Lecture Notes on Evasiveness of Graph Properties

Lectures by
László Lovász
Computer Science Department, Princeton University,
Princeton, NJ 08544, USA,
and Computer Science Department, Eõtvõs Loránd University,
Budapest, Hungary H-1088.

Notes by
Neal Young
Computer Science Department, Princeton University,
Princeton, NJ 08544, USA.

April 14, 1991

Contents

Abstract

These notes cover the first eight lectures of the class *Many Models of Complexity* taught by László Lovász at Princeton University in the Fall of 1990. The first eight lectures were on evasiveness of graph properties and related topics; subsequent lectures were on communication complexity and Kolmogorov complexity and are covered in other sets of notes.

The fundamental question considered in these notes is, given a function, how many bits of the input an algorithm must check in the worst case before it knows the value of the function. The algorithms considered are deterministic, randomized, and non-deterministic. The functions considered are primarily graph properties — predicates on edge sets of graphs invariant under relabeling of the edges.

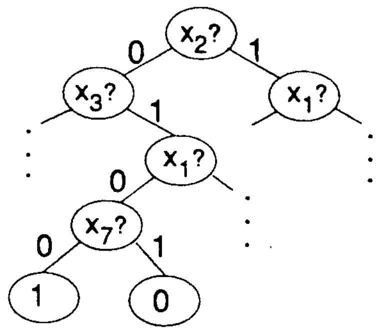

Figure 1: A Simple Decision Tree

1 Decision Trees and Evasive Properties

The goal of this course is to examine various ways of measuring the complexity of computations. In this lecture, we discuss the decision tree complexity of functions. We begin a characterization of which functions require that for any deterministic algorithm for computing the function, there is some input for which the algorithm checks all the bits of the input.

1.1 Decision Trees

decision tree

A *decision tree* is a tree representing the logical structure of certain algorithms on various inputs. The nodes of the tree represent branch points of the computation — places where more than one outcome are possible based on some predicate of the input — and the leaves represent possible outcomes. Given a particular input, one starts at the root of the tree, performs the test at that node, and descends accordingly into one of the subtrees of the root. Continuing in this way, one reaches a leaf node which represents the outcome of the computation.

simple decision tree

Given a function $f : \{0,1\}^n \rightarrow \{0,1\}$, a *simple decision tree* for the function is a binary tree whose internal nodes have labels from $\{1, 2, \ldots, n\}$ and whose leaves have labels from $\{0,1\}$. If a node has label i, then the test performed at that node is to examine the ith bit of the input. If the result is 0, one descends into the left subtree, whereas if the result is 1, one descends into the right subtree. The label of the leaf so reached is the value of the function on the input.

While it is clear that any such function f has a simple decision tree, we will be interested in simple decision trees for f which have minimal depth $D(f)$. $D(f)$ is called the *decision tree complexity of* f.

decision tree complexity of f

It is clear that $D(f)$ is at most the number of variables of f. A simple example which achieves this upper bound is the parity function $f(x_1, \ldots, x_n) = x_1 + x_2 + \cdots + x_n \bmod 2$. For this function every leaf of any simple decision tree for f has depth n, because if the value of some x_i has not been examined by the time a leaf is reached for an input x, the tree gives the same answer when x_i is flipped, so the function computed is not parity.

1.2 An Evasive Function

evasive

A function f with $D(f)$ equal to the number of variables is said to be ***evasive***. A less trivial example of an evasive function is

$$f(x_{ij} : i, j \in \{1, \ldots, n\}) = \bigwedge_i \bigvee_j x_{ij},$$

that is f is 1 iff every row of the matrix with entries x_{ij} has at least one 1.

adversary argument

To show f is evasive, we use an ***adversary argument***. We simulate the computation of some decision tree, except instead of checking the bits of the input directly, we ask the adversary. The adversary, when asked for the value of x_{ij}, responds 0 as long as some other variable in the row remains undetermined, and 1 otherwise. In this way the adversary maintains that the value of the function is undetermined until all variables have been checked. Note that in the case we show only that *some* leaf is of depth n^2.

1.3 A Non-Evasive Function

Next we give a non-trivial example of a non-evasive function. Given players $1, 2, \ldots, n$, let $x_{ij} : 1 \leq i < j \leq n$ be 1 if player i will beat player j if they play each other and 0 if j will beat i. (No draws allowed. Note that this is not necessarily a transitive relation.) The function is 1 iff there is some player who will beat everyone.

The object is to determine f without playing all possible matches. To do this, first play a "knockout tournament" — have 1 and 2 play, have the winner play 3, have the winner play 4, etc. until every player but some player i has lost to somebody. Now play i against everyone he hasn't played. If i wins all his matches, f is 1, otherwise f is 0. The number of matches played in the first stage is $n - 1$, and at most $n - 2$ are played in the second, so $D(f) \leq 2n - 3$. (How can one redesign the first stage to show $D(f) \leq 2n - \lfloor \log_2 n \rfloor$?)

