lemma holds trivially for $n \leq d$. For higher values of n we prove it by induction. We show that

$$|\mathcal{R}| = \left|\mathcal{R}^{(1)}\right| + \left|\mathcal{R}^{(2)}\right| \ , \tag{2.3}$$

where $(X \setminus \{x_n\}, \mathcal{R}^{(1)})$ and $(X \setminus \{x_n\}, \mathcal{R}^{(2)})$ are range spaces on $n-1$ elements of maximum VC-dimension d and $d-1$, respectively. The lemma then follows from (2.2) by induction.

Consider a range R on $X \setminus \{x_n\}$ and the two possible 'augmented ranges' on X, namely R and $R \cup \{x_n\}$. R is in $\mathcal{R}^{(1)}$ if *at least one* augmented range is in $\mathcal{R}$. R is in $\mathcal{R}^{(2)}$ if *both* augmented ranges are in $\mathcal{R}$. It is not hard to see that these definitions satisfy (2.3).

The range space $(X \setminus \{x_n\}, \mathcal{R}^{(1)})$ must have VC-dimension at most d. The range space $(X \setminus \{x_n\}, \mathcal{R}^{(2)})$ must have VC-dimension at most $d-1$. To see this, consider a set $Y \subseteq X \setminus \{x_n\}$ that is shattered by $\mathcal{R}^{(2)}$. $Y \cup \{x_n\}$ must be shattered by $\mathcal{R}$, so Y must have size at most $d-1$. This concludes the proof. □

Vapnik and Chervonenkis used VC-dimension in their work in learning theory (98). The law of large numbers tells us that we can learn the probability of an event from a long enough sequence of independent Bernoulli trials — the sample average for the event converges almost surely to the probability of the event. Vapnik and Chervonenkis sought to extend this from a single event to a class of events.

Consider a range space $(X, \mathcal{R})$ along with a probability measure μ on X. Let $S \in X^k$ be a series of k independent trials drawn according to μ. We use $R_{(k)}$ to denote the sample average of a range R, *i.e.*, $R_{(k)} = |R \cap S| / k$. The *VC-theorem* (98) essentially tells us that we can learn *all* probabilities $\{\mu(R) : R \in \mathcal{R}\}$ from S if k is large enough. We present their main theorem in the context of range spaces and with implicit application of Lemma 2.2.

Theorem 2.1 (VC-theorem). *Let $(X, \mathcal{R})$ be a range space having VC-dimension at most d and let $S \in X^k$ be a series of k independent trials drawn according to μ. Then for any 'error' threshold $t > 0$ we have*

$$\begin{aligned} \mathbb{P}\left\{ \sup_{R \in \mathcal{R}} \left| \mu(R) - R_{(k)} \right| \geq t \right\} &\leq \Phi_d(2k) \cdot 4 \cdot \exp\left(\frac{-kt^2}{8} \right) \\ &= \mathcal{O}\left(\exp\left(d \ln 2k - \frac{kt^2}{8} \right) \right) . \end{aligned}$$

Another version of the inequality due to Devroye (25) states that

$$\begin{aligned}\mathbb{P}\left\{\sup_{R\in\mathcal{R}}\left|\mu(R)-R_{(k)}\right|\geq t\right\} &\leq \Phi_d(k^2)\cdot 4e^{4t+4t^2}\cdot\exp\left(-2kt^2\right)\\ &= \mathcal{O}\big(\exp\left(2d\ln k-2kt^2\right)\big)\ .\end{aligned}$$

It is difficult to overstate the significance of this theorem. It bounds the 'error', *i.e.*, the difference between the estimated probability and the true probability, for all ranges simultaneously, *even when the number of ranges is infinite.* It is not the number of ranges that affects this probability bound, but rather the complexity of their interaction as quantified by VC-dimension.

Small ε-Nets via Random Sampling

The VC-theorem can be used to show that, for range spaces having small VC-dimension, small ε-nets can be obtained via random sampling. The following is a straightforward consequence of the VC-theorem.

Corollary 2.1. *For a range space $(X,\mathcal{R})$ of dimension d and a constant failure probability $\delta>0$, a random sample of k elements from X is an ε-net with probability at least $1-\delta$ for some $k=\mathcal{O}\big(\frac{d}{\varepsilon^2}\log\frac{d}{\varepsilon}\big)$.*

In their paper introducing ε-nets, Haussler and Welzl (54) proved a significantly stronger result.

Theorem 2.2. *For a range space $(X,\mathcal{R})$ of dimension d and a constant failure probability $\delta>0$, a random sample of k elements from X is an ε-net with probability at least $1-\delta$ for*

$$k\geq\max\left(\frac{4}{\varepsilon}\ln\frac{2}{\delta}\ ,\ \frac{8d}{\varepsilon}\ln\frac{8d}{\varepsilon}\right)\ .$$

In particular, this means that random sampling can be used to obtain ε-nets of size $\mathcal{O}\big(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\big)$ in $\mathcal{O}\big(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\big)$ expected time.

However, this bound of $\mathcal{O}\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$ is not quite tight. An improved upper bound of $\mathcal{O}\left(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ was given by Blumer *et al.* (15). The proof is essentially the same but uses the following tighter bound on the function $\Phi_d(n)$:

Proposition 2.1.

$$\Phi_d(n) = \mathcal{O}\left(\frac{n^d}{d!}\right) = \mathcal{O}\left(\left(\frac{en}{d}\right)^d\right) \ .$$

Proof sketch. The first inequality is proved combinatorially by induction. The second inequality is a consequence of Stirling's approximation. A complete proof appears in (15, pp. 957–958). □

Lower Bounds for ε-Nets

Though the analysis used to bound the size of ε-nets obtained via random sampling is quite involved, the sampling method itself is trivial. It is therefore natural to ask if another method can do better. Komlós *et al.* (66) proved that in general this is not possible.

Lemma 2.3. *For any $d \geq 2$ there exists a range space of dimension d such that, for any sufficiently small ε, any ε-net must contain at least $\Omega\left(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ points.*

The proof of this lemma involves a randomly constructed range space $\mathcal{S} = (X, \mathcal{R})$, where X contains $\Theta\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ points and each of the $2^{|X|}$ possible ranges is added to $\mathcal{R}$ independently with probability p. For an appropriate choice of p, they are able to prove that if ε is sufficiently small, then with high probability $\mathcal{S}$ has VC-dimension at most d and $\mathcal{S}$ does not admit an ε-net of size t for some $t = \Theta\left(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon}\right)$.

One drawback of the random construction of Komlós *et al.* is that it has no natural geometric interpretation. For many years it was an open problem whether range spaces with natural geometric interpretations exist that do not admit ε-nets of size $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$. This was recently answered in the affirmative by Alon (9).

Lemma 2.4. *Let $\mathcal{S}(X) = (X, \mathcal{R})$ be a range space defined by a set X of points in $\mathbb{R}^2$ where $\mathcal{R}$ contains all intersections of lines with X. Then for every positive constant c there exists a set X and some $\varepsilon > 0$ such that any ε-net for $\mathcal{S}(X)$ is larger than c/ε.*

Alon's lower bound only grows like $\Theta\big(\frac{1}{\varepsilon} \cdot w\big(\frac{1}{\varepsilon}\big)\big)$, where w is a version of the inverse Ackermann function. It is therefore still unknown whether natural geometric range spaces can match the lower bound of $\Theta\big(\frac{d}{\varepsilon} \log \frac{1}{\varepsilon}\big)$.

Parameterized Complexity

Fixed-parameter tractability concerns NP-complete problems that have polynomial time algorithms under certain restrictions. The theory of fixed-parameter tractability, and more generally the theory of parameterized complexity, arose in the 1990s and was solidified by the work of Downey and Fellows (38).

Definition 2.9 (**Fixed-parameter tractable).** A problem is fixed-parameter tractable for a specified parameter if it can be solved in time

$$f(k) \cdot n^{\mathcal{O}(1)} \ ,$$

where n is the length of the input, k is the value of the parameter, and f is a function depending only on k (in particular $f(k)$ has no dependence on n).

Optimization problems are often parameterized by OPT, the size of the optimal solution. It may be desirable to have multiple parameters; in this case we can adhere to the above definition by using the sum of parameters as a single parameter. The following example generalizes vertex cover.

Example 2.4. Consider a range space $\mathcal{S} = (X, \mathcal{R})$ such that every range contains at most k elements. Hitting set is fixed-parameter tractable when parameterized by $(\text{OPT} + k)$.

Proof sketch. This problem admits a simple algorithm that uses the bounded search tree method frequently seen in parameterized algorithms. Consider an

algorithm that starts with $H = \emptyset$ and augments H at each step by choosing the lowest-indexed range R_i that is not hit by H, then adding an element from R_i to H. An optimum hitting set is built by such an algorithm as long as the right element from R_i is chosen at each step.

Since each range has at most k elements, the algorithm has at most k options at each step. Since there is some hitting set of size OPT, a breadth-first search of the algorithm's decision tree finds an optimal hitting set at level OPT of the tree. Exhaustive breadth-first search of the decision tree can therefore find an optimum hitting set in $\mathcal{O}(k^{\text{OPT}})$ time, along with overhead that is polynomial in the input size. □

The complexity class FPT contains all fixed-parameter tractable problems. The W hierarchy is a collection of complexity classes that contain FPT but are not believed to be contained in FPT, where

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \text{W[3]} \subseteq \ldots \subseteq \text{W[P]} .$$

FPT and W[P] are parameterized analogues of P and NP respectively, and FPT $\neq$ W[P] if and only if P $\neq$ NP.

The more interesting classes are at the bottom of the hierarchy. It is known that FPT $\neq$ W[1] unless NP $\subseteq$ DTIME($2^{o(n)}$). Therefore proving W[1]-hardness suggests that a problem is unlikely to be fixed-parameter tractable. When parameterized by OPT, the independent set and hitting set problems are complete for W[1] and W[2] respectively. The W hierarchy is discussed in detail by Downey and Fellows (38).

We are interested in the fixed-parameter tractability of problems restricted to range spaces of bounded VC-dimension. In particular we are interested in the hitting set problem parameterized by OPT. The existence of a general parameterized algorithm for hitting set in bounded VC-dimension would imply that a multitude of geometric hitting problems are in FPT. The following observation is discouraging for range spaces of dimension ≥ 4.

Observation 2.1. The hitting set problem is W[1]-hard when parameterized by OPT, even when restricted to range spaces of VC-dimension $d \geq 4$.

Proof. This lemma is a simple consequence of a related hardness result. Dom *et al.* (37) recently proved that the problem of stabbing axis-parallel rectangles in $\mathbb{R}^2$ with axis-parallel lines (both vertical and horizontal) is W[1]-complete when parameterized by OPT. This problem is an instance of hitting set in a range space $\mathcal{S} = (X, \mathcal{R})$, where each $x \in X$ contains a line and each $R \in \mathcal{R}$ represents a rectangle. To prove the lemma we must show that the VC-dimension of $\mathcal{S}$ is at most 4. We consider a set of 5 lines and prove that they cannot be shattered. Assume without loss of generality 3 or more of the lines are vertical; otherwise 3 or more are horizontal and the proof is the same. A set of 3 vertical lines, call them ℓ_1, ℓ_2, ℓ_3 from left to right, cannot be shattered since any rectangle hitting ℓ_1 and ℓ_3 must also hit ℓ_2. □

Unfortunately the above lemma does not tell us whether hitting set is actually in W[1] when the VC-dimension is restricted to 4; neither does it tell us whether hitting set is in FPT when restricted to some lower dimension.

Observation 2.2. The hitting set problem is in P when restricted to range spaces of VC-dimension 1.

Proof. Let $\mathcal{S}$ be a range space of dimension 1. If the incidence matrix of $\mathcal{S}$ is totally balanced (see, *e.g.*, (55)) then a minimum hitting set can be found in polynomial time.

If the matrix is not totally balanced, the rows and columns can be permuted so that the submatrix $\left[\begin{smallmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{smallmatrix}\right]$ appears. This submatrix is induced by 2 points $\{x_i, x_j\}$ and 3 ranges. If some range contains neither x_i nor x_j then $\{x_i, x_j\}$ is shattered. But this is impossible since $\mathcal{S}$ has dimension 1, so $\{x_i, x_j\}$ must hit every range. Therefore a minimum hitting set can be found in polynomial time by checking all sets of size 1 and 2. □

2.4 Notes

Contributions

This chapter does not contain any significant contributions. Observations 2.1 and 2.2 are possibly new but are easy. The rectangle stabbing problem in

the proof of Observation 2.1 was suggested by Michael Fellows (47) as having constant VC-dimension while being W[1]-hard. The VC-dimension in Example 2.3 is often mentioned but seems to be folklore. Our proof was developed independently but is completely straightforward and probably very similar to previous proofs.

Chapter 3

Guarding Terrains

This chapter deals entirely with the terrain guarding problem. We first discuss the range spaces associated with terrain guarding and give a tight bound on the maximum VC-dimension of such range spaces. We then give an overview of approximation algorithms for the problem. Finally, we resolve a long-standing open problem by proving that terrain guarding is NP-complete.

Contents

3.1 Terrain Guarding Preliminaries

An instance of the *terrain guarding problem* consists of a terrain T that is an x-monotone polygonal chain. An x-monotone chain in $\mathbb{R}^2$ is a chain that intersects any vertical line at most once. The terrain is given by its set of vertices $V = \{v_1, v_2, ..., v_n\}$, where $v_i = (x_i, y_i)$. The vertices are ordered such that $x_i < x_{i+1}$. There is an edge connecting each (v_i, v_{i+1}) pair where $i = 1, 2, ..., n-1$. We say a point p on the terrain *sees* another point q on the terrain if no point on the line segment $\overline{pq}$ lies strictly below the terrain T. We denote this by $p \sim q$.

These terrains are sometimes called 1.5-dimensional terrains since they exist in $\mathbb{R}^2$ but do not have full 2-dimensional freedom like unrestricted polygonal chains. The analogous structures in $\mathbb{R}^3$ are called 2.5-dimensional terrains, or *polyhedral terrains*. These are polyhedral terrains of genus 0 (*i.e.*, having no holes) that intersect any vertical line at most once. All terrains we deal with in this chapter are 1.5-dimensional unless clearly stated otherwise.

A set G of points on a terrain is called a *guarding set* if every point on the terrain is seen by some point in G. The optimization version of the terrain guarding problem is the problem of finding a minimum guarding set for a given terrain.

Motivation for guarding terrains comes from scenarios that include covering a road with street lights or security cameras. Other applications include finding a configuration for line-of-sight transmission networks for radio broadcasting, cellular telephony and other communication technologies (12). A terrain is a natural model for an approximately linear border whose altitude varies, for example the large portion of the Canada/U.S. border that follows the 49th parallel.

For points a, b on a terrain, we say that $a < b$ if a is strictly to the left of b. One of the most important structural restrictions of terrains is stated by the *order claim*, first noted by Ben-Moshe *et al.* (12).

Claim 3.1 (**Order Claim**). *Let a, b, c, d be four points on the terrain such that $a < b < c < d$. If a sees c and b sees d, then a sees d.*

Proof. Line segments $\overline{ac}$ and $\overline{bd}$ cross and must intersect at some point p. No point of the triangle $\triangle apd$ can be below the terrain; since this triangle includes the line segment $\overline{ad}$, a must see d. □

3.2 Range Spaces and Approximation

Let X be the set of points on the terrain that must be guarded and let G be the set of potential guard locations. For a guard $g \in G$, define the range $R(g) \subseteq T$ as $\{x \in X : g \sim t\}$. Guarding the points in X with guards from G is equivalent to set cover on the range space $(X, \{R(g) : g \in G\})$, or equivalent to hitting set on the dual range space. This range space can also be defined by the visibility matrix for X and G in which there is a row for each $x \in X$, a column for each $g \in G$, and the cell for the pair (x, g) contains a 1 if $x \sim g$ and a 0 otherwise.

We are particularly interested in the case where X is a finite set and $X = G$; in this case we say that the range space is *induced* by the set X of points on the terrain. Such a range space is self-dual since its incidence matrix is self-transpose, or symmetric, by symmetry of visibility. This case is most interesting when X contains a minimum guarding set for T and any set guarding X also guards T. In this case we say that X constitutes a *parsimonious* discretization since optimization algorithms can use X instead of T without adverse effects.

Discretization

There are two standard versions of the terrain guarding problem: a vertex version and a continuous version. The vertex version allows us to place guards only at the vertices of the terrain. The continuous version, which we have defined above, allows guards to be placed anywhere on the terrain. In other versions a subset of points on the terrain to guard is given with the input.

Before proceeding, we define notation for ray shooting. For a vertex v and arbitrary points p, q on the terrain, consider the ray emanating from p and passing through v. If p sees v and q is the first point in T hit by the ray after v, then we say that the ray shot from p through v hits the terrain at q. We denote this $p \rightsquigarrow v = q$ and note that this defines a partial map from $T \times V$ to T. For sets $X \subseteq T$ and $Y \subseteq V$, we use $X \rightsquigarrow Y$ to denote the set $\{x \rightsquigarrow y : x \in X, y \in Y\}$. Clearly $|X \rightsquigarrow Y| \leq |X| \cdot |Y|$. Note that $p \rightsquigarrow v = q$ implies that either $p < v < q$ or $q < v < p$, where $p < q$ means that p is to the left of q.

Ben-Moshe *et al.* (12, §6) provide a fairly simple discretization technique. Discretization is performed as follows. Let $U' = V \cup (V \rightsquigarrow V)$. U' partitions T into $\mathcal{O}(n^2)$ intervals such that all points in an interval are seen by the same subset of V; let U'' contain one representative point from each of these intervals. Any set of vertices guarding $U = U' \cup U''$ must guard the entire terrain. The authors then note that any guard on an edge (v_i, v_{i+1}) is dominated by the endpoints $\{v_i, v_{i+1}\}$, so if the minimum guarding set has size OPT there is a guarding set of size $2 \cdot \text{OPT}$ contained in V.

This factor of 2 is prohibitive for some applications of discretization. We eliminate this problem with the following lemma.

Lemma 3.1. *$W = V \cup (V \rightsquigarrow V) \cup ((V \rightsquigarrow V) \rightsquigarrow V)$ is a set of $\mathcal{O}(n^3)$ points that contains an optimum guarding set for T.*

Proof. It is clear that $|W| = \mathcal{O}(n^3)$. We say that a guarding set for T is *forced right* if no non-vertex guard can slide to the right. That is, no guard in the relative interior of an edge can be replaced by another guard on the same edge that is further to the right. There exists an optimum guarding set that is forced right, and we show that any guarding set that is forced right is contained in W.

Let G be a guarding set for T that is forced right. Assume for the sake of contradiction that $G \setminus W$ is non-empty and let g be the rightmost guard in $G \setminus W$. If g is not a vertex and cannot be moved to the right then there must be a vertex v and a point $p \in T \setminus U$ such that $g \rightsquigarrow v = p$ and $g < v < p$. g sees the half edge (p, v_{i+1}) and some other guard to the right of p must see the

other half edge (v_i, p). Let g' be the rightmost guard that sees p. There is a vertex v' such that $p \rightsquigarrow v' = g'$ and $p < v' < g'$.

We know that g' cannot be a vertex; otherwise g would be in W. Since we cannot slide g' to the right, there must be a vertex v'' such that $g' \rightsquigarrow v'' = p'$ and $g' < v'' < p'$. Some guard g'' is responsible for the left half-edge ending at p'. But now we have a contradiction. We know that neither p' nor g' is a vertex. Therefore the quadrilateral (p, g', p', g'') is either a line or is convex. We know that the lower hull of the quadrilateral does not pass below the terrain since $p \sim g' \sim p' \sim g''$. Therefore $p \sim g''$, but g' is defined as the rightmost guard that sees p. Having arrived at a contradiction it must be the case that $G \subseteq W$. □

The technique of Ben-Moshe *et al.*, starting from W instead of V, can now be used to create a complete parsimonious discretization using $\mathcal{O}(n^4)$ points. The resulting set, call it Z, is such that any set of points that guards Z guards the terrain, and Z contains an optimum guarding set for T.

The closely related yet more troublesome issue of discretizing polygons is discussed in Section 4.1.

VC-Dimension of Terrain Guarding

Here we prove the following lemma, a result of independent research (61).

Lemma 3.2. *The VC-dimension of a range space induced by points on a terrain is at most 4 and this bound is tight.*

Proof. To prove that a monotone chain can have VC-dimension 4, we simply provide an example of a terrain with 4 points that are shattered by 16 guards. Such a terrain is shown in Figure 3.1.

We use the order claim to argue that no set P of 5 points on a terrain can be shattered by a set G of 32 guards. This gives us the upper bound of 4 for the VC-dimension.

Let $P = \{a, b, c, d, e\}$ and let, for example, $g(b, d)$ denote the guard in G that sees b and d but not a, c, or e. Note that neither P nor G can contain duplicate points, and that if $g = p$ for some $g \in G$, $p \in P$, then $g \sim p$. Assume

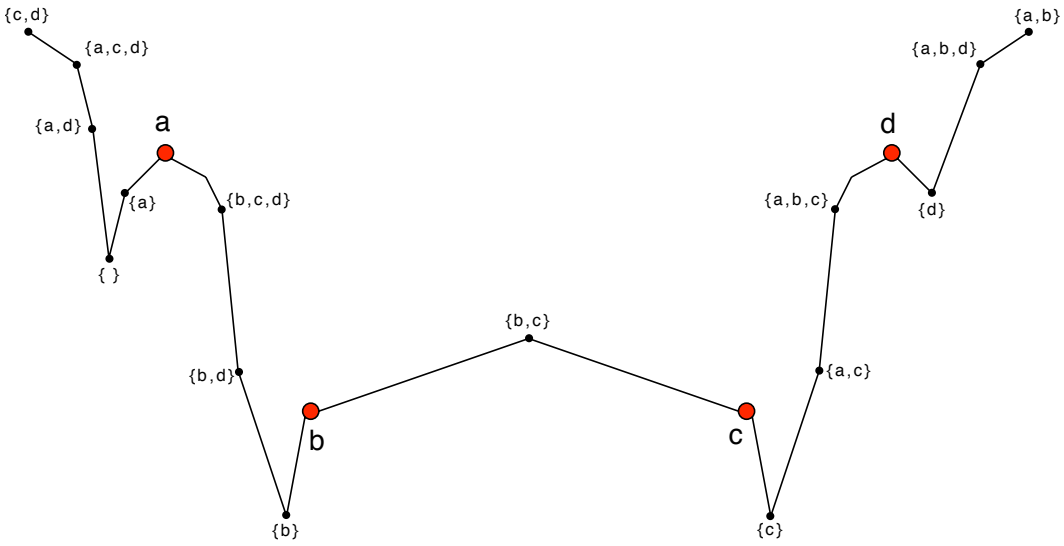


Figure 3.1: A monotone chain with 4 points, a, b, c, d, that are shattered by 16 guards. The guard seeing $\{a, b, c, d\}$ is not pictured, but a very high vertex on the left end of the terrain would see all other vertices. Each of the other 15 guards is labeled with the subset of $\{a, b, c, d\}$ that it sees.

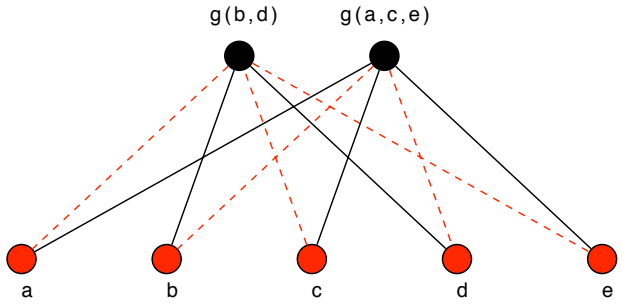
without loss of generality that $a < b < c < d < e$. We can see (Figure 3.2 may help) that $g(a, c, e)$ and $g(b, d)$ contradict the order claim unless either

- $g(b, d) < c < d < g(a, c, e)$, or
- $g(a, c, e) < b < c < g(b, d)$.

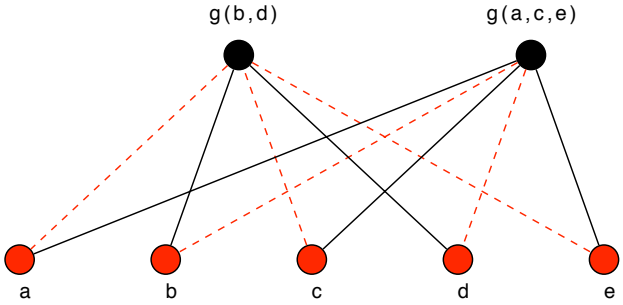
Noting that these two cases are symmetric, we assume the former without loss of generality. Now consider $g(b, c, e)$. There are four potential ranges that we consider placing $g(b, c, e)$ in:

- left of $g(b, d)$
- between $g(b, d)$ and d
- between d and $g(a, c, e)$
- right of $g(a, c, e)$.

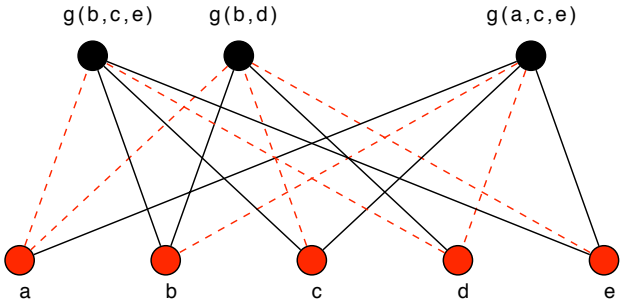
It is not difficult to verify that placing $g(b, c, e)$ in any of these four ranges would contradict the order claim (see Figure 3.2 for an example). Therefore 5 points on a monotone chain cannot be shattered and no monotone chain can have VC-dimension greater than 4. □



(a) In this configuration the Order Claim is contradicted by $g(b,d)$, $g(a,c,e)$, d, and e.



(b) In this configuration the Order Claim is not contradicted.



(c) The Order Claim is now contradicted by the addition of $g(b,c,e)$, regardless of its position. In this configuration the Order Claim is contradicted by $g(b,c,e)$, $g(b,d)$, c, and d.

Figure 3.2: Examples of configurations of G and P for the proof that no 5 points on a 1.5D terrain can be shattered. Solid lines indicate clear lines of sight. Dashed lines indicate blocked lines of sight.

A Note on Polyhedral Terrains

We now give a simple reduction proving that polyhedral (*i.e.*, 2.5-dimensional) terrains have unbounded VC-dimension. This was first observed by Efrat and Har-Peled (41).

Lemma 3.3. *The VC-dimension of range spaces associated with polyhedral terrains is unbounded.*

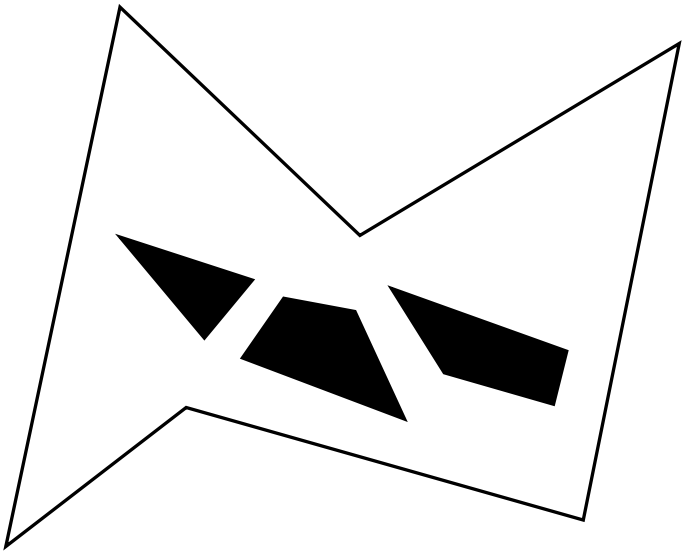
Proof. Set cover can be reduced to the problem of guarding the perimeter of a polygon with holes using guards on the perimeter (see Eidenbenz *et al.* (43, §4)). As a direct consequence, for any finite range space $(X_1, \mathcal{R}_1)$, there exists a polygon with holes whose associated range space is $(X_2, \mathcal{R}_2)$ such that $X_1 \subseteq X_2$ and $\mathcal{R}_1 \subseteq \mathcal{R}_2$. This implies that a polygon with holes can have arbitrarily large VC-dimension.

For any polygon A with holes we show how to construct a polygonal terrain of equal or greater VC-dimension. The idea behind building T is simple. Lines of sight between points on A are blocked by the exterior of A. On our terrain T we build corresponding mountains to block lines of sight.

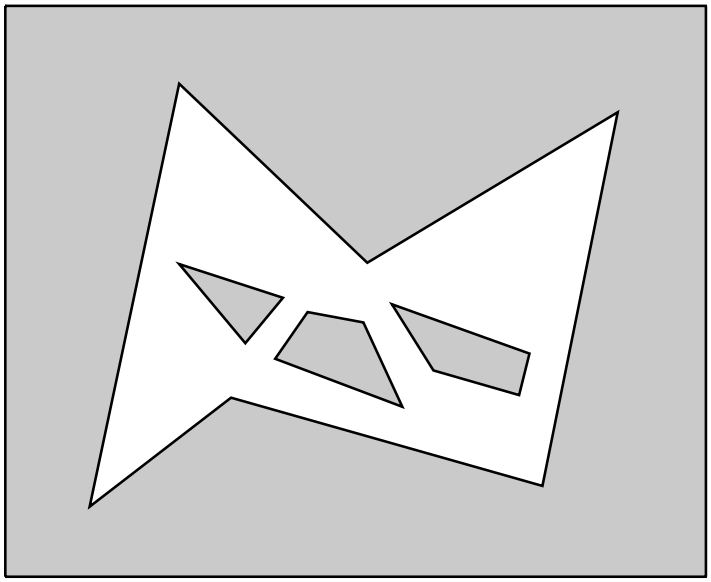
We start with T as a horizontal rectangle at altitude 0 that acts as a bounding box for A. We then trace the perimeter of A on this rectangle and call it A_T. A_T partitions T into two open sets, T^- which corresponds to the interior of A and T^+ which corresponds to the exterior of A, including the holes.

In terms of vertical projections, A_T, T^- and T^+ remain fixed as we change T. However, T^- is lowered and T^+ is raised. There are many ways to perform this raising and lowering, but perhaps the most elegant is the method of raising roofs from *straight skeletons* (Aichholzer and Aurenhammer (4), in particular §4). We raise T^+ based on its straight skeleton and lower T^- based on its straight skeleton. The result is that every point in T^+ has positive altitude and every point in T^- has negative altitude. Only A_T and the rectangular perimeter of T are at altitude 0. See Figure 3.3 for an example.

We can now verify that two points p, q on A_T see each other if and only if the corresponding points p', q' on A see each other. Since p and q are both



(a) The polygon A with holes indicated in black.



(b) A simplified top view of the associated terrain T. Black lines indicate A_T and the terrain's perimeter, both at altitude 0. T^- (white) has negative altitude while T^+ (shaded) has positive altitude.

Figure 3.3: A polygon A and a top view of the associated terrain T.

at altitude 0, all of (p, q) is at altitude 0. If p sees q then the open line segment (p, q) contains no point below T so no point on (p, q) can be the vertical projection of a point in T^+. The corresponding open line segment (p', q') therefore cannot intersect the exterior of A, so p' and q' must see each other. Therefore p sees q if and only if p' sees q', and the converse can be proved similarly.

From any polygon with holes, we have shown how to construct a 2.5-dimensional terrain with equal or greater VC-dimension, so 2.5-dimensional terrains have unbounded VC-dimension. □

Approximation Algorithms

Approximation of terrain guarding has been an open problem of interest since 1995, when an NP-completeness proof was proposed but never completed by Chen *et al.* (20). With the problem's hardness strongly suspected but not known, a series of approximation algorithms have been developed over the last decade. In this section we give an overview of these algorithms.

Generic Approximation via Hitting Set

The terrain guarding problem can be formulated as an instance of hitting set. Let X be the set of potential guard locations and let $\mathcal{R}$ contain the subsets of X that are seen by points on the terrain. The terrain guarding problem is now equivalent to hitting set for $\mathcal{S} = (X, \mathcal{R})$. Lemma 3.2 tells us that $\mathcal{S}$ has VC-dimension at most 4. Generic approximation techniques for range spaces of bounded dimension can then be used to obtain an $\mathcal{O}(\log \text{OPT})$-approximation algorithm.

Constant Factor Approximations

The first constant factor approximation for the terrain guarding problem was given by Ben-Moshe *et al.* (12). Their primary strategy was divide and conquer. They worked towards recursively breaking the terrain into subterrains that could be handled independently. When it is known that two disjoint

subterrains both require internal guards (*i.e.*, the relative interiors of both subterrains must contain guards), a constant number of guards can be used to 'separate' them, *i.e.*, ensure that future work towards guarding one of them can safely ignore the other. They did not explicitly bound the approximation factor of their algorithm. A guaranteed approximation factor of 5 was attained by King (60); the algorithm repeatedly searches for a particular unguarded point x, then places a set of 5 guards that are guaranteed to dominate any guard that sees x.

Considering the LP

Elbassioni *et al.* (44) considered several LPs associated with terrain guarding. In particular, they considered the restricted problem in which guards can only look left. Chen *et al.* (20) proved that an optimal solution for this problem can be found by a simple greedy algorithm. Elbassioni *et al.* revisit this problem, considering the 'visibility looking left' incidence matrix for a set X of potential guard locations and a set Y of points to guard. They prove that if X and Y are disjoint, the matrix is totally balanced. Balanced matrices, introduced by Berge (14), generalize edge-vertex incidence matrices of bipartite graphs while totally balanced matrices, a subclass, generalize those of acyclic graphs. Linear programs with totally balanced constraint matrices can be solved combinatorially in polynomial time (55).

The general terrain guarding problem can be approximated using left-looking guards and right-looking guards, but only if we know which points should be guarded from the left and which should be guarded from the right. Elbassioni *et al.* used linear programming to determine this. After finding an optimum fractional guarding set, they partition points into those guarded mostly from the right and those guarded mostly from the left. For each set they then find an optimum one-sided guarding set. Putting the pieces of their algorithm together they achieve an approximation factor of 4 for most cases; they also handle non-uniform cost functions on the guards as well as budgeted variants of the terrain guarding problem.

A PTAS via Local Search

Recently a PTAS for the terrain guarding problem was given by Gibson *et al.* (53). In fact, the algorithm is a trivial local search algorithm that starts with any valid guarding set and, while possible, improves the guarding set by removing up to b guards and replacing them with a smaller number of guards. The running time is bounded by $n^{\mathcal{O}(b)}$.

The interesting part is not the algorithm but rather the analysis. This follows the paradigm recently introduced by Chan and Har-Peled (18) and Mustafa and Ray (82). For an appropriately chosen $b = \mathcal{O}(1/\epsilon^2)$ the algorithm is a $(1+\epsilon)$-approximation. The proof of this fact relies on the existence of a planar graph relating the current guarding set X and an optimum guarding set Y. Let $X' = X \setminus Y$ and $Y' = Y \setminus X$. Consider the bipartite graph whose vertex set is $X' \cup Y'$ and in which $x \in X'$ is adjacent to $y \in Y'$ if and only if x and y see a common point that is not guarded by $X \cap Y$. Gibson *et al.* proved that this graph is planar.

This graph can then be considered in the light of separator theorems for planar graphs (*e.g.*, the original by Lipton and Tarjan (71) or the more useful generalization of Frederickson (49)). The conclusion is that, if no local improvement exists, then X' is not much bigger than Y' and therefore X is not much bigger than Y. It is worth emphasizing that the graph is never constructed; its existence is enough to guarantee the approximation factor of the naïve search algorithm.

3.3 NP-Completeness

In this section we present the NP-completeness result developed in joint work with Erik Krohn (63).

In order to prove NP-completeness of terrain guarding we develop a sophisticated reduction from planar 3SAT that overcomes the obstacle presented by the order claim. Therefore an exact polynomial time algorithm is not possible unless P = NP. We actually prove that terrain guarding is *strongly* NP-complete, which means in particular that, while there might be a PTAS for the

problem (and in this case there is), there cannot be an FPTAS unless P = NP (50).

The order claim is crucially exploited by all approximation algorithms for the problem. In this section we develop a construction that overcomes the order claim obstacle and shows that the terrain guarding problem is NP-hard. Therefore, an exact polynomial time algorithm is not possible unless P = NP. The NP-hardness result is shown for the standard discrete and continuous variants of the problem. Here we state the result and sketch the proof; the rest of the section is dedicated to the full proof.

Theorem 3.1. *Minimum terrain guarding is strongly NP-hard.*

Proof. Let $\Phi = (X, C)$ be a Boolean formula in 3-CNF with $|X| = n$ and $|C| = m$. Specifically, we require that Φ is a *planar* 3-CNF formula (see Definition 3.1). In our reduction we construct in polynomial time a terrain T_Φ that can be guarded by $f(\Phi)$ guards if and only if Φ is satisfiable. Here f is a function mapping planar 3-CNF formulae to the natural numbers, such that $f(\Phi)$ is polynomial in n and is computable in time polynomial in n.

T_Φ is built in several steps. First, we build a path representation of Φ, denoted P_Φ, in which each node stores a list of variables. The relationship between variable lists in adjacent nodes is strictly defined. Some nodes are specially marked as clause nodes or deletion nodes. This path representation, along with the PLANAR 3-SAT problem, is discussed shortly.

From the path representation P_Φ we then construct T_Φ. As this is an intricate process we separate the construction into two parts. First we explain how to construct a terrain for n variables such that any minimum guarding set corresponds to a consistent truth assignment of the variables. Secondly we explain how additional gadgets are incorporated into such a terrain to construct T_Φ such that any guarding set of size $f(\Phi)$ corresponds to a consistent truth assignment satisfying Φ. Our reduction has polynomial time complexity, and Lemmas 3.4 and 3.5 tell us that T_Φ can be embedded on a grid of polynomial size. □

Planar 3SAT and Path Representations

In this subsection we introduce the PLANAR 3-SAT problem. We then describe how to express an instance of this problem in a format that lends itself naturally to embedding in a terrain using our truth assignment propagation framework. The clause and inversion gadgets used for the embedding process are described last.

Defining PLANAR 3-SAT

Many variants of 3-SAT are NP-hard; reductions from restricted variants are sometimes far simpler than reductions from general 3-SAT. In particular, PLANAR 3-SAT is often used to prove NP-hardness of geometric problems (see, *e.g.*, (81)). We adapt the following definition from Mulzer and Rote (81).

Definition 3.1 (PLANAR 3-SAT)**.** Let Φ be a Boolean formula in 3-CNF. The *formula graph* of Φ, $G(\Phi)$, has one variable-vertex v_x for each variable x and one clause-vertex v_C for each clause C. The variable-vertices v_x are connected by edges to form a *variable cycle*, and for each clause-vertex v_C an edge (v_C, v_x) is added if C contains either literal x or $\overline{x}$. We say Φ is **planar** iff $G(\Phi)$ is planar. The PLANAR 3-SAT problem is equivalent to the 3-SAT problem restricted to planar formulae.

Theorem 3.2 (Lichtenstein (68))**.** *PLANAR 3-SAT is NP-complete.*

The variable cycle divides the plane into two regions, the interior and exterior of the cycle. Let $\mathcal{C}_-$ (resp. $\mathcal{C}_+$) be the set of clauses on the interior (resp. exterior) of the variable cycle. Let n be the number of variables. The most convenient way for us to now visualize Φ is that given by Knuth and Raghunathan[1] (65) in which the variables $x_1, \ldots, x_n$ are laid out from top to bottom on a vertical line with three-legged clauses laid out to the left and right of this line. The edges are rectilinear, the clauses from $\mathcal{C}_-$ all lie to the left of the variables, and the clauses from $\mathcal{C}_+$ all lie to the right of the variables (see Figure 3.4).

[1]The layout described by Knuth and Raghunathan actually has the variables on a horizontal line, but having them on a vertical line is more convenient for our explanation.

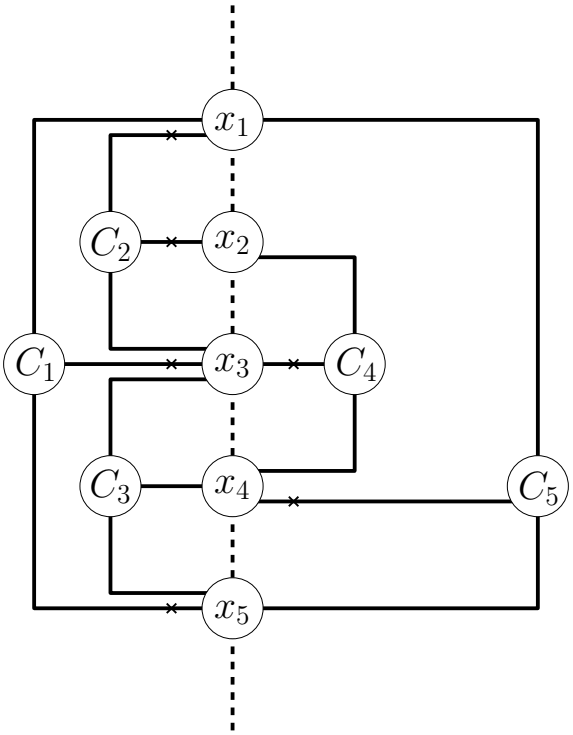


Figure 3.4: An instance Φ with three-legged clauses laid out to the left and right of the variables. Crosses on the lines indicate negations. For example, $C_1 = (x_1 \vee \overline{x_3} \vee \overline{x_5})$ and $C_4 = (x_2 \vee \overline{x_3} \vee x_4)$.

Motivation for using PLANAR 3-SAT

The only way we have been able to propagate a truth assignment around a terrain is in a linear 'highway', with each variable living in a 'lane' of the highway. We have been unable to reorder the variables in this highway, and we have only developed very restricted clause gadgets. Because of this, difficulties arose because each 3-CNF clause gadget would act like a 'roadblock' for the middle of the three variables involved, after which we could not continue to propagate that variable's truth assignment. Our solution is to arrange the variables and clauses in a way that ensures that each variable is the middle variable in at most two clauses, and that the variable is only used in the length of the highway that is between these two clauses.

If, for some $i \in [2, n-1]$, x_i does not appear as the middle variable in a

clause in $\mathcal{C}_-$, we add a *deletion node* to $G(\Phi)$ that is adjacent only to v_{x_i} and lies on the left side of the variable line. We do the same for $\mathcal{C}_+$ on the right side of the variable line. An example can be seen in Figure 3.5. This deletion node is used in the following description of a *removal ordering*.

A removal ordering for Φ

Considering Φ as laid out in Figure 3.4, it is not difficult to see that the clauses can be removed in an order such that a clause being removed has nothing 'between its legs'. We call such an ordering a *removal ordering* and order the sets $\mathcal{C}_-$ and $\mathcal{C}_+$ separately. Without loss of generality we can assume that x_1 and x_n are used in two common clauses, one in $\mathcal{C}_-$ and one in $\mathcal{C}_+$. If this is not the case, we can replace Φ with a new formula Φ' by adding a variable x_{n+1}, a clause $(x_1 \vee x_n \vee x_{n+1})$ in $\mathcal{C}_-$, and a clause $(\overline{x_1} \vee x_n \vee x_{n+1})$ in $\mathcal{C}_+$. An assignment of TRUE to x_{n+1} satisfies both new clauses without affecting any other clauses, so $\Phi' \Leftrightarrow \Phi$ and the size of Φ' is linear in the size of Φ. Therefore it is safe to assume that x_1 and x_n are used in two common clauses and this additional clause is unnecessary.

We describe the removal ordering for $\mathcal{C}_-$. There is an associated list of variables from which one variable is removed at each step. At the beginning of the process the variable list contains all variables. At each step we can remove:

- a clause with nothing between its legs, along with the clause's middle variable,
- a deletion node whose associated variable is not used in any remaining clauses in $\mathcal{C}_-$, along with its associated variable.

To avoid ambiguity, we always perform the action that removes the variable with the lowest index. At the end of the process the variable list contains only x_1 and x_n, and no clauses remain.

The key property of such a removal ordering is that, whenever a clause is removed, it involves three consecutive variables from the variable list. This allows our reduction to work even with our extremely restricted clause gadgets.

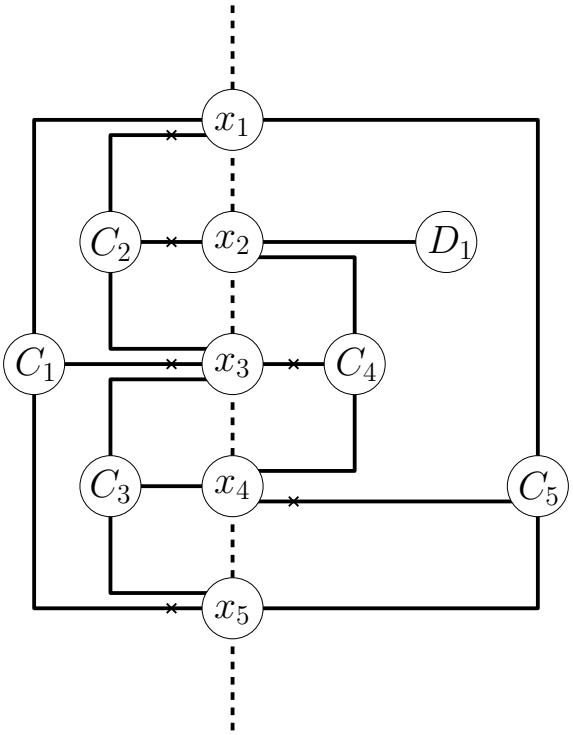


Figure 3.5: The same layout as in Figure 3.4 with an additional deletion node. The respective removal orderings are (C_2, C_3, C_1) and (C_4, D_1, C_5). Each variable is only used in between the two clauses that use it as the middle variable, so the issue of clauses acting as 'roadblocks' for middle variables is not a problem.

Building a path from a removal ordering

We construct two sequences α and β of variable lists corresponding to removal orderings of $\mathcal{C}_-$ and $\mathcal{C}_+$ respectively. For $0 \leq i \leq n-2$, the list α_i contains the variables remaining after the first i removals in the removal ordering for $\mathcal{C}_-$. In particular, this means that $\alpha_0 = (x_1, x_2, \ldots, x_n)$ and $\alpha_{n-2} = (x_1, x_n)$. β is built similarly from the removal ordering for $\mathcal{C}_+$.

We construct the path P_Φ based on the variable lists in the order $\alpha_{n-2}, \ldots, \alpha_1, \alpha_0, \beta_0, \beta_1, \ldots, \beta_{n-2}$. Such a path is shown in Figure 3.6. This path is a basic representation of how Φ can be turned into a linear 'highway' so it can be embedded in T_Φ. The deletion nodes ensure that, for each variable other than x_1 and x_n, the variable's lane is actually bounded by two

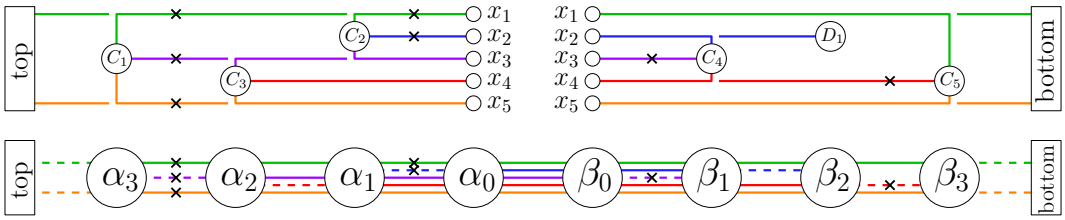


Figure 3.6: A horizontal layout of Φ (above) that illustrates its treatment as a variable highway. The path representation P_Φ (below). The variables $x_1, \ldots, x_5$ are shown as the lines from top to bottom, with dashed lines representing variables being deleted. Crosses on the lines indicate negations. Note in the α sequence that x_1 is negated because the negative literal $\overline{x_1}$ appears in C_2, and x_1 is later negated again because the positive literal x_1 appears in C_1.

'roadblocks'.

Running time

A planar embedding of a planar graph can be found in linear time (56; 79). The other tasks involved in constructing T_Φ from Φ can be performed trivially in polynomial time.

Truth assignment propagation with clause evaluation

We can think of the variable assignment as starting in the middle of P_Φ and being propagated out to the left and right. Our technique for propagating a consistent truth assignment in a variable highway is discussed next. This includes standard variable gadgets used for propagation, as well as deletion gadgets used as endpoints for variable lanes. For a reduction from SAT we need to determine if there is a consistent truth assignment that satisfies the clauses of Φ. Two of the main types of gadgets we need are for evaluating α-clauses while a truth assignment is propagated upwards and for evaluating β-clauses while a truth assignment is propagated downwards. These gadget types, along with the inversion gadget that inverts a variable (swaps the positions of guards representing TRUE and FALSE), are discussed last. The locations of the gadgets is determined by P_Φ.

Variable lanes and general layout

To propagate a variable assignment around the terrain, our reduction 'reflects' the assignment back and forth over a main valley. Each *reflector* has n *slots* – one for each variable lane. The slots in a reflector are stacked with the slot for x_1 being the highest and the slot for x_n being the lowest. Most reflectors do not transmit information about all n variables since most variable lists in P_Φ do not contain all variables. When a reflector does not transmit information about a variable x_i, the slot for x_i is empty, *i.e.* it is a straight line segment (see, *e.g.*, Figure 3.9). Thus empty slots act as space holders, and the positions of variable slots do not depend on which (or how many) slots are active in a given reflector.

Generally, multiple reflectors (though always a constant number) may be required to implement each step in the path P_Φ. Each reflector takes up the same amount of space, *i.e.* has the same size rectilinear bounding box. Interacting gadgets near the bottom of the terrain are closer to each other than interacting gadgets at the top of the terrain. To ensure that slopes of important lines of sight are of the same order of magnitude, we can 'pad' the lower part of the main valley so that all of the reflectors are in the top half of the terrain. In this way, gadgets that interact with each other are always the same horizontal distance apart up to a constant multiplicative factor. We can also add padding to the walls between reflectors so that gadgets that interact with each other are always the same vertical distance apart up to a constant multiplicative factor. Both types of padding increase the size of the terrain by at most a constant factor; neither type is shown in our figures.

Propagating a Truth Assignment

When studying the computational complexity of a problem it is often useful to consider the problem's locality. If a local change in a terrain can have a global effect on the optimal solution we may be able to exploit this nonlocal behavior to transmit information in a reduction. Specifically, we may be able to use it to transmit a truth assignment to different clauses. With this in mind, our first goal is simply to *propagate a truth assignment* around a terrain. Our greatest

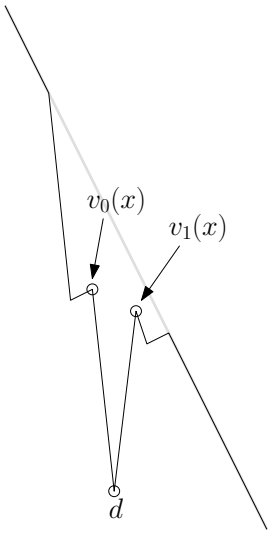


Figure 3.7: A variable gadget. Any point that sees the distinguished point d is dominated by at least one of $v_0(x)$ and $v_1(x)$.

concern is ensuring that the truth assigment is *consistent*, *i.e.* that variables have the same value wherever they are represented in the terrain.

In this section we deal with two types of terrains that introduce some of the important principles used in our full reduction terrains. First we consider *truth assignment propagation terrains*. In these terrains, we have a variable highway for n variables, and every variable slot is active in every reflector. Our main conclusion for these terrains is given as Observation 3.1. We then introduce deletion gadgets so that variable lanes can have endpoints other than the top and bottom reflectors. Our main conclusion for these terrains with deletion gadgets is given as Observation 3.2.

Encoding a truth assignment

We start out as simply as possible, encoding a single Boolean variable without any propagation. An example is shown in Figure 3.8. The *variable gadget* (see Figure 3.7) has a distinguished point, d, that can be seen from only two other vertices. These two vertices, call them $v_0(x_1)$ and $v_1(x_1)$, respectively represent an assignment of FALSE and TRUE to the variable x_1. Any point that sees d is dominated by either $v_0(x_1)$ or $v_1(x_1)$. Therefore, for any minimum guarding

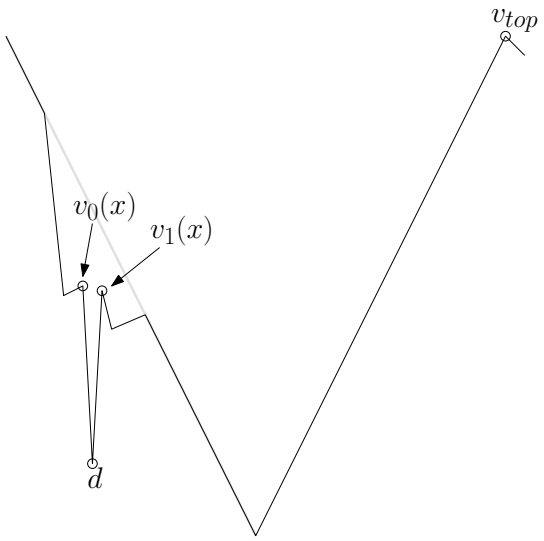


Figure 3.8: The simplest 'truth assignment propagation terrain' with one variable and no propagation.

set there exists a corresponding guarding set of the same size that contains either $v_0(x_1)$ or $v_1(x_1)$. We can assume without loss of generality that any minimum guarding set for the terrain contains a guard on at least one of these points. Similarly, any point that sees the rightmost vertex is dominated by v_{top} so we can assume that any minimum guarding set contains v_{top}. We make these assumptions in order to discuss minimum guarding sets more cleanly; later on we make similar assumptions without mention.

To encode an arbitrary number of variables, still without propagation, we simply stack variable gadgets on top of each other to create a basic reflector called an *assignment gadget*. An example is shown in Figure 3.10. A minimum guarding set for that terrain contains v_{top} as well as one guard for each of the three variable gadgets, corresponding to any truth assignment we want. This can be generalized to a truth assignment for any number of variables.

Distinguished points and internal guards

The distinguished point d in a variable gadget cannot be seen from outside the gadget. This ensures that any guarding set contains at least one point in each variable gadget. This is essential for proving correctness of our reduction.

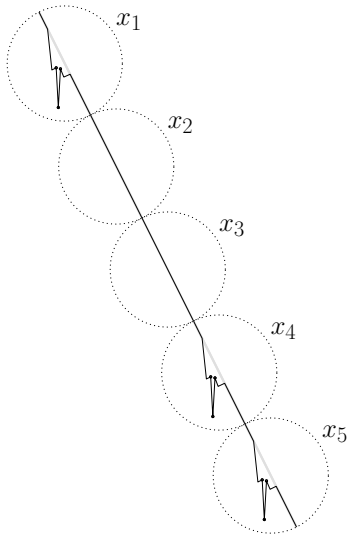


Figure 3.9: An assignment gadget. The slots corresponding to lanes for x_2 and x_3 are empty.

Certain gadgets require internal guards, between 0 and 2 depending on the gadget type. If all internal guards in the terrain interact in the right way, *i.e.* if they correspond to a consistent truth assignment satisfying the clauses of Φ, then they are sufficient to guard the entire terrain. If Φ is not satisfiable, we require the same number of internal guards to guard the distinguished points, but at least one additional guard is required to guard the rest of the terrain. Thus $f(\Phi)$ is simply the number of internal guards required, and is trivially computable from the numbers of gadgets of each type. In this accounting we consider the guards required at v_{top} and v_{bottom} to be internal guards.

Propagating a truth assignment

Now that we can encode an arbitrary truth assignment, we want to be able to propagate it consistently around the terrain. We do this by *reflecting the assignment* back and forth across a central valley. Each assignment gadget interacts with two assignment gadgets on the opposite side of the valley, taking input from the one above and giving output to the one below. The assignment gadgets on the right side of the valley are mirror images of those on the left side, though the positions of guards representing TRUE and FALSE are swapped. An

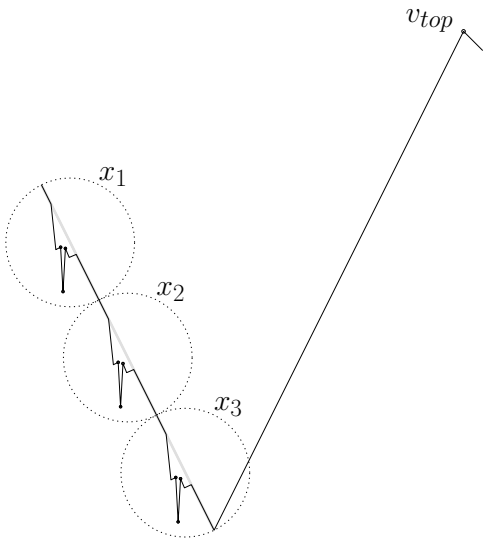


Figure 3.10: Another 'truth assignment propagation terrain', now with three variables, still no propagation.

example of the reflecting behavior is shown in Figure 3.11, a variable interaction is shown in Figure 3.12, and the details of the variable interaction are shown in Figure 3.13. The way a variable's assignment is propagated down the terrain holds the key to understanding the complexity of terrains, and is based on the relationship between what on the opposite side of the valley can be seen by guards at $v_0(x)$ or $v_1(x)$. It is possible for $v_0(x)$ to see things $v_1(x)$ cannot because $v_1(x)$ is 'too low'. Similarly, it is possible for $v_1(x)$ to see things $v_0(x)$ cannot because $v_0(x)$ is 'too far to the left'. We explain the details in Figure 3.13 shortly.

It is important to point out that the direction in which a truth assignment is propagated is simply a matter of perspective. We can think of the truth assignment as starting at the top and being propagated downwards, as starting at the bottom and being propagated upwards, or as starting in the middle and being propagated both upwards and downwards.

Variable gadget interaction

A variable gadget only interacts with other gadgets representing the same variable. This interaction is shown coarsely in Figure 3.11. The necessary guard

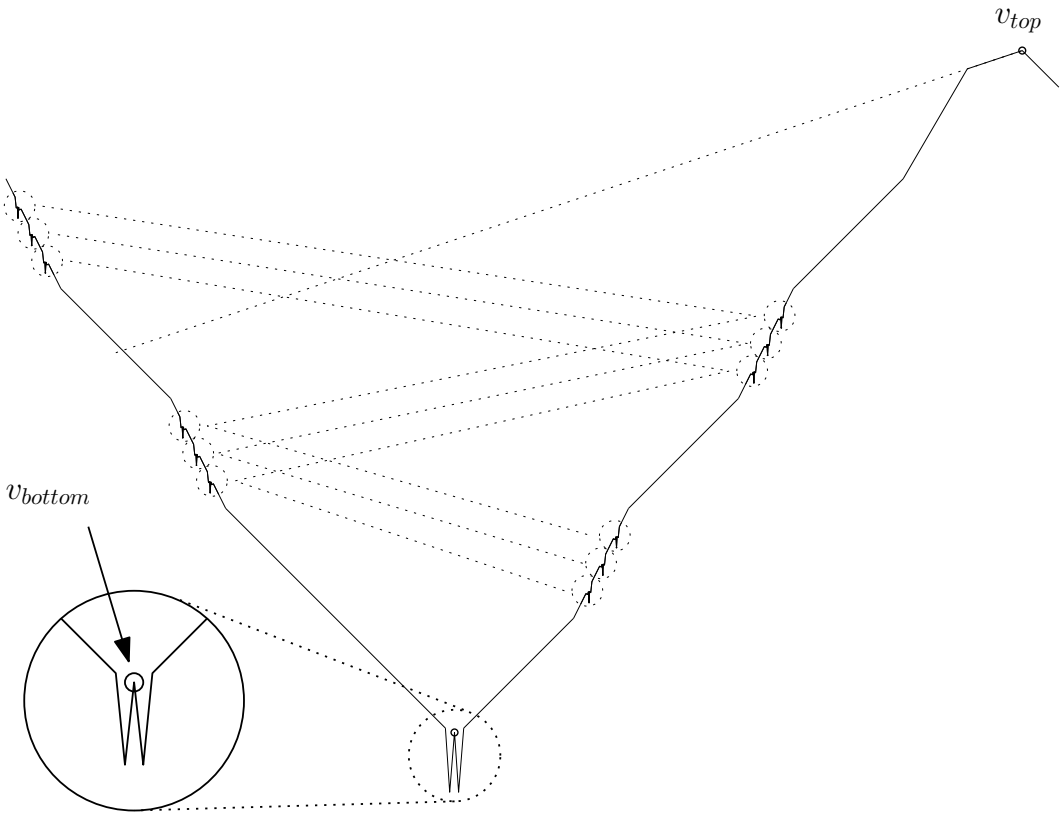


Figure 3.11: An assignment of three variables being propagated using four reflectors. Note that in an actual terrain used in our reduction, the top and bottom assignment gadgets have every variable slot empty except for x_1 and x_n.

at v_{top} sees enough of the first (*i.e.* top left) assignment gadget that the first assignment gadget can be optimally guarded by n guards corresponding to any truth assignment for the n variables. This is the output of the first assignment gadget. After that, zig-zagging down the terrain, the interactions of the variable gadgets are designed to ensure that an assignment gadget can be guarded by n guards if and only if it encodes a truth assignment (its output) that matches the truth assignment encoded in the assignment gadget above (its input). Finally, a guard at v_{bottom} is necessary and sufficient to guard everything below the final assignment gadgets. Thus a 'truth assignment propagation terrain' (see, *e.g.*, Figure 3.11) with k assignment gadgets propagating n variables can be guarded with $nk + 2$ guards if and only if the truth assignment

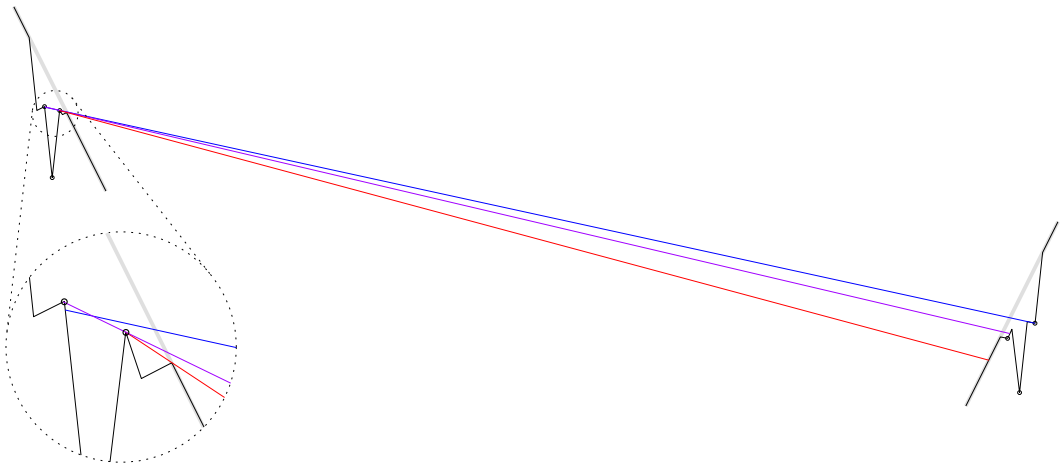


Figure 3.12: A variable interaction. The inset detail is exaggerated to show how specific lines of sight interact. For more detail see Figure 3.13.

is consistent; furthermore, this works for any of the 2^n possible truth assignments. This count of $nk+2$ only works for these truth assignment propagation terrains because every assignment gadget has a slot for each of the n variables so there are nk total variable gadgets; this need not be true in general.

In the detail in Figure 3.13 the four points $\{d, d', p, q\}$ are of particular interest. To guard d and d' we need at least one of $\{v_0(x), v_1(x)\}$ and at least one of $\{u_0(x), u_1(x)\}$ in our guarding set. The gadgets are configured such that, of these four potential guards, only $v_0(x)$ and $u_1(x)$ see q, and only $v_1(x)$ and $u_0(x)$ see p. Therefore the only pairs of guards that see $\{d, d', p, q\}$ are $\{v_0(x), u_0(x)\}$ and $\{v_1(x), u_1(x)\}$. These pairs correspond to the variable x being set to FALSE or TRUE respectively.

Special care is taken to ensure that guards in a guarding set of size $nk + 2$ do not interfere with the wrong gadgets. For each of the nk variable gadgets we have points of type p and q (see Figure 3.13). v_{top} can see all such points in the first assignment gadget, but none from any other assignment gadget. v_{bottom} cannot see any at all. The $v_0(x)$ and $v_1(x)$ type guards can only see the appropriate points in their own variable gadget and in the variable gadget below that their variable gadget interacts with. The 'lip' on the gadget ensures $v_0(x)$ and $v_1(x)$ cannot see any variable gadgets further down, and they cannot

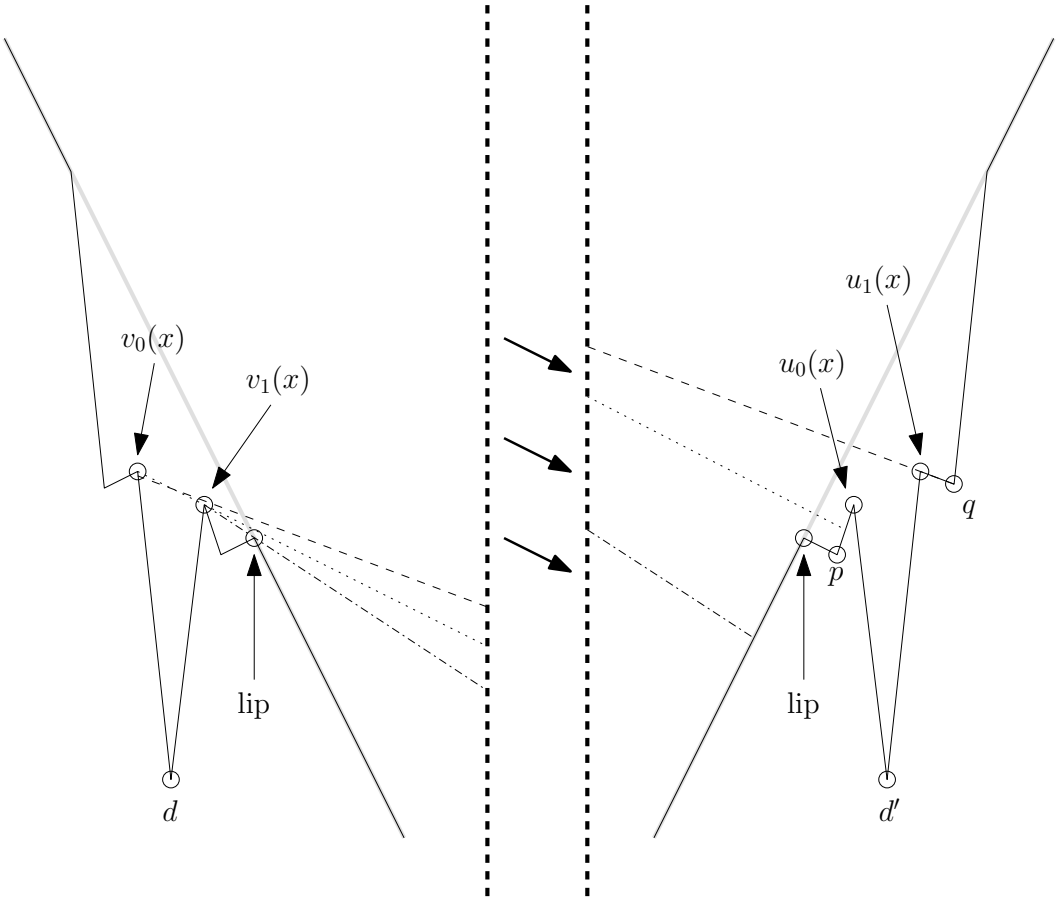


Figure 3.13: Interaction of variable gadgets. $v_0(x)$ cannot see p because the line of sight is blocked by $v_1(x)$. $v_1(x)$ cannot see q because the line of sight is blocked by $u_1(x)$. One internal guard is required for each gadget.

see any higher points of type p and q because those points sit in 'dimples' that are appropriately steep.

Managing lines of sight

For each pair of variable gadgets that interact, two tweaking phases need to be performed to ensure that important lines of sight either exist or do not exist, as required; tweaking is done by moving certain vertices vertically by small amounts. The first phase is done for each gadget, starting at the bottom of the terrain and proceeding upwards. Then the second phase is done for

each gadget, starting at the top of the terrain and proceeding downwards. We describe these phases in reference to the interaction shown in Figure 3.13.

In the first phase, the two vertices to be adjusted vertically are $v_1(x)$ and the lip vertex to the right of $v_1(x)$. First $v_1(x)$ is adjusted so that the line of sight from $v_0(x)$ through $v_1(x)$ hits the terrain on the opposite side between $u_0(x)$ and p. After that, the lip vertex to the right of $v_1(x)$ is adjusted so that the line of sight from $v_1(x)$ passing through this lip vertex hits the terrain on the opposite side below the opposite lip vertex but above any variable gadget below.

In the second phase, the vertices to be adjusted vertically are p and q. p is adjusted so that the line of sight from p through the adjacent lip vertex hits the terrain on the opposite side near the bottom of the relevant variable gadget. q is adjusted more precisely so the line of sight from q through $u_1(x)$ hits the terrain on the opposite side on the line segment $(d, v_0(x))$. The line of sight passes just over $v_1(x)$ and hits just below $v_0(x)$.

If the tweaking is done in this order, the tweaking process does not disturb variable gadgets that have already been tweaked. The other gadget types can be tweaked similarly. The tweaking process is discussed in more detail later.

We reiterate our main point regarding these truth assignment propagation terrains. Again, this only holds for these demonstrative truth assignment propagation terrains.

Observation 3.1. A truth assignment propagation terrain with k assignment gadgets propagating n variables can be guarded using $nk + 2$ guards corresponding to any consistent truth assignment. Any guarding set that does not correspond to a consistent truth assignment requires more guards.

Deletion gadgets

In the truth assignment propagation terrains shown thus far, each variable has a lane in the variable highway that spans every assignment gadget, from the top to the bottom. However, for our reduction we need to be able to place 'roadblocks' to manage the endpoints of each variable's lane. To end a lane, we use *deletion gadgets*. We need a *downward deletion gadget* for the bottom

endpoint of a lane and an *upward deletion gadget* for the top endpoint of a lane.

Both downward and upward deletion gadgets are essentially simplified variable gadgets. A downward deletion gadget is actually just a flat region where a variable gadget would be; in a sense it is a variable gadget that has been simplified to a straight line (see Figure 3.14). An upward deletion gadget is only slightly more complicated; it requires a single guard that functions similarly to v_{top} but only for a single lane (see Figure 3.15).

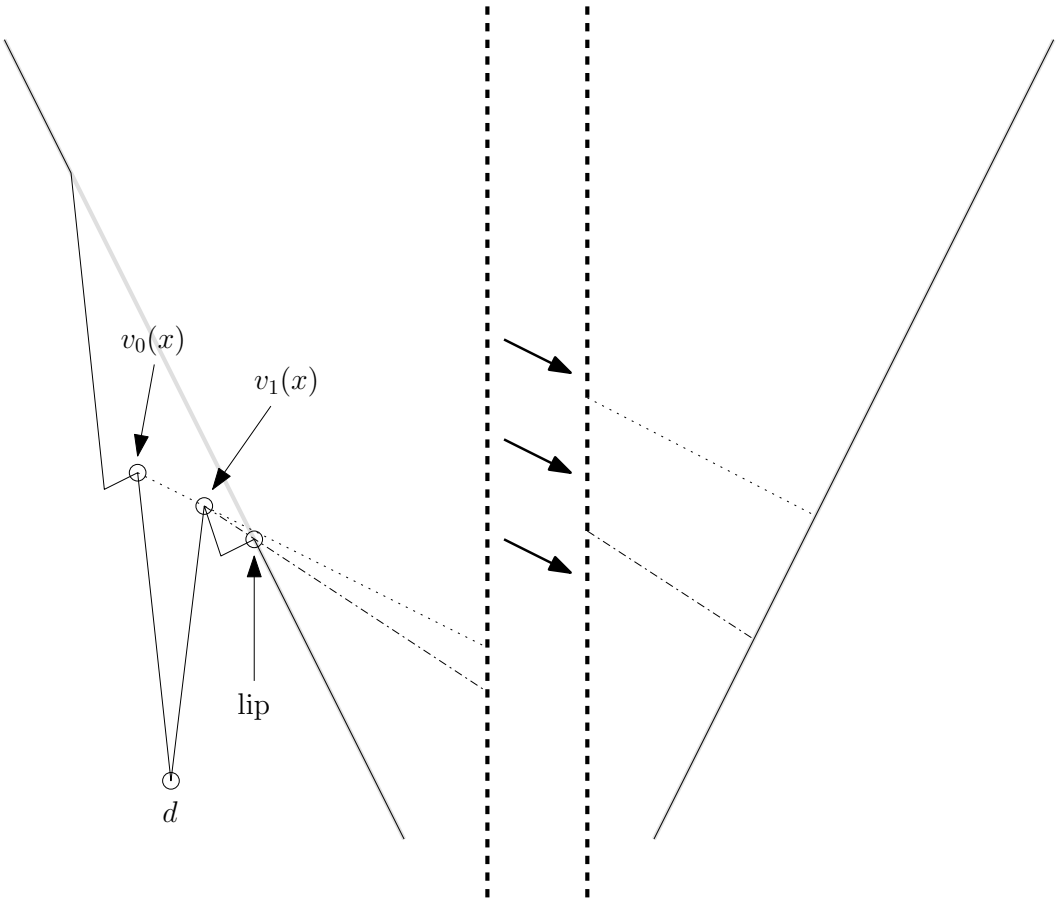


Figure 3.14: The bottom variable gadget in a variable lane (left) interacts with a downward deletion gadget (right). The downward deletion gadget is simply a flattened region. $v_1(x)$ and the lip vertex are positioned so that neither $v_0(x)$ nor $v_1(x)$ can see gadgets lower down the terrain. One internal guard is required on the left side, no internal guard is required on the right.

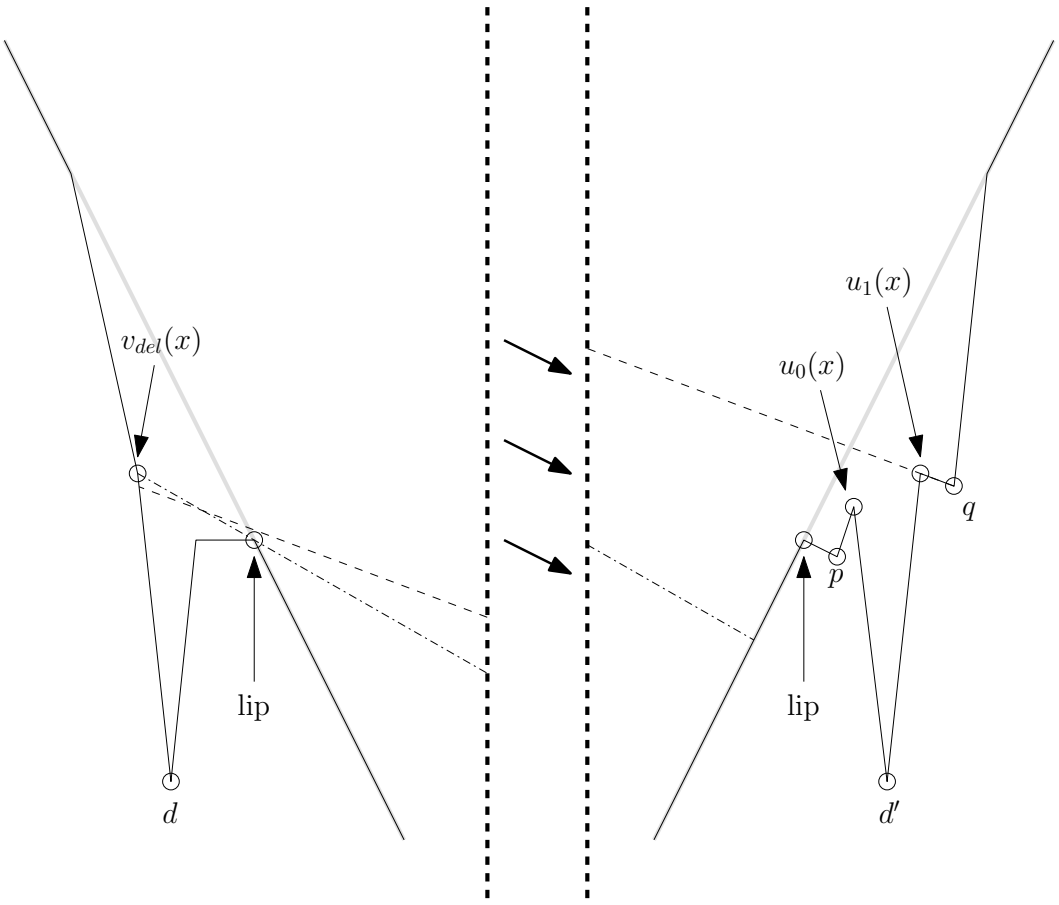


Figure 3.15: An upward deletion gadget (left) interacts with the top variable gadget in a variable lane (right). The upward deletion gadget is a simplified variable gadget. $v_{del}(x)$ can see both p and q and is used to guard the distinguished point d. The lip vertex near v_{del} ensures that v_{del} cannot interact with variable gadgets lower down the terrain. One internal guard is required on the left side and one is required on the right.

Observation 3.2. A truth assignment propagation terrain with deletion gadgets with k_v total variable gadgets and k_{del} upward deletion gadgets can be guarded using $k_v + k_{del} + 2$ guards corresponding to any consistent truth assignment. Any guarding set that does not correspond to a consistent truth assignment requires more guards.

Preliminary embedding on a grid

At this point we should begin to address the issue of the precision required by our reduction. To prove that terrain guarding is *strongly* NP-hard we must show that any terrain generated by our reduction can be represented by a unary string of polynomial length. This means, in particular, that we must be able to represent the location of a vertex with a unary string of polynomial length. The standard way to do this in a geometric NP-hardness proof is to ensure that all vertices can be placed on a grid of polynomial size.

We begin explaining the grid embedding now, before things get more complicated. The following lemma handles the general ideas and we handle additional details (*i.e.* the clause and inversion gadgets that have not yet been discussed) after they are presented.

Lemma 3.4. *A truth assignment propagation terrain with deletion gadgets, propagating an assigment of n variables using $\mathcal{O}(n)$ reflectors, can be embedded in a grid of size $\mathcal{O}(n^6) \times \mathcal{O}(n^6)$.*

Proof. We argue that the vertices can be placed on a grid of polynomial size using a hierarchical grid structure. Our level 1 grid has polynomial size, each level 1 grid square contains a level 2 grid of polynomial size, each level 2 grid square contains a level 3 grid of polynomial size, and each level 3 grid square contains a level 4 grid of polynomial size. Our primary concern is the precision required when aiming key lines of sight between interacting gadgets.

In level 1 of the grid, each square is sized such that an assignment gadget spans a rectangle, made up of 2 grid squares stacked vertically, from corner to opposite corner (this is illustrated in Figure 3.16).

For the sake of simplicity we ensure that important lines of sight between interacting gadgets all have similar slopes (specifically, slopes of $\pm\Theta(1/n)$). We can guarantee this with some simple padding that increases the size of the terrain by only a constant factor. This padding is *not* shown in Figure 3.16. Firstly, we ensure that, vertically speaking, interacting reflectors are separated by one level 1 grid square. Thus in any two interacting gadgets, any point from one gadget is separated vertically from any point in the other gadget by

$\Theta(1)$ level 1 grid squares. The second type of padding we do is padding the bottom of the terrain so that no reflectors are placed below the level at which there are n level 1 grid squares between the two sides of the main valley. This guarantees that any two reflectors are separated horizontally by $\Theta(n)$ level 1 squares. These two padding processes ensure that lines of sight between interacting gadgets always have slopes of $\pm\Theta(1/n)$.

The size of our level 1 grid is $\Theta(n) \times \Theta(n)$. At the cost of only a constant multiplicative factor to the grid's size, the minor gadgets containing v_{top} and v_{bottom} can also be implemented in the level 1 grid.

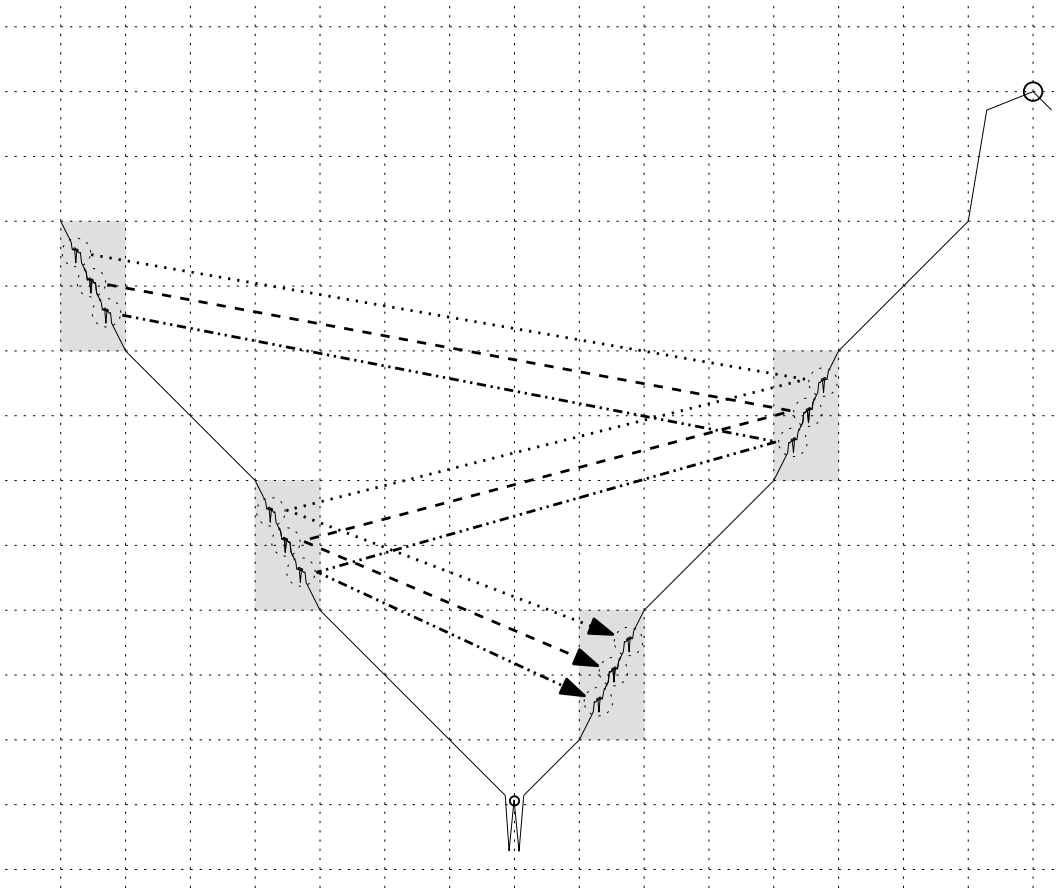


Figure 3.16: Level 1 of the grid structure. The padding used to ensure lines of sight between interacting gadgets have slope $\pm\Theta(1/n)$ is not shown.

Level 2 of the grid is sized such that the space for each slot in an assignment

gadget spans a rectangle, made up of a constant number of grid squares, from corner to opposite corner. Thus we fill a level 1 grid square with a level 2 grid of size $\Theta(n) \times \Theta(n)$. Each variable gadget or deletion gadget, by itself, can be embedded in a grid of constant size if it does not need to be tweaked. So, at the cost of a constant multiplicative factor to the size of each level 2 grid, we can embed the untweaked gadgets in the first two grid levels.

Before the variable gadgets and the deletion gadgets are tweaked, all vertices except lip vertices are in the correct horizontal locations but not necessarily in the correct vertical positions. Thus the tweaking only moves vertices vertically, except for lip vertices that slide along the line of slope ± 2. These configurations each use a constant size grid per slot gadget. An untweaked variable gadget is shown in Figure 3.17.

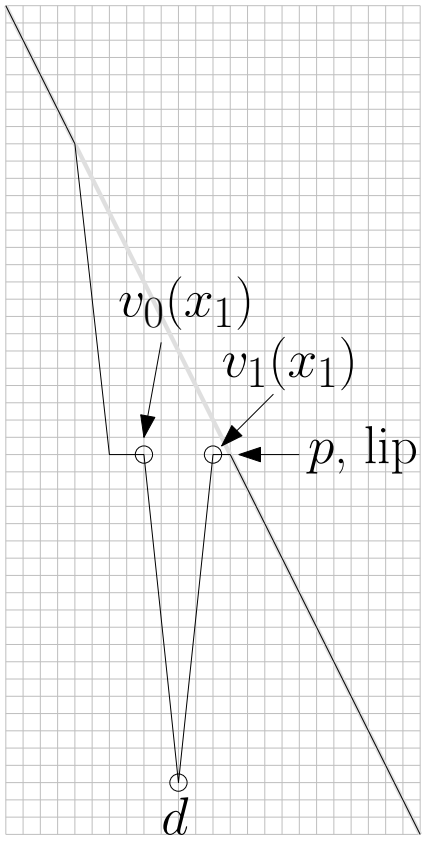


Figure 3.17: An untweaked variable gadget with level 2 grid lines shown.

The tweaking is then done in two stages, as previously discussed, where each stage uses a new grid level. In the first tweaking phase, taking place in level 3 of the grid structure, we move three vertices in each variable gadget – those of type v_1, p, and the lip vertex. These vertices can be lowered but not raised. We do the tweaking from the bottom up, so that, as a variable gadget is being tweaked, the variable gadget to which it sends output has already been tweaked.

We first lower v_1 until the ray emanating from v_0 and passing through v_1 hits the output variable gadget below u_0 but above p. This ray might be occluded *within the gadget being tweaked* but we ignore that for now as the occlusion is removed in the next step. Now we move p and the lip vertex downwards in unison so that they remain horizontally level, but diverge as p moves straight downwards and the lip vertex slides down along the guideline of slope ± 2. We lower p and the lip vertex until the ray emanating from v_1 and passing through the lip vertex hits the output variable gadget just below its lip vertex.

To determine the required size of each level 3 grid we must determine the precision required by the two key line segments. Each spans $\Theta(n^2)$ level 2 squares horizontally and $\Theta(n)$ level 2 squares vertically. We can consider the tweaking process to 'aim' the lines of sight, where the targets are in the output variable gadget and have size $\Theta(1)$ level 2 squares. Since, for either line of sight, the distance between the two 'aiming' vertices is $\Theta(1)$ level 2 squares, and each line of sight has length $\Theta(n^2)$ level 2 grid squares, the amount of precision required in the tweaking of the aiming vertices is $\Theta(n^{-2})$ level 2 squares. Thus it suffices for each level 2 square to contain a level 3 grid of size $\Theta(n^2) \times \Theta(n^2)$.

In the second tweaking step, corresponding to the level 4 grids, we move the vertices of type p and q. In this stage we start at the highest variable gadget and proceed downwards. The main point of this tweaking stage is to aim the lines of sight of interest emanating from p and q in the variable gadgets. First we address the line of sight emanating from the q vertex and passing through the adjacent u_1 vertex. We are doing the tweaking in the output variable gadget now, aiming a line of sight at the input variable gadget above. The line

of sight needs to pass above the v_1 vertex and hit the input gadget just below the v_0 vertex. For this purpose, it needs to pass just above v_1, but within $\mathcal{O}(1)$ level 3 squares, so the target has size $\Theta(1)$ level 3 squares. Again, the line of sight has length $\Theta(n^2)$ level 2 squares, or $\Theta(n^4)$ level 3 squares. The distance between the two vertices responsible for the aiming is $\Theta(n^2)$ level 3 squares. Therefore the amount of precision required is $\Theta(n^{-2})$ level 3 squares. The line of sight emanating from p has a larger target; in fact it only needs to hit the relevant variable gadget on the line segment below the lip vertex so the target is $\Theta(1)$ level 2 squares, so even less precision is required. So it suffices for each level 3 square to contain a level 4 grid of size $\Theta(n^2) \times \Theta(n^2)$.

This completes the tweaking of gadgets, and thus completes the embedding of the terrain on a grid. The total size of the grid used is polynomial, specifically $\Theta(n^6) \times \Theta(n^6)$. □

Evaluating clauses

Our basic truth assignment propagation process can be thought of as behaving in the following way. A truth assignment for the variables is set in one of the assignment gadgets (it does not matter which). Call this reflector the *starting gadget*. The way the assignment gadgets interact ensures that, in a minimum guarding set, this truth assignment is propagated consistently both upwards and downwards from the starting gadget.

With the starting gadget fixed, we can consider each assignment gadget to have an input truth assignment and an output truth assignment (though the starting gadget has no input and the topmost and bottommost assignment gadgets have no output). Each assignment gadget above the starting gadget takes input from below and sends its output upwards. Each assignment gadget below the starting gadget takes input from above and sends its output downwards. An assignment gadget can be thought of as an identity gadget since, in a minimum guarding set, its output is the same as its input.

When building T_Φ we also have a starting gadget, and other gadgets still have input and output. However, we need more than just an identity gadget. In this section we describe the types of gadgets required to propagate a truth

assignment that satisfies Φ. The three gadget types in this section are:

1. **Inversion gadget** – in the variable lane for a variable x_i, this gadget switches the position of the guards representing TRUE and FALSE assignments.

2. **Upward clause gadget** – for three variables x_i, x_j, x_k, with $i < j < k$ and with x_j the only non-empty lane between x_i and x_k, asserts that the clause $(x_i \vee x_j \vee \overline{x_k})$ is satisfied, otherwise at least one extra guard is required. The x_j lane must be empty above this gadget.

3. **Downward clause gadget** – for three variables x_i, x_j, x_k, with $i < j < k$ and with x_j the only non-empty lane between x_i and x_k, asserts that the clause $(x_i \vee \overline{x_j} \vee \overline{x_k})$ is satisfied, otherwise at least one extra guard is required. The x_j lane must be empty below this gadget.

These gadgets are sufficient to complete our reduction from PLANAR 3-SAT.

Gadget input and output

Our reduction ensures that in a guarding set of size $f(\Phi)$, guards can only be on certain special types of internal vertices[2]:

1. v_{top} and v_{bottom}

2. vertices of type $v_{del}(x)$ in upward deletion gadgets

3. vertices of type $v_0(x)$, $v_1(x)$, $u_0(x)$, and $u_1(x)$[3] in standard and modified[4] variable gadgets

4. vertices of type $u_0(x)$, $u_1(x)$, $w_0(x)$ and $w_1(x)$ in inversion gadgets.

[2]More precisely, in a guarding set of size $f(\Phi)$, any guard not on one of these point types can be replaced by a guard on one of these point types with no loss in visibility.

[3]In variable gadgets, vertices of type $u_0(x)$ and $u_1(x)$ are actually also vertices of type $v_0(x)$ and $v_1(x)$.

[4]Modified variable gadgets are used in clause gadgets.

The output of a gadget only has to be valid in a guarding of size $f(\Phi)$, otherwise it is not necessarily valid and all assumptions are allowed to fall apart since a minimum guarding set must correspond to a truth assignment satisfying Φ if and only if it has size $f(\Phi)$. $f(\Phi)$ internal guards are still necessary, but they do not necessarily correspond to a consistent and satisfying truth assignment if there are additional guards.

Inversion gadget

For a single variable, an *inversion gadget* swaps the positions of guards representing the TRUE and FALSE truth assignments. By default, in a variable gadget the vertex $v_0(x)$ (representing FALSE) is above the vertex $v_1(x)$ (representing TRUE). However, if in the lane for x there are an odd number of inversion gadgets between the variable gadget in question and the start gadget, $v_1(x)$ is above $v_0(x)$. An inversion gadget replaces a single variable gadget in the lane corresponding to the variable being inverted.

A standard variable gadget has two possible minimum guarding sets: $\{v_0(x)\}$ and $\{v_1(x)\}$ (see Figure 3.7). The special variable slot in an inversion gadget also has two possible minimum guarding sets, though each has two guards instead of just one. From each minimum guarding set, however, only one guard affects the output of the gadget.

An inversion gadget is shown in Figure 3.18, with a larger view shown in Figure 3.19. A detailed description to accompany Figure 3.18 is as follows. Input is shown above and output is shown below, with the same inversion gadget on the top right and bottom right. There are only two minimum guarding sets that include exactly one of the input guards $v_0(x)$ and $v_1(x)$. These sets are $\{v_0(x), u_0(x), w_0(x), v'_0(x)\}$ and $\{v_1(x), u_1(x), w_1(x), v'_1(x)\}$. Two internal guards are required in the inversion gadget (*i.e.* in the range $[u_1(x), u_0(x)]$), since r is not seen by anything outside the range $[u_1(x), w_0(x)]$ and s is not seen by anything outside the range $[w_1(x), u_0(x)]$. t is not seen by anything outside the range $[w_0(x), w_1(x)]$. Though it is difficult to see, p is seen by $v_1(x)$ and $w_0(x)$ but not by $u_1(x)$. q is seen by $v_0(x)$ and $w_1(x)$ but not by $w_0(x)$. r is seen by $u_1(x)$ and $w_0(x)$. The two potential output guards are $u_0(x)$ and

$u_1(x)$. $w_0(x)$ and $w_1(x)$ are useless outside this gadget.

Upward clause gadget

An upward clause gadget takes as input (from below) a variable assignment and outputs a variable assignment with one variable removed. These gadgets are used to implement clauses in the α sequence of P_Φ. For three variables x_i, x_j, x_k that are adjacent[5] in the input highway ($i < j < k$), the gadget deletes the middle variable x_j. The gadget can be guarded with a minimum number of internal guards if and only if the following two conditions hold:

1. Each variable in the input (except x_j) must have the same value in the output.

2. The clause $(x_i \vee x_j \vee \overline{x_k})$ is satisfied by the input.

Another way of saying that the clause $(x_i \vee x_j \vee \overline{x_k})$ is satisfied by the input is to say that the input can include $\overline{x_j}$ only if $(x_i \vee \overline{x_k})$ evaluates to TRUE in the input and output.

In an upward clause gadget involving the variables x_i, x_j, x_k, the variable x_j is deleted from the assignment. This takes place in a single reflector, in which all active variable slots except x_j contain variable gadgets. Assuming the clause being evaluated is $(x_i \vee x_j \vee \overline{x_k})$, we explain what is put in the place of a variable gadget for x_j. The key is the special point q_j. Of the 'output' points $v_0(x_i)$, $v_1(x_i)$, $v_0(x_k)$ and $v_1(x_k)$, only $v_1(x_i)$ and $v_0(x_k)$ can see q_j. Of the 6 'input' points, only $u_1(x_j)$ can see q_j. Since the variables x_i and x_k have the same output as input, $\overline{x_j}$ can be in the input if and only if x_i or $\overline{x_k}$ is in the input. Thus a minimum guarding of the gadget ensures that the clause $(x_i \vee x_j \vee \overline{x_k})$ is satisfied.

Downward clause gadget

A downward clause gadget takes as input (from above) a variable assignment and outputs a variable assignment with one variable removed. These gadgets

[5]By adjacent we mean that there are no active lanes in the input between x_i and x_j or between x_j and x_k.

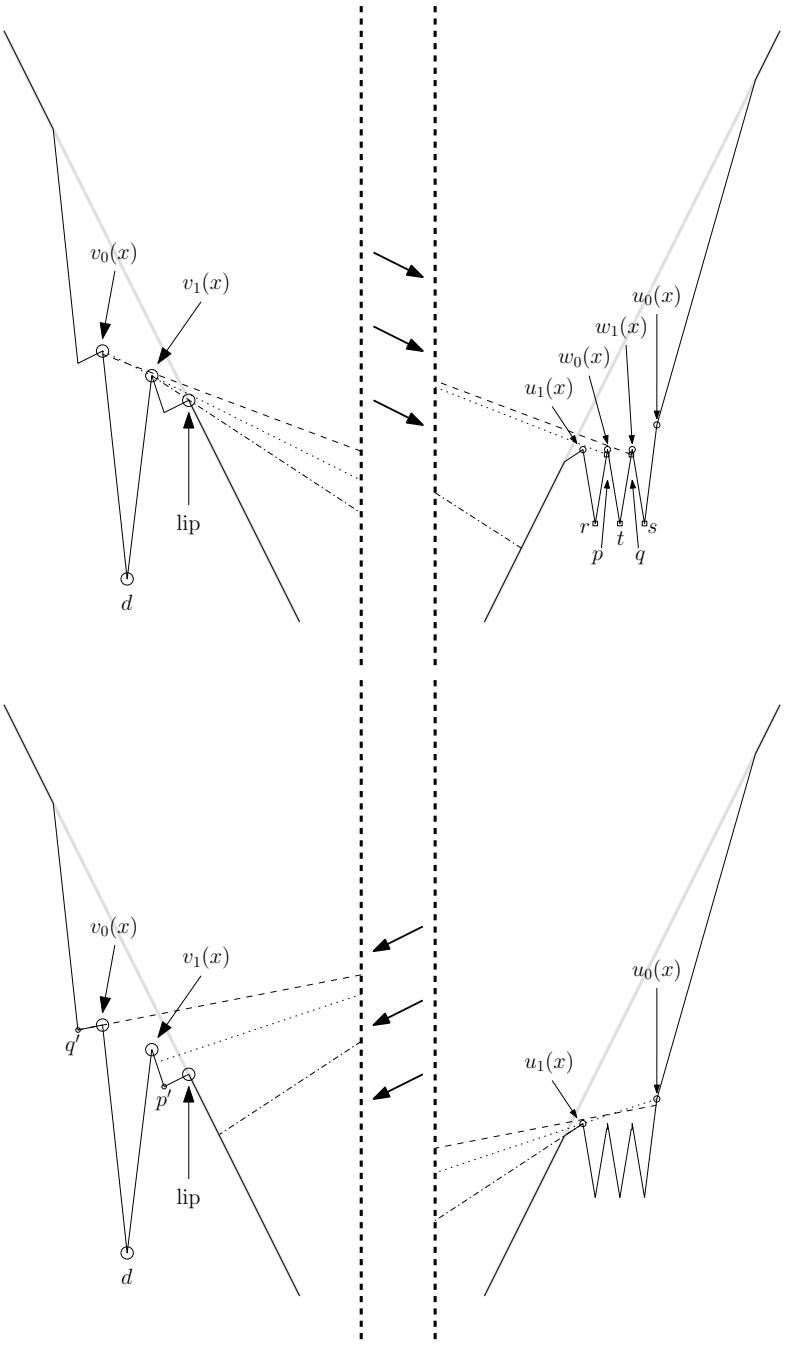


Figure 3.18: Inversion gadget interaction. For more detail of the inversion gadget see Figure 3.19, particularly the inset.

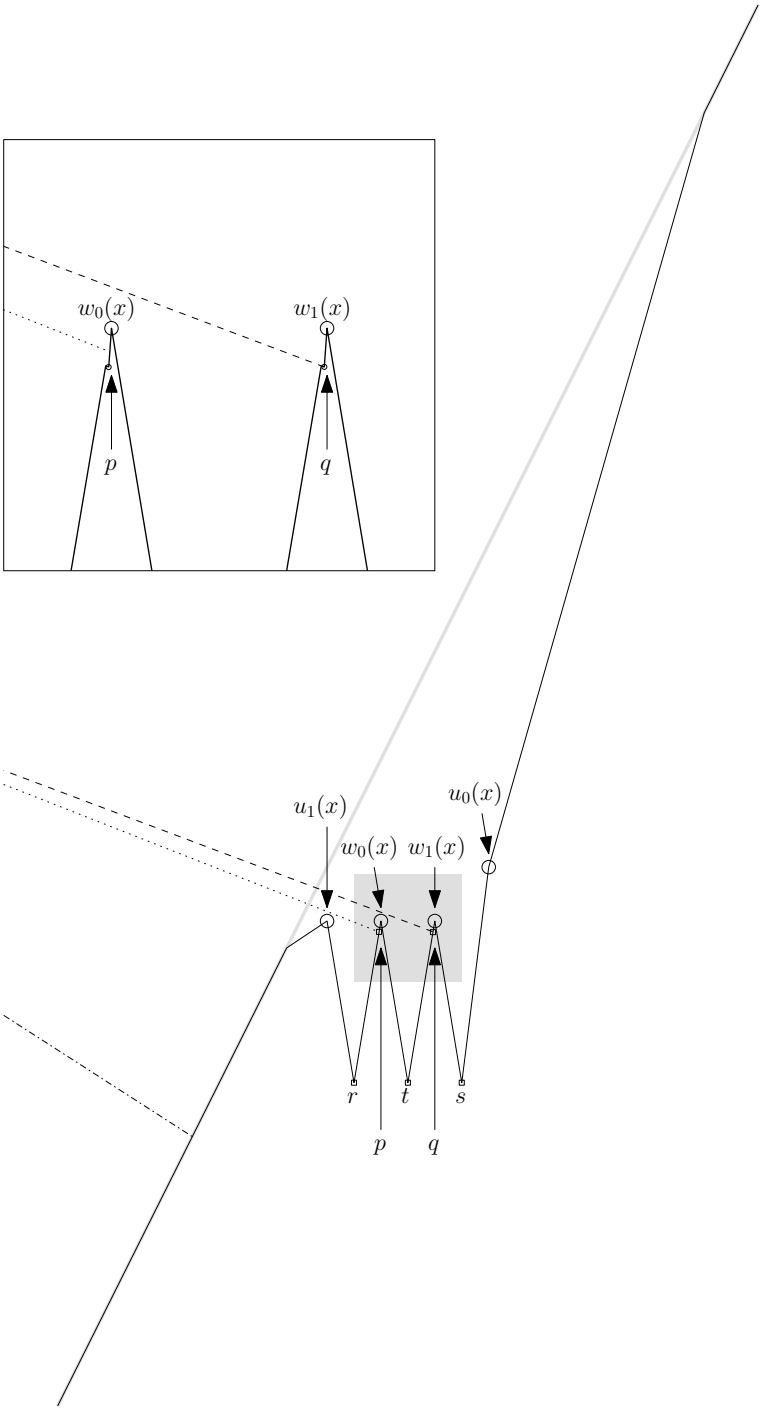


Figure 3.19: Detail of inversion gadget. See Figure 3.18 for details of its interaction with variable gadgets above and below it. The inset shows a magnification of the shaded region.

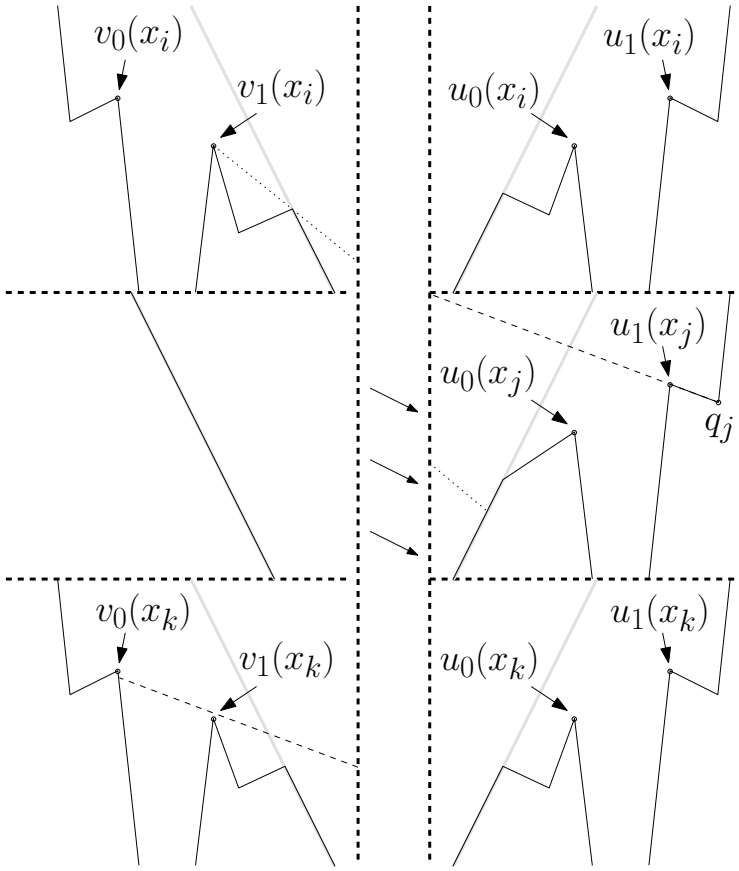


Figure 3.20: Upward clause gadget interaction. In every reflector above this gadget the x_j slot is empty. Each of the 6 slots shown, except the middle left, requires one internal guard. No extra guards are required iff the clause $(x_i \vee x_j \vee \overline{x_k})$ is satisfied, since the only internal guards that see q_j are $v_1(x_i)$, $u_1(x_j)$, and $v_0(x_k)$. Note that, in the top left slot, the lip vertex has been lowered so that $v_1(x_i)$ can see q_j. Also, q_j has been adjusted so that $v_0(x_k)$ can see it. The two lines of sight relevant to these adjustments are shown.

are used to implement clauses in the β sequence of P_Φ. For three variables x_i, x_j, x_k that are adjacent in the input highway, the gadget deletes the middle variable x_j. The gadget can be guarded with a minimum number of internal guards if and only if the following two conditions hold:

1. Each variable in the input (except x_j) must have the same value in the output.

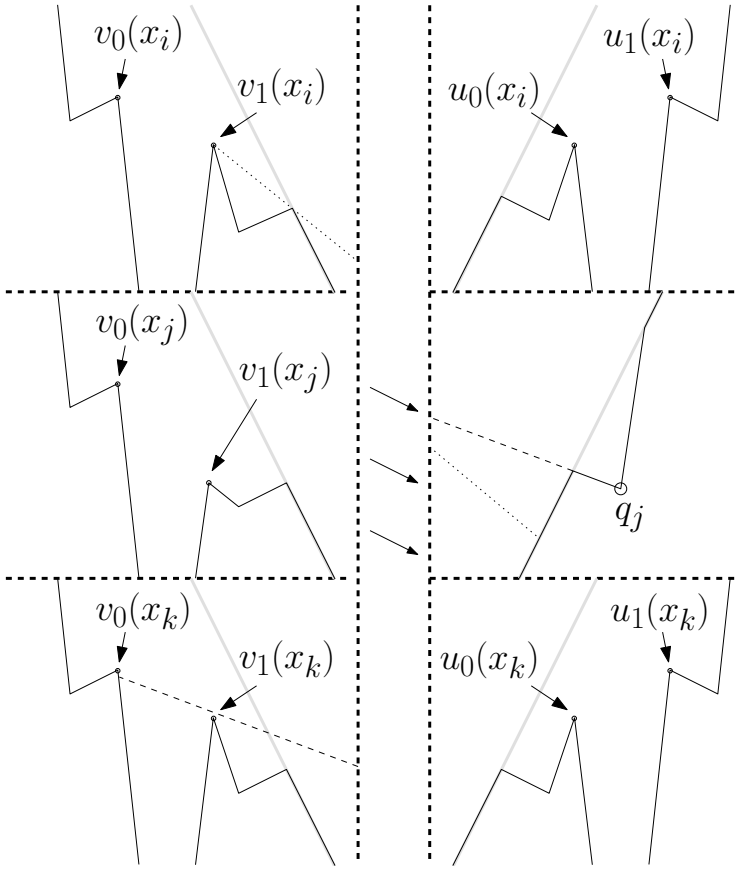


Figure 3.21: Downward clause gadget interaction. In every reflector below this gadget the x_j slot is empty. Each of the 6 slots shown, except the middle right, requires one internal guard. No extra guards are required iff the clause $(x_i \vee \overline{x_j} \vee \overline{x_k})$ is satisfied, since the only internal guards that see q_j are $v_1(x_i)$, $v_0(x_j)$, and $v_0(x_k)$. Note that, in the top left slot, the lip vertex has been lowered so that $v_1(x_i)$ can see q_j. Also, q_j has been adjusted so that $v_0(x_k)$ can see it. The two lines of sight relevant to these adjustments are shown.

2. The clause $(x_i \vee \overline{x_j} \vee \overline{x_k})$ is satisfied by the input.

Another way of saying that the clause $(x_i \vee \overline{x_j} \vee \overline{x_k})$ is satisfied by the input is to say that the input can include x_j only if $(x_i \vee \overline{x_k})$ evaluates to TRUE in the input and output.

In a downward clause gadget involving the variables x_i, x_j, x_k, the variable x_j is deleted from the assignment. This takes place in a single reflector, in which all active variable slots except x_j contain variable gadgets. Assuming the clause being evaluated is $(x_i \vee \overline{x}_j \vee \overline{x}_k)$, we explain what is put in the place of the variable gadget for x_j. The key is the special point q_j. Of the 'input' points $v_0(x_i)$, $v_1(x_i)$, $v_0(x_j)$, $v_1(x_j)$, $v_0(x_k)$ and $v_1(x_k)$, only $v_1(x_i)$, $v_0(x_j)$ and $v_0(x_k)$ can see q_j. Thus a minimum guarding of the gadget ensures that the clause $(x_i \vee \overline{x}_j \vee \overline{x}_k)$ is satisfied. All variables except x_j have the same output as input.

Final embedding on a grid

We previously introduced Lemma 3.4 that states that a truth assignment propagation terrain can be embedded in a grid of polynomial size. Here we extend that lemma to show that any terrain generated by our reduction from PLANAR 3-SAT can be embedded in a grid of polynomial size.

Lemma 3.5. *For any instance Φ of PLANAR 3-SAT, T_Φ can be embedded in a grid of polynomial size.*

Proof. In the proof of Lemma 3.4 we dealt with truth assignment propagation terrains that do not contain inversion gadgets or clause gadgets. Here we handle the embedding of inversion gadgets and clause gadgets. The reader should read the proof of Lemma 3.4 before proceeding. Note that the number of reflectors in T_Φ is $\Theta(n)$.

As well as extending the embedding method from Lemma 3.4 we refine it slightly. Whereas before we had a level 3 grid of size $\Theta(n^2) \times \Theta(n^2)$ in each level 2 grid square, we now have a level 3 grid of size $\Theta(n^3) \times \Theta(n^3)$ in each level 2 grid square. This is because the lines of sight tweaked in the first tweaking

phase need to be aimed more precisely. The size of the targets they need to hit, in terms of level 2 squares, is now $\Theta(1/n)$ rather than $\Theta(1)$.

For the clause gadgets, the lines of sight are aimed the same way as in the variable gadgets; the only real difference is that the lines of sight have different targets.

For the inversion gadgets, the same is essentially true though the process is more complicated. In the first tweaking phase, we first lower u_1, w_0, and w_1 in unison to aim the line of sight emanating from u_0 and passing through u_1. We then lower the lip vertex adjacent to u_1 to ensure that u_1 can see the output variable gadget but nothing lower.

In the second phase we must place the points p and q. p must be dug out just below w_0 towards the middle of the valley and q must be dug out just below w_1 towards the middle of the valley. Recall from the proof of Lemma 3.4 the key lines of sight between the inversion gadget and the input gadget above have slope $\pm\Theta(1/n)$. Since u_1, w_0, and w_1 are separated by a horizontal distance of $\Theta(1)$ level 2 grid squares, we can have p and q placed $\Theta(1/n)$ level 2 grid squares below u_1, w_0, and w_1. Now, when p and q are placed, we also place a lip vertex next to each to aim lines of sight to and from p and q. These lip vertices can be respectively placed $\Theta(1/n)$ level 2 grid squares away from p and q. When p and q are being 'aimed', they must hit targets of size $\Omega(1)$ level 3 squares, and these are $\Theta(n)$ level 1 squares away, or $\Theta(n^2)$ level 2 squares away, or $\Theta(n^5)$ level 3 squares away. Thus the error allowed in the slope is $\Theta(n^{-5})$. When adjusting p and q, along with their adjacent lip vertices, precision of $\Theta(n^{-6})$ level 2 squares, or $\Theta(n^{-3})$ level 3 squares, is sufficient. This corresponds to a level 4 grid of size $\Theta(n^3) \times \Theta(n^3)$ in each level 3 grid square.

Thus the additional precision required by the inversion gadgets can be handled by increasing our overall grid size from $\Theta(n^6) \times \Theta(n^6)$ to $\Theta(n^8) \times \Theta(n^8)$. □

3.4 Notes

Contributions

The parsimonious discretization method (Lemma 3.1) and the VC-dimension bound (Lemma 3.2) were proved independently by the author. The VC-dimension bound has been published (61).

The major contribution of this chapter is the NP-completeness result (Theorem 3.1). The truth assignment propagation technique was developed independently by the author, then conveyed to Erik Krohn who completed the proof of weak NP-hardness. We subsequently strengthened the proof in order to prove strong NP-hardness. The weak NP-hardness result has been published (63) and the strong NP-hardness result has been submitted.

Future Directions

We have shown that terrain guarding is NP-hard. With the PTAS for terrain guarding given by Gibson *et al.* (53), this essentially resolves the approximability of the problem. The biggest remaining question regarding the complexity of terrain guarding is whether or not it is fixed-parameter tractable. With the size of the minimum guarding set as a fixed parameter, the parameterized version of the problem asks, "For a given input terrain T with n vertices and parameter k, does T have a guarding set of size at most k?" Determining the parameterized complexity would essentially resolve the complexity of terrain guarding.

We do not know if the discretization method in Lemma 3.1 can be improved upon in terms of the number of points generated. Our method can require $\Theta(n^3)$ points. Smaller point sets might be sufficient — perhaps there exists a better parsimonious discretization method that generate sets of size $\mathcal{O}(n^2)$.

Chapter 4

Guarding Polygons

We now move our attention to guarding problems involving polygons, commonly known as *art gallery problems*. Our emphasis is on optimization problems, *i.e.*, finding minimum guarding sets for specific input polygons. We discuss the range spaces associated with these problems, paying special attention to the issue of discretization. We then give an approximation algorithm for guarding simple polygons with perimeter guards; this algorithm improves the state of the art.

Contents

4.1 Polygon Guarding Preliminaries

In computational geometry, art gallery problems, *i.e.*, polygon guarding problems, are motivated by the question, "How many security cameras are required to guard an art gallery?" The art gallery is modeled as a connected polygon P. A camera, which we henceforth call a *guard*, is modeled as a point in the polygon, and we say that a guard g *sees* a point q in the polygon if the line segment $\overline{gq}$ is contained in P. The visibility polygon of a point p, denoted $\mathrm{Vis}(p)$, contains all points in P that see p. We call a set G of points a *guarding set* if every point in P is seen by some $g \in G$. Let $V(P)$ denote the vertex set of P and let ∂P denote the boundary of P. We assume that P is closed and non-degenerate so that $V(P) \subset \partial P \subset P$.

We consider the minimization problem that asks, given an input polygon P with n vertices, for a minimum guarding set for P. Variants of this problem typically differ based on what points in P must be guarded and where guards can be placed, as well as whether P is simple or contains holes. Typically we want to guard either P or ∂P, and our set of potential guards is typically $V(P)$ (vertex guards), ∂P (perimeter guards), or P (point guards). For results on art gallery problems not related to minimization problems we direct the reader to O'Rourke's book (83), which is available for free online.

The problem was proved to be NP-complete first for polygons with holes by O'Rourke and Supowit (84). For guarding simple polygons it was proved to be NP-complete for vertex guards by Lee and Lin (67); their proof was generalized to work for point guards by Aggarwal (3). This raises the question of approximability. There are two major hardness results. First, for guarding simple polygons, Eidenbenz (42) proved that the problem is APX-complete, meaning that we cannot do better than a constant-factor approximation algorithm unless P = NP. Subsequently, for guarding polygons with holes, Eidenbenz *et al.* (43) proved that the minimization problem is as hard to approximate as set cover in general if there is no restriction on the number of holes. It therefore follows from results about the inapproximability of set cover (see

Section 2.1) that, for polygons with holes, it is NP-hard to find a guarding set of size $o(\log n)$. These hardness results hold whether we are dealing with vertex guards, perimeter guards, or point guards.

Range Spaces

Guarding problems can naturally be expressed as instances of set cover or hitting set. We wish to model an instance of a guarding problem as an instance of hitting set. The desired range space $\mathcal{S} = (S_G, \mathcal{R})$ is constructed as follows. S_G contains the potential guard locations. For each point p that needs to be guarded, R_p is the set of potential guards that see p. Now $\mathcal{R} = \{R_p : p \in S_T\}$, where S_T is the set of points that must be guarded.

It is known that range spaces associated with the guarding of simple polygons with point guards have constant VC-dimension. This was first proved by Kalai and Matoušek, who were able to apply a Ramsey-type result after using, among other arguments, the following order claim for simple polygons:

Claim 4.1 (Polygon Order Claim). *Consider points a, b, c, d, e, f in clockwise order on the perimeter of a simple polygon P. If $a \sim c$, $b \sim d$, $e \sim a$, and $f \sim d$, then $a \sim d$. The same also holds for the corresponding points in a polygon P' that contains P.*

Proof. This claim is essentially two copies of the order claim for terrains (Claim 3.1), glued together along the line segment $\overline{ad}$. Line segments $\overline{ac}$ and $\overline{bd}$ intersect at a point p and line segments $\overline{ea}$ and $\overline{fd}$ intersect at a point q. Now the points $\{a, p, d, q\}$ form a convex quadrilateral containing the line segment $\overline{ad}$; the interior of the quadrilateral is disjoint from ∂P so $a \sim d$. □

Bounds on the maximum VC-dimension were improved by Valtr (96), who showed that it is at least 6 and at most 23. The lower bound is given by example and the upper bound is given by a combinatorial argument. A sketch of the upper bound argument for the maximum dimension d is as follows. Consider a set Y of d points that are shattered by a set Z of 2^d points. Decompose the plane by cutting along $\mathcal{O}(d^2)$ lines defined in some way by the points in Y. The decomposition ensures that points inside a single region see 'similar'

subsets of Y; this means no region can contain too many points from Z. Since the decomposition has only $\mathcal{O}(d^4)$ regions, Valtr is able to prove that Z cannot fit in the decomposition for any $d \geq 24$.

The upper bound of 23 applies *a fortiori* to range spaces for perimeter guards and vertex guards. Thus when guarding simple polygons we can construct ε-nets of size $\mathcal{O}\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ using general techniques. In a polygon with h holes the VC-dimension is $O(\log h)$ (96) and therefore ε-nets of size $\mathcal{O}\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\log h\right)$ can be constructed using general techniques.

Discretization

If either S_G or S_T is infinite, some discretization may be necessary to find a hitting set for $\mathcal{S}$, or even to construct $\mathcal{S}$ in the first place. Discretization is the task of building an appropriate finite range space $\mathcal{S}'$ defined by finite subsets $S'_G \subseteq S_G$ and $S'_T \subseteq S_T$. The goal is for $\mathcal{S}'$ to approximate $\mathcal{S}$; we clarify this shortly. Once we have constructed a range space $\mathcal{S}'$ that satisfies these criteria, we can find an approximately minimum hitting set H for $\mathcal{S}'$, for example using general techniques (see Section 2.1). Let OPT and OPT$'$ be the sizes of optimum hitting sets for $\mathcal{S}$ and $\mathcal{S}'$ respectively, and assume our approximation algorithm run on $\mathcal{S}'$ guarantees a hitting set of size $f(\text{OPT}')$.

The first criterion for $\mathcal{S}'$ is that OPT$'$ must not be much larger than OPT. Ideally OPT$'$ = OPT, but in many cases OPT$'$ = $\mathcal{O}$(OPT) is also acceptable. This criterion ensures that our discretization does not ruin the guaranteed approximation factor. The second criterion is that any hitting set for $\mathcal{S}'$ must also be a hitting set for $\mathcal{S}$. In other words, any subset of S'_G that guards S'_T must also guard S_T. This criterion ensures that our discretization does not break correctness.

The process of constructing $\mathcal{S}'$ and finding an approximate hitting set H for $\mathcal{S}'$ is now an approximation algorithm for hitting set on $\mathcal{S}$, and the approximation factor is

$$\frac{f(\text{OPT}')}{\text{OPT}}.$$

The technique of Brönniman and Goodrich (see Section 2.2) does not actually require S_T to be finite, so long as the net finder and verifier run in

polynomial time. For the problem of guarding polygons this polynomial behaviour is easy to achieve. Unfortunately S_G must be finite regardless to ensure that the algorithm stops after a finite number of iterations.

Equivalence Cells For a Finite Set

Consider the case where S_G is finite; we assume S_G guards the entire polygon. For points $p, q \in S_T$, we say that p and q are equivalent if and only if $R_p = R_q$. In a decomposition of a polygon into cells, we say that a cell is an *equivalence cell* if all points are equivalent. A natural discretization strategy is to partition S_T into a finite number of sets that are closed under equivalence, and then to build a subset S'_T by taking one representative point from each set. A subset of S_G guards S_T if and only if it guards S'_T.

Ghosh (51) did this for the vertex guarding problem in which $S_G = V$ and $S_T = P$. His algorithm decomposes the input polygon into a polynomial number of cells such that each point in a given cell is seen by the same set of vertices. For $p \in S_G$ and vertex v that see each other, consider the ray shot from p through v. The set of all such rays decomposes the polygon into a polynomial number of cells. We call this the *ray shooting decomposition* and denote it $\mathcal{D}_R(P, S_G)$.

It is not hard to see that each cell is closed under equivalence. If two points p and q are not equivalent there is a vertex v that sees one but not the other; assume without loss of generality that v sees p. Consider the geodesic from p to q. Of all the points on this geodesic seen by v, let p' be the closest to q. Then p' must lie on a ray shot from v through another vertex; this ray separates p from q so p and q cannot be in the same cell. We can also consider the visibility polygon for v. Any point on the boundary of v's visibility polygon must either be on the polygon's perimeter or on a ray shot from v through another vertex. Therefore a point moving around P cannot enter or leave v's visibility polygon without crossing one of the rays used to decompose the polygon.

Ghosh originally used his discretization technique, along with the greedy set cover approximation algorithm (see Section 2.1), to provide a $\mathcal{O}(\log n)$-approximation algorithm; the approximation factor improves to $\mathcal{O}(\log \textsc{opt})$ if

more recent methods are applied, or $\mathcal{O}(\log h \log \text{OPT})$ for vertex guarding a polygon with h holes.

A ray shooting decomposition can be generalized to work for any finite set S_G — simply shoot a ray from every point $p \in S_G$ through every vertex seen by p. In each resulting cell any two points see the same subset of S_G.

Using a simple inequality for general line arrangements it can be seen that the number of cells in $\mathcal{D}_R(P, S_G)$ is $\mathcal{O}(n^2|S_G|^2)$. Bose *et al.* (16) introduced a minimal decomposition with fewer cells. Let the *visibility decomposition*, denoted $\mathcal{D}_V(P, S_G)$, be the minimum decomposition of a polygon into equivalence cells with regard to S_G. It is minimum in that the union of cell boundaries in this decomposition is exactly equal to $\bigcup_{p \in S_G} \partial(\text{Vis}(p))$. The decomposition can be constructed as follows. For a point $p \in S_G$ that sees a vertex v, instead of cutting along the entire ray shot from p through v, we leave the line segment $\overline{pv}$ and cut only from v until we hit ∂P. Bose *et al.* call such a cut a *window of point* p; it is not difficult to see that the boundary of $\text{Vis}(p)$ consists only of windows of p and parts of ∂P. Considering simple polygons, they proved the following upper bound for the number of cells and provided an example showing the bound is tight (16, Figure 10). A straightforward modification to their proof extends the upper bound to polygons with holes.

Lemma 4.1. *For a polygon P with h holes and a finite set S_G, $\mathcal{D}_V(P, S_G)$ contains $\mathcal{O}((h+1)n|S_G|^2)$ cells.*

Proof. The key to bounding the number of cells in $\mathcal{D}_V(P, S_G)$ is counting the number of points at which windows cross each other. For a fixed window w and a fixed point $p \in S_G$, if P is simple then at most 2 windows of p can cross w; more generally, for a polygon with h holes, this bound is $2(h+1)$. To see this, recall that crossing a window of p toggles visibility from p. Thus if more than $2(h+1)$ windows of p cross w, the points on w seen by p must form at least $h+2$ disconnected intervals. Between any two of these intervals there must be a hole blocking visibility from p. Since the polygon has fewer than $h+1$ holes, w must be crossed by at most $2(h+1)$ windows from p.

Each window is crossed by at most $2(h+1)|S_G|$ other windows. The total number of windows is $\mathcal{O}(n|S_G|)$, so the number of points at which windows

cross is $\mathcal{O}((h+1)n|S_G|^2)$. The lemma follows by applying Euler's formula for planar graphs. □

Discretizing Guard Locations

Discretization is more complicated when neither S_T nor S_G is a finite set. One cannot simply determine equivalence cells when the number of equivalence classes is infinite. The problems in which $(S_G, S_T) \in \{P, \partial P\}^2$ are very natural, yet no fully polynomial discretization has yet been achieved.

Pseudopolynomial Discretization

Pseudopolynomial discretization was achieved by Deshpande *et al.* (24). Given a triangulated decomposition we say that a triangle is 'good' if any set of guards in the triangle is dominated by three guards at the triangle's vertices. Their algorithm starts with a triangulation of the visibility decomposition $\mathcal{D}_V(P, V)$. They then perform additional decomposition steps until all triangles are good. The decomposition algorithm returns the triangle vertices as S_G. The number of decomposition steps and the size of S_G are pseudopolynomial in that they may be linear in the polygon's *spread*, *i.e.*, the ratio between the longest and shortest distances between two vertices. This ratio can be exponential in the input size. The decomposition is such that $\textsc{opt}' \leq 3 \cdot \textsc{opt}$; any subset of S_G that guards S_T can be replaced by at most three times as many guards in S'_G.

Considering a Grid

Efrat and Har-Peled (41) developed a discretization technique that defines S_G as the points of a very fine grid. They then use standard approximation techniques based on ε-nets to find a guarding set of size $(\textsc{opt}')\log(\textsc{opt}')$. This is a heuristic — the authors do not bound $\textsc{opt}'$ as a function of $\textsc{opt}$.

For a $k \times k$ grid, the algorithm's dependence on k is only $\mathcal{O}\big(\log^2 k\big)$. This is achieved through careful implicit representation of the grid points and their weights. It is likely that, for some k that is linear in the polygon's spread, there must exist a triangulated decomposition based on the grid points that contains only good triangles. This would imply that $\textsc{opt}' \leq 3 \cdot \textsc{opt}$. Thus,

while the discretization bound would be pseudopolynomial, the approximation algorithm would actually run in fully polynomial time.

4.2 Improved Approximation for Perimeter Guards

When guarding simple polygons we can construct ε-nets of size $\mathcal{O}\big(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\big)$ using general techniques for range spaces of constant VC-dimension. Using techniques specific to vertex guarding or perimeter guarding a simple polygon, it is possible to break through the general $\Theta\big(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon}\big)$ lower bound to build smaller ε-nets. The following theorem is the result of joint work with David Kirkpatrick (62).

Theorem 4.1. *For the problem of guarding a simple polygon with vertex guards or perimeter guards, there exist polynomial-time algorithms to construct ε-nets of size $\mathcal{O}\big(\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon}\big)$, and we describe such an algorithm explicitly.*

Proof outline. In the first part of the proof we introduce the basic ideas that allow the construction of ε-nets of size $\mathcal{O}(1/\varepsilon^2)$. In the second part we give a more complicated, hierarchical technique that lets us construct ε-nets of size $\mathcal{O}\big(\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon}\big)$. □

Using standard techniques such as those given in Section 2.2, the algorithm we provide in the proof of Theorem 4.1 becomes a $\mathcal{O}(\log\log \textsc{opt})$-approximation algorithm whose running time is polynomial in n and the number of potential guard locations. This is the best approximation factor obtained for vertex guards and perimeter guards. If no finite set of guard locations is given, we use the discretization technique of Deshpande *et al.* and our algorithm is polynomial in n and Δ, where Δ is the ratio between the longest and shortest distances between vertices.

A similar result for a different problem was recently obtained by Aronov *et al.* (11), who proved the existence of ε-nets of size $\mathcal{O}\big(\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon}\big)$ when $\mathcal{S}$ is either a set of axis-parallel rectangles in $\mathbb{R}^2$ or axis-parallel boxes in $\mathbb{R}^3$.

Building Quadratic ε-Nets

Here we show how to build an ε-net using $\mathcal{O}(1/\varepsilon^2)$ guards. This result is not directly useful to us but we use this as an opportunity to perform the geometric leg work, and hopefully provide some intuition, without worrying about the hierarchical decomposition to be described later. It should be clear that these ε-nets can be constructed in polynomial time.

Subdividing the Perimeter.

For the construction both of the ε-nets here and those later on we subdivide the perimeter into a number of *fragments*. Fragment endpoints always lie on vertices, but the weight of a guard location may be split between multiple fragments and a fragment may consist of a single vertex. The key difference between the construction of the ε-nets here and those later is the method of fragmentation. Here, the perimeter is simply divided into $m = 4/\varepsilon$ fragments each having weight $\frac{\varepsilon}{4}w(G)$. For our purposes, $1/\varepsilon$ is always an integer so m is always an integer.

Placing Extremal Guards.

For two fragments A_i and A_j we place guards at *extreme points of visibility*. Those are the first and last points on A_i seen from A_j and the first and last points on A_j seen from A_i. For a contiguous fragment we define the first (resp. last) point of the segment according to the natural clockwise ordering on the perimeter. We use $G(A_i, A_j)$ to denote the set of up to 4 extremal guards placed between A_i and A_j.

These extreme points of visibility might not lie on vertices. In fact, it is entirely possible that two fragments A_i and A_j see each other even if no vertex of A_i sees A_j and vice versa. If an extreme point of visibility is not a potential guard location, we simply ignore it. Our proofs, in particular the proof of Lemma 4.3, only require guards on extreme points of visibility that either lie on vertices or on fragment endpoints.

All Pairs Extremal Guarding.

Our current aim is to build an ε-net by placing extremal guards for every pair (A_i, A_j) of fragments. We denote this set of guards with

$$S_{AP} = \bigcup_{i \neq j} G(A_i, A_j) \ .$$

Note that $|S_{AP}| \leq 4\binom{m}{2} = \mathcal{O}(1/\varepsilon^2)$. Also note that every fragment endpoint is included in S_{AP}.

Lemma 4.2. *Any point not guarded by S_{AP} sees at most 4 fragments.*

Corollary 4.1. *S_{AP} is an ε-net of size $\mathcal{O}(1/\varepsilon^2)$.*

For the proof of Lemma 4.2 we need to present additional properties of the fragments that can be seen by a point. For a point x, the fragments seen by x are ordered clockwise in the order they appear on the boundary of P. We need to consider lines of sight from x, and what happens when a transition is made from seeing one fragment A_i to seeing the next fragment A_j. There are three possibilities:

1. $j = i + 1$ and x sees the guard at the common endpoint of A_i and A_j
2. A_j occludes A_i, in which case we say that x has a *left tangent* to A_j (see Figure 4.1)
3. A_i was occluding A_j, in which case we say that x has a *right tangent* to A_i (see Figure 4.1).

We say a fragment A *owns* a point x if x sees A in a sector of size at least π. We assume any point x is owned by at most one fragment; if x is a fragment endpoint it is itself a guard, and otherwise if x is owned by two fragments then only those two fragments can see it.

Lemma 4.3. *Let A_i, A_j, A_k be fragments that are seen by x consecutively in clockwise order. If x has a left tangent to A_j, and the combined angle of A_j and A_k at x is no more than π, then x sees a guard in $G(A_j, A_k)$. Symmetrically, if x has a right tangent to A_j, and the combined angle of A_i and A_j at x is no more than π, then x sees a guard in $G(A_i, A_j)$.*

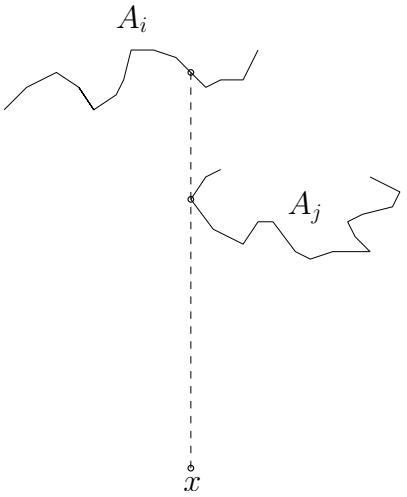


Figure 4.1: The point x has a left tangent to A_j.

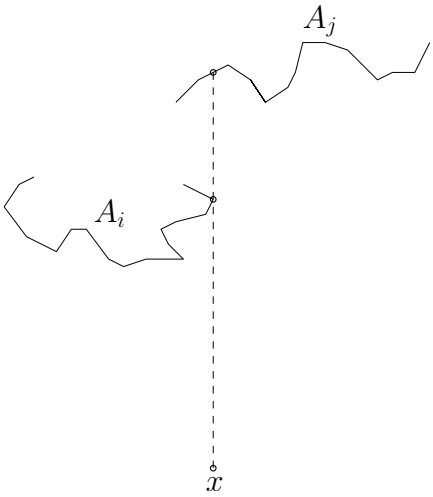


Figure 4.1: The point x has a right tangent to A_i.

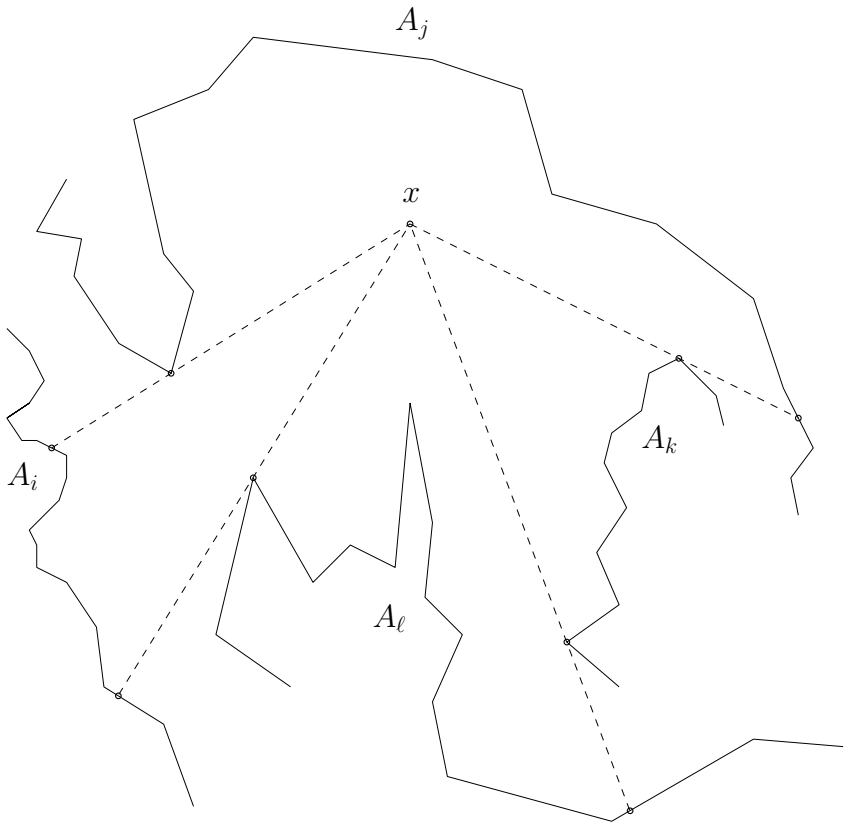


Figure 4.2: The point x has no tangent to A_i, a left tangent to A_j, both a left and right tangent to A_k, and a right tangent to A_ℓ. A_j owns x.

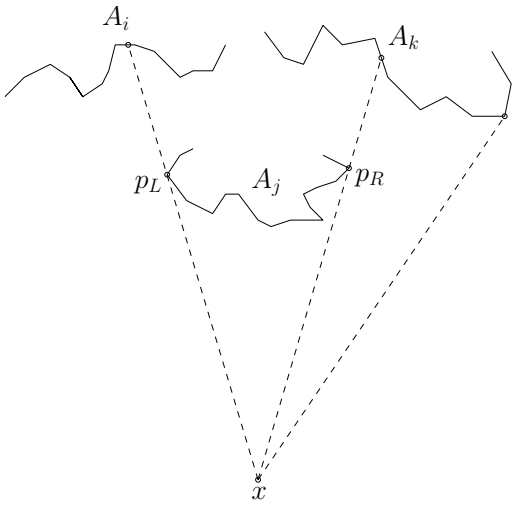


Figure 4.3: Case 1 in the proof of Lemma 4.3. The point x has a left tangent and a right tangent to A_j.

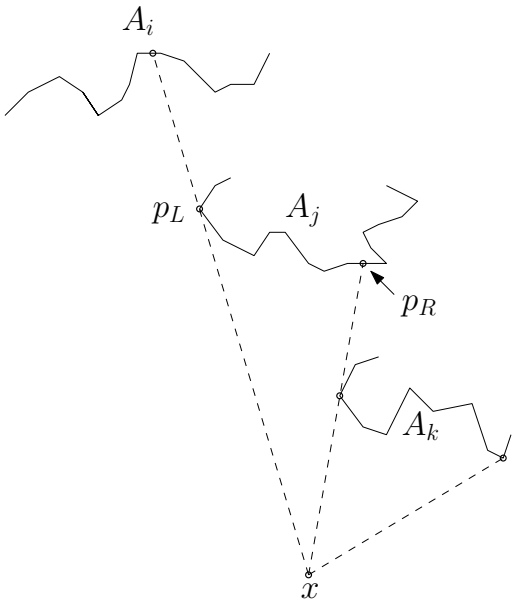


Figure 4.3: Case 2 in the proof of Lemma 4.3. The point x has left tangents to both A_j and A_k.

Proof. We can assume w.l.o.g. that x has a left tangent to A_j since the proof of the other case is symmetric. There are now two cases we have to deal with, depending on whether x has a right tangent to A_j (case 1) or a left tangent to A_k (case 2). Define p_L and p_R respectively as the first and last points on A_j seen by x. Observe that x must see every vertex on the geodesic between p_L and p_R. Let q be the first point on A_j seen from A_k. In both cases 1 and 2 (see Figures 4.3 and 4.3), q must be a vertex of the geodesic between p_L and p_R. This can be shown by contradiction; if q lies between consecutive vertices of this geodesic then those two consecutive vertices must also be seen from A_k, and one of them comes before q.

The restriction that the combined angle of A_j and A_k at x is no more than π is necessary to ensure that the geodesic of interest from A_k to A_j does not 'pass behind' x to see a point on A_j before p_L.

It should be emphasized that, since there is a left tangent to A_j, p_L is always a vertex. Also, if p_R is not a vertex it cannot be the first point on A_j seen from A_k. □

The proof of Lemma 4.2 is now fairly straightforward.

Proof of Lemma 4.2. Let x be a point that sees at least 5 fragments. Assume x is not a fragment endpoint, otherwise it is itself a guard in S_{AP}. If we have a directed graph whose underlying undirected graph is a cycle, then either we have a directed cycle or we have a vertex with in-degree 2. By the same principle, either some fragment seen by x has no tangent from x, or every fragment seen by x has a left tangent from x (or every one has a right tangent, which can be handled symmetrically).

If a fragment seen by x has no tangent from x, call such a fragment A_0 and let A_{-2}, A_{-1}, A_0, A_1, A_2 be fragments seen by x in clockwise order. If the combined angle at x of A_{-2} and A_{-1} is more than π, the combined angle of A_1 and A_2 is less than π. So we can apply Lemma 4.3 with one of the two pairs of fragments to show that x is seen by a guard.

If every fragment seen by x has a left tangent from x, then we can apply Lemma 4.3 using two consecutive fragments with a combined angle at x of less than π.

□

Before we move on we prove one more helpful lemma.

Lemma 4.4. *The number of fragments seen by an unguarded point x that do not have a tangent from x is at most 1.*

Proof. Assume the contrary and let A_0 and A_i be two such fragments. If one such fragment owns x, assume it is A_0 and call the next two fragments seen by x in the clockwise direction A_1 and A_2 respectively. By Lemma 4.3, x is seen by a guard in $G(A_1, A_2)$ so we reach a contradiction. If no such fragment owns x then assume w.l.o.g. that, over the fragments seen by x between A_0 and A_i going clockwise, the combined angle at x is less than π (if this is not true it must be true going counterclockwise). Again, x is seen by a guard in $G(A_1, A_2)$ so we reach a contradiction. □

Smaller ε-Nets via Hierarchical Fragmentation

Previously we showed how a quadratic number of guards (*i.e.*, $\mathcal{O}(1/\varepsilon^2)$) could be placed to ensure that any unguarded point sees at most 4 fragments. Now we discuss how hierarchical fragmentation can be used to reduce the number of guards required to $\mathcal{O}\big(\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon}\big)$. We use S_{HF} to denote the guarding set constructed here. It should be clear that these ε-nets can be constructed in polynomial time.

We can consider the hierarchy as represented by a tree. At the root there is a single fragment representing the entire perimeter of the polygon. This root fragment is broken up into a certain number of child fragments. Fragmentation continues recursively until a specified depth t is reached. We set $t = \big\lceil \log\log\frac{1}{\varepsilon}\big\rceil$. The *fragmentation factor* (equivalently, the branching factor of the corresponding tree) is not constant, but rather depends on both t and the level in the hierarchy. The fragmentation factor generally decreases as the level of the tree increases. Specifically, if b_i is the fragmentation factor at the i^{th} step, we have

$$b_i = \begin{cases} 2^{2^{t-1}+1} \cdot 4t \cdot 2^{1-t} \cdot \alpha & , \quad i = 1 \\ 2^{2^{t-i}+1} & , \quad 1 < i \leq t \ , \end{cases}$$

where $\alpha \leq 1$ is a term introduced only to deal with an issue arising from ceilings and double exponentials, namely the fact that $2^{2^{\lceil \log\log 1/\varepsilon \rceil}}$ is not in $\mathcal{O}(1/\varepsilon)$. We specify α shortly (see (4.3)).

If f_i is the total number of fragments after the i^{th} fragmentation step, this gives us

$$f_i = \begin{cases} 1 & , \quad i = 0 \\ 4t \cdot 2^{2^t - 2^{t-i} - t + i + 1} \cdot \alpha & , \quad 0 < i \leq t \\ 4t \cdot 2^{2^t} \cdot \alpha & , \quad i = t \ , \end{cases}$$

since

$$\begin{aligned} f_i &= \prod_{j=1}^{i} b_j \\ &= 4t \cdot 2^{1-t} \cdot \alpha \cdot \prod_{j=1}^{i} 2^{2^{t-j}+1} \\ &= 4t \cdot 2^{1-t+\sum_{j=1}^{i}(2^{t-j}+1)} \cdot \alpha \\ &= 4t \cdot 2^{2^t - 2^{t-i} - t + i + 1} \cdot \alpha \quad . \end{aligned}$$

Our algorithm places guards for all pairs of *sibling fragments*, *i.e.*, fragments having the same parent fragment. For the purposes of this guard placement, the complement of the parent fragment, *i.e.*, the subset of G outside the parent fragment, is considered a dummy child fragment. That is, it is considered a child fragment when placing guards, but not when counting the number of child fragments seen from some point x as in the statement of Corollary 4.2 or in the proof of Lemma 4.5. To denote the complement of a fragment A we use $\overline{A}$. Considering $\overline{A}$ to be a child of A when placing guards allows us to consider the children of A as if they were fragments with guards placed for all pairs. For example, we can obtain the following corollary from Lemmas 4.2 and 4.4.

Corollary 4.2. *For an unguarded point x and a fragment A, the number of child fragments of A seen by x is at most 3, and at most one of these child fragments does not have a tangent from x.*

The total number of guards placed is

$$|S_{HF}| \leq 4\sum_{i=1}^{t} \binom{b_i + 1}{2} f_{i-1} \leq 4\sum_{i=1}^{t} b_i^2 f_{i-1} \quad .$$

If $t \geq 6$ we have $b_i \leq 2^{2^{t-i}+1}$ for all values of i. This gives us

$$\begin{aligned} |S_{HF}| &\leq 4\alpha \sum_{i=1}^{t} 2^{2\left(2^{t-i}+1\right)} \cdot 4t \cdot 2^{2^t - 2^{t-i+1} - t + i} \\ &= 16t\alpha \sum_{i=1}^{t} 2^{2^t - t + i + 2} \\ &= 16t\alpha \cdot 2^{2^t - t + 3}(2^t - 1) \\ &< 16t\alpha \cdot 2^{2^t + 3} \\ &= 128t\alpha \cdot 2^{2^t} \ . \end{aligned}$$

Recall that $t = \left\lceil \log\log \frac{1}{\varepsilon} \right\rceil$. We need to define α in a way that ensures b_1 is an integer and also ensures the following two equations hold:

$$|S_{HF}| = \mathcal{O}\left(\frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}\right) \tag{4.1}$$

$$\frac{f_t}{4t} \geq \frac{1}{\varepsilon} \ . \tag{4.2}$$

To satisfy these three criteria, it suffices to set

$$\alpha = \frac{\left\lceil 2^{2^{t-1}+1} \cdot 4t \cdot 2^{-t} \cdot 2^{\log(1/\varepsilon) - 2^t} \right\rceil}{2^{2^{t-1}+1} \cdot 4t \cdot 2^{-t}} = \frac{\left\lceil 4t \cdot 2^{\log(1/\varepsilon) + 1 - t - 2^{t-1}} \right\rceil}{4t \cdot 2^{2^{t-1}+1-t}} \ . \tag{4.3}$$

We must now provide a generalization of Lemma 4.2 that works with our hierarchical fragmentation.

Lemma 4.5. *Any point that is not guarded by S_{HF} sees at most $4i$ fragments at level i.*

Applying this with $i = t$ and using (4.1) and (4.2), we get

Corollary 4.3. *S_{HF} is an ε-net of size $\mathcal{O}\left(\frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}\right)$.*

Proof of Lemma 4.5. Let x be a point that does not see any guard in S_{HF}. From the tree associated with the hierarchical fragmentation, we consider the subtree of fragments that see x. We define a *branching fragment* as a fragment with multiple children seen by x and we claim that at any level there are at

most 2 branching fragments. Corollary 4.2 tells us that any fragment has at most 3 children seen by x. At level 1 there are at most 4 fragments seen by x, so it follows that the number of fragments seen by x at level i is at most $4i$. We must now prove our claim that there are at most 2 branching fragments at any level.

First we note that a branching fragment either has no tangent from x or owns x. To see this, consider a fragment A that has a tangent from x and does not own x. Assume w.l.o.g. that x has a left tangent to A and call the point of tangency p_L. x must then also have a left tangent to the child fragment A_0 of A that contains p_L. A_0 must be the leftmost child fragment of A seen by x. If x sees another child fragment A_1 of A to the right of A_0, then by Lemma 4.3 it is seen by a guard in $G(A_0, A_1)$.

Consider now the following possibilities for a given fragment A.

1. A is not seen by x. Clearly x cannot see any child fragments of A.

2. A does not own x, and x has a tangent to A. A then has exactly one child fragment that sees x, and this fragment is of type (2).

3. A does not own x, and x does not have a tangent to A. By Corollary 4.2, x can see at most 3 child fragments of A. At most one of these children is of type (3) and all others must be of type (1) or (2).

4. A owns x and has no tangents from x, *i.e.* $\overline{A}$ has two tangents from x. If a child of A owns x it must be the only child of A that sees x, and this child is also of type (4). Otherwise, A would have a child fragment A_i that is seen by x, does not own x, and is adjacent to $\overline{A}$. x would then be seen by a guard in $G(A_i, \overline{A_P})$. Thus A has at most one child that is not of type (1) or (2).

5. A owns x and has two tangents from x. Because $\overline{A}$ is, in a sense, a 'dummy' child of type (3), A cannot have a real child of type (3) by the proof of Lemma 4.4. Further, if A has a child A_0 that owns x, this child must also be of type (5). Otherwise assume w.l.o.g. that A_1, immediately clockwise from A_0, has a left tangent from x. Then, using A_2 to denote the fragment clockwise from A_1 (A_2 might be $\overline{A_P}$), x is seen by $G(A_1, A_2)$.

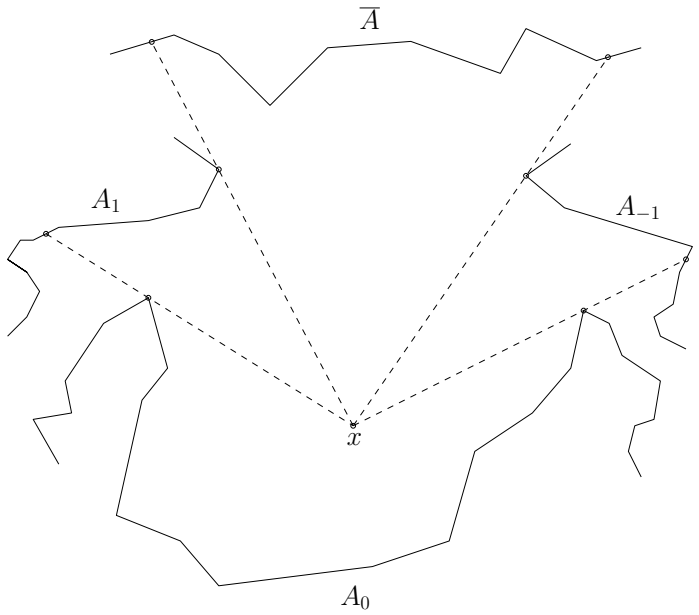


Figure 4.4: The only way a fragment of type (5) can have three children seen by x.

6. A owns x and has exactly one tangent from x (see Figure 4.5). We consider how A can have multiple children seen by x. Assume w.l.o.g. that $\overline{A}$ has a right tangent. If A_{-1} is the child of A seen by x immediately counterclockwise from $\overline{A}$ then A_{-1} must own x, otherwise x is seen by $G(A_{-1}, \overline{A})$. If A_1 is the child of A seen by x immediately clockwise from $\overline{A}$ then A_1 cannot have a tangent from x otherwise x would be seen by $G(\overline{A}, A_1)$. If x can see A_2, a child of A between A_1 and A_{-1}, then x must have two tangents to A_{-1} otherwise it would be seen by $G(A_1, A_2)$.

   Therefore if A has more than one child seen by x, there must one of type (3) and one of type (5), plus (possibly) a child of type (2).

We call a non-root fragment *fruitful* if it or one of its descendants is a branching fragment. Only fragments of type (3-6) can be fruitful. Only fragments of type (6) can have more than one fruitful child, and they can have at most two fruitful children. No non-root fragment can have a child fragment of type (6). Also, if the root has a child fragment of type (6), the root cannot have a child of type (3). Therefore any level has at most 2 fruitful fragments.

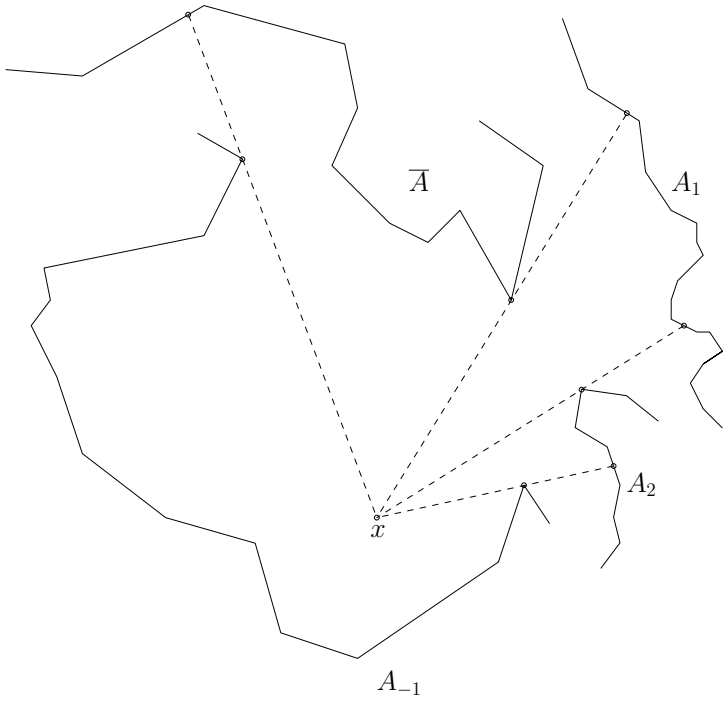


Figure 4.5: The only way a fragment of type (6) can have three children seen by x.

We can now state the following:

- Level 1 has at most 4 child fragments that see x, at most 2 of which are fruitful.

- A fruitful fragment has at most 3 child fragments that see x, at most 1 of which is fruitful.

- A non-fruitful fragment has at most 1 child fragment that sees x.

Therefore any level has at most 2 fruitful fragments and the number of fragments at level i that see x is at most $4i$.

□

4.3 Notes

Contributions

Lemma 4.1 and its proof are due to Bose, Lubiw, and Munro (16). However, they considered only simple polygons. We made a small modification to their proof that generalizes the result to polygons with holes.

Section 4.2 is the result of joint work with David Kirkpatrick, who independently obtained and published a preliminary version of the results in 2000 (64). These results did not include any connections to ε-nets or approximation algorithms. The significance of the results was pointed out by the thesis author, who suggested that they be given a more thorough treatment. The task of adding generalizations and more rigorous analysis was undertaken jointly, with the majority of additional proofs and writing done by the thesis author.

Future Directions

We have obtained a $o(\log \textsc{opt})$-approximation factor for vertex guards and perimeter guards, so it is natural to ask if we can do the same for point guards. Different techniques would be required since points guards do not have a natural cyclic ordering like perimeter guards. We would also like to do better than $\mathcal{O}(\log \log \textsc{opt})$ for perimeter guards. In particular, is there a constant factor approximation algorithm to match the hardness of approximation result of Eidenbenz (42)? And can the lower bounds be improved? It seems likely that Alon's lower bound (see Lemma 2.4) could imply a lower bound for point guards. Is this the case, and would this imply an improved inapproximability result for point guards?

There are certain natural variants of the problem of guarding polygons that should be explored. One variant is when redundancy is required, *i.e.*, when points must be guarded by more than one guard. Another variant is when guards have a restricted angle of visibility or finite range.

For simple polygons, the range spaces associated with point guards have maximum VC-dimension at least 6 and at most 23 (96); it is believed that the true value is closer to the lower end of this range, perhaps even 6 (58). The

upper bound of 23 holds *a fortiori* for range spaces associated with perimeter guards but the lower bound of 6 does not. A lower bound of 4 follows from a trivial modification to an example for monotone chains (61); we can increase this bound to 5 without too much difficulty (see Figure 4.6). Can range spaces associated with perimeter guards actually have VC-dimension as high as 6? And can the upper bound of 23 be improved? It seems that improving the upper bound would be easier for perimeter guards than for point guards. It is also possible that Valtr's bound of 23 could be improved by using a better decomposition. His proof uses an upper bound for the number of cells that comes from general line arrangements; replacing this with the decomposition and analysis of Bose *et al.* (16) might yield a better upper bound on d.

Ghosh (52) used the visibility decomposition of Bose *et al.* (16) to find approximately optimal sets of vertex guards. After discretization he treats the problem as set cover and greedily obtains a $\mathcal{O}(\log n)$-approximation. The algorithm runs in $\mathcal{O}(n^4)$ time for simple polygons and $\mathcal{O}(n^5)$ time for polygons with holes. It seems likely that the algorithm could be improved in two significant ways. Firstly, an approximation factor of $\mathcal{O}(\log \textsc{opt})$ can be obtained, possibly without a significant increase in the running time. Secondly, Bose *et al.* actually proved that there are only $\mathcal{O}(|S_G|^2)$ cells in the decomposition $\mathcal{D}_V(P, S_G)$ that have *minimal* visibility. If the vertices are in general position, the sets of guards associated with any two neighbouring cells are such that one is a subset of the other. To guard the entire polygon it should be sufficient to guard all cells with minimal visibility; this fact could potentially be exploited to improve the running time.

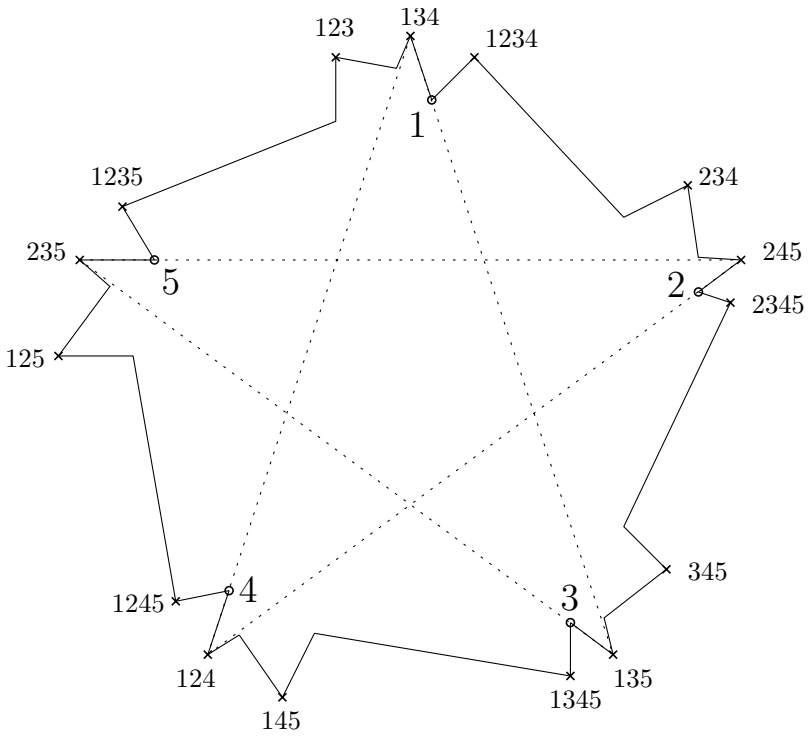
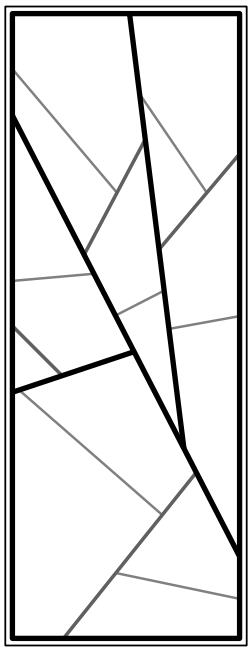


Figure 4.6: A polygon with a set S of 5 points on the perimeter. The points in $S = \{1, 2, 3, 4, 5\}$ are marked with circles and labeled with large numbers. Each point in S sees all of S, and each guard seeing a subset of S of size 3 or 4 is marked with a cross and labeled with small numbers indicating which points in S it sees. Guards seeing the 16 subsets of S of size 0, 1, or 2 are not shown. Adding these is a simple matter of adding nooks with very small angles of visibility, thus we can construct a polygon with 5 points on the perimeter shattered by 2^5 perimeter guards. Such a polygon can also be obtained via a fairly straightforward modification of the example of Kalai and Matoušek for point guards (58).

Part II
Geometric Split Trees



Chapter 5

Probabilistic Groundwork

In this chapter we present some rudimentary material related to random variables and probability distributions. We also define discrete (*i.e.*, quantized) versions of beta distributions and Dirichlet distributions that are closely related to our subsequent analysis of random hyperplane splits.

Contents

Concept	Definition
uniformly at random (u.a.r.)	Given a set S, we say that a subset $X \subseteq S$ of k elements is generated by sampling *u.a.r.* **without** replacement from S iff X is distributed uniformly over all k-element subsets of S. We say that a k-tuple $X \in S^k$ is generated by sampling u.a.r. **with** replacement from S iff X is distributed uniformly over all k-tuples in S^k.
i.i.d.	The abbreviation i.i.d. stands for *independent and identically distributed*. We say random variables $X_1, X_2, \ldots$ are generated i.i.d. iff each X_i is sampled independently from the same distribution.
order statistic (*e.g.*, $X_{(k)}$)	Given a sample of n real-valued random variables $X_1, \ldots, X_n$, the k^{th} *order statistic* is the k^{th} smallest of the n values. We typically denote order statistics using bracketed subscripts, *e.g.*, $X_{(k)}$.
convergence in law (*e.g.*, $X_n \xrightarrow{\mathcal{L}} X$)	Let X be a random variable and let $X_1, X_2, \ldots$ be a sequence of random variables where the distribution of X_i may depend on i. We say that X_n converges in law to X (denoted $X_n \xrightarrow{\mathcal{L}} X$) as $n \to \infty$ if, for all x, $\lim_{n\to\infty} F_{X_n}(x) = F_X(x)$.

Table 5.1: Definitions relevant to probability distributions and the sampling of random variables.

5.1 Basic Distributions and Concepts

Definitions

In Table 5.1 we define some concepts related to probability distributions and sampling. In Table 5.2 we define some properties of distributions.

Property	**Notation**	**Definition**
cumulative distribution function (c.d.f.)	$F_X(x)$ $F(x)$	The *c.d.f.* (cumulative distribution function) of X, denoted $F_X(x)$ or simply $F(x)$, is defined as $F_X(x) = \mathbb{P}\{X \leq x\}$.
probability density function (p.d.f.)	$f_X(x)$ $f(x)$	If X is absolutely continuous, then there exists a function $f \geq 0$, the density (or p.d.f.) of X, such that $\mathbb{P}\{X \in A\} = \int_A f(x)\,dx$ for all Borel sets A. In particular, $\mathbb{P}\{X \in [a,b]\} = F(b) - F(a)$, and $f(x) = F'(x)$ at almost all x.
probability mass function (p.m.f.)	$f_X(x)$ $f(x)$	If X is discrete, the *p.m.f.* (probability mass function) of X, denoted $f_X(x)$ or simply $f(x)$, is defined by $f_X(x) = \mathbb{P}\{X = x\}$. The p.m.f. gives us $\mathbb{P}\{x_1 < X \leq x_2\} = \sum_{x_1 < x \leq x_2} f_X(x) = F_X(x_2) - F_X(x_1)$.
mean	μ_X μ $\mathbb{E}\{X\}$	The *mean* of X, denoted μ_X or simply μ, is also called the expected value of X, denoted $\mathbb{E}\{X\}$. If X is a discrete random variable then $\mathbb{E}\{X\} = \sum_x x f_X(x)$ if the sum is finite. If X is a continuous random variable then $\mathbb{E}\{X\} = \int_{-\infty}^{\infty} x f_X(x)\,dx$ if the integral is finite.
variance	$\mathbf{Var}\{X\}$ σ^2 σ_X^2	The *variance* of X, denoted $\mathbf{Var}\{X\}$, σ_X^2, or simply σ^2, is equal to $\mathbb{E}\{(X - \mu_X)^2\}$, the expected squared difference between X and its mean, assuming $\mathbb{E}\{X^2\}$ is finite.
standard deviation	$\text{stdev}\{X\}$ σ σ_X	The *standard deviation* of X, denoted $\text{stdev}\{X\}$, σ_X, or simply σ, is the square root of $\mathbf{Var}\{X\}$.
support	$\text{supp}(X)$	The *support* of X is the set of values taken by X with positive probability, *i.e.*, $\text{supp}(X) = \{x \in \mathbb{R} : f(x) > 0\}$.
quantile function	$F_X^{-1}(p)$	The *quantile function* of X, denoted $F_X^{-1}(p)$, is the inverse function of $F_X(x)$ if F_X is strictly increasing and continuous. Otherwise it is defined as $F_X^{-1}(p) = \inf\{x \in \mathbb{R} : F_X(x) \geq p\}$.

Table 5.2: Basic distribution properties of a real-valued random variable X.

Notable Univariate Distributions

Here we define some notable real-valued univariate distributions. The discrete distributions are summarized in Table 5.3 and the continuous distributions are summarized in Table 5.4.

Discrete Uniform Distribution Let $S = \{x_1, \ldots, x_n\}$ be a finite set of n reals. We use Uniform(S) to denote the discrete uniform distribution on S. The p.m.f. is given by

$$f(k) = \begin{cases} 1/|S| & , \quad k \in S \\ 0 & \text{otherwise} . \end{cases}$$

One example of a discrete uniform random variable is the value of a fair k-sided die, which is distributed like Uniform($\{1, \ldots, k\}$).

Bernoulli Distribution We use Bernoulli(p) to denote the Bernoulli distribution with probability p. This discrete distribution models a biased coin flip. A Bernoulli(p) random variable takes value 1 with probability p and value 0 with probability $1 - p$. The p.m.f. can be written as a single function that is valid for the distribution's support, *i.e.*, the values 0 and 1:

$$f(x) = (1 - p) \left(\frac{p}{1 - p} \right)^x .$$

Binomial Distribution

Continuous Uniform Distribution We use U(a, b) to denote the continuous uniform distribution on the range $[a, b]$. The p.d.f. $f(x)$ is given by

$$f(x) = \begin{cases} \frac{1}{b-a} & , \quad x \in [a, b] \\ 0 & \text{otherwise} . \end{cases}$$

The most common continuous uniform distribution we use is U(0, 1). Because U(0, 1)-distributed random variables are so ubiquitous we sometimes simply

	Uniform (Discrete)	**Bernoulli**	**Binomial**
Notation	Uniform(S)	Bernoulli(p)	Bin(n, p)
Support	S	$\{0, 1\}$	$\{0, \dots, n\}$
$f(x)$ (p.m.f.)	$\frac{1}{\lvert S\rvert}$	$(1-p)\left(\frac{p}{1-p}\right)^x$	$\binom{n}{x}(p)^x(1-p)^{n-x}$

Table 5.3: Notable discrete univariate distributions. For the discrete uniform distribution the set S must be a finite subset of $\mathbb{R}$.

call them *uniform random variables* without specifying the parameters or the fact that it is continuous.

Normal Distribution We use $\mathcal{N}(\mu, \sigma^2)$ to denote the normal distribution (also known as the Gaussian distribution) with mean μ and variance σ^2. The p.d.f. is given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Beta Distribution We use beta(a, b) to denote the beta distribution with positive parameters a and b. The p.d.f. is given by

$$f(x) = \begin{cases} \frac{1}{\mathrm{B}(a,b)}\, x^{a-1}(1-x)^{b-1} & , \quad x \in [0,1] \\ 0 & \text{otherwise}, \end{cases}$$

where $\mathrm{B}(a, b) = \int_0^1 u^{a-1}(u-1)^{b-1}\, du$, known as the Euler beta function, acts as a normalizing constant. We discuss beta distributions and their multivariate generalizations (*i.e.*, Dirichlet distributions) in Section 5.2.

Domination and Coupling

For real-valued random variables X and Y, it is possible that all values taken by X are greater than or equal to all values taken by Y. In this case, we can say that $X \geq Y$ regardless of any correlation between X and Y. If this is not

	Uniform (Continuous)	**Normal (a.k.a. Gaussian)**	**Beta**
Notation	$\mathrm{U}(a, b)$	$\mathcal{N}(\mu, \sigma^2)$	$\mathrm{beta}(a, b)$
Support	$[a, b]$	$\mathbb{R}$	$[0, 1]$
$f(x)$ (p.d.f.)	$\frac{1}{b-a}$	$\frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}$	$\frac{x^{a-1}(1-x)^{b-1}}{\mathrm{B}(a,b)}$

Table 5.4: Notable continuous univariate distributions. The Euler beta function $\mathrm{B}(a, b)$ is a normalizing integral.

the case, it is still possible to compare X and Y with a meaningful relation. Stochastic domination is a common way to do this.

Definition 5.1 (**Stochastic Domination**)**.** For real-valued random variables X and Y, we say X (stochastically) dominates Y, denoted $X \geq^s Y$, iff

$$\mathbb{P}\{X \geq t\} \geq \mathbb{P}\{Y \geq t\} \quad \text{for all } t \in \mathbb{R}.$$

Coupling. Closely related to stochastic domination is the concept of *coupling*. Coupling is the introduction of an artificial dependency between two random variables for the purpose of analysis. Coupling two random variables X and Y often involves generating both simultaneously (or generating an ordered pair (X, Y)) in a way that preserves a desired relation. The marginal probabilities of X and Y must be distributed correctly, but subject to these constraints we are free to manipulate the conditional probabilities $X|Y$ and $Y|X$.

Proposition 5.1. *For real-valued random variables X and Y, we have $X \geq^s Y$ if and only if there exists a coupling of X and Y such that $X \geq Y$ deterministically.*

Proof. If the coupling exists then stochastic domination is obvious. If we have $X \geq^s Y$ then we can couple X and Y using the *probability integral transform* from a single uniform $[0, 1]$ random variable. Define the map ψ with

$$\psi : [0,1] \to \text{supp}(X) \times \text{supp}(Y)$$
$$\psi(u) = \left(\ F_X^{-1}(u) \ , \ F_Y^{-1}(u) \ \right) \ .$$

The relation $X \geq^s Y$ implies that $F_X^{-1}(u) \geq F_Y^{-1}(u)$ for all $u \in [0,1]$. If $U \sim \text{U}(0,1)$ then $\psi(U) = (X,Y)$ is a coupling with the desired properties. □

Example 5.1 (**Coupled Dice**)**.** In this example we wish to relate a 6-sided die and a 12-sided die; we represent the outcomes of the dice with random variables $D_6 \sim \text{Uniform}(\{1,\ldots,6\})$ and $D_{12} \sim \text{Uniform}(\{1,\ldots,12\})$ respectively. We can couple the dice so that $D_{12} \geq D_6$ deterministically. Define $X \sim \text{Uniform}(\{1,\ldots,6\})$ and $Y \sim \text{Bernoulli}(1/2)$. It is easy to verify that the coupling $(D_6, D_{12}) = (X, 2X - Y)$ generates the desired marginal distributions for both D_{12} and D_6, and the coupling ensures that $D_{12} \geq D_6$ deterministically.

Vector Domination

Marshall and Olkin (75) investigated inequalities involving the application of a convex function to components of a vector. The key lies in the comparison of the sum of the largest elements of each vector.

Definition 5.2 (**Vector Domination**)**.** For vectors $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, let $a_{(i)}$ and $b_{(i)}$ denote, respectively, the i^{th} smallest components of $\boldsymbol{a}$ and $\boldsymbol{b}$. We say $\boldsymbol{a}$ dominates $\boldsymbol{b}$, denoted $\boldsymbol{a} \succeq \boldsymbol{b}$, iff the following inequalities are satisfied:

$$\sum_{i=j}^{n} a_{(i)} \ \geq \ \sum_{i=j}^{n} b_{(i)} \quad \forall \ j \in \{1,\ldots,n\} \ .$$

Proposition 5.2 (Marshall and Olkin (75))**.** *For vectors* $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$

and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ *and a function* ψ*, consider the inequality*

$$\sum_{i=1}^{n} \psi(a_i) \;\geq\; \sum_{i=1}^{n} \psi(b_i) \;.$$

The inequality is satisfied for every *function* ψ *that is non-negative, non-decreasing, and non-concave* if and only if $\boldsymbol{a} \succeq \boldsymbol{b}$*.*

Vector domination is a binary relation. It is reflexive (*i.e.* $\boldsymbol{a} \succeq \boldsymbol{a}$) and transitive (*i.e.* $(\boldsymbol{a} \succeq \boldsymbol{b}) \wedge (\boldsymbol{b} \succeq \boldsymbol{c}) \implies (\boldsymbol{a} \succeq \boldsymbol{c})$). However, it is not a total relation — given two vectors one does not always dominate the other. For example, given the vectors $(0.5, 0.25, 0.25)$ and $(0.4, 0.4, 0.2)$, neither dominates the other.

In the next chapter we use vector domination for analysis involving random vectors. To this end, we now extend the definition to a stochastic setting.

Definition 5.3 (**Vector Domination in Expectation**)**.** For random vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, we say $\boldsymbol{A}$ dominates $\boldsymbol{B}$ in expectation, denoted $\boldsymbol{A} \succeq^{E} \boldsymbol{B}$, if and only if, for any function ψ that is non-negative, non-decreasing, and non-concave, we have

$$\mathbb{E}\left\{\sum_{i} \psi(A_i)\right\} \;\geq\; \mathbb{E}\left\{\sum_{i} \psi(B_i)\right\} \;.$$

This stochastic definition is of great use in the comparison of random trees in the next chapter.

Observation 5.1. For random vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ taking values in $\mathbb{R}^n$, if $\boldsymbol{A}$ and $\boldsymbol{B}$ can be coupled so that $\boldsymbol{A} \succeq \boldsymbol{B}$ deterministically, then $\boldsymbol{A} \succeq^{E} \boldsymbol{B}$. This is particularly useful in the following situations:

1. For $n = 2$, when $\max(A_1, A_2) \geq^{s} \max(B_1, B_2)$ and the sum of components is fixed for each vector with $A_1 + A_2 \geq B_1 + B_2$. This arises naturally when dealing with random split trees that are binary (see Model 6.2).

2. When $\boldsymbol{A}$ and $\boldsymbol{B}$ can be coupled so that, for each i, $A_i \geq B_i$ deterministically. For example, this is the case in the next section in which we compare discrete and continuous Dirichlet distributions (see Proposition 5.4).

3. When $\boldsymbol{A}$ and $\boldsymbol{B}$ can be coupled naturally so that $\boldsymbol{B}$ is obtained by splitting the components of A. This can arise when partial splits are introduced for the sake of comparison (see, *e.g.*, Section 8.2).

5.2 Betas and Dirichlets

In this section we discuss beta distributions in more depth and discuss Dirichlet distributions for the first time. We also introduce and analyze discrete analogues for both families of distributions (subject to a restriction on the parameters).

Beta Distributions

Standard Beta Distributions

Let $a > 0$ and $b > 0$ be fixed constants. Using $f(x; a, b)$ to denote the probability density function of beta(a, b), let us recall that

$$f(x; a, b) \;=\; \frac{x^{a-1}(1-x)^{b-1}}{\mathrm{B}(a,b)} \;=\; \frac{x^{a-1}(1-x)^{b-1}}{\int_0^1 t^{a-1}(1-t)^{b-1}\,dt}\,.$$

Since a and b are fixed, the denominator $\mathrm{B}(a, b)$ is constant. It is therefore easy to see that a beta$(1, 1)$ is distributed like U$(0, 1)$.

For any positive values of a and b, the density function is unimodal. If $a > 1$ and $b > 1$ then the density function is strictly concave with the maximum (*i.e.*, the distribution's mode) at $\frac{a-1}{(a-1)+(b-1)}$.

Betas and Uniforms When a and b are integers, beta(a, b) has a very natural relationship with uniform variables. Imagine we have a sequence of $a + b - 1$ random variables $\{U_i : i = 1 \ldots a + b - 1\}$ distributed i.i.d. like

$\mathrm{U}(0,1)$. We use $U_{(k)}$ to denote the k^{th} order statistic, *i.e.*, the variable in the sequence taking the k^{th} smallest value. It is well-known (see, *e.g.*, (26, p. 431)) that the a^{th} order statistic $U_{(a)}$ is distributed like $\mathrm{beta}(a,b)$. This is a generalization of the fact that $\mathrm{beta}(1,1) \sim \mathrm{U}(0,1)$. Perhaps the next simplest example is when $a = b = 2$. In this case, we have that the median of three uniforms is distributed like $\mathrm{beta}(2,2)$.

Observation 5.2. For integer $d \geq 1$, consider the median of d uniforms. To avoid ambiguity, if d is even we define the unique median as chosen u.a.r. from the two medians. With this random but unambiguous definition, the median is distributed like

$$\mathrm{beta}\left(\left\lfloor \frac{d+1}{2} \right\rfloor, \left\lfloor \frac{d+1}{2} \right\rfloor\right) .$$

Proof. For odd d, this is well-known, so we consider the case where $d = 2k$ for some integer k. In this case we define the median to be either $U_{(k)}$ or $U_{(k+1)}$, chosen with a fair coin flip. We now have the median distributed like $\mathrm{beta}(k, k+1)$ with probability $1/2$ and like $\mathrm{beta}(k+1, k)$ otherwise. It can be proven analytically that this distribution is equal to the $\mathrm{beta}(k,k)$ distribution, but we use a simple coupling argument that may be more intuitive.

We note that $U_{(k)}$ is distributed like U_I, where $I \sim \mathrm{Uniform}(\{1, \ldots, 2k\})$ depends on the ordering of $U_1, \ldots, U_{2k}$ but is independent of the actual values. We couple the coin flip such that $U_{(k)}$ is chosen as the median iff $I \geq k+1$, *i.e.*, iff U_{2k} is strictly greater than $U_{(k)}$. This happens with probability $1/2$ and is independent of the values of $U_{(1)}, \ldots, U_{(2k)}$ so the coupling is valid. With this coupling, the median we choose for $\{U_1, \ldots, U_{2k}\}$ is equal to the median of $\{U_1, \ldots, U_{2k-1}\}$. Thus our coupling shows that the median of $2k$ uniforms, breaking the tie with a fair coin flip, is distributed like $\mathrm{beta}(k,k)$. □

Let us now examine another distribution involving $2k-1$ uniforms. Our variables $\{U_1, \ldots, U_{2k-1}\} = \{U_{(1)}, \ldots, U_{(2k-1)}\}$ split the open interval $(0,1)$ into $2k$ open intervals which can be naturally indexed from 1 to $2k$. Each of these intervals is identically distributed. A consequence of this is that the sum of the first k intervals is distributed like the sum of the odd intervals. Each of these sums is distributed like $\mathrm{beta}(k,k)$.

Cumulative Distribution Functions of Beta Distributions The cumulative distribution function for a beta(a, b) distribution is given by the *regularized incomplete beta function*, denoted $I_x(a, b)$. We are particularly interested in distributions of the type beta(a, b) for integers a and b. In this case, for $X \sim \text{beta}(a, b)$ we have

$$F_X(x) \;=\; I_x(a,b) \;=\; \sum_{j=a}^{a+b-1} \binom{a+b-1}{j} x^j (1-x)^{a+b-1-j} \;. \qquad (5.1)$$

This looks like the c.d.f. of a binomial distribution for a reason. Recall that beta(a, b) is distributed like the a^{th} order statistic of $a + b - 1$ uniforms. This means that $\mathbb{P}\{X \le x\}$ is equal to the probability that at least a out of $a+b-1$ independent uniforms fall at or below x. The number falling at or below x is binomially distributed like $\text{Bin}(a + b - 1, x)$.

Discrete Beta Distributions

Our major motivation for the discussion of beta distributions is the analysis of random splits of finite point sets. A split of n points cannot take a continuous distribution for finite n; rather it is quantized with steps of size $1/n$. We define discrete variants of beta distributions that correspond exactly to certain random splits. For integers a, b the beta(a, b) distribution can be generated based on splitting the interval $(0, 1)$ by $a + b - 1$ cuts at uniform random points in the interval. Now imagine that the $(0, 1)$ interval is split into n chunks of size $1/n$.

For fixed positive integers a, b, and $n \ge a + b - 1$, we define the discrete beta distribution $\text{beta}_n(a, b)$ by giving a generation method. We first choose $a + b - 1$ integers u.a.r. without replacement from $\{1, \ldots, n\}$. Call the chosen integers, sorted in ascending order, $X_{(1)}, \ldots, X_{(a+b-1)}$; for homogeneity we also define $X_{(0)} = 0$ and $X_{(a+b)} = n + 1$. The integers $X_{(i)}$ partition the set $\{1, \ldots, n\} \setminus \{X_{(1)}, \ldots, X_{(a+b-1)}\}$ into $a+b$ (possibly empty) sets corresponding to intervals of the form $(X_{(i)}, X_{(i+1)})$. We define $\text{beta}_n(a, b)$ as the distribution of the number of integers in the first a sets, divided by n.

We now give several useful properties of discrete beta distributions.

Proposition 5.3. *For fixed positive integers a and b, we have the following:*

1. $\text{beta}_n(a,b) \;\sim\; \dfrac{n-(a+b-1)}{n} \;-\; \text{beta}_n(b,a)$.
2. $\text{beta}_n(a,b) \;\leq^s\; \text{beta}(a,b)$.
3. $\text{beta}_n(a,b) \;\xrightarrow{\mathcal{L}}\; \text{beta}(a,b)$ *as* $n \to \infty$.
4. $X \sim \text{beta}_n(a,b)$ *and* $Y \sim \text{beta}(a,b)$ *can be coupled so that, deterministically, we have* $\Big(X\ ,\ \frac{n-(a+b-1)}{n} - X\Big) \preceq (Y\ ,\ 1-Y)$ *and, even stronger,* $X \leq Y$ *and* $\frac{n-(a+b-1)}{n} - X \;\leq\; 1-Y$ *simultaneously.*
5. $\text{beta}_n(a,b) \;\sim\; \dfrac{1}{n}\,\text{Bin}\Big(\, n-(a+b-1)\ ,\ \text{beta}(a,b)\,\Big)$.

Proof. Property (1) follows from the distribution's definition. Property (5) comes from an argument about the ranks of i.i.d. uniform random variables $U_1, \ldots, U_n$. Let I be the index such that U_I has rank a among $\{U_1, \ldots, U_{a+b-1}\}$. U_I is distributed as $\text{beta}(a,b)$ and the number of the variables $U_{a+b}, \ldots, U_n$ falling in the range $(0, U_I)$ is distributed like $\text{Bin}\big(\, n-(a+b-1)\ ,\ U_I\,\big)$. By its definition, $\text{beta}_n(a,b)$ is distributed identically since the ranks of $\{U_1, \ldots, U_{a+b-1}\}$ among $\{U_1, \ldots, U_n\}$ are distributed like $a+b-1$ integers sampled u.a.r. without replacement from $\{1, \ldots, n\}$.

We prove properties (2), (3), and (4) with a coupling argument. A random variable $X \sim \text{beta}_n(a,b)$ is most naturally generated with a set of $a+b-1$ indices sampled u.a.r. without replacement from $\{1, \ldots, n\}$. In terms of analysis, however, we prefer to sample *with* replacement so we can deal with independent variables. Fortunately we can generate X using $a+b-1$ integers Z_i, $i \in \{1, \ldots, a+b-1\}$, chosen i.i.d. and u.a.r. (with replacement) from $\{0, \ldots, n-(a+b-1)\}$. The i^{th} order statistic, denoted $Z_{(i)}$, corresponds to the number of the $n-(a+b-1)$ unchosen indices that fall in the first i intervals. Thus the Z_i's give us a corresponding set $\big\{Z_{(i)} + i : i \in \{1, \ldots, a+b-1\}\big\}$ of unique indices from $\{1, \ldots, n\}$ distributed as if we sampled u.a.r. without replacement.

We can express $\mathbb{P}\{X \leq x\}$ as the probability that a or more of the Z_i's fall at or below xn. We generate $Y \sim \text{beta}(a,b)$ from i.i.d. $[0,1]$ uniforms

$U_1, \ldots, U_{a+b-1}$ in the usual way so that $\mathbb{P}\{Y \le k\}$ is the probability that a or more of the U_i's fall at or below k/n.

When generating X and Y from the desired distributions, we couple the Z_i's with the U_i's so that $Z_i = \lfloor (n-(a+b-1)+1)U_i \rfloor$. For every i, this gives us

$$\begin{aligned} 0 \;\le\; U_i - \frac{Z_i}{n} \;&=\; U_i - \frac{\lfloor (n-(a+b-1)+1)U_i \rfloor}{n} \\ &\le\; \frac{1 + nU_i - (n-(a+b-1)+1)U_i}{n} \\ &=\; \frac{1 + (a+b-2)U_i}{n} \\ &\le\; \frac{a+b-1}{n} \;. \end{aligned}$$

There is some i such that $X = Z_{(a)}/n = Z_i/n$ and $Y = U_{(a)} = U_i$. Our coupling therefore gives us $0 \le Y - X \le (a+b-1)/n$ which implies property (4). It also implies, for any $x \in \mathbb{R}$,

$$\mathbb{P}\{Y \le x\} \;\le\; \mathbb{P}\{X \le x\} \;\le\; \mathbb{P}\left\{ Y \le x + \frac{a+b-1}{n} \right\} \;.$$

This proves property (2), which is equivalent to the left inequality. To see that the upper and lower bounds converge in the limit as $n \to \infty$, we note that, for fixed a and b, the density function of Y is bounded by some constant $c = c(a,b)$. This gives us

$$\begin{aligned} \mathbb{P}\{Y \le x\} - \lim_{n\to\infty} \mathbb{P}\{X \le x\} \;&\le\; \lim_{n\to\infty} \mathbb{P}\left\{ x \le Y \le x + \frac{a+b-1}{n} \right\} \\ &\le\; \lim_{n\to\infty} \frac{c(a+b-1)}{n} \\ &=\; 0 \;, \end{aligned}$$

completing the proof of property (3). □

We now resume our discussion of the $a+b$ sets created by our generation method for the $\text{beta}_n(a,b)$ distribution.

Observation 5.3. For positive integers $n \geq d$, let $X_{(1)}, \ldots, X_{(d)}$ be d indices chosen u.a.r. without replacement from $\{1, \ldots, n\}$ and sorted in ascending order. Let $I_0, \ldots, I_d$ be the intervals of integers with $I_0 = (0, X_{(1)})$, $I_d = (X_{(d)}, n+1)$, and $I_i = (X_{(i)}, X_{(i+1)})$ for $1 \leq i \leq d-1$. Consider a set of intervals indexed by a subset A of the indices $\{0, \ldots, d\}$. Of the $n-d$ unchosen integers, the number falling in the intervals indexed by A is distributed like

$$\sum_{i \in A} |I_i| \;\sim\; n \cdot \mathrm{beta}_n\big(|A|, d+1-|A|\big) \;\; .$$

Alternating intervals are of particular interest to us in subsequent chapters, particularly in the proof of Lemma 7.2.

Observation 5.4. Let A contain the intervals of a random parity, *i.e.*, A is the set of odd indices with probability $1/2$ and is the set of even indices otherwise. In this case, Observation 5.2, along with the connection with binomial distributions given by Proposition 5.3, implies that

$$\sum_{i \in A} \big|I_i\big| \;\sim\; \mathrm{Bin}\Big(n-d \;,\; \mathrm{beta}\big(\lceil d/2 \rceil \,, \lceil d/2 \rceil \big) \Big) \;\; .$$

Dirichlet Distributions

Before jumping into the Dirichlet family of distributions we define two closely related distributions.

Categorical Distribution Let $\boldsymbol{p} = (p_1, \ldots, p_K)$ be a probability vector with $K \geq 2$ components. We use Categorical($\boldsymbol{p}$) to denote the categorical distribution on the integers $\{1, \ldots, K\}$. This discrete univariate distribution models the outcome of a biased K-sided die, generalizing Bernoulli distributions and discrete uniform distributions. The p.m.f. is given by

$$f(x) = \begin{cases} p_x & , \quad x \in \{1, \ldots, K\} \\ 0 & \quad \text{otherwise} \;\; . \end{cases}$$

Multinomial Distribution Let $\boldsymbol{p} = (p_1, \ldots, p_K)$ be a probability vector with $K \geq 2$ components. We use $\text{Mult}(n, \boldsymbol{p})$ to denote the multinomial distribution with n trials on the probability vector $\boldsymbol{p}$. This discrete distribution on $\{0, \ldots, n\}^K$ is a multivariate generalization of the binomial distribution. Let $X_1, \ldots, X_n$ be i.i.d. Categorical($\boldsymbol{p}$) trials and, for $1 \leq i \leq K$, let Z_i be the frequency of outcome i in the sample, *i.e.*, $Z_i = \big|\{j : X_j = i\}\big|$. Then the joint distribution of the frequencies is multinomial with $(Z_1, Z_2, \ldots, Z_K) \sim \text{Mult}(n, \boldsymbol{p})$. For $\boldsymbol{x} = (x_1, \ldots, x_K) \in \{0, \ldots, n\}^K$, the p.m.f. is given by

$$f(\boldsymbol{x}) = \begin{cases} n! \prod_{i=1}^K (p_i^{x_i}/x_i!) & , \quad \sum_{i=1}^K x_i = n \\ 0 & \text{otherwise} . \end{cases}$$

Dirichlet Distribution Let $\boldsymbol{a} = (a_1, \ldots, a_K)$ be a vector of $K \geq 2$ positive real parameters. We use $\text{Dir}(\boldsymbol{a})$ to denote the Dirichlet distribution with parameter vector $\boldsymbol{a}$. The Dirichlet family of distributions are a multivariate generalization of the beta family of distributions. The distribution is over a subset of $\mathbb{R}^K$, specifically the standard $K-1$ simplex

$$\triangle^{K-1} = \left\{(x_1, \ldots, x_K) \in \mathbb{R}^K \ : \ \textstyle\sum_{i=1}^K x_i = 1, \ x_i \geq 0 \ \forall \ i\right\} .$$

The last component x_K is redundant since $x_K = 1 - \sum_{i=1}^{K-1} x_i$. Though it is sometimes omitted from the definition, we choose to include it for notational convenience and homogeneity. For $\boldsymbol{x} = (x_1, \ldots, x_K) \in \mathbb{R}^K$, the p.m.f. is given by

$$f(\boldsymbol{x}) = \begin{cases} \frac{1}{\text{B}(\boldsymbol{a})} \prod_{i=1}^K x_i^{a_i - 1} & , \quad \boldsymbol{x} \in \triangle^{K-1} \\ 0 & \text{otherwise} , \end{cases}$$

where $\text{B}(\boldsymbol{a})$, the multivariate beta function, acts as a normalizing constant.

If $\boldsymbol{a}$ is a vector of positive integers, the connection between betas and uniforms and the definition of discrete beta distributions generalize naturally to the Dirichlet family. Let $\alpha_i = \sum_{j=1}^i a_j$ and consider i.i.d. uniforms $\{U_i :$

$1 \le i \le \alpha_K - 1\}$. Using $U_{[i]}$ to denote $U_{(\alpha_i)}$,[1] we have

$$\big(U_{[1]}\,,\, U_{[2]} - U_{[1]}\,,\, \ldots\,,\, U_{[K]} - U_{[K-1]}\big) \;\sim\; \mathrm{Dir}(\boldsymbol{a})\ .$$

In other words, the uniforms U_i cut the interval $[0,1]$ into α_K intervals; the first component is distributed as the sum of the first a_1 intervals, the second component is distributed as is the sum of the next a_2 intervals, and so on. For the discrete version of the distribution, which we denote $\mathrm{Dir}_n(\boldsymbol{a})$, consider discrete random variables $\{Z_i : 1 \le i \le \alpha_K - 1\}$ drawn u.a.r. without replacement from $\{1,\ldots,n\}$. Using $Z_{[i]}$ to denote $Z_{(\alpha_i)}$, we have

$$\frac{1}{n}\Big(Z_{[1]} - a_1\ ,\ Z_{[2]} - Z_{[1]} - a_2\ ,\ \ \ldots\ \ ,\ Z_{[K]} - Z_{[K-1]} - a_K\Big) \;\sim\; \mathrm{Dir}_n(\boldsymbol{a})\ .$$

Proposition 5.4. *For a vector $\boldsymbol{a}$ of fixed positive integers, random vectors $\boldsymbol{X} \sim \mathrm{Dir}_n(\boldsymbol{a})$ and $\boldsymbol{Y} \sim \mathrm{Dir}(\boldsymbol{a})$ satisfy the following:*

1. *The first components are distributed like $X_1 \sim \mathrm{beta}_n(a_1, \alpha_K - a_1)$ and $Y_1 \sim \mathrm{beta}(a_1, \alpha_K - a_1)$. If $K > 2$ the vectors $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ containing the remaining $K-1$ components are distributed like $\boldsymbol{X}' \sim \frac{n - nX_1 - a_1}{n}\mathrm{Dir}_{n-nX_1-a_1}((a_2,\ldots,a_K))$ and $\boldsymbol{Y}' \sim (1-Y_1)\mathrm{Dir}((a_2,\ldots,a_K))$.*

2. *There exists a coupling of $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that, deterministically, $\boldsymbol{X} \preceq \boldsymbol{Y}$ and, even stronger, $X_i \le Y_i$ for every i simultaneously.*

3. $\mathrm{Dir}_n(\boldsymbol{a}) \xrightarrow{\mathcal{L}} \mathrm{Dir}(\boldsymbol{a})$ *as* $n \to \infty$.

4. $\mathrm{Dir}_n(\boldsymbol{a}) \;\sim\; \dfrac{1}{n}\,\mathrm{Mult}\Big(n - (\alpha_K - 1)\ ,\ \mathrm{Dir}(\boldsymbol{a})\Big)$.

Proof. By induction on K, property (1) follows from definitions and property (4) follows from (1) due to the specific binomial distribution of a multinomial's first component. The proof of the other properties is an extension of the proof of Proposition 5.3. This proof requires slightly more careful coupling.

For $1 \le i \le \alpha_K - 1$ we generate coupled random variables $U_{(i)}$ and $Z_{(i)}$, where both sequences are non-decreasing in i. We again use $U_{[i]}$ and $Z_{[i]}$ to

[1] We use this notation largely to avoid the use of subsubscripts.

denote $U_{(\alpha_i)}$ and $Z_{(\alpha_i)}$ respectively. We use dummy variables so the sequences start with $U_{(0)} = Z_{(0)} = 0$ and end with $U_{[K]} = n$ and $Z_{[K]} = n - \alpha_K + 1$. For indices $i = 1, \ldots, \alpha_K - 1$, we use independently generated β_i variables to define $U_{(i)}$ and $Z_{(i)}$. We have

$$\begin{aligned} \beta_i &\sim \text{beta}(1, \alpha_K - i) \\ (U_{(i)}|U_{(i-1)}, \beta_i) &= U_{(i-1)} + \beta_i \cdot (n - U_{(i-1)}) \\ (Z_{(i)}|Z_{(i-1)}, \beta_i) &= Z_{(i-1)} + \left\lfloor \beta_i \cdot (n - \alpha_K + 2 - Z_{(i-1)}) \right\rfloor \end{aligned}$$

We can make the following observations for $1 \leq i \leq \alpha_K - 1$:

$$U_{(i)} \sim n \cdot \text{beta}(i, \alpha_K - i) \tag{5.2}$$

$$Z_{(i)} \sim n \cdot \text{beta}_n(i, \alpha_K - i) \tag{5.3}$$

$$U_{(i)} \leq Z_{(i)} + \alpha_K - 1 \leq U_{(i)} + \alpha_K - 1 \tag{5.4}$$

Here (5.2) and (5.3) follow from inductive definitions of continuous and discrete beta distributions. We postpone proof of (5.4) until the end of the larger proof.

The coupling idea is that we use the $Z_{(i)}$'s to define the discrete Dirichlet and the $U_{(i)}$'s to define the continuous Dirichlet. The discrete Dirichlet $\boldsymbol{X}$ has components $X_j = \frac{1}{n}(Z_{[j]} - Z_{[j-1]})$ and the continuous Dirichlet $\boldsymbol{Y}$ has components $Y_j = \frac{1}{n}(U_{[j]} - U_{[j-1]})$.

We wish to show that $X_j \leq Y_j$ for all $1 \leq j \leq K$ simultaneously. It follows from (5.4) that $Z_{(i)} - Z_{(i-1)} \leq U_{(i)} - U_{(i-1)}$ for $i < \alpha_K$. This tells us that $X_j \leq Y_j$ for all $j < K$. For the special case of $j = K$ we have $X_K = \frac{1}{n}(n - \alpha_K + 1 - Z_{[K-1]})$ and $Y_K = \frac{1}{n}(n - U_{[K-1]})$, so $X_K \leq Y_K$ also follows from (5.4). Thus our coupling satisfies property (2).

We show convergence in law just as we did for beta distributions. We have $X_j \leq Y_j \leq X_j + \frac{\alpha_K - 1}{n}$ for any component. Recall that $\text{Dir}(\boldsymbol{a})$ is a distribution on $\mathbb{R}^{K-1}$. For any continuity set A, define the fat boundary $A^* = \{\boldsymbol{x} \in \mathbb{R}^{K-1} : ||\boldsymbol{x} - \boldsymbol{y}||_\infty \leq \frac{\alpha_K - 1}{n}, \boldsymbol{y} \in \partial A\}$ containing all points that are within $\frac{\alpha_K - 1}{n}$ of ∂A

in every component. Unless $\boldsymbol{Y} \in A^*$, $\boldsymbol{X}$ and $\boldsymbol{Y}$ must either both be in A or both be outside of A. Finally we note that the density function for $\mathrm{Dir}(\boldsymbol{a})$ is bounded by a constant $c = c(\boldsymbol{a})$. We have

$$\begin{aligned}\mathbb{P}\{\boldsymbol{Y} \in A\} - \lim_{n\to\infty} \mathbb{P}\{\boldsymbol{X} \in A\} &\leq \lim_{n\to\infty} \mathbb{P}\{\boldsymbol{Y} \in A^*\} \\ &\leq \mu(\partial A) + \lim_{n\to\infty} c \cdot \left(\frac{\alpha_K - 1}{n}\right)^{K-1} \\ &= 0\ ,\end{aligned}$$

completing the proof of property (3).

We now prove (5.4) as promised. It follows from the stronger inequality

$$Z_{(i)} \ \leq\ U_{(i)} \ \leq\ \left(\frac{n}{n-\alpha_K+2}\right)\left(Z_{(i)}+i\right)\ . \tag{5.5}$$

We use induction to prove (5.5), with the base case of $i = 1$ satisfied since $Z_{(1)} = \lfloor U_{(1)} \rfloor$. For general $i \geq 2$, assume (5.5) holds for $i-1$. The left inequality $Z_{(i)} \leq U_{(i)}$ follows from the recursive definitions of $Z_{(i)}$ and $U_{(i)}$ since $Z_{(i-1)} \leq U_{(i-1)}$ and

$$n - U_{(i-1)} \ \geq\ n - (Z_{(i-1)} + i - 1) \ \geq\ n - \alpha_K + 2 - Z_{(i-1)}\ .$$

For the right inequality our induction hypothesis gives us

$$U_{(i)} \ =\ (U_{(i-1)})(1-\beta_i)+\beta_i \cdot n \ \leq\ \left(\frac{n}{n-\alpha_K+2}\right)\left(Z_{(i-1)}+i-1\right)(1-\beta_i)+\beta_i \cdot n\ .$$

From here the right inequality of (5.5) follows by rearranging terms:

$$\begin{aligned} U_{(i)} &\leq \left(\frac{n}{n-\alpha_K+2}\right)\Big((Z_{(i-1)}+i-1)(1-\beta_i)+\beta_i\cdot(n-\alpha_K+2)\Big) \\ &\leq \left(\frac{n}{n-\alpha_K+2}\right)\Big(Z_{(i-1)}+i-1+\beta_i\cdot(n-\alpha_K+2-Z_{(i-1)})\Big) \\ &\leq \left(\frac{n}{n-\alpha_K+2}\right)\Big(i+Z_{(i-1)}+\big\lfloor\beta_i\cdot(n-\alpha_K+2-Z_{(i-1)})\big\rfloor\Big) \\ &= \left(\frac{n}{n-\alpha_K+2}\right)(Z_{(i)}+i)\ . \end{aligned}$$

□

5.3 Notes

Contributions

The majority of material in this chapter is rudimentary and/or well-known. Propositions 5.3 and 5.4 were proved independently but are not novel (see, *e.g.*, Devroye (34, Lemma 2)). We are not aware of the discrete beta and Dirichlet distributions existing in the literature under these names — where we have seen them they are defined in terms of uniform random variables.

Chapter 6

Random Split Trees

In this chapter we discuss models of random tree data structures. The central model of interest is the well-studied *random split tree* model. We start with a discussion of the highly restricted *random binary search tree* model. We then generalize this to the random split tree model, then generalize further to a bounded model that lets us bound the structural distribution of a random tree with that of a random split tree.

Contents

6.1 Random Binary Search Trees

We begin the discussion of split trees with a simple example, the *random binary search tree*.

Model 6.1 (**Random Binary Search Tree**)**.** Given a fully ordered set S of n distinct elements, consider the set $\mathcal{T}$ of all possible binary search trees storing S. Only the *rank* of an element in S is relevant to the structure of a BST so we assume w.l.o.g. that $S = \{1, \ldots, n\}$. A *random binary search tree* is simply a tree drawn from a particular probability distribution on $\mathcal{T}$. Rather than specifying the distribution explicity we describe the generation process. Two of the most natural descriptions are as follows:

1. Let $X_1, \ldots, X_n$ be a random permutation of the integers $1, \ldots, n$ distributed uniformly over the $n!$ members of the symmetric group S_n. Starting with an empty binary search tree, build a tree by inserting the elements in the order given by $X_1, \ldots, X_n$.

2. Choose an element u.a.r. to be the root element, *i.e.*, let I be the rank of the root element where I is uniform on the indices $1, \ldots, n$. Let S_1 (respectively, S_2) be the subset of S containing the elements with index less than (respectively, greater than) I. Recursively, construct the subtrees of I from the sets S_1 and S_2.

The two descriptions of the generation process are equivalent. The first is perhaps more useful for motivating the model, since on-line construction of a binary search tree is very natural and, in practice, data oftens appears in random order. The second definition provides a natural segue into our discussion of a more general model.

These random binary search trees are well studied. In analyzing the structure of such a tree storing n elements we are interested in the height of the tree and the average node depth; these are random variables which we denote H_n and D_n respectively. Interest in these variables is easy to motivate — H_n

and D_n correspond respectively to the worst-case and average-case query time in the tree.

It turns out that both H_n and D_n have expected values that grow logarithmically with n. The height and average depth in a binary search tree are minimized when the tree is perfectly balanced, and in this case both values are equal to $\log_2 n$ modulo a lower order term. Therefore, assuming the following limits exist, we have

$$\lim_{n\to\infty} \frac{\mathbb{E}\{H_n\}}{\ln n} \;\geq\; \lim_{n\to\infty} \frac{\mathbb{E}\{D_n\}}{\ln n} \;\geq\; \frac{\log_2 n}{\ln n} \;\approx\; 1.44270\ldots .$$

Depth Analysis

For the average depth, analysis has been via the depth of the last element inserted into a tree storing n elements; we denote this D_n^*. The following was proved first by Mahmoud and Pittel (74).

Theorem 6.1.

$$\lim_{n\to\infty} \frac{\mathbb{E}\{D_n^*\}}{\ln n} \;=\; \lim_{n\to\infty} \frac{\mathbf{Var}\{D_n^*\}}{\ln n} \;=\; 2\ .$$

Devroye (29) gave a more elegant proof of this using the theory of records in random sequences. His proof is based on the following lemma which defines D_n^* as a sum of independent Bernoulli random variables.

Lemma 6.1. *Let $B_2, B_3, \ldots, B_n$ be independent random variables with B_i distributed as* Bernoulli$(2/i)$. *Then we have*

$$D_n^* \;\stackrel{\text{def}}{=}\; \sum_{i=2}^{n} B_i\ .$$

Proof. The binary search tree is constructed by starting with an empty tree, then inserting the elements $1,\ldots,n$ in the order determined by a random permutation $X_1,\ldots,X_n$. Thus D_n^* is the depth of element X_n.

To consider the depth of X_n it is useful for us to consider the tree as constructed by a different but equivalent process. We build the tree by inserting

the elements in the order given by $X_n, X_1, X_2, \ldots, X_{n-1}$, after each insertion performing a single rotation (if necessary) to ensure that X_n is a leaf. After inserting X_i, a rotation is necessary if and only if X_i is a child of X_n; this happens if and only if the ranks of X_n and X_i in the set $\{X_n, X_1, X_2, \ldots, X_i\}$ differ by 1. Since each rotation increases the depth of X_n by 1, D_n^* is simply the number of rotations performed.

Let R_i be the rank of X_n in the set $\{X_n, X_1, X_2, \ldots, X_{i-1}\}$ and let R_i' be the rank of X_i in the set $\{X_n, X_1, X_2, \ldots, X_i\}$. A rotation is performed if and only if $R_i' \in \{R_i, R_i + 1\}$. As a property of uniformly random permutations, we have that R_i' is uniform on $\{1, \ldots, i+1\}$ and all R_i' are independent.

With $\{B_i\}$ defined as in the statement of the lemma, we can now define D_n^* as a sum of indicator variables.

$$D_n^* \stackrel{\text{def}}{=} \sum_{i=1}^{n-1} \mathbb{1}_{\{\text{inserting } X_i \text{ forces a rotation}\}} \stackrel{\text{def}}{=} \sum_{i=1}^{n-1} \mathbb{1}_{\{R_i' \in \{R_i, R_i+1\}\}} \stackrel{\text{def}}{=} \sum_{i=2}^{n} B_i \ .$$

□

Height Analysis

For the height, Robson (90) proved that the limit of $\mathbb{E}\{H_n\}/\ln n$ does exist and gave an upper bound. Devroye (27) proved a matching lower bound. Let $\eta(c) = 1 - c \ln \frac{2e}{c}$. There are two solutions to $\eta(c) = 0$; one solution is less than 2 and one is greater than 2. Let $\alpha = 4.31107\ldots$ be the unique solution greater than 2 to $\eta(c) = 0$. With convergence in probability we have

$$\lim_{n\to\infty} \frac{H_n}{\ln n} \ = \ \alpha \ .$$

We prove the upper bound for the limit, also obtaining exponential upper tail bounds.

Theorem 6.2. *Let $c > \alpha$. Then $\eta(c) > 0$ and*

$$\mathbb{P}\{H_n \geq c \ln n\} \leq n^{-\eta(c)}.$$

Proof. Let $t \geq 0$. By a union bound, $\mathbb{P}\{H_n \geq t\} \leq n\mathbb{P}\{D_n^* \geq t\}$, where D_n^* is the depth of the last node inserted in a random binary search tree on n nodes. By Lemma 6.1, D_n^* is distributed as $\sum_{i=2}^{n} B_i$, where B_i is Bernoulli$(2/i)$ and the B_i are independent. Thus, for $s > 0$, by the Chernoff bound,

$$\begin{aligned}
\mathbb{P}\{D_n^* \geq t\} &\leq e^{-st} \prod_{i=2}^{n} \mathbb{E}\{e^{sB_i}\} \\
&= e^{-st} \prod_{i=2}^{n} (1 + 2(e^s - 1)/i) \\
&\leq e^{-st} \prod_{i=2}^{n} \exp(2(e^s - 1)/i) \\
&\leq \exp(2(e^s - 1)\ln n - st)
\end{aligned}$$

and upon taking s such that $2e^s \ln n = t$, we get the bound

$$\mathbb{P}\{H_n \geq t\} \leq n \cdot \exp\left(t - 2\ln n - t\ln\left(\frac{t}{2\ln n}\right)\right) .$$

Letting $t = c\ln n$ for constant c, the upper bound becomes

$$\exp\left(\ln n\left(c - c\ln\left(\frac{c}{2}\right) - 1\right)\right) = n^{c - c\ln\left(\frac{c}{2}\right) - 1} = n^{-\eta(c)}.$$

To see that $\eta(c)$ is positive and increasing for $c > \alpha$, we note that $\eta(\alpha) = 0$ and observe that $\eta'(c) = \ln\frac{c}{2}$, which is strictly positive for $c > 2$. □

Analysis of H_n was subsequently tightened by Reed (89), who determined the second order term of the expectation and proved constant variance:

Theorem 6.3 (Reed (89))**.** *Let $\alpha \approx 4.31107\ldots$ be the unique solution greater than 2 of the equation $\alpha \ln\frac{2e}{\alpha} = 1$. Let $\beta = \frac{3}{2\ln(\alpha/2)} \approx 1.95303$. Then*

$$\mathbb{E}\{H_n\} = \alpha\ln n - \beta\ln\ln n + \mathcal{O}(1)$$

and $\mathbf{Var}\{H_n\} = \mathcal{O}(1)$.

Coupling With Random k-d Trees

Random binary search trees are useful randomized data structures for storing one dimensional keys which, in geometric terms, we can consider to be points in $\mathbb{R}$. But how can they be generalized in a natural way to store a set S of n points in $\mathbb{R}^d$ for $d > 1$?

Perhaps the most obvious approach is to ignore all dimensions except the first. This projects our point set onto a line, after which we can build a binary search tree based on the projected point set[1]. More generally we can consider any line ℓ such that the projection of S onto ℓ contains n unique points. If we build a random BST based on the projected points, the height and average depth are distributed as before. What we have done is partition $\mathbb{R}^d$ using hyperplanes perpendicular to ℓ.

Another approach is that taken by k-d trees[2], first described by Bentley (13). A BST splits at each node based on a single dimension; a k-d tree does the same but considers different dimensions at different levels of the tree. In $\mathbb{R}^2$ a k-d tree alternates between using vertical and horizontal lines to partition the space (see, *e.g.*, Figure 6.1). More generally, at a node v in d dimensions, a k-d tree splits based on dimension $1 + (\text{depth}(v) \bmod d)$, using a hyperplane perpendicular to that axis.

One advantage of k-d trees over binary search trees is the efficiency of orthogonal range searching. An orthogonal range query can be performed in $\mathcal{O}\big(n^{(d-1)/d} + k\big)$ time with a balanced k-d tree whereas a linear scan may be necessary if the data set is stored in a lower dimensional BST. The improvement comes from the fact that any hyperplane (or bounding facet of an orthogonal range polytope) can intersect only $\mathcal{O}\big(n^{(d-1)/d}\big)$ of the cuts made by a k-d tree. See, *e.g.*, de Berg *et al.* (23) for details. The expected cost of an orthogonal range query for a k-d tree storing points uniformly distributed in a hypercube was analyzed by Chanzy *et al.* (19); this expected cost is worse than the cost for a perfectly balanced k-d tree.

[1] For simplicity we assume all points have unique coordinates in the first dimension.

[2] Though 'k-d tree' is short for 'k-dimensional tree', for the sake of continuity we avoid using k to denote the dimension.

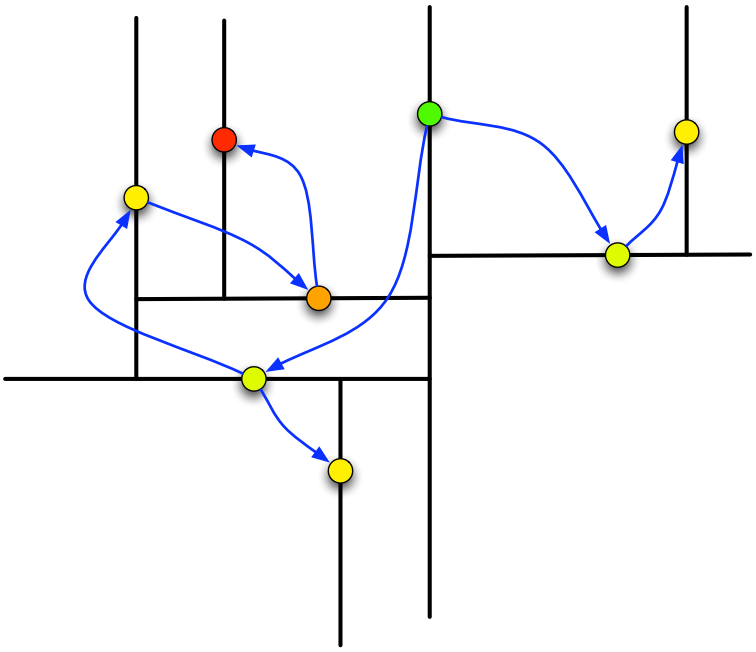


Figure 6.1: A k-d tree storing 8 points in $\mathbb{R}^2$. Tree edges are indicated by blue arrows directed from parent to child.

Now we use a simple coupling argument (see, *e.g.*, Lindvall (69)) to show that the structure of a random k-d tree is distributed identically to that of a random BST. By 'structure' we mean the underlying rooted acyclic graph obtained by ignoring any differentiation between left and right children. The definition of a random k-d tree is analogous to that of a random BST; such a tree is generated by inserting the elements according to a random permutation.

Proposition 6.1. *A random BST and a random k-d tree, each storing n elements, have identically distributed structure.*

Proof Sketch. The argument uses recursive coupling to generate a random BST and a random k-d tree that have identical structure. The two random trees are completely dependent, but each is drawn from the correct distribution. The key observation is that the subtree sizes of the two roots are identically distributed. The rank of the root (in a k-d tree this is w.r.t. the splitting dimension) is uniformly distributed on $\{1, \ldots, n\}$ for each tree. We simply

couple these random variables so that the roots split their data sets in the same proportions. The argument continues recursively in non-empty subtrees. □

Random b-ary Search Trees

It is natural to generalize the random binary search tree model to trees with branching factor $b \geq 2$. In a random b-ary search tree storing $n \geq k-1$ elements, the root stores $b-1$ elements sampled u.a.r. without replacement. The remaining $n-(b-1)$ elements are sent to the subtrees of the root; an element is sent to the i^{th} subtree if it is greater than exactly $i-1$ of the elements in the root. A leaf node stores between 0 and $b-2$ elements.

For such a tree we are most interested in the height and average element depth, which we denote $H_{n,b}$ and $D_{n,b}$ respectively. As in the binary case, $\mathbb{E}\{H_{n,b}\}$ and $\mathbb{E}\{D_{n,b}\}$ are both logarithmic in n. For fixed b, let c_b be the positive constant such that $1/c_b = \sum_{i=2}^{b} 1/i$. The following theorems generalize limits for random BSTs.

Theorem 6.4 (Mahmoud and Pittel (74))**.**

$$\lim_{n\to\infty} \frac{\mathbb{E}\{D_{n,b}\}}{\ln n} \;=\; c_b \;.$$

Theorem 6.5 (Devroye (34))**.**

$$\lim_{n\to\infty} \frac{\mathbb{E}\{H_{n,b}\}}{\ln n} \;=\; \gamma(b) \;,$$

where $\gamma(b)$ *is defined as*

$$\gamma(b) \;=\; \inf\left\{ c > c_b \;:\; t + c\ln(b!) - c\sum_{i=1}^{b-1} \ln(t+i) \;\leq\; 0 \right\} \;,$$

and $t = t(c) > 0$ *is the unique solution of the equation* $\dfrac{1}{c} = \displaystyle\sum_{i=1}^{b-1} \frac{1}{t+i}$.

6.2 A General Model

The random BST model, while easily motivated and well understood, is very specific. Devroye (31) proposed and analyzed the far more general *random split tree* model.

Model 6.2 (**Random Split Tree**)**.** A *random split tree* is a random tree storing a data set of n elements. The distribution is determined by the following parameters which must be specified:

1. $\mathcal{V}$ – the random split vector prototype.
2. b – the branching factor.
3. s_0 – the number of elements in each internal node.
4. s_1 – the minimum number of elements in each leaf node.
5. s – the maximum number of elements in each leaf node.

The prototype split vector $\mathcal{V} = (V_1, V_2, \ldots, V_b)$ is the most interesting parameter (noting that it also specifies b). Each tree node is given a *split vector* that is drawn from the distribution specified by $\mathcal{V}$. We emphasize that all split vectors in the tree are identically distributed. When generating a random split tree, a node's split vector is generated when the node is created. Devroye considers an equivalent definition that uses an infinite skeleton tree (31, p. 409).

Elements are inserted into the split tree one at a time. Insertion starts at the root, recursively proceeding downward until a leaf node is reached. Each recursive step chooses a subtree randomly according to the current node's split vector – the i^{th} subtree is chosen with probability V_i. Thus a node's split vector defines a probability distribution on its subtrees and we must have $\sum_i V_i = 1$ and $V_i \geq 0$.

When an insertion path reaches a leaf, if the leaf is under capacity (*i.e.* contains fewer than s elements) the element lives in the leaf. If the leaf is already at capacity it splits, becoming an internal node and spawning b children. Of the $s + 1$ elements, s_0 (chosen u.a.r.) stay in the node, s_1 (chosen u.a.r.) are sent to each child, and each of the $s + 1 - s_0 - bs_1$

remaining elements is sent to a random child according to the split vector ($s + 1 - s_0 - bs_1$ must therefore be nonnegative).

Let T be a random split tree on n elements defined by parameters $\mathcal{V}, b, s_0, s_1, s$. If $n \leq s_0$, T consists of a single root node. Otherwise, let Λ_i denote the number of elements in the i^{th} subtree of T's root and let $\Lambda(T)$ denote the vector

$$\Lambda(T) \;=\; \big(\Lambda_1 - s_1, \Lambda_2 - s_1, \ldots, \Lambda_b - s_1\big)\;.$$

Observation 6.1. $\Lambda(T) \sim \text{Mult}(n - s_0 - bs_1, \mathcal{V})\,.$

Uniform Split Vectors

Here we define uniform splits in terms of split vectors defined by Dirichlet distributions (see Section 5.2). We then show that the random split tree model includes random b-ary search trees.

Definition 6.1 (**Uniform Split Vector**)**.** A *uniform b-ary split vector* is a random probability vector distributed like $\text{Dir}(\mathbf{1})$, where $\mathbf{1}$ denotes the ones vector of length b. The components of such a vector are jointly distributed like the interval lengths obtained by cutting the $[0, 1]$ interval at $b - 1$ points distributed i.i.d. like $\text{U}(0, 1)$.

Proposition 6.2. *Let $\mathcal{V}$ be a uniform b-ary split vector. A random b-ary search tree storing n elements is distributed like the random split tree, also storing n elements, with prototype split vector $\mathcal{V}$ and parameters $(b, s_0, s_1, s) = (b, b - 1, 0, b - 1)$.*

Proof. Using T_b and T_{split} to denote the two trees we wish to show that T_b and T_{split} are identically distributed. For $n \leq k - 1$ the proposition is true since each tree is a single node. We proceed from this base case by induction on n.

We want to show that $\Lambda(T_b)$ and $\Lambda(T_{\text{split}})$ are identically distributed[3]. From the definition of discrete Dirichlet distributions in Section 5.2 it can be seen

[3]Here the meaning of $\Lambda(T_b)$ should be clear despite the fact that we have not proven T_b to be a random split tree.

that $\Lambda(T_b) \sim \text{Dir}_n(\mathbf{1})$. By Observation 6.1, $\Lambda(T_{\text{split}})$ is distributed like $\text{Mult}(\mathcal{V})$. Since $\mathcal{V} \sim \text{Dir}(\mathbf{1})$, Proposition 5.4 tells us that $\text{Dir}_n(\mathbf{1})$ and $\text{Mult}(\mathcal{V})$ have the same distribution, so $\Lambda(T_b)$ and $\Lambda(T_{\text{split}})$ are identically distributed. The proposition now follows from a recursive coupling argument as in the proof of Proposition 6.1. □

The random BST is a nice example of a random split tree because it is so simple. Unfortunately there is an important issue that is easy to miss. Consider the following statements:

1. Each split vector is distributed like $(V_1, V_2, \ldots, V_b)$.
2. The i^{th} subtree of the root contains $s_1 + (n - s_0 - bs_1)V_i \pm \mathcal{O}(1)$ elements.

For a random BST both statements are true. However, it is not the case that both statements are true for all random split trees. The number of elements in the root's i^{th} subtree is distributed like $s_1 + \text{Bin}(n - s_0 - bs_1, V_i)$. The reason both statements are true for a random BST is that, for $U \sim \text{U}(0, 1)$, we have $\text{Bin}(n-1, U) \sim \lfloor nU \rfloor$. Consider, on the other hand, a random split tree defined by $\mathcal{V} = \left(\frac{1}{2}, \frac{1}{2}\right)$. The number of elements in each subtree is distributed like $\text{Bin}\left(n-1, \frac{1}{2}\right)$. Thus perfectly balanced split vectors do not generate perfectly balanced trees; in fact perfectly balanced trees do not belong to the random split tree model.

Limit Laws

In Section 6.1 we analyzed H_n and D_n, the height and average depth of a random binary search tree. Ideally the same could be done for the more general model of random split trees. Let T_n be a random split tree storing n elements and let $H(T_n)$ and $D(T_n)$ respectively denote be the maximum and average element depth in T_n. It is important to emphasize that we are dealing with depths of *elements*, not tree nodes. This was a non-issue for a random BST since each node contains exactly one element.

Devroye (31) proved limit laws for both height and depth in a random split tree that are valid under reasonable assumptions. Let V be a component of

$\mathcal{V}$ chosen u.a.r., *i.e.* $V \sim V_I$ where i is uniform on $\{1, \ldots, b\}$. Devroye's limit laws depend only on b and the distribution of V.

Theorem 6.6. *Let T_n be a random split tree storing n elements whose parameters satisfy the following reasonable assumptions:*

1. *The parameters b, s_0, s_1, and s are fixed constants.*
2. *$\mathcal{V}$ is well-behaved — $\mu > 0$, $\sigma > 0$, and $\mathbb{P}\{V = 1\} = 0$.*

Let μ and σ respectively denote the mean and standard deviation of $V \ln(1/V)$. Let γ be another constant depending only on V and b. With convergence in probability,

$$\lim_{n\to\infty} \frac{H(T_n)}{\ln n} = \gamma \qquad \text{and} \qquad \lim_{n\to\infty} \frac{D^*(T_n)}{\ln n} = \frac{1}{\mu} .$$

Furthermore, $D^*(T_n)$ exhibits convergence in distribution, with

$$\frac{D^*(T_n) - (\ln n)/\mu}{\sigma\sqrt{(\ln n)/\mu^3}} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1) .$$

Theorem 6.6 is powerful. For a reasonably behaved random split tree we need only provide the prototype split vector and the theorem tells us the most important properties of the tree's limiting structure. The value of γ can be expressed explicity (though not as cleanly as μ and σ) as a function of V and b (31, pp. 411–412).

Geometric Examples

We are particularly interested in random split trees that arise in geometric settings. Such a tree T typically has the following properties:

- The tree stores a set S of n points in $\mathbb{R}^d$.
- The root node partitions $\mathbb{R}^d$ into convex regions according to a geometric split.

And for an internal node v in the tree where $T(v)$ is the subtree rooted at v,

- $T(v)$ corresponds to a convex region $R(v)$ of $\mathbb{R}^d$.
- $T(v)$ contains the points in $S \cap R(v)$.
- v partitions $T(v)$ according to a geometric split.

Thus a geometric split tree corresponds to a recursive decomposition of $\mathbb{R}^d$.

A random BST is one such example. We can consider it as storing a set of points on the real line $\mathbb{R}$ and recursively decomposing $\mathbb{R}$ into convex regions (in this case, line segments and rays) using hyperplanes (in this case, points). A random k-d tree is a generalization of this to higher dimensional point sets. From Propositions 6.1 and 6.2 we know that random k-d trees are random split trees as per Model 6.2.

Devroye considered several other geometric examples belonging to the random split tree model; we give several examples along with splitter distributions (for a table see (31, p. 414)). For arbitrary point sets in an interval of $\mathbb{R}$, two examples (that both generalize a random BST) are:

- *b-ary search tree*: select $b-1$ points u.a.r. and partition into the b sub-intervals defined by these points. $V \sim \text{beta}(1, b-1)$.
- *Median of* $(2k+1)$ *search tree*: select $2k+1$ points u.a.r. and partition into the 2 sub-intervals defined by the median of these $2k+1$ points. $V \sim \text{beta}(k+1, k+1)$.

For point sets *uniformly distributed* in some bounding polytope in $\mathbb{R}^d$, two examples (that also both generalize a random BST) are:

- *Simplex tree*: for points in a simplex in $\mathbb{R}^d$, sample one point p u.a.r. and partition into $d+1$ sub-simplices, each defined as the convex hull of p and a facet of the original simplex. $V \sim \text{beta}(1, d)$.
- *Quadtree*: for points in a box[4] in $\mathbb{R}^d$, sample one point p u.a.r. and partition into 2^d sub-rectangles by cutting along the d axis-parallel hyperplanes passing through p. For i.i.d. uniforms U_i we have $V \sim \prod_{i=1}^{d} U_i$.

[4] A d-dimensional rectangular prism which we can assume w.l.o.g. to be axis-parallel.

For a uniformly distributed point set in a bounding region R, the probability of a newly sampled point falling in $T(v)$ is proportional to the volume of $R(v)$; specifically it is $\text{vol}(R(v))/\text{vol}(R)$. Thus V is distributed like the relative volume of a child region chosen u.a.r. For simplex trees and quadtrees, these relative volumes are distributed identically at every node; this is a necessary condition for membership in the random split tree model.

One may ask why b-ary search trees and median of $(2k+1)$ search trees fit the model even when S is a deterministic point set. The reason is that every set of n distinct points in $\mathbb{R}$ has the same combinatorial structure with regard to the splits we use. Consider, for example, the range space $\mathcal{S} = (S, \mathcal{R})$ where S is a set of n distinct points in $\mathbb{R}^d$ and $\mathcal{R}$ contains intersections of S with open half-spaces. For $d = 1$ the range space is identical (modulo relabeling) for any set S, but the same cannot be said for $d > 1$. Thus in $\mathbb{R}$ the point set doesn't matter and, for example, can be assumed to be a set of n uniformly distributed points in the interval $[0, 1]$.

6.3 Bounding With Reference Trees

An important property of random split trees is that all split vectors are i.i.d. like the prototype split vector $\mathcal{V}$. Random geometric trees do not necessarily fit this requirement of the random split tree model. In this case our approach is to analyze a geometric split tree using bounds on the distributions of split vectors.

Model 6.3 (**Relaxed Random Split Tree**). A *random split tree* in the *relaxed* model is a random tree storing a data set of n elements. The relaxed model differs from the standard random split tree model (see Model 6.2) only in the distribution of split vectors. In the standard model, $\mathcal{V} = (V_1, \ldots, V_b)$ is a fixed multivariate distribution and each split vector is i.i.d. like $\mathcal{V}$. In the relaxed model, $\mathcal{V}$ is replaced by a family of distributions

$\widehat{\mathcal{V}}$. Each split vector in the tree is distributed independently (but not necessarily identically) like some distribution in $\widehat{\mathcal{V}}$. Thus the parameters specify a *family* of random trees since degrees of freedom remain with regard to the assignment of distributions from $\widehat{\mathcal{V}}$ to tree nodes.

This relaxed model is not all-inclusive. However, even if a random tree T_n storing n elements does not fit the relaxed model, the theory of random split trees can be used to bound the distributions of T_n's structural properties. To this end, we introduce another generalization of the random split tree model.

Model 6.4 (**Split-Bounded Random Tree**)**.** A *random tree* in the *split-bounded* model is a random tree storing a data set of n elements. This model, like Model 6.2, is defined with parameters $\mathcal{V}, b, s_0, s_1, s$. A tree T in this model satisfies the following five criteria:

1. Every internal node has branching factor b.
2. Every internal node contains s_0 elements.
3. Every leaf node contains at least s_1 elements.
4. Every leaf node contains at most s elements.

We define additional notation before stating the final criterion. For a node v in T, let $X(v)$ denote the number of elements in v's subtree. For an internal node v with children $c_1, \ldots, c_b$, let $\Lambda(v)$ be the b-ary vector whose i^{th} component is $X(c_i) - s_1$. The last criterion is:

5. For every internal node v, the distribution of $\Lambda(v)$ is such that

$$\left(\frac{\Lambda(v)}{X(v) - s_0 - bs_1} \,\middle|\, X(v) \right) \preceq^E \text{Mult}(\mathcal{V}) \ .$$

Observation 6.2. Let $T(n)$ be a split-bounded random tree on n elements from Model 6.4 with parameters $\mathcal{V}, b, s_0, s_1, s$. Let $T^{(+)}(n)$ be the random split tree on n elements from Model 6.2, also with parameters $\mathcal{V}, b, s_0, s_1, s$. If the

children of the root of $T(n)$ have subtree sizes $N_1, \ldots, N_b$ and the children of the root of $T^{(+)}(n)$ have subtree sizes $N_1^{(+)}, \ldots, N_b^{(+)}$, then $(N_1, \ldots, N_b) \preceq^E (N_1^{(+)}, \ldots, N_b^{(+)})$.

Depth Domination

We consider the average depth in a random tree T from the split-bounded model. We bound the depth distribution $D(T)$ using a reference tree from the standard random split tree model.

For given parameters $\mathcal{V}, b, s_0, s_1, s$, let $\mathcal{T}$ be the set of all split-bounded random trees from Model 6.4 with parameters $\mathcal{V}, b, s_0, s_1, s$. Let $\mathcal{T}(n)$ be the set of trees in $\mathcal{T}$ containing n elements. Let the reference tree $T^{(+)}(n)$ be the random split tree containing n elements defined by Model 6.2 with parameters $\mathcal{V}, b, s_0, s_1, s$.

Lemma 6.2 (Depth Domination Lemma). *For any tree $T \in \mathcal{T}(n)$ we have domination of average element depth given by*

$$D(T) \ \leq^s \ D(T^{(+)}(n)) \ .$$

Proof. For any $T \in \mathcal{T}(n)$, $n \geq 0$, and $t \leq 0$ we have

$$\mathbb{P}\{D(T) \geq t\} \ = \ \mathbb{P}\big\{D(T^{(+)}(n)) \geq t\big\} \ = \ 1 \ .$$

We now need to prove that, for all $t \geq 0$,

$$\sup_{T \in \mathcal{T}(n)} \mathbb{P}\{D(T) \geq t+1\} \ \leq \ \mathbb{P}\big\{D(T^{(+)}(n)) \geq t+1\big\} \ .$$

We do this by induction on n. Consider the split at the root of T, keeping in

mind that each subtree belongs to the family $\mathcal{T}$. For $t \geq 0$, $n \geq 1$,

$$\sup_{T \in \mathcal{T}(n)} \mathbb{P}\{D(T) \geq t+1\} \leq \mathbb{E}\left\{ \sum_{i=1}^{b} \left(\frac{N_i}{n}\right) \left(\sup_{T_i \in \mathcal{T}(N_i)} \mathbb{P}\{D(T_i) \geq t\} \right) \middle| N_1, \ldots, N_b \right\} \quad (6.1)$$

where the expectation is over $N_1, \ldots, N_b$, the numbers of elements sent to the subtrees of T's root.

We argue by induction on n that for all $t \geq 0$,

$$\sup_{T \in \mathcal{T}(n)} \mathbb{P}\{D(T) \geq t+1\} \leq \mathbb{P}\left\{D(T^{(+)}(n)) \geq t+1\right\} .$$

For $n \leq 0$, this is obvious, as $\mathbb{P}\{D(T(n)) \geq t+1\} = 0$. For $n \geq 1$, we can see from (6.1) and the induction hypothesis that

$$\sup_{T \in \mathcal{T}(n)} \mathbb{P}\{D(T) \geq t+1\} \leq \mathbb{E}\left\{ \sum_{i=1}^{b} \left(\frac{N_i}{n}\right) \left(\mathbb{P}\left\{D(T^{(+)}(N_i)) \geq t\right\} \right) \middle| N_1, \ldots, N_b \right\} .$$

We argue that the right hand side is bounded from above by

$$\mathbb{E}\left\{ \sum_{i=1}^{b} \mathbb{P}\left\{D(T^{(+)}(N_i^{(+)})) \geq t\right\} \frac{N_i^{(+)}}{n} \right\}, \quad (6.2)$$

where now, $N_1^{(+)}, \ldots, N_b^{(+)}$ are the sizes of the subtrees of the root of $T^{(+)}(n)$. If this bound is true, then we are done, because (6.2) is $\mathbb{P}\left\{D(T^{(+)}(n)) \geq t\right\}$.

Consider first the function

$$\psi(n) = n\mathbb{P}\left\{D(T^{(+)}(n)) \geq t\right\} = n\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\{D_i^* \geq t\} = \sum_{i=1}^{n} \mathbb{P}\{D_i^* \geq t\} ,$$

where D_i^* is the depth of the last element inserted in a split tree, drawn from the same distribution as $T^{(+)}$, on i elements. Clearly ψ is non-negative and

non-decreasing. Further, $\psi(n+1) - \psi(n) = \mathbb{P}\{D^*_{n+1} \geq t\}$, which is trivially non-decreasing in n for fixed t, implying that ψ is also non-concave.

We now apply Observation 6.2, which tells us that $(N_1, \ldots, N_b) \preceq^E (N_1^{(+)}, \ldots, N_b^{(+)})$. Since ψ is non-negative, non-decreasing, and non-concave, this suffices to conclude (see Definition 5.3) that

$$\mathbb{E}\left\{\sum_i \psi(N_i)\right\} \;\leq\; \mathbb{E}\left\{\sum_i \psi\left(N_i^{(+)}\right)\right\} .$$

This proves the bound in (6.2), completing the proof. □

6.4 Notes

Contributions

The novel contributions in this chapter include the relaxed and split-bounded models (Models 6.3 and 6.4), and the depth domination lemma (Lemma 6.2). The depth domination lemma generalizes a theorem from a paper cowritten with Luc Devroye and Colin McDiarmid (35, Theorem 1.1). The proof of the original theorem was developed jointly with Luc Devroye. The statement of Lemma 6.2 and the generalization of the original theorem's proof are independent contributions of the thesis author.

Future Directions

The most pressing open problem arising from this chapter is whether or not the depth domination lemma can be complemented by an analogous height domination lemma. That is, do the conditions of Lemma 6.2 imply domination of height as well as depth? To prove height bounds we currently use depth domination to piggyback onto existing proofs. This is possible for height bounds, such as the bound given by Theorem 6.2, derived solely from depth bounds. This strategy is inelegant and does not always work. A height domination lemma would give us a clean and reliable way to prove height bounds.

Chapter 7

Random Hyperplane Splits

In this chapter we discuss random hyperplane splits and the closely related issue of counting k-facets. We show that random hyperplane splits are no worse than (*i.e.*, stochastically at least as balanced as) uniformly distributed splits. We then show that, for certain point sets, random hyperplane splits are distributed according to discrete beta distributions, and this is conjectured to be the worst case. Finally we give lower bounds on the number of $(\leq k)$-sets; this bounds random hyperplane splits away from perfectly even splits in a non-trivial way.

Contents

In this chapter we consider *random hyperplane splits* as a way of partitioning a set of points in $\mathbb{R}^d$.

Definition 7.1 (**Random Hyperplane Split**)**.** A random hyperplane split of a set S of n points in general position in $\mathbb{R}^d$, $n \geq d$, is defined as follows. Choose d points u.a.r. from S and let H be the unique hyperplane passing through these points. Orient H randomly with a fair coin flip. H defines two open half-spaces H^+ and H^-. The random hyperplane split is the partition of $S \setminus H$ into $S \cap H^+$ and $S \cap H^-$.

We consider a point set to be in general position if its combinatorial structure is robust with regard to infinitessimal perturbations. We state the exact definition that we use in this chapter, and more generally in the context of hyperplane splits.

Definition 7.2 (**General Linear Position**)**.** We say a set S of n points in $\mathbb{R}^d$ is in *general linear position*, or simply *general position*, if and only if, for any $i \in \{0, \ldots, d-1\}$, no i-flat contains $i+2$ or more points from S.

This general position criterion is equivalent, for $n \geq d+1$, to the requirement that any hyperplane contains at most d points. It is sufficient (but not necessary) to ensure that d points from S define a unique hyperplane.

7.1 Competing with Uniform Splits

In this section we compare a random hyperplane split in $\mathbb{R}^d$ with a uniform binary split (see Model 6.1). We show that hyperplane splits are at least as even as uniform splits. In particular, this implies domination in expectation for the corresponding split vectors (see Definition 5.3).

We consider a random hyperplane split on a set S of n points in general position in $\mathbb{R}^d$. Let N_1 and N_2 be the number of points on either side of the random hyperplane split. Let N_1^* and N_2^* be the number of points on either

side of a uniform split on n points. We know that $N_1^* \sim \text{Uniform}(\{0, \dots n-1\})$ and $N_2^* = n - 1 - N_1^*$.

Lemma 7.1. $\max(N_1, N_2) \leq^s \max(N_1^*, N_2^*)$.

And from Observation 5.1 we obtain the following:

Corollary 7.1. $(N_1, N_2) \preceq^E (N_1^*, N_2^*)$.

Proof of Lemma 7.1. Let A be the set containing the first $d-1$ points chosen for the random hyperplane. Imagine a hyperplane H rotating around the axis defined by A. Let H^+ and H^- be the open half-spaces defined by H, where H^+ and H^- rotate along with H. As H rotates in a fixed direction it intersects the points in $S \setminus A$ one after another in a rotational order. In this order label these points $p_1, \dots, p_{n-d+1}$. Use $H_{(i)}$ to denote the hyperplane passing through $A \cup \{p_i\}$ and use $H^+_{(i)}$ (resp. $H^-_{(i)}$) to denote the intersection of $S \setminus A$ and H^+ (resp. H^-), when $H = H_{(i)}$. Our definitions are rotational and can therefore be cyclic, so if we extend our definitions of $H^+_{(i)}$ and $H^-_{(i)}$ to allow indices greater than $n - d + 1$, we have $H^+_{(i)} = H^-_{(n-d+1+i)}$.

Consider the difference between $H^+_{(i)}$ and $H^+_{(i+1)}$. When H rotates from $H_{(i)}$ to $H_{(i+1)}$, p_i is added to one side of H (because H no longer passes through it) while p_{i+1} is removed from one side of H (because H now passes through it). We can now see that, over the values of i from 1 to $2(n-d+1)$, $|H^+_{(i)}|$ does a walk taking only steps of $+1$, 0, or -1. It starts, by convention, at its minimum value with $i = 1$, goes up to its maximum value at $i = n-d+1$, and returns to at most 1 plus its minimum value at $i = 2(n-d+1)$. Note that this means $|H^+_{(i)}|$ hits each value strictly between its minimum and its maximum at least twice. See Figures 7.1 and 7.2 for an example of a point set and the corresponding walk.

Given A, we have N_1 distributed like $|H^+_{(I)}|$, where I is uniform on $1, \dots, 2(n-d+1)$. Therefore, for integer k with $\min_i |H^+_{(i)}| \leq k \leq \max_i |H^+_{(i)}|$, we have

$$\mathbb{P}\{N_1 = k\} \;\geq\; \frac{1}{n-d+1}\,.$$

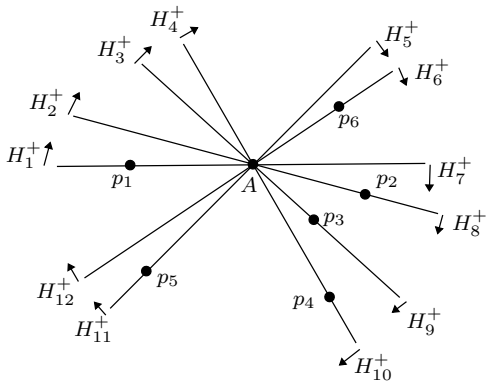


Figure 7.1: A line indicates each $H_{(i)}$. Each $H_{(i)}^+$ is indicated by an arrow and a label. A contains only the central labeled point.

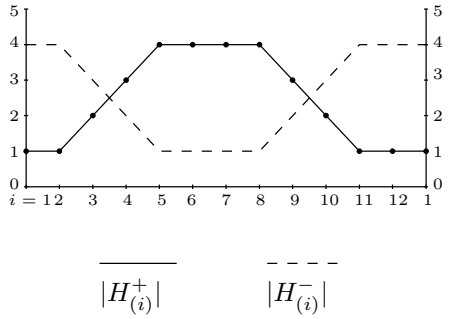


Figure 7.2: The walk done by $|H_{(i)}^+|$ is indicated with a solid line. The corresponding walk done by $|H_{(i)}^-|$ is shown with a dashed line. Values for i are given along the bottom axis and values for the two walks $|H_{(i)}^+|$ and $|H_{(i)}^-|$ appear along the side axes.

For $k > \frac{n-d}{2}$, it is impossible that $N_1 = N_2 = k$, so we have $\mathbb{P}\{\max(N_1, N_2) = k\} = \mathbb{P}\{N_1 = k\} + \mathbb{P}\{N_2 = k\}$.

$$\mathbb{P}\{\max(N_1, N_2) = k\} \;\geq\; \begin{cases} \frac{1}{n-d+1} & , \quad k = \frac{n-d}{2} \\ \frac{2}{n-d+1} & , \quad \frac{n-d}{2} < k < \max_i |H_{(i)}^+| \; . \end{cases} \tag{7.1}$$

For a uniform split on n points, let N_1^* and N_2^* be the number of points on either side of the split. N_1^* and N_2^* are both uniform on the integers $0, \ldots, n-1$, with the additional constraint that $N_1^* + N_2^* = n - 1$. Therefore, again for

integer k, we have

$$\mathbb{P}\{\max(N_1^*, N_2^*) = k\} \quad = \quad \begin{cases} 1/n & , \quad k = \frac{n-1}{2} \\ 2/n & , \quad \frac{n-1}{2} < k \leq n-1 \ . \end{cases} \tag{7.2}$$

Note that (7.1) and (7.2) are valid regardless of the parities of $n-d$ and $n-1$. The rest of the proof follows trivially.

□

Consider the case where S is a set of n points in convex position in $\mathbb{R}^2$, *e.g.*, a set of n distinct points on a parabola. It is not difficult to see that $N_1 \sim \text{Uniform}(\{0, \ldots, n-2\})$. The conditional distribution $N_1|A$, *i.e.*, the distribution of N_1 given the first point chosen to form the hyperplane split, is $\text{Uniform}(\{0, \ldots, n-2\})$, regardless of A. This is the most uneven distribution possible in $\mathbb{R}^2$. In the next section we elaborate on this special (and well-known) case and generalize it to higher dimensions.

7.2 Connections with k-Facets

In this section we focus on the problem of counting *k-facets* and how this problem is intimately related to the analysis of random hyperplane splits.

Consider a set of S points in general position in $\mathbb{R}^d$. Any d points from S define a unique hyperplane that partitions the remaining $n-d$ points. Informally, the d points form a k-facet if one of the open halfspaces contains exactly k points.

Here we define the closely related notions of k-facets and k-sets.

Definition 7.3 (k-Facet). Let S be a set of $n \geq d$ points in general position in $\mathbb{R}^d$. Let σ be an oriented $(d-1)$-simplex spanned by d points from S. The affine hull of σ is a hyperplane that defines two open halfspaces, and the orientation of σ designates one of these halfspaces, call it σ^+, as

positive. We say that σ is a k-facet iff $|S \cap \sigma^+| = k$. We say that σ is a **($\leq k$)-facet** iff σ is a j-facet for some $j \leq k$.

Definition 7.4 (k-Set). Let S be a set of $n \geq d$ points in $\mathbb{R}^d$ (we do not require S to be in general position). Let S' be a subset of S. We say that S' is a k-set iff $|S'| = k$ and there exists a hyperplane separating S' from $S \setminus S'$. We say that S' is a **($\leq k$)-set** iff S' is a j-set for some $j \leq k$.

A set of d points defines two oriented hyperplanes; thus in a set S of n points, any subset of d points defines a k-facet for exactly one value of k between 0 and $\lfloor (n-d)/2 \rfloor$.

Let $e_k(S)$ denote the number of k-facets in the point set S. We use $e_{\leq k}(S) = \sum_{i=0}^{k} e_k(S)$ to denote the number of ($\leq k$)-facets. For points in general position, the total number of oriented d-tuples from S is $e_{\leq n-d}(S) = 2\binom{n}{d}$. The set S is sometimes implicit in the notation. We follow the vector notation of Welzl (102), with $(e_i)_i = (e_0, e_1, \ldots, e_{n-d})$ and $(e_{\leq i})_i = (e_0, e_{\leq 1}, \ldots, e_{\leq n-d})$. Note that $(e_i)_i$ determines $(e_{\leq i})_i$ and vice versa.

The connection between k-facets and random hyperplane splits is fairly clear. Using K to denote the number of points on one side of a random oriented hyperplane split and X to denote the proportion of points on one side of a random oriented hyperplane split, we can define the p.m.f. and c.d.f. of X, denoted $f(x)$ and $F(x)$ respectively, using

$$f(x) \;=\; \mathbb{P}\{X = x\} \;=\; \mathbb{P}\{K = xn = k\} \;=\; \frac{e_k(S)}{2\binom{n}{d}}$$

and

$$F(x) \;=\; \mathbb{P}\{X \leq x\} \;=\; \mathbb{P}\{K \leq xn = k\} \;=\; \frac{e_{\leq k}(S)}{2\binom{n}{d}} \;.$$

Thus determining the distribution of a random hyperplane split is equivalent to counting k-facets, *i.e.*, determining the vectors $(e_i)_i$ and $(e_{\leq i})_i$.

To show that a random hyperplane split is 'good', *i.e.*, that it splits the set S evenly, we want to give an upper bound for the values $e_{\leq k}(S)$ for $k < (n-d)/2$.

In $\mathbb{R}^1$, all sets of n distinct points are combinatorially equivalent and we have $e_k = 2$ and $e_{\leq k} = 2(k+1)$. In $\mathbb{R}^2$, for any set S in general convex position we have $e_k(S) = n$ and $e_{\leq k}(S) = n(k+1)$. In the previous section we showed that points in convex position yield the most unbalanced random hyperplane splits; this is equivalent to the statement that for *any* set S of n points in general position in $\mathbb{R}^2$, $e_{\leq k}(S) \leq n(k+1)$.

Dual and Spherical Interpretations

Here we briefly mention spherical point sets, arrangements of halfspaces and hemispheres, and polar duality. For a more in-depth treatment we direct the reader to Wagner's survey (101), particularly §1.2 and §2.

Spherical Point Sets Thus far we have defined and discussed k-facets in an affine setting, *i.e.*, with regard to affine halfspaces defined by points in $\mathbb{R}^d$. Natural analogues exist in spherical settings. Let $\mathbb{S}^d$ denote the standard d-dimensional unit sphere in $\mathbb{R}^{d+1}$, given by $\mathbb{S}^d = \{U \in \mathbb{R}^{d+1} : ||u|| = 1\}$. Hemispheres in $\mathbb{S}^d$ are analogous to halfspaces in $\mathbb{R}^d$.

We say that a set S of n points in $\mathbb{S}^d$ is in general (spherical) position if and only if $S \cup \{\mathbf{0}\}$ is in general linear position in $\mathbb{R}^{d+1}$. For such a point set, any set of d points from S defines a unique great $(d-1)$-sphere that contains them. Thus a set of d points from S, along with an orientation, defines a unique hemisphere. If the interior of this hemisphere contains exactly k points from S, we say the oriented set of d points is a *spherical k-facet*.

Levels in Arrangements Let $\mathcal{A}_A$ be an arrangement of n closed affine halfspaces in $\mathbb{R}^d$. We consider only simple arrangements, in which a point $x \in \mathbb{R}^d$ belongs to the boundary of at most d halfspaces of $\mathcal{A}_A$. The *level* of a point $x \in \mathbb{R}^d$ is the number of halfspaces of $\mathcal{A}_A$ that do not include x. The ℓ-level of $\mathcal{A}_A$ is the boundary of the set of points having level $\leq \ell$. The definition of ℓ-levels carries over naturally to arrangements of hemispheres in $\mathbb{S}^d$. We say an arrangement of halfspaces in $\mathbb{R}^d$ (respectively, hemispheres in $\mathbb{S}^d$) is feasible if the halfspaces (respectively, hemispheres) have non-empty intersection.

Polar Duality Let S be a set of n points in $\mathbb{R}^d$. Polar point-hemisphere duality is a bijection that maps the set S to a feasible arrangement of n hemispheres in $\mathbb{S}^d$. If the point set S also comes with a specified origin $\mathbf{0}$, polar point-halfspace duality is a bijection that maps S to a feasible arrangement of n halfspaces in $\mathbb{R}^d$ such that $\mathbf{0}$ is contained in the feasible region. These two duality transforms are well-known; the treatment given by Wagner (101, §1.2) is particularly relevant to the study of k-facets.

For a point set S in $\mathbb{R}^d$ with a given origin, let S_S^* denote the dual feasible hemisphere arrangment in $\mathbb{S}^d$ and let S_A^* denote the dual affine halfspace arrangment in $\mathbb{R}^d$. A set of d points in S corresponds to a vertex v_A in S_A^* and a pair of antipodal vertices v_S^+ and v_S^- in S_S^*. If the d points in S define a k-facet and a $n-d-k$-facet, the corresponding antipodal vertices in S_S^* have levels k and $n-d-k$. The level of v_A is either k or $n-d-k$; only one orientation is preserved. Let $v_{\leq \ell}(S_A^*)$ denote the number of vertices in the $(\leq \ell)$-levels in S_A^*. We have $v_{\leq k}(S_A^*) \leq e_{\leq k}(S)$. We also have $e_{\leq k}(S) \leq 2 \cdot v_{\leq k}(\widehat{\mathcal{A}}_A(k))$, where $\widehat{\mathcal{A}}_A(k)$ is an arrangement of n halfspaces maximizing $v_{\leq k}(\widehat{\mathcal{A}}_A(k))$. For justification of these claims see, *e.g.*, Wagner (100; 101).

7.3 Upper Bounds for $(\leq k)$-Facets

The Moment Curve

In a search for point sets yielding uneven hyperplane cuts in dimension $d > 2$, the most natural step is to generalize sets in convex position in $\mathbb{R}^2$. For this we introduce a simple curve in $\mathbb{R}^d$ that generalizes a parabola in $\mathbb{R}^2$. We make frequent use of sets of distinct points on this curve.

Definition 7.5 (**Moment Curve**)**.** The *moment curve* in $\mathbb{R}^d$ is the parametric curve $\gamma = \{\boldsymbol{\gamma}(t) : t \in \mathbb{R}\}$ where $\boldsymbol{\gamma}(t)$ is the column vector $(t, t^2, \ldots, t^d)^T$.

Definition 7.6 (**Moment Curve Point Set**)**.** We use $\mathcal{C}(n, d)$ to denote a set of n distinct points on the moment curve in $\mathbb{R}^d$. All such point sets are combinatorially equivalent (and in general position), but for concreteness we specify $\mathcal{C}(n, d) = \{\boldsymbol{\gamma}(i) : i = 1, \ldots, n\}$.

Definition 7.7 (**Cyclic Polytope**). A *cyclic polytope* with n vertices in $\mathbb{R}^d$ is a polytope that is combinatorially equivalent to the convex hull of $\mathcal{C}(n,d)$.

A *neighbourly polytope* in $\mathbb{R}^d$ (see, *e.g.*, (105)) is a polytope such that any set of $\lfloor d/2 \rfloor$ vertices form a face. Cyclic polytopes are neighbourly polytopes that are particularly amenable to analysis.

Lemma 7.2. *For a set S of n points in $\mathbb{R}^d$, let $K = K(S)$ be the random variable representing the number of points on one side of a random hyperplane cut in S. For any $n \geq d \geq 1$ there exist point sets (of which the moment curve point set is a canonical example) such that*

$$K \;=\; K\big(\mathcal{C}(n,d)\big) \;\sim\; \begin{cases} \mathrm{beta}_n\big(\lceil \frac{d+1}{2} \rceil, \lfloor \frac{d+1}{2} \rfloor\big) & \textit{w.p.} \quad 1/2 \\ \mathrm{beta}_n\big(\lfloor \frac{d+1}{2} \rfloor, \lceil \frac{d+1}{2} \rceil\big) & \textit{w.p.} \quad 1/2 \; . \end{cases}$$

Proof. Let $S = \mathcal{C}(n,d)$ and let H be the random hyperplane defining the random cut. H can be expressed as $H = \big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{a}^T\boldsymbol{x} = b\big\}$ where $\boldsymbol{a} = (a_1, \ldots, a_d)^T$ is a column vector and b is a scalar. H defines positive and negative (open) halfspaces $H^+ = \big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{a}^T\boldsymbol{x} > b\big\}$ and $H^- = \big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{a}^T\boldsymbol{x} < b\big\}$. The orientation of H is chosen randomly, so we have

$$K \;=\; \begin{cases} \big|\mathcal{C}(n,d) \cap H^+\big| & \text{w.p.} \quad 1/2 \\ \big|\mathcal{C}(n,d) \cap H^-\big| & \text{w.p.} \quad 1/2 \; . \end{cases}$$

We wish to consider how γ and H interact. If $\boldsymbol{x} = \boldsymbol{\gamma}(t) \in |\gamma \cap H|$ is a point of intersection then $t \in \mathbb{R}$ satisfies

$$\boldsymbol{a}^T\boldsymbol{x} \;=\; \boldsymbol{a}^T\boldsymbol{\gamma}(t) \;=\; \boldsymbol{a}^T \cdot (t, t^2, \ldots, t^d)^T \;=\; \sum_{i=1}^{d} a_i t^i \;=\; b \; .$$

More generally, for arbitrary $t \in \mathbb{R}$, consider the difference polynomial

$$f_H(t) \;=\; -\,b + \sum_{i=1}^{d} a_i t^i \; .$$

Note that $f_H(t)$ is a polynomial of degree no more than d and therefore has at

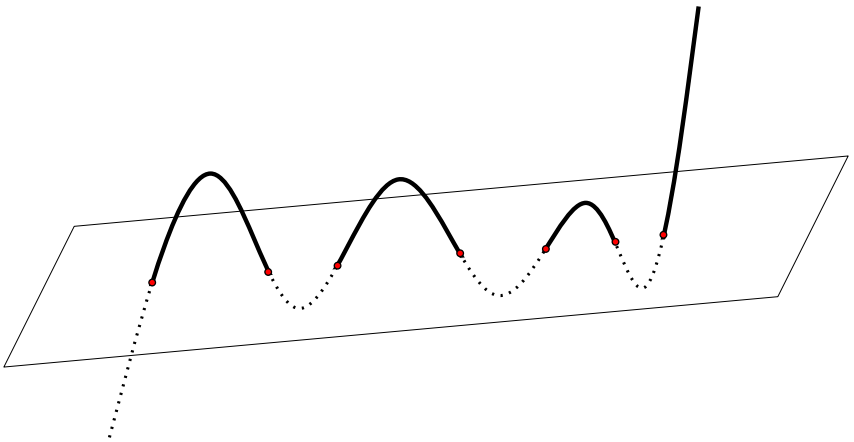


Figure 7.3: A conceptual sketch of a hyperplane H interacting with the moment curve in $\mathbb{R}^7$. Even intervals of the curve lie below H and are dotted while odd intervals lie above H and are solid.

most d roots. The sign of $f_H(t)$ is most important. We have

$$\boldsymbol{\gamma}(t) \in \begin{cases} H^+ & \text{if} \quad f_H(t) > 0 \\ H & \text{if} \quad f_H(t) = 0 \\ H^- & \text{if} \quad f_H(t) < 0 \ . \end{cases}$$

Since H is a hyperplane cut for $\mathcal{C}(n,d)$ it must touch γ at d distinct points $\boldsymbol{\gamma}(t_1), \boldsymbol{\gamma}(t_2), \ldots, \boldsymbol{\gamma}(t_d)$, for some $t_1 < t_2 < \ldots < t_d$. Therefore $f_H(t)$ has roots $t_1, \ldots, t_d$. This tells us several things. Firstly, $f_H(t)$ cannot have any roots other than $t_1, \ldots, t_d$. So in each of the $d+1$ open intervals $(-\infty, t_1), (t_1, t_2)$, $\ldots, (t_d, \infty)$, $f_H(t)$ is either strictly positive or strictly negative. For future reference we refer to these intervals as $I_0, \ldots, I_d$ respectively. Secondly, if $f_H(t)$ is positive in I_i for some $i \in \{0, \ldots, d-1\}$ then it is negative in I_{i+1} and vice versa. Otherwise γ would have to 'kiss' H at a point without crossing H, so $f_H(t)$ would have a critical point at one of its roots. However, $f_H(t)$ cannot have more than $d-1$ critical points (since $f'_H(t)$ is a degree $d-1$ polynomial) and it must have a critical point strictly between any two of its d distinct roots. In a sense, $f_H(t)$ has no spare critical point that can exist at a root.

We now turn our attention to the analysis of a random hyperplane cut in $\mathcal{C}(n,d)$. Let us choose d of these points to define our splitting hyperplane

H, and let us abuse notation by calling these points $\boldsymbol{\gamma}(t_1), \ldots, \boldsymbol{\gamma}(t_d)$ with $t_1 < t_2 < \ldots < t_d$. Here the values t_i are actually random variables taking integer values from 1 to n. As before, the points $\boldsymbol{\gamma}(t_i)$ cut the curve γ into $d+1$ intervals $I_0, \ldots, I_d$. Of the other $n-d$ points in $\mathcal{C}(n,d)$, the cutting hyperplane H separates the points in the even intervals from those in the odd intervals. Since H is oriented randomly the lemma now follows from Observation 5.3. □

For moment curve point sets, exact values for e_k are easily obtained. These values are well-known (see, *e.g.*, Andrzejak and Welzl (10)).

Proposition 7.1. *Let $q = \lfloor (d+1)/2 \rfloor$. The numbers of k-facets in a moment curve point set are given by*

$$\begin{aligned} e_k(\mathcal{C}(n,d)) &= 2\binom{k+q-1}{q-1}\binom{n-q-k}{q-1} \cdot \left(\frac{n+1}{2}\right)^{d+1-2q} \\ &= \Theta\left(\binom{n-k+1}{\lfloor d/2 \rfloor}\binom{k+1}{\lceil (d-2)/2 \rceil}\right) \\ &= \mathcal{O}\left(n^{\lfloor d/2 \rfloor}(k+1)^{\lceil (d-2)/2 \rceil}\right) , \end{aligned}$$

where the constants do not depend on d, k, or n.

Proof. We continue with the concepts and terminology from the proof of Lemma 7.2. For a k-facet σ, instead of considering intervals as being even or odd we now consider intervals as being positive or negative based on membership in σ^+ or σ^-. We focus on the lengths of intervals as given by an ordered pair of sequences: a 'positive sequence' for the positive interval lengths and a 'negative sequence' for the negative interval lengths. We call such an ordered pair a *length sequence*.

Consider a sequence of q non-negative integers whose sum is k, and let $L(k,q)$ denote the set of all such sequences. Let A denote the set of length sequences of k-facets in odd dimension $d = 2q-1$. A length sequence is in A if and only if its positive sequence is in $L(k,q)$ and its negative sequence is in $L(n-d-k,q)$. Thus we can define A as the Cartesian product of these sets, *i.e.*, $A = L(k,q) \times L(n-d-k,q)$. Similarly, let B denote the set of length sequences of k-facets in even dimension $d = 2q$. In this case we can either have

$q+1$ positive intervals and q negative intervals, or q positive intervals and $q+1$ negative intervals. Thus we have

$$B \;=\; \Big(L(k,q+1)\times L(n-d-k,q)\Big)\cup\Big(L(k,q)\times L(n-d-k,q+1)\Big)\;.$$

In even dimension, the respective lengths of the positive sequence and the negative sequence differ by 1, and the longer of the two sequences determines the sign of the first interval. A k-facet is therefore uniquely determined by its length sequence. In odd dimension, a length sequence defines two k-facets: one starting with a positive interval and one starting with a negative interval. Therefore $e_k(\mathcal{C}(n,2q-1)) = 2|A|$ and $e_k(\mathcal{C}(n,2q)) = |B|$.

There is a natural bijection mapping elements of $|L(k,q)|$ to binary strings containing k zeros and $q-1$ ones. The q intervals of zeros (some of which may be empty) have a combined length of k and are delimited by the $q-1$ ones. The number of such strings is $|L(k,q)| = \binom{k+q-1}{q-1}$. Working out the sizes of A and B gives us the values as stated by Andrzejak and Welzl (10):

$$\begin{aligned} e_k(\mathcal{C}(n,2q-1)) &= 2|A| &&= 2\tbinom{k+q-1}{q-1}\tbinom{n-q-k}{q-1} \\ e_k(\mathcal{C}(n,2q)) &= |B| &&= \tbinom{k+q}{q}\tbinom{n-q-k}{q-1} + \tbinom{k+q-1}{q-1}\tbinom{n-q-k+1}{q}\;. \end{aligned}$$

We can express $e_k(\mathcal{C}(n,2q))$ more cleanly, at the same time highlighting a connection between k-facets in dimensions $2q$ and $2q-1$. We have

$$\begin{aligned} e_k(\mathcal{C}(n,2q)) &= (n+1)\tbinom{k+q-1}{q-1}\tbinom{n-q-k}{q-1} \\ &= \tbinom{n+1}{2}\, e_k(\mathcal{C}(n,2q-1))\;. \end{aligned}$$

To be concrete as we explain this identity, we consider k-facets with labeled intervals. For $d = 2q-1$ each k-facet has $(2d)!$ possible labelings. For $d = 2q$ we always give the first and last interval the same label, so again each k-facet has $(2d)!$ possible labelings.

Consider a pair of length sequences, each of length q, along with the $(2q)!$ possible labelings. The pair maps to two labeled k-facets in $\mathcal{C}(n,2q-1)$ — one that starts with the positive sequence and one that starts with the negative

sequence. We can modify the pair of length sequences for use in $\mathcal{C}(n, 2q)$ by splitting one of the $2q$ intervals. There are $n+1$ possible ways to insert a new split — $k+q$ ways in the positive list and $n-2q-k+1$ ways in the negative list. After one of these splits one sequence is longer and this determines the sign of the first interval. The pair of sequences therefore maps to $n+1$ labeled k-facets in $\mathcal{C}(n, 2q)$, which gives us $2(2d)!e_k(\mathcal{C}(n, 2q)) = (n+1)(2d)!e_k(\mathcal{C}(n, 2q-1))$. □

Generalized Upper Bound Conjectures

It is widely believed that moment curve point sets, and more generally vertex sets of neighbourly polytopes, yield the most uneven random hyperplane splits. We state two closely related conjectures. These conjectures are called the spherical- and affine- generalized upper bound conjectures, abbreviated SGUBC and AGUBC respectively. The SGUBC was made independently by Eckhoff (39), Linhart (70), and Welzl (102); the AGUBC was made by Linhart.

Let $\mathcal{C}^*(n, d)$ denote an arrangement of hemispheres that is the polar dual of a spherical neighbourly polytope with n vertices in $\mathbb{S}^d$.

Conjecture 7.1 (**SGUBC**). *Let $\mathcal{A}_S$ be an arrangement of n closed hemispheres in $\mathbb{S}^d$. For all $0 \leq \ell < (n-d)/2$,*

$$v_{\leq \ell}(\mathcal{A}_S) \leq v_{\leq \ell}(\mathcal{C}^*(n, d)) \ .$$

Moreover, equality is attained whenever $\mathcal{A}_S$ is the polar dual of a spherical neighbourly polytope.

Conjecture 7.2 (**AGUBC**). *Let $\mathcal{A}_A$ be an arrangement of n halfspaces in $\mathbb{R}^d$. For all $0 \leq \ell \leq n-d$, we have*

$$v_{\leq \ell}(\mathcal{A}_A) \ \leq \ v_{\leq \ell}(\mathcal{C}^*(n, d)) \ .$$

Moreover, equality is attained whenever S is the vertex set of a neighbourly polytope.

Partial results are known for these conjectures. The SGUBC was proved for $d = 2$ by Peck (86) and Alon and Győry (7). Welzl proved the following theorem, which is equivalent to the SGUBC for feasible arrangements in $\mathbb{S}^3$.

Theorem 7.1 (Welzl (102)). *Let S be a set of n points in $\mathbb{R}^3$. For all $0 \leq k < (n-3)/2$, we have*

$$e_{\leq k}(S) \;\leq\; e_{\leq k}(\mathcal{C}(n,d)) \;.$$

Linhart (70) proved the AGUBC for $d \leq 4$. Wagner (100) proved the following relaxation for all d:

Theorem 7.2 (Wagner (100)). *Let $\mathcal{A}_A$ be an arrangement of n halfspaces in $\mathbb{R}^d$. For all $0 \leq \ell \leq n-d$, we have*

$$v_{\leq \ell}(\mathcal{A}_A) \;\leq\; 2 \cdot v_{\leq \ell}(\mathcal{C}^*(n,d)) \;.$$

Corollary 7.2. *Let S be a set of n points in $\mathbb{R}^d$. For all $0 \leq k < (n-d)/2$, we have*

$$e_{\leq k}(S) \;\leq\; 4 \cdot e_{\leq k}(\mathcal{C}(n,d)) \;.$$

Proof. It is known that $e_{\leq k}(S) \;\leq\; 2 \cdot v_{\leq k}(\widehat{\mathcal{A}}_A(k))$, where $\widehat{\mathcal{A}}_A(k)$ is the arrangement of n halfspaces that maximizes $v_{\leq k}(\widehat{\mathcal{A}}_A(k))$. It is also known (see, *e.g.*, Linhart (70)) that values for $v_k(\mathcal{C}^*(n,d))$ are equal to the corresponding values for $e_k(\mathcal{C}(n,d))$ as given in Proposition 7.1. The corollary then follows from the above theorem. □

7.4 Lower Bounds for ($\leq k$)-Facets

In this section we discuss lower bounds for numbers of ($\leq k$)-facets. We wish to show that if d is a fixed constant, a set of n points in general position in $\mathbb{R}^d$ cannot yield a random hyperplane split that is perfectly balanced.

Upper Bounds for Halving Facets

A halving facet is defined as a k-facet where $k \in \left\{ \lfloor (n-d)/2 \rfloor \, , \, \lceil (n-d)/2 \rceil \right\}$. Let $e_{1/2}(S)$ denote the number of halving facets in a set S. The best known upper bounds for $e_{1/2}$ are given for $d = 2$ by Dey (36), for $d = 3$ by Sharir *et al.* (94), for $d = 4$ by Matoušek *et al.* (76), and for $d \geq 5$ by Alon *et al.* (6).

An upper bound for $e_{1/2}(S)$ simply provides a lower bound for the probability that a random hyperplane split is not perfectly balanced. Unfortunately such a bound is not particularly meaningful — we would prefer upper bounds for $e_k(S)$ for all k close to $(n-d)/2$. Agarwal *et al.* (2) show that an upper bound on the number of halving facets implies an upper bound for e_k that is sensitive to k. Using their method along with the various upper bounds for halving facets, we obtain the following:

Theorem 7.3 ((2; 36; 94; 76; 6)). *For any set S of n points in general position in $\mathbb{R}^d$ we have*

$$e_k(S) \;=\; \begin{cases} \mathcal{O}\big(n(k+1)^{1/3}\big) & , \quad d = 2 \\ \mathcal{O}\big(n(k+1)^{3/2}\big) & , \quad d = 3 \\ \mathcal{O}\big(n^2(k+1)^{88/45}\big) & , \quad d = 4 \\ \mathcal{O}\Big(n^{\lfloor d/2 \rfloor}(k+1)^{\lceil d/2 \rceil - (4d-3)^{-d}}\Big) & , \quad d \geq 5 \, , \end{cases}$$

where the constant in the general $d \geq 5$ case depends on d.

Lower Bounds for ($\leq k$)-Facets

The upper bounds for $e_k(S)$ given in Theorem 7.3 are not strong enough to bound random hyperplane splits away from perfect splits as $n \to \infty$. Even when $d = 2$, for which the bounds are strongest, the split distribution could obey these bounds while the proportion of points on the larger side of a split is $(1/2) + \mathcal{O}\big(n^{-1/3}\big)$ with high probability.

In contrast, the following result, proved by Ábrego and Fernández-Merchant (1) and Lovász *et al.* (73) is meaningful in this regard:

Lemma 7.3. *For any set S of n points in general position in $\mathbb{R}^2$ and any $k < (n-2)/2$,*

$$e_{\leq k} \;\geq\; 3\binom{k+2}{2},$$

and this is tight for $k \leq n/3$.

This result was extended to higher dimensions by Aichholzer *et al.* (5), but only for certain values of k.

Lemma 7.4. *For any set S of n points in general position in $\mathbb{R}^d$, $d \geq 1$, and any $k < (n-d)/(d+1)$,*

$$e_{\leq k} \;\geq\; (d+1)\binom{k+d}{d},$$

and this is tight for $k < (n-d)/(d+1)$.

In the remainder of this section we extend this result to any $k < (n-d)/2$.

An Inductive Formulation of Hyperplane Splits

Here we introduce an inductive formulation of random hyperplane splits. In the statement and proof of Lemma 7.2, along with our discussion of discrete beta distributions in Section 5.2, we analyzed random hyperplane splits of moment curve point sets using a formulation that changes naturally as the dimension increases. We wish to formulate random hyperplane splits of general point sets similarly.

Let $\text{aff}(S_P)$ and $\text{span}(S_V)$ denote, respectively, the affine hull of a set S_P of points and the linear span of a set S_V of vectors. For $1 \leq i \leq d$, let $\boldsymbol{e_i}$ denote the unit vector in $\mathbb{R}^d$ with a 1 in the i^{th} coordinate and a 0 in every other coordinate. Consider a set $P = \{P_1, \ldots, P_d\}$ of d points sampled u.a.r. without replacement from S. For $2 \leq i \leq d$ we use $\boldsymbol{P}_i$ to denote the vector $\overrightarrow{P_{i-1}P_i}$.

For this formulation we assume our point set $S = \{p_1, p_2, \ldots, p_n\}$ satisfies a definition of general position that is stronger than general linear position (see Definition 7.2). S is in general linear position if and only if any d of the

vectors $\{\overrightarrow{p_i p_j} : 1 \leq i \leq j \leq n\}$ are linearly independent. Here we assume that S satisfies the stronger requirement that any d of the vectors in $\big(\{\overrightarrow{p_i p_j} : 1 \leq i \leq j \leq n\} \cup \{\boldsymbol{e}_i : 1 \leq i \leq d\}\big)$ are linearly independent.

Since S satisfies our strengthened definition of general position, any d of the $2d-1$ vectors in $\big(\{\boldsymbol{P}_j : 2 \leq j \leq i\} \cup \{\boldsymbol{e}_j : i+1 \leq j \leq d\}\big)$ must be linearly independent. In particular this means that, for any $1 \leq i \leq d$, the affine subspace

$$\begin{aligned} H_i &= \Big\{ p+q \ : \ p \in \mathrm{aff}\big(\{P_1, \ldots, P_i\}\big) \ , \ q \in \mathrm{span}\big(\{\boldsymbol{e}_j : i+1 \leq j \leq d\}\big) \Big\} \\ &= P_1 \ + \ \mathrm{span}\Big(\ \{\boldsymbol{P}_j : 2 \leq j \leq i\} \ \cup \ \{\boldsymbol{e}_j : i+1 \leq j \leq d\} \ \Big) \end{aligned} \tag{7.3}$$

is a hyperplane in $\mathbb{R}^d$ since $H_i - P_1$ is the linear span of $d-1$ linearly independent vectors. H_i is the hyperplane containing every point that can be obtained from a point in $\mathrm{aff}(\{P_1, \ldots, P_i\})$ by changing the last $d-i$ coordinates. Each H_i is given a random orientation according to a fair coin flip; this specifies the open halfspaces H_i^+ and H_i^-. Our final random hyperplane split is defined by $H_d = H$.

We define a partition of $\mathbb{R}^d$ into three sets related to the symmetric difference[1] (denoted with $\triangle$) of H_{i-1}^+ and H_i^+. Using $\sqcup$ to denote the disjoint union[2] of sets, we have $\mathbb{R}^d = \big(W_i \sqcup W_i^+ \sqcup W_i^-\big)$. The closed set W_i and the open sets W_i^+ and W_i^- are defined as

$$\begin{aligned} W_i &= H_{i-1} \cup H_i \\ W_i^+ &= (H_{i-1}^+ \cap H_i^+) \cup (H_{i-1}^- \cap H_i^-) \ = \ (S \setminus W_i) \cap (H_{i-1}^+ \triangle H_i^-) \\ W_i^- &= (H_{i-1}^+ \cap H_i^-) \cup (H_{i-1}^+ \cap H_i^-) \ = \ (S \setminus W_i) \cap (H_{i-1}^+ \triangle H_i^+) \ . \end{aligned}$$

The above definition is only for indices $2 \leq i \leq d$; for $i = 1$ we define $W_1 = H_1$, $W_1^+ = H_1^+$, and $W_1^- = H_1^-$. Note that our strengthened assumption of general position guarantees that $(S \cap W_i) = \{P_1, \ldots, P_i\}$.

These d partitions define a map from points to sign vectors, *i.e.*, a map

[1]The *symmetric difference* of sets A and B is $A \triangle B = (A \cup B) \setminus (A \cap B)$.
[2]Here *disjoint union* simply means the union of sets that are disjoint.

$\psi : \mathbb{R}^d \to \{-1, 0, 1\}^d$ with

$$\psi(p) = \Big(\psi_1(p), \psi_2(p), \ldots, \psi_d(p)\Big) \quad : \quad \psi_i(p) = \begin{cases} 1 & , \quad p \in W_i^+ \\ 0 & , \quad p \in W_i \\ -1 & , \quad p \in W_i^- \end{cases} .$$

It is not difficult to verify by induction on i that, for a point $p \in S$,

$$\prod_{j=1}^{i} \psi_j(p) = \begin{cases} 1 & , \quad p \in H_i^+ \\ 0 & , \quad p \in H_i \\ -1 & , \quad p \in H_i^- \end{cases} .$$

In other words, $p \in (S \setminus \{P_1, \ldots, P_i\})$ is in H_i^- if and only if p is in an odd number of the sets $\{ W_j^- : 1 \le j \le i \}$. This implies the following fact which we use later in this section:

Observation 7.1.

$$\min\Big(|S \cap H^-| \ , \ |S \cap H^+|\Big) \ \le \ \sum_{i=1}^{d} \min\Big(|S \cap W_i^-| \ , \ |S \cap W_i^+|\Big) \ .$$

To exploit this fact we must analyze the distribution of $|S \cap W_i^-|$.

Proposition 7.2. $|S \cap W_i^-| \ \sim \ \text{Uniform}(\{0, \ldots, n-i\})$.

Proof. This proof is very similar to the analysis of Section 7.1. The definition of H_i in (7.3) implies that the intersection of H_{i-1} and H_i does not depend on P_i. Rather it is fully specified by $P_1, \ldots, P_{i-1}$ as the $(d-2)$-flat given by

$$H_{i-1} \cap H_i \ = \ P_1 \ + \ \text{span}\Big(\ \{\boldsymbol{P}_j : 2 \le j \le i-1\} \ \cup \ \{\boldsymbol{e}_j : i+1 \le j \le d\} \ \Big) \ .$$

On the other hand, H_i certainly does depend on P_i. Given $P_1, \ldots, P_{i-1}$, H_i must pass through the $(d-2)$-dimensional flat $H_{i-1} \cap H_i$ but has one degree of rotational freedom around it. With this degree of freedom we can imagine H_i rotating repeatedly around its axis, hitting the points in $S \setminus \{P_1, \ldots, P_{i-1}\}$ in a fixed cyclic order. In each rotational period, H_i contains a point from $S \setminus \{P_1, \ldots, P_{i-1}\}$ at exactly $2(n-(i-1))$ positions. The position of H_i,

determined when P_i is sampled, is distributed uniformly among these $2(n-(i-1))$ positions.

As H_i rotates while H_{i-1} is fixed, $S \cap W_i^-$ changes. The points in $S \setminus \{P_1, \ldots, P_{i-1}\}$ are added to W_i^- in some order, then removed in the same order. This means that, for each $j \in \{0, \ldots, n-i\}$, exactly 2 of the $2(n-(i-1))$ positions have $|S \cap W_i^-| = j$. $|S \cap W_i^-|$ is therefore distributed like a Uniform($\{0, \ldots, n-i\}$) random variable. □

An Extended Lower Bound for ($\leq k$)-Facets

We now use our inductive formulation to generalize the lower bound of Lemma 7.4 to all $k \leq \lfloor \frac{n-d}{2} \rfloor$. Let R_i denote $|S \cap W_i^-|$. The sequence $R = (R_1, \ldots, R_d)$ can be any sequence of natural numbers for which $R_i \in \{0, \ldots, n-i\}$. Let $\mathcal{R}$ be the set of all such sequences and note that there is a bijection ψ mapping a sequence $R \in \mathcal{R}$ to an ordered set of d unique indices from $\{1, \ldots, n\}$. Under ψ, R_i is mapped to the R_i'th element in $\{1, \ldots, n\} \setminus \{R_1 + 1, \ldots, R_{i-1} + 1\}$.

Note that R bijectively defines a set $P = P(R)$ of d points from S, along with an ordering on the points in P. Each set of d points, and therefore each k-facet for $0 \leq k \leq \lfloor \frac{n-d}{2} \rfloor$, corresponds to exactly $d!$ sequences in $\mathcal{R}$ that give the $d!$ different orderings of the points. Define $K(R)$ as the unique value $k \leq \lfloor (n-d)/2 \rfloor$ for which the points $P(R)$ define a k-facet. Define $M(R)$ as

$$M(R) \;=\; \sum_{i=1}^{d} \min\Big(R_i\,,\, n-d-R_i\Big) \;\geq\; K(R)\;,$$

where the inequality $M(R) \geq K(R)$ follows from Observation 7.1.

Theorem 7.4. *For a set S of n points in general position in $\mathbb{R}^d$, for any $0 \leq k \leq \lfloor \frac{n-d}{2} \rfloor$ we have*

$$e_{\leq k}(S) \geq (d+1)\binom{k+d}{d}\;.$$

By Aichholzer et al. (5), this is tight for $k \leq \lfloor \frac{n-d}{d+1} \rfloor$.

Proof. Define $\mathcal{R}(k) = \{R \in \mathcal{R} : M(R) = k\}$. Each $R \in \mathcal{R}(k)$ corresponds to a $(\leq k)$-facet (specifically, a $K(R)$-facet). Since at most $d!$ sequences can correspond to the same k-facet, we have

$$e_{\leq k}(S) \;\geq\; \frac{1}{d!}\left|\bigcup_{j\leq k}\mathcal{R}(j)\right| \;=\; \frac{1}{d!}\sum_{j=0}^{k}\left|\mathcal{R}(j)\right| .$$

To complete the proof it is sufficient to show that $|\mathcal{R}(k)| = (d+1)!\binom{k+d-1}{d-1}$, since

$$\frac{1}{d!}\sum_{j=0}^{k}(d+1)!\binom{j+d-1}{d-1} \;=\; (d+1)\binom{k+d}{d} .$$

Each $R \in \mathcal{R}$ corresponds bijectively to an ordered set of d unique indices from $\{1,\ldots,n\}$. These d indices (regardless of their ordering) split the remaining $n-d$ indices into $d+1$ intervals. The sequence of interval lengths is a sequence of natural numbers whose sum is $n-d$. For $0 \leq k \leq \lfloor\frac{n-d}{2}\rfloor$, R is in $\mathcal{R}(k)$ if and only if one of the intervals has length $n-d-k$ and the other d intervals have a combined length of k. Let $L^*(k,d)$ denote the set of all such sequences. A length sequence from $L^*(k,d)$, along with a permutation of length d, corresponds bijectively to a sequence in $\mathcal{R}(k)$.

As in the proof of Proposition 7.1, we define $L(k,d)$ as the set of all sequences of d natural numbers whose sum is k, noting that $|L(k,d)| = \binom{k+d-1}{d-1}$. We can consider a sequence in $L^*(k,d)$ as defined by the position of the large element (of value $n-d-k$), along with a sequence from $L(k,d)$ to fill the remaining d positions. This is a bijection, so we have

$$\left|\mathcal{R}(k)\right| \;=\; d!\left|L^*(k,d)\right| \;=\; d!(d+1)\left|L(k,d)\right| \;=\; (d+1)!\binom{k+d-1}{d-1} ,$$

completing the proof. □

7.5 Notes

Contributions

For the most part, Section 7.1 is from a paper cowritten with Luc Devroye and Colin McDiarmid (31). The author of this thesis was involved in the fine-tuning of the proof of Lemma 7.1 but had little to do with the lemma's conceptual development. The lemma, equivalent to the SGUBC for $d = 2$, has actually been known since it was proved independently by Peck (86) and Alon and Győry (7) in the 1980s. This was not known to us until after publication.

More generally, k-facets and the problem of counting them were not known to the author of this thesis until fairly recently. Lemma 7.2, which relates discrete beta distributions and random hyperplane splits in moment curve point sets, was obtained by the thesis author before this k-facet revelation. Luc Devroye and Colin McDiarmid provided guidance. We did not calculate the actual values given by Proposition 7.1 independently.

Theorem 7.4, which extends the bound given by Aichholzer *et al.* (5) to all (meaningful) values of k, is a new result that was obtained independently. The inductive formulation of hyperplane splits is another novel contribution, developed by the author of this thesis as a byproduct of attempts to prove the general upper bound conjectures.

From a wider point of view, a significant contribution of this chapter is that it sheds new light on old results. For example, the values given by Proposition 7.1 have little intuitive meaning compared to the statement of Lemma 7.2, which draws the connection between a random hyperplane split and the median of d uniform random variables. Thus, though the specific values implied by Lemma 7.2 were already known, the lemma offers a novel interpretation of them.

Future Directions

The generalized upper bound conjectures (Conjectures 7.1 and 7.2) remain open problems of great interest. Improving lower bounds for the number of $(\leq k)$-facets is another open problem. The lower bounds have not garnered as

much interest, but improvements seem more accessible. We would also like to put known lower bounds through the analytical machinery of Devroye (31) to determine what the bounds would imply for the random trees to be introduced in the final chapter.

Chapter 8

Hyperplane Search Trees

We now apply the analysis of the previous three chapters to two data structures: random hyperplane search trees and random arrangement trees. Both data structures are space partition trees that recursively split a data set based on random hyperplanes. We give new bounds involving the structural distributions of these trees.

Contents

8.1 Hyperplane Search Trees

A hyperplane search tree is a binary tree used to store a set S of n d-dimensional data points. In a random hyperplane search tree for S, the root represents a hyperplane defined by d data points drawn uniformly at random from S. The remaining data points are split by the hyperplane, and the definition is used recursively on each subset. We assume that the data are points in general position in $\mathbb{R}^d$. We show that uniformly over all such data sets S, the expected height of the hyperplane tree is not worse than that of the ordinary one-dimensional random binary search tree (see Model 6.1).

Hyperplane Search Trees A hyperplane search tree is defined as follows. Given is a set $S = \{x_1, \ldots, x_n\}$ of points in general position[1] in $\mathbb{R}^d$. The root node is formed by $X_1, \ldots, X_d$, obtained by uniform random sampling without replacement from $x_1, \ldots, x_n$. The hyperplane through these points is denoted by $H = H(X_1, \ldots, X_d)$. It partitions $\mathbb{R}^d \backslash H$ into two sets H^+ and H^-, with some rule to choose which is which. The $n - d$ remaining data points are split according to membership in H^+ and H^-. The subtrees are defined recursively from there, and are randomly labeled as the left and right subtrees of the root. A set of cardinality less than d is not split: it occupies a leaf in the tree. Leaves correspond thus to collections of cardinality between 0 and $d - 1$. For $d = 1$, therefore, all points in S lie in internal nodes and all leaves are empty. Figure 8.1 shows a hyperplane tree in $\mathbb{R}^2$ and the partition of the plane into disjoint polygons defined by it.

For $d \geq 1$ and $n \geq d$, define

$$\mathcal{S}_{n,d} \;=\; \left\{\, S \;:\; S \subseteq \mathbb{R}^d\,,\; |S| = n\,,\; S \text{ is in general position} \right\}.$$

For $d \geq 2$ and a given set $S \in \mathcal{S}_{n,d}$, let $T(S)$ denote the random hyperplane search tree based on S. For $d = 1$, the hyperplane tree depends only on $|S|$, not on the elements of S. Thus, it makes sense to drop the set, and simply write

[1] We assume general linear position according to Definition 7.2.

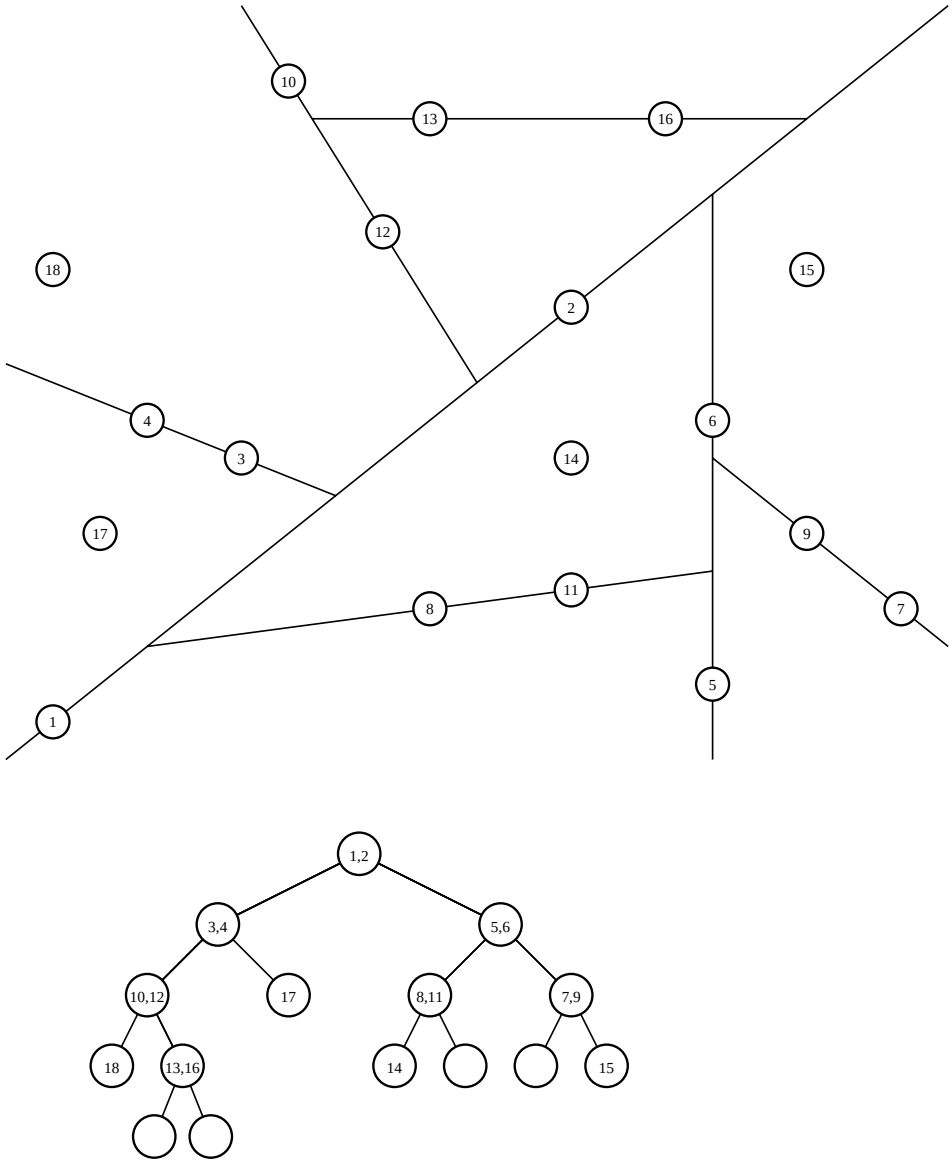


Figure 8.1: (Devroye) The figure shows the partition of the plane induced by the hyperplane tree. Internal nodes of the tree store two data points each.

$T_{|S|}$ or T_n. With this definition, the structure[2] of the usual random binary search tree on n distinct random keys is the same as that of T_n.

[2]The structure ignores any labeling of a node's children, *e.g.*, no distinction is made between a left child and a right child in a binary tree.

Related Structures

Application areas of multidimensional search trees include graphics, computational geometry, pattern recognition, and tree classification. The k-d tree (Bentley (13)), obtained by letting data points define splits that are perpendicular to one of the axes, creates a structure that is exactly distributed like the ordinary one-dimensional binary search tree if split points are picked randomly from the data (see Section 6.1). The properties are independent of the underlying distribution. Quadtrees (Samet (91)) are also based on the premise that one data point defines a split. However, the tree is 2^d-ary, as each quadrant defined by the data point corresponds to a subset of the tree. The properties of these trees depend heavily on the distribution. For the uniform density on the unit hypercube, it is known that the height H_n satisfies

$$\frac{H_n}{\ln n} \to \frac{\alpha}{d}$$

almost surely, where $\alpha = 4.311\ldots$ is as for the one-dimensional binary search tree (Devroye (28)). For additional analysis, see Flajolet, Gonnet, Puech and Robson (48) or Devroye and Laforest (33). Hyperplane search trees have a formidable property: their shapes are invariant under rotations and indeed under linear transformations in general. Rotations do alter the form of k-d trees or quadtrees, for example. This may be important in statistical applications where often one applies an appropriate linear transformation to the data to make them more manageable. Furthermore, queries such as point location (see Mehlhorn (78), for definitions) can be performed in $\mathcal{O}(\log n)$ time on the average. Unfortunately, while insertion is rather simple, and deletion in $\mathcal{O}(\log n)$ expected amortized time is achievable via lazy delete (see Cormen, Leiserson and Rivest (22) for definitions), ordinary deletion in $\mathcal{O}(\log n)$ expected time may take some extra care. Nevertheless, this too can be handled in logarithmic expected time per operation. If a constraint check (to see on which side of a hyperplane a point falls) is performed in one unit of time, then Theorem 8.1 and Corollary 8.2 show that hyperplane trees are more interesting than ordinary k-d trees. This would no longer be true if constraint checks

would cost d time units. Under a suitable vector calculus model, hyperplane trees may thus lead to improved search times, as both the quadtree and the k-d tree are based on coordinate-wise comparisons.

Trees that recursively decompose a space using hyperplanes are common in computer science. *Tree classifiers* based on hyperplane splitting have been widely used and analyzed because of their importance in pattern recognition. Closely related are *BSP trees* (binary space partition trees) used in graphics applications. See §20 of Devroye *et al.* (32) for a partial survey.

In computational geometry such trees are ubiquitous. See for example the survey of Edelsbrunner and Van Leeuwen (40), or the work of Haussler and Welzl (54) on half-space and simplex range queries. One may also consult Overmars and Van Leeuwen (85), Willard (103) or Mulmuley (80).

The partition trees obtained by Haussler and Welzl (54) generalize hyperplane trees very nicely. Instead of taking d points at random to partition a convex set into two parts, one selects $k > d$ points at random and considers the partition defined by all $\binom{k}{d}$ hyperplanes defined by subsets of size d from the k points. The sets in the partition are further partitioned in the same manner. The expected height and average depth of such trees appears not to have been studied to date; in Section 8.2 we perform analysis that relates these trees to random b-ary search trees (see Section 6.1).

Membership in Random Tree Models

In order to best apply the analysis of Chapter 7, we must determine how random hyperplane search trees fit into the random tree models from Chapter 6.

Proposition 8.1. *For a set S of n distinct points on the moment curve in $\mathbb{R}^d$, the random hyperplane search tree $T(S)$ belongs to the standard random split tree model (Model 6.2) with parameters $(b, s_0, s_1, s) = (2, d, 0, d-1)$ and prototype split vector $\mathcal{V} = (X, 1-X)$ with $X \sim \text{beta}\big(\lfloor \frac{d+1}{2} \rfloor, \lfloor \frac{d+1}{2} \rfloor\big)$.*

Proof. We focus on the distribution of the number of points falling on one side of a random hyperplane split, from which the proposition can be verified in

a straightforward manner. All sets of n points on the moment curve in $\mathbb{R}^d$ are combinatorially equivalent with regard to random arrangement splits. We therefore only concern ourselves with the ranks of the points in S as sorted by first coordinate[3]. The distribution of $\mathcal{V}$ follows from Lemma 7.2 our analysis of discrete beta distributions in Section 5.2. Specifically, Lemma 7.2 and Observation 5.4 together tell us that the number of points in one open halfspace of a random splitting hyperplane H in S (the halfspace is chosen with a fair coin flip) is distributed like

$$\mathrm{Bin}\Big(n-d\,,\mathrm{beta}\big(\,\lfloor\tfrac{d+1}{2}\rfloor\,,\,\lfloor\tfrac{d+1}{2}\rfloor\,\big)\Big)\ ,$$

as required. □

Proposition 8.2. *For each set $S \in \mathcal{S}_{n,d}$, the random hyperplane search tree $T(S)$ belongs to the split-bounded tree model (Model 6.4) with parameters $(b, s_0, s_1, s) = (2, d, 0, d-1)$ and bounding split vector $\mathcal{V} = (U, 1-U)$ with $U \sim \mathrm{U}(0,1)$.*

Proof. The split vector $\mathcal{V} = (U, 1-U)$ defines a uniform binary split. By Corollary 7.1 we know that such a split dominates any random hyperplane split. The rest of the proof is straightforward. □

Proposition 8.3. *For $S \in \mathcal{S}_{n,d}$, $d \leq 3$, the random hyperplane search tree $T(S)$ belongs to the split-bounded tree model (Model 6.4) with parameters $(b, s_0, s_1, s) = (2, d, 0, d-1)$ and bounding split vector $\mathcal{V} = (X, 1-X)$ with $X \sim \mathrm{beta}\big(\,\lfloor\tfrac{d+1}{2}\rfloor\,,\,\lfloor\tfrac{d+1}{2}\rfloor\,\big)$.*

Proof. For $d \leq 2$ this is implied by the previous proposition, so we consider the case $d = 3$. The split vector $\mathcal{V} = (X, 1-X)$ with $X \sim \mathrm{beta}\big(\,\lfloor\tfrac{d+1}{2}\rfloor\,,\,\lfloor\tfrac{d+1}{2}\rfloor\,\big)$ defines a random hyperplane split for the set $\mathcal{C}(n,d)$. Theorem 7.1 implies that a random hyperplane split in S is stochastically dominated by a random hyperplane split in $\mathcal{C}(n,d)$. The rest of the proof is straightforward. □

The SGUBC would imply that the above proposition also holds for $d \geq 4$. We can use Wagner's relaxation to bound the behaviour of random hyperplane

[3]The first coordinate of a point $\boldsymbol{\gamma}(x)$ is simply x.

search trees. We define an appropriate split vector $\mathcal{V} = (W, 1-W)$ with $W^* = \min\{W, 1-W\}$ distributed on $[0, 1/2]$ according to its c.d.f.

$$\begin{aligned}\mathbb{P}\{W^* \le x\} &= \min\left\{1, 4\cdot\mathbb{P}\left\{\tfrac{1}{2} - \left|\tfrac{1}{2} - \mathrm{beta}\left(\left\lfloor\tfrac{d+1}{2}\right\rfloor, \left\lfloor\tfrac{d+1}{2}\right\rfloor\right)\right| \le x\right\}\right\} \\ &= \min\left\{1, 8\cdot\mathbb{P}\left\{\mathrm{beta}\left(\left\lfloor\tfrac{d+1}{2}\right\rfloor, \left\lfloor\tfrac{d+1}{2}\right\rfloor\right) \le x\right\}\right\} \\ &= \min\left\{1, 8\cdot I_x\left(\left\lfloor\tfrac{d+1}{2}\right\rfloor, \left\lfloor\tfrac{d+1}{2}\right\rfloor\right)\right\},\end{aligned}$$

where I_x is the regularized incomplete beta function. As $I_x(\lfloor\frac{d+1}{2}\rfloor, \lfloor\frac{d+1}{2}\rfloor)$ becomes more tightly concentrated around $1/2$ as $d \to \infty$, so too does W.

Proposition 8.4. *For $S \in \mathcal{S}_{n,d}$, $d \ge 4$, the random hyperplane search tree $T(S)$ belongs to the split-bounded tree model (Model 6.4) with parameters $(b, s_0, s_1, s) = (2, d, 0, d-1)$ and bounding split vector $\mathcal{V} = (W, 1-W)$, where W is as defined above.*

Proof. This follows from Corollary 7.2 and the definition of W. □

Consequences

For a point set $S \in \mathcal{S}_{n,d}$, our results compare the trees $T(S)$ and T_n, where $T(S)$ denotes the random hyperplane search tree on S. The respective heights of the trees are denoted $H(S)$ and H_n. The depth of an element (not a node) sampled u.a.r. from $T(S)$ (respectively T_n) is denoted $D(S)$ (respectively D_n).

Theorem 8.1. *For each set $S \in \mathcal{S}_{n,d}$, we have $D(S) \le^s D_n$.*

Proof. This follows from Lemma 6.2 and Proposition 8.2. □

There are similar results for other measures to compare the trees $T(S)$ and T_n that follow trivially from this result. For one example, we consider the IPL (internal path length) of a tree, defined as the sum of the depths of all points (not nodes) stored in the tree.

Corollary 8.1. $\sup_{S\in\mathcal{S}_{n,d}} \mathbb{E}\{\mathrm{IPL}(T(S))\} \le \mathbb{E}\{\mathrm{IPL}(T_n)\}$.

We also have an ordering on the moments:

Corollary 8.2. *For all $r > 0$,*

$$\sup_{S \in \mathcal{S}_{n,d}} \mathbb{E}\{(D(S))^r\} \leq \mathbb{E}\{(D_n)^r\} \ ,$$

and

$$\sup_{S \in \mathcal{S}_{n,d}} \mathbb{E}\left\{e^{rD(S)}\right\} \leq \mathbb{E}\left\{e^{rD_n}\right\} \ .$$

For $c \geq 2$ let $\eta(c) = 1 - c \ln \frac{2e}{c}$. Let $\alpha = 4.31107\ldots$ be the unique solution at least 2 to $\eta(c) = 0$.

Corollary 8.3. *Let $c > \alpha$. Then $\eta(c) > 0$ and for each set $S \in \mathcal{S}_{n,d}$, $d \geq 2$ and $n \geq d$, we have*

$$\sup_{S \in \mathcal{S}_{n,d}} \mathbb{P}\{H(S) \geq c \ln n\} \leq n^{-\eta(c)}.$$

Proof. The proof of Theorem 6.2, which gives the analogous result for H_n, uses a union bound involving D_n. The bound on $H(S)$ follows by replacing D_n with $D(S)$ in that proof. □

The next theorem involves random median-of-3 trees. These are split trees, storing a fully ordered set of elements, that generalize random binary search trees. A split in a random median-of-3 tree chooses 3 elements u.a.r. and splits the remaining points based on the median of the 3 selected points. The prototype split vector is $\mathcal{V} = (X, 1 - X)$ with $X \sim \text{beta}(2, 2)$. Analytical results and historical context are given by Devroye (31).

Theorem 8.2. *Let $D_{n,3}$ denote the depth of an element sampled u.a.r. from a random median-of-3 tree storing n elements. For each set S of n points in general position in $\mathbb{R}^3$, we have $D(S) \leq^s D_{n,3}$.*

Proof. This follows from Lemma 6.2 and Proposition 8.3. □

Poblete and Munro (87) showed that $\lim_{n\to\infty} D_{n,3}/\ln n = 12/7$, and Devroye (30) showed that $\lim_{n\to\infty} H_{n,3}/\ln n \approx 3.192570\ldots$.

8.2 Arrangement Trees

We have investigated the properties of random hyperplane search trees, in which a point set in $\mathbb{R}^d$ is split into two subsets by a single hyperplane defined by d of the points chosen uniformly at random. One way to generalize this splitting process is to choose k points at random from the set for some $k \geq d$. Each d-tuple of these points defines a hyperplane, so the k points generate an *arrangement* of $\binom{k}{d}$ hyperplanes. Though we mention some basic facts about hyperplane arrangements, we direct the interested reader to Stanley's in-depth treatment (95).

An arrangement $\mathcal{A}$ of hyperplanes defines a number of regions, *i.e.*, connected components of $\mathbb{R}^d - \bigcup_{H \in \mathcal{A}} H$. We want a simple bound on the number of regions, denoted $r(\mathcal{A})$. We restate the definition of $\Phi_d(m)$, first introduced in Definition 2.8 in the context of range spaces.

$$\Phi_d(m) = \begin{cases} \sum_{i=0}^{d} \binom{m}{i} & , \quad m \geq d \\ 2^m & , \quad m \leq d \ . \end{cases}$$

It is a classical result of Schläfli (93) that an arrangement of m hyperplanes splits the space into no more than $\Phi_d(m)$ regions; this upper bound is tight if and only if the hyperplanes are in general position. For our purposes, the hyperplanes are in general position if and only if $k = d$ or $k = d+1$. However, the upper bound of $\Phi_d\left(\binom{k}{d}\right)$ always holds.

A *random arrangement tree* is a random tree that defines a recursive decomposition of $\mathbb{R}^d$. At each node, the splitting is done according to the hyperplane arrangement defined by $k \geq d$ points sampled u.a.r. without replacement. The k points are stored in the tree node and the remaining points are sent to the appropriate subtrees corresponding to the new regions of the decomposition. When $k = d$ this is the same as a random hyperplane search tree.

Random arrangement trees are not new. To our knowledge they were first described by Haussler and Welzl (54), who used them to perform efficient halfspace queries.

Membership in Random Tree Models

We prove two results for random arrangement trees. Here we let $T(S)$ denote the random arrangement tree with parameter $k \geq d+1$ for a point set S. For consistency with the random split tree model we define $T'(S)$ to be a pruned version of $T(S)$ obtained by removing empty leaf nodes so that each internal node has at least k children, and no internal node with more than k children has an empty leaf as a child.

Proposition 8.5. *For a set S of n distinct points on the moment curve in $\mathbb{R}^d$, the random arrangement tree $T'(S)$ belongs to the standard random split tree model (Model 6.2) with parameters $(b, s_0, s_1, s) = (k^*, k, 0, k-1)$ and prototype split vector $\mathcal{V}$ which is a uniform k^*-ary split vector. Here k^* is equal to k if d is even and $k+1$ if d is odd. $\mathcal{V} \sim \mathrm{Dir}(\mathbf{1})$, where $\mathbf{1}$ is the ones vector of length k^*.*

Proof. A random arrangement split is generated by k points $X_1, \ldots, X_k$ chosen u.a.r. from S. The remaining points are partitioned according to the regions defined by the arrangement

$$\mathcal{A} = \left\{ \; H_A = \mathrm{aff}(A) \;\; : \;\; A \subset \{X_1, \ldots, X_k\} \; , \; |A| = d \; \right\} .$$

All sets of n points on the moment curve in $\mathbb{R}^d$ are combinatorially equivalent with regard to random arrangement splits. We can therefore assume w.l.o.g. that $S = \{\boldsymbol{\gamma}(U_i) : 1 \leq i \leq n\}$ for i.i.d. $(0,1)$ uniforms $U_1, \ldots, U_n$. We can also assume w.l.o.g. that, for $1 \leq i \leq k$, $X_i = \boldsymbol{\gamma}(U_i)$.

We extend our analysis from the proof of Lemma 7.2. The values $U_1, \ldots, U_k$ cut the interval $(0,1)$ into $k+1$ subintervals $I_0, \ldots, I_k$; the moment curve γ is cut into corresponding intervals $\gamma_i = \{\boldsymbol{\gamma}(x) : x \in I_i\}$. No hyperplane in $\mathcal{A}$ can separate two points in the same interval of γ. A hyperplane $H_A \in \mathcal{A}$ separates intervals γ_i and γ_j if and only if A contains an odd number of points between them. For γ_0 and γ_k, all d points from A must fall between them, so they are separated by H_A if and only if d is odd. For any other pair γ_i and γ_j, $i < j < i+d$, some $H_A \in \mathcal{A}$ separates γ_i and γ_j. This follows from the existence of an odd integer a such that $a \leq j-i$ and $d-a \leq k-(j-i)$.

Thus, if d is odd, each of the intervals is contained in a different region of $\mathbb{R}^d - \mathcal{A}$. If d is even, the same is true except for γ_0 and γ_k which are contained in the same region. We define subsets $J_0, \ldots, J_{k^*-1}$ of $(0,1)$, where $J_0 = I_0 \cap I_k$ if d is even, and $J_i = I_i$ if $i \geq 1$ or d is odd. There are k^* regions of $\mathbb{R}^d - \mathcal{A}$ that can potentially contain points. We use $R_0, \ldots, R_{k^*-1}$ to denote the potentially non-empty regions of $\mathbb{R}^d - \mathcal{A}$, where

$$R_i \cap \{\boldsymbol{\gamma}(x) : x \in (0,1)\} = \{\boldsymbol{\gamma}(x) : x \in J_i\} \ .$$

Let $\boldsymbol{J}$ denote the random vector $\big(|J_1|, \ldots, |J_{k^*-1}|\big)$. When d is odd it is plain to see that $\boldsymbol{J} \sim \mathrm{Dir}(\mathbf{1})$ since $\boldsymbol{J}$ is generated by cutting the $(0,1)$ interval at points given by i.i.d. uniforms $U_1, \ldots, U_k$. We also have $\boldsymbol{J} \sim \mathrm{Dir}(\mathbf{1})$ when d is even. To see this we consider the unit circle instead of the line segment $(0,1)$. If we cut this circle at k points chosen u.a.r. and i.i.d., we get $k^* = k$ intervals whose lengths are jointly distributed like $\mathrm{Dir}(\mathbf{1})$. But $\boldsymbol{J}$ is distributed identically since the points at which we cut the circle map naturally[4] to i.i.d. uniforms $U_1, \ldots, U_k$.

To complete the proof we need only point out that $\boldsymbol{J}$ acts as a split vector; the $n-k$ remaining points fall in $R_0, \ldots, R_{k^*-1}$ with frequencies jointly distributed like $\mathrm{Mult}(\boldsymbol{J})$. □

The feasible case of the SGUBC (Conjecture 7.1) would imply that, for $d \geq 1$, vertex sets of neighbourly polytopes yield random hyperplane splits that, stochastically speaking, are the most imbalanced. This extends naturally to random arrangement splits.

Conjecture 8.1. *For $S \in \mathcal{S}_{n,d}$, $d \geq 2$, the random arrangement tree $T(S)$ belongs to the split-bounded model (Model 6.4) with parameters $(b, s_0, s_1, s) = (\Phi_d\left(\binom{k}{d}\right), k, 0, k-1)$ and prototype split vector $\mathcal{V}$ which is a uniform k^*-ary split vector. Here k^* is equal to k if d is even and $k+1$ if d is odd.*

We prove a variation that is equivalent for $d = 2$ but strictly weaker for $d \geq 3$. If the above conjecture is true, our analysis essentially wastes $d-2$ splitting points.

[4]Simply map a point $(\sin\theta, \cos\theta) \neq (1,0)$ on the circle to the point $\frac{\theta}{2\pi} - \left\lfloor \frac{\theta}{2\pi} \right\rfloor$ in $(0,1)$.

Proposition 8.6. *For $S \in \mathcal{S}_{n,d}$, $d \geq 2$, the random arrangement tree $T(S)$ belongs to the split-bounded model (Model 6.4) with parameters $(b, s_0, s_1, s) = (\Phi_d\left(\binom{k}{d}\right), k, 0, k-1)$ and prototype split vector $\mathcal{V}$ which is a uniform $(k-d+2)$-ary split vector.*

Proof. This proposition is true if a split at the root of a random $(k-d+2)$-ary tree dominates a random arrangement split; this domination follows from Lemma 8.1. □

Fan Splits We define a random fan split of a point set $S \subset \mathbb{R}^d$ as follows. Let $X_1, \ldots, X_k$ be k points chosen u.a.r. from S. We first split $\mathbb{R}^d$ with the hyperplane $\text{aff}(\{X_1, \ldots, X_d\})$. Then, for $i = d+1 \ldots k$, we split the region containing X_i (and only that region) with the hyperplane defined by $\text{aff}(\{X_1, \ldots, X_i\})$. Thus we split $\mathbb{R}^d$ into $k-d+2$ regions using one hyperplane and $k-d$ half-hyperplanes.

We prove that a random fan split is dominated by a random $(k-d+2)$-ary split. This implies that a random arrangement split of is dominated by a random $(k-d+2)$-ary split since a random arrangment split is obtained by subdividing regions created by a fan split.

Lemma 8.1. *For any $S \in \mathcal{S}_{n,d}$ a random fan split of S based on k points is dominated by a random $(k-d+2)$-ary split of n elements.*

Proof. Consider the sizes of the subsets defined by the two random splits. Let $\{N_i^F : 1 \leq i \leq k-d+2\}$ be the subset sizes for the fan split and let $\{N_i : 1 \leq i \leq k-d+2\}$ be the subset sizes for the $(k-d+2)$-ary split. To prove the lemma we must show that, for any function ψ that is non-negative, non-decreasing, and non-concave,

$$\mathbb{E}\left\{\sum_{i=1}^{k-d+2} \psi(N_i^F)\right\} \leq \mathbb{E}\left\{\sum_{i=1}^{k-d+2} \psi(N_i)\right\} . \tag{8.1}$$

Let $\widehat{N}_1^*$ and $\widehat{N}_2^*$ be the respective numbers of points on either side of the hyperplane $\text{aff}(\{X_1, \ldots, X_d\})$ from the fan split. Let $\widehat{N}_1$ and $\widehat{N}_2$ be the subset sizes obtained from a uniform split, *i.e.*, uniform on the integers $0 \ldots n-1$

with the additional constraint that $\widehat{N}_1 + \widehat{N}_2 = n - 1$. By Corollary 7.1 and the definition of vector domination, we know that

$$\mathbb{E}\Big\{ f(\widehat{N}_1^*) + f(\widehat{N}_2^*) \Big\} \leq \mathbb{E}\Big\{ f(\widehat{N}_1) + f(\widehat{N}_2) \Big\}$$

for any function f that is non-negative, non-decreasing, and non-concave. We define a function that exploits this fact.

At this point we note that in a fan split the uniform cuts are independent of the point set — only the hyperplane cut depends on the point set. It is therefore reasonable to aim to express $\mathbb{E}\Big\{ \sum_{i=1}^{k-d+2} \psi(N_i) \Big\}$ as a function of $\widehat{N}_1$ and $\widehat{N}_2$ and express $\mathbb{E}\Big\{ \sum_{i=1}^{k-d+2} \psi(N_i^*) \Big\}$ as a function of $\widehat{N}_1^*$ and $\widehat{N}_2^*$. For $i \geq 0$ and integers $m \geq x \geq 0$ we define $f_{(i)}(m, x)$ with the goal that it satisfies the following identities:

$$\begin{aligned}
\mathbb{E}\left\{ \sum_{i=1}^{k-d+2} \psi(N_i) \right\} &= \mathbb{E}\Big\{ f_{(k-d)}(n-1, \widehat{N}_1) + f_{(k-d)}(n-1, \widehat{N}_2) \Big\} \text{ and} \\
\mathbb{E}\left\{ \sum_{i=1}^{k-d+2} \psi(N_i^*) \right\} &= \mathbb{E}\Big\{ f_{(k-d)}(n-d+1, \widehat{N}_1^*) + f_{(k-d)}(n-d+1, \widehat{N}_2^*) \Big\} \ .
\end{aligned}$$

If we define $f_{(i)}(m, x) = 0$ for $m < x$ it can be verified by induction on k that these identities are satisfied by the following inductive definition of $f_{(i)}(m, x)$:

$$\begin{aligned}
f_{(0)}(m, x) &= \psi(x) \ , \\
f_{(i+1)}(m, x) &= \left(\frac{m - x}{m - 1} \right) f_{(i)}(m - 1, x) + \left(\frac{2}{m - 1} \right) \sum_{j=0}^{x-1} f_{(i)}(m - 1, j) \ .
\end{aligned} \tag{8.2}$$

Thus (8.1) holds if $f_{(k-2)}(m, x)$ is non-negative, non-decreasing, and non-concave in x for fixed m. This is clearly the case for $f_{(0)}(m, x) = \psi(x)$. It remains to extend this to $f_{(i)}(m, x)$ for $i \geq 1$.

Though $f_{(i)}(m, x)$ is bivariate and is only defined for integers m and x, we abuse notation by defining $f'_{(i)}(m, x) = f_{(i)}(m, x + 1) - f_{(i)}(m, x)$ and $f''_{(i)}(m, x) = f'_{(i)}(m, x + 1) - f'_{(i)}(m, x)$. We prove that $f_{(i)}(m, x)$, $f'_{(i)}(m, x)$, and $f''_{(i)}(m, x)$ are all non-negative for $m \geq x$ and $i \geq 0$. We assume this is

true for all indices up to and including i and prove it for $i+1$.

We have

$$\begin{aligned} f'_{(i+1)}(m,x) &= \left(\tfrac{m-x-1}{m-1}\right) f_{(i)}(m-1,x+1) - \left(\tfrac{m-x}{m-1}\right) f_{(i)}(m-1,x) \\ &\quad + \left(\tfrac{2}{m-1}\right) \sum_{j=0}^{x} f_{(i)}(m-1,j) - \left(\tfrac{2}{m-1}\right) \sum_{j=0}^{x-1} f_{(i)}(m-1,j) \\ &= \left(\tfrac{m-x-1}{m-1}\right) f'_{(i)}(m-1,x) + \left(\tfrac{1}{m-1}\right) f_{(i)}(m-1,x) \ , \end{aligned}$$

which is non-negative by our induction hypothesis.

$$\begin{aligned} f''_{(i+1)}(m,x) &= \left(\tfrac{m-x-2}{m-1}\right) f'_{(i)}(m-1,x+1) - \left(\tfrac{m-x-1}{m-1}\right) f'_{(i)}(m-1,x) \\ &\quad + \left(\frac{f_{(i)}(m-1,x+1)}{m-1}\right) - \left(\frac{f_{(i)}(m-1,x)}{m-1}\right) \\ &= \left(\tfrac{m-x-2}{m-1}\right) f''_{(i)}(m-1,x) \ , \end{aligned}$$

which is non-negative by our induction hypothesis. This completes the proof. □

8.3 Notes

Contributions

Random hyperplane search trees were the starting point for the branch of research upon which the second part of this thesis is based. The joint paper with Luc Devroye and Colin McDiarmid (31) did not consider membership in random tree models, but did include Theorem 8.1 and its corollaries. Theorem 8.2 is a novel contribution obtained with guidance from Luc Devroye. The generalization of random hyperplane search trees to random arrangement trees was undertaken independently. All results from Proposition 8.5 onwards are novel results.

Future Directions

As $d \to \infty$, random hyperplane splits converge in law to perfectly balanced splits. We have made this statement here informally and without proof, but it

is an important point that deserves more rigorous treatment, including analysis of the rate of convergence. This is the subject of a joint paper with Luc Devroye, currently in preparation.

A problem of less importance is finding a tight upper bound for the branching factor in an arrangement tree. This is a function of k and d; $\Phi_d(\binom{k}{d})$ is an upper bound, but it is only tight for $k \leq d+1$. For higher values of k the arrangement of hyperplanes is not in general position. Thus far, we have not found an expression for $r(\mathcal{A})$ that is even remotely elegant, and we have not found any expression that works for general k and d.

Chapter 9

Conclusion and Summary

9.1 Thesis Contributions

At the end of each chapter we have summarized the novel contributions. Here we reiterate the most significant progress put forward in this thesis.

Part I Our research on guarding problems has resulted in two significant contributions: one for terrains and one for polygons. For guarding terrains, in joint work with Erik Krohn we have proved that the decision problem is strongly NP-complete. This resolves a problem of significant interest that was open for the past 15 years. This result is given in Section 3.3. For the problem of guarding polygons, in joint work with David Kirkpatrick we have developed a new polynomial-time approximation algorithm for guarding simple polygons with guards on the perimeter. The approximation guarantee is $\mathcal{O}(\log \log \text{OPT})$. This is the first algorithm for guarding polygons to beat the $\mathcal{O}(\log \text{OPT})$ guarantee obtained from general methods for range spaces of bounded VC-dimension. Our algorithm is given in Section 4.2.

Part II Our research on random hyperplane splits and random geometric trees has led to new bounds for the structural distributions of several random trees. These new bounds, obtained for random hyperplane search trees and random arrangement trees in joint work with Luc Devroye and Colin McDiarmid, are given in Chapter 8. Most of these bounds are a consequence of domination results comparing the depths of average elements in a range of random trees. In addition to these bounds, Part II describes a useful connection between random hyperplane splits, or equivalently the problem of counting k-facets, and well-known distributions generated from uniform random variables.

For example, random hyperplane splits for a certain family of d-dimensional point sets (the vertices of cyclic polytopes) are distributed like splits based on the median of d uniforms. This sheds new light on the problem of counting k-facets and may help resolve a major conjecture that these splits are as uneven as random hyperplane splits can get.

Some of the results herein have already been published or submitted for publication. These include VC-dimension bounds for terrain guarding (sole author, 2008 (61)), NP-hardness of terrain guarding (with Erik Krohn, 2010 (63)), an improved approximation algorithm for guarding polygons from the perimeter (with David Kirkpatrick, 2010 (62)), and analysis of random hyperplane search trees (with Luc Devroye and Colin McDiarmid, 2009 (35)).

With the exception of Figure 8.1, all figures are original and were created independently by the author. Figure 8.1 was created by Luc Devroye.

9.2 Future Directions

Along with the novel contributions, we note open problems and future research directions at the end of each chapter. For guarding polygons, approximation algorithms still do not match inapproximability bounds. This is perhaps the clearest direction for future work on guarding problems. Our work connecting k-facets and natural probability distributions leaves unresolved issues that are more exciting. The distributions of random hyperplane splits become more tightly concentrated around perfect splits as d increases. This blessing of dimensionality is somewhat surprising. With Luc Devroye we are currently giving the issue the rigorous treatment that did not make its way into this thesis. Lower bounds for $(\leq k)$-facets are another area that we are working to improve.

Bibliography

[1] B. Ábrego and S. Fernández-Merchant. A lower bound for the rectilinear crossing number. *Graphs and Combinatorics*, 21(3):293–300, 2005.

[2] P. Agarwal, B. Aronov, T. Chan, and M. Sharir. On Levels in Arrangements of Lines, Segments, Planes, and Triangles. *Discrete and Computational Geometry*, 19(3):315–331, 1998.

[3] A. Aggarwal. *The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects*. PhD thesis, The Johns Hopkins University, 1984.

[4] O. Aichholzer and F. Aurenhammer. Straight skeletons for general polygonal figures in the plane. In *Computing and Combinatorics*, pages 117–126, 1996.

[5] O. Aichholzer, J. García, D. Orden, and P. Ramos. New results on lower bounds for the number of $(\leq k)$-facets. *European Journal of Combinatorics*, 30(7):1568–1574, 2009.

[6] N. Alon, I. Bárány, Z. Füredi, and D. Kleitman. Point selections and weak ε-nets for convex hulls. *Combinatorics, Probability and Computing*, 1(03):189–200, 1992.

[7] N. Alon and E. Győri. The number of small semispaces of a finite set of points in the plane. *Journal of Combinatorial Theory, Series A*, 41(1):154–157, 1986.

[8] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):177, 2006.

[9] N. Alon. A non-linear lower bound for planar epsilon-nets. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 2010. To Appear.

[10] A. Andrzejak and E. Welzl. In between k-sets, j-facets, and i-faces: (i, j)-partitions. *Discrete and Computational geometry*, 29(1):105–131, 2003.

[11] B. Aronov, E. Ezra, and M. Sharir. Small-size ε-nets for axis-parallel rectangles and boxes. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 639–648. ACM, 2009.

[12] B. Ben-Moshe, M. Katz, and J. Mitchell. A constant-factor approximation algorithm for optimal 1.5D terrain guarding. *SIAM Journal on Computing*, 36(6):1631–1647, 2007.

[13] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[14] C. Berge. Balanced matrices. *Mathematical Programming*, 2(1):19–31, 1972.

[15] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[16] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. *Computational Geometry: Theory and Applications*, 23(3):313–335, 2002.

[17] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14(1):463–479, 1995.

[18] T. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Proceedings of the 25th annual Symposium on Computational Geometry*, pages 333–340. ACM, 2009.

[19] P. Chanzy, L. Devroye, and C. Zamora-Cura. Analysis of range search for random k-d trees. *Acta Informatica*, 37(4):355–383, 2001.

[20] D. Z. Chen, V. Estivill-Castro, and J. Urrutia. Optimal guarding of polygons and monotone chains. In *Proceedings of the 17th Canadian Conference on Computational Geometry*, pages 133–138, 1995.

[21] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[22] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Boston, MA., 1990.

[23] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York Inc, 2008.

[24] A. Deshpande, T. Kim, E. Demaine, and S. Sarma. A pseudopolynomial time $O(\log n)$-approximation algorithm for art gallery problems. *Lecture Notes in Computer Science*, 4619:163–174, 2007.

[25] L. Devroye. Bounds for the uniform deviation of empirical measures. *Journal of Multivariate Analysis*, 12(1):72–79, 1982.

[26] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[27] L. Devroye. A note on the height of binary search trees. *Journal of the ACM*, 33:489–498, 1986.

[28] L. Devroye. Branching processes in the analysis of the heights of trees. *Acta Informatica*, 24:277–298, 1987.

[29] L. Devroye. Applications of the theory of records in the study of random trees. *Acta Informatica*, 26:123–130, 1988.

[30] L. Devroye. On the expected height of fringe-balanced trees. *Acta Informatica*, 30(5):459–466, 1993.

[31] L. Devroye. Universal limit laws for depths in random trees. *SIAM Journal on Computing*, 28:409–432, 1999.

[32] L. Devroye, L. Györi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer Verlag, New York, 1996.

[33] L. Devroye and L. Laforest. An analysis of random d-dimensional quadtrees. *SIAM Journal on Computing*, 19:821–832, 1990.

[34] L. Devroye. On the height of random m-ary search trees. *Random Struct. Algorithms*, 1(2):191–204, 1990.

[35] L. Devroye, J. King, and C. McDiarmid. Random hyperplane search trees. *SIAM Journal on Computing*, 38(6):2411–2425, 2009.

[36] T. Dey. Improved bounds for planar k-sets and related problems. *Discrete and Computational Geometry*, 19(3):373–382, 1998.

[37] M. Dom, M. Fellows, and F. Rosamond. Parameterized complexity of stabbing rectangles and squares in the plane. In *Proceedings of the 3rd International Workshop on Algorithms and Computation*, pages 298–309. Springer-Verlag, 2009.

[38] R. Downey and M. Fellows. *Parameterized Complexity*. Springer New York, 1999.

[39] J. Eckhoff. Helly, Radon, and Carathéodory type theorems. *Handbook of Convex Geometry*, pages 389–448, 1993.

[40] H. Edelsbrunner and J. van Leeuwen. Multidimensional data structures and algorithms: a bibliography. Technical report, Technische Universität Graz, 1983.

[41] A. Efrat and S. Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006.

[42] S. Eidenbenz. Inapproximability results for guarding polygons without holes. *Lecture Notes in Computer Science*, 1533:427–436, 1998.

[43] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[44] K. Elbassioni, E. Krohn, D. Matijević, J. Mestre, and D. Ševerdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, pages 1–13, 2009. 10.1007/s00453-009-9358-4.

[45] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Information Processing Letters*, 95(2):358–362, 2005.

[46] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[47] M. R. Fellows. Personal communication, 2010.

[48] P. Flajolet, G. Gonnet, C. Puech, and J. M. Robson. The analysis of multidimensional searching in quad-trees. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 100–109, Philadelphia, 1991.

[49] G. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987.

[50] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., New York, 1979.

[51] S. Ghosh. Approximation algorithms for art gallery problems. In *Proceedings of the Canadian Information Processing Society Congress*, pages 429–434, 1987.

[52] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718 – 722, 2010.

[53] M. Gibson, G. Kanade, E. Krohn, and K. Varadarajan. An Approximation Scheme for Terrain Guarding. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 140–148. Springer, 2009.

[54] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete and Computational Geometry*, 2:127–151, 1987.

[55] A. J. Hoffman, A. W. J. Kolen, and M. Sakarovitch. Totally-balanced and greedy matrices. *SIAM Journal on Algebraic and Discrete Methods*, 6:721, 1985.

[56] J. Hopcroft and R. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[57] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[58] G. Kalai and J. Matoušek. Guarding galleries where every point sees a large area. *Israel Journal of Math*, 101(1):125–139, 1997.

[59] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations: Proceedings*, page 85, 1972.

[60] J. King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. *Lecture Notes in Computer Science*, 3887:629–640, 2006.

[61] J. King. VC-dimension of visibility on terrains. In *Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 27–30, 2008.

[62] J. King and D. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. Submitted.

[63] J. King and E. Krohn. Terrain guarding is NP-hard. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1580–1593, 2010.

[64] D. Kirkpatrick. Guarding galleries with no nooks. In *Proceedings of the 12th Canadian Conference on Computational Geometry*, pages 43–46, 2000.

[65] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.

[66] J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for ε-Nets. *Discrete and Computational Geometry*, 7(1):163–173, 1992.

[67] D. Lee and A. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

[68] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[69] T. Lindvall. *Lectures on the Coupling Method*. Dover Publications, New York, 2002.

[70] J. Linhart. The Upper Bound Conjecture for arrangements of halfspaces. *Contributions to Algebra and Geometry*, 35(1):29–35, 1994.

[71] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[72] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

[73] L. Lovász, K. Vesztergombi, U. Wagner, and E. Welzl. Convex Quadrilaterals and k-Sets. *Towards a Theory of Geometric Graphs*, 342:139, 2004.

[74] H. Mahmoud and B. Pittel. On the most probable shape of a search tree grown from a random permutation. *SIAM Journal on Algebraic and Discrete Methods*, 5(1):69–81, 1984.

[75] A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, New York, 1979.

[76] J. Matoušek, M. Sharir, S. Smorodinsky, and U. Wagner. On k-sets in four dimensions. *Discrete and Computational Geometry*, 35(2):177–191, 2006.

[77] J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: Small ε-nets for disks and halfspaces. In *Proceedings of the 6th Annual Symposium on Computational Geometry*, pages 16–22. ACM, 1990.

[78] K. Mehlhorn. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*. Springer-Verlag, Berlin, 1984.

[79] K. Mehlhorn and P. Mutzel. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica*, 16(2):233–242, 1996.

[80] K. Mulmuley. Randomized multidimensional search trees: dynamic sampling. In *Proceedings of the 7th Annual Symposium on Computational Geometry*, pages 121–131, North Conway, 1991.

[81] W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2):1–29, 2008.

[82] N. Mustafa and S. Ray. PTAS for geometric hitting set problems via local search. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 17–22. ACM, 2009.

[83] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987. http://maven.smith.edu/~orourke/books/ArtGalleryTheorems/art.html.

[84] J. O'Rourke and K. J. Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–189, 1983.

[85] M. H. Overmars and J. van Leeuwen. Dynamic multidimensional data structures based on quad- and k-d trees. *Acta Informatica*, 17:265–287, 1982.

[86] G. Peck. On k-sets in the plane. *Discrete Mathematics*, 56(1):73–74, 1985.

[87] P. Poblete and J. Munro. The analysis of a fringe heuristic for binary search trees. *Journal of Algorithms*, 6(3):336–350, 1985.

[88] R. Raz and S. Safra. A sub-constant error-probability low-degree-test and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[89] B. Reed. The height of a random binary search tree. *Journal of the ACM*, 50:306–332, 2003.

[90] J. M. Robson. The height of binary search trees. *The Australian Computer Journal*, 11:151–153, 1979.

[91] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16:187–260, 1984.

[92] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

[93] L. Schläfli. Theorie der vielfachen Kontinuität. *Denkschriften der Schweizerischen naturforschenden Gesellschaft*, 38:1–237, 1901.

[94] M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for k-sets in three dimensions. *Discrete and Computational Geometry*, 26(2):195–204, 2001.

[95] R. P. Stanley. An introduction to hyperplane arrangements. In *Geometric Combinatorics*, pages 389–496. American Mathematical Society, 2007.

[96] P. Valtr. Guarding galleries where no point sees a small area. *Israel Journal of Mathematics*, 104(1):1–16, 1998.

[97] V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.

[98] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of Probability and its Applications*, volume 16, pages 264–280, 1971.

[99] V. Vazirani. *Approximation Algorithms*. Springer Verlag, New York, 2001.

[100] U. Wagner. On a geometric generalization of the upper bound theorem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 635–645, 2006.

[101] U. Wagner. k-sets and k-facets. *Contemporary Mathematics*, 453:443, 2008.

[102] E. Welzl. Entering and leaving j-facets. *Discrete and Computational Geometry*, 25(3):351–364, 2001.

[103] D. E. Willard. Polygon retrieval. *SIAM Journal on Computing*, 11:149–165, 1982.

[104] N. Young. Randomized rounding without solving the linear program. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 170–178, 1995.

[105] G. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer, New York, 1995.