1.4 Non-Deterministic Complexity

The basic idea behind these two examples is that for most functions (what are the two exceptions?) there are proper subsets of the variables whose values can determine the value of the function irrespective of the values of the other variables. The goal in minimizing decision tree depth is to discover the partial

of f

assignments as quickly as possible, while the goal in showing large decision tree complexity is to show this is not possible. Define

$$\begin{aligned} D_1(f) &= \max_{x:f(x)=1} \min\{k : \exists i_1,\ldots,i_k,\epsilon_1,\ldots,\epsilon_k : f|_{x_{i_1}=\epsilon_1,\ldots,x_{i_k}=\epsilon_k} \equiv 1\}, \\ D_0(f) &= \max_{x:f(x)=0} \min\{k : \exists i_1,\ldots,i_k,\epsilon_1,\ldots,\epsilon_k : f|_{x_{i_1}=\epsilon_1,\ldots,x_{i_k}=\epsilon_k} \equiv 0\}. \end{aligned}$$

That is, $D_i(f)$ is the least k so that from every assignment we can pick k variables such that assigning only these k values already forces the function to be i. Alternatively, $D_i(f)$ corresponds to the non-deterministic decision tree complexity of verifying $f(x) = i$, and $\max\{D_0(f), D_1(f)\}$ is the non-deterministic decision tree complexity of computing f. (A non-deterministic computation may be considered as an ordinary computation augmented by the power to to make lucky guesses.)

1.5 $D(f) \le D_0(f)D_1(f)$

For boolean x let x^ϵ denote x if $\epsilon = 0$ and $\bar{x}$ if $\epsilon = 1$. The representation

$$f(x_1,\ldots,x_n) = \bigvee_l \bigwedge_{i\in S_l} x_i^{\epsilon_{il}}$$

of f in terms of the disjunction of a number of elementary conjunctions of literals[1] is called a ***disjunctive normal form*** (DNF) of f.

literal

disjunctive normal form

If we can represent f in DNF so that every elementary conjunction has at most k terms, then $D_1(f) \le k$, because if any partial assignment of variable forces f to be 1, it must force some elementary conjunction to be 1. Conversely, there exists a DNF for f in which every elementary conjunction has at most $D_1(f)$ terms: for $\epsilon : f(\epsilon) = 1$ let S_ϵ be the indices of the minimum set of (at most $D_1(f)$) variables whose assignment $x_i = \epsilon_i$ forces f to 1. Then

$$f(x) = \bigvee_{\epsilon:f(\epsilon)=1} \bigwedge_{i\in S_\epsilon} x_i^{1-\epsilon_i}.$$

One can similarly correlate D_0 and the ***conjunctive normal form*** CNF of f.

conjunctive normal form

Next, we show the surprising relation $D(f) \le D_1(f)D_0(f)$. Write f simultaneously in DNF and CNF so that the sizes of the elementary conjunctions (disjunctions for CNF) do not exceed $D_1(f)$ ($D_0(f)$). To determine the value of f on an input x, we use the following strategy. We choose the first variable x_i in the first elementary conjunction of the DNF, and query its value ϵ_i. We then substitute the value ϵ_i for the variable x_i in the DNF and the CNF and simplify, obtaining a DNF and CNF for $f' = f|_{x_i=\epsilon_i}$. Since each elementary conjunction in the new DNF has size at most $D_1(f)$, $D_1(f') \le D_1(f)$. Similarly, $D_0(f') \le D_0(f)$.

[1]A *literal* is a boolean variable or its negation.

The crucial observation is that ***each elementary disjunction in the CNF has a variable (in fact a literal) in common with each elementary conjunction in the DNF.*** (Otherwise the variables in the elementary disjunction and the elementary conjunction can be simultaneously set to force the function to 0 and 1.) Thus by continuing the above process, by the time we have queried all of the at most $D_1(f)$ variables in the first elementary conjunction, we have reduced the size of every elementary disjunction by at least 1. It follows that we can query at most the variables in the first $D_0(f)$ elementary conjunctions before we have determined the value of the function. Thus $D(f) \leq D_0(f)D_1(f)$.

Recalling the earlier remark about non-determinism, $D_1(f)$, and $D_0(f)$, one might say that the above shows that in this model $\mathrm{NP} \cap \mathrm{co}-\mathrm{NP} = \mathrm{P}$.

1.6 The Aanderaa-Karp-Rosenberg Conjecture

We can represent functions on graphs by encoding the adjacency matrix in the input to the function. For an undirected graph with n nodes, we let $x_{ij}^G : 1 \leq i < j \leq n$ represent the presence or absence of the edge (i,j) by taking the value 1 or 0 respectively.

graph properties

S_n

In this way we can represent arbitrary functions on graphs. Generally, however, we will restrict our attention to ***graph properties*** — boolean functions whose values are independent of the labeling of the nodes of the graph. Technically, $f : \{x_{ij} : 1 \leq i < j \leq n\} \rightarrow \{0,1\}$ is a graph property if for any $\Pi \in S_n$,[2] and for any x,

$$f(\ldots, x_{ij}, \ldots) = f(\ldots, x_{\Pi(i)\Pi(j)}, \ldots).$$

monotone

The Aanderaa-Karp-Rosenberg (AKR) Conjecture is that any monotone[3] , non-trivial graph property is evasive. It is known to be true for n a prime power, and counter-examples are known if the monotonicity requirement is dropped.

weakly symmetric

transitive

A generalization of this conjecture follows. F is *weakly symmetric* if there exists a transitive[4] group $G \subseteq S_n$ such that for all $g \in G$, $f(\ldots, x_i, \ldots) = f(\ldots, x_{g(i)}, \ldots)$. The generalized conjecture is that any monotone, non-trivial, weakly symmetric boolean function is evasive.

For example, suppose $f \equiv$ "graph G has no isolated node". First, observe that for general f, if $\#\{x \in \{0,1\}^n : f(x) = 1\}$ is odd, then f is evasive. To see this, observe that for any x_i the above property is maintained for either $f|_{x_i=0}$ or $f|_{x_i=1}$, so that the adversary can answer queries so as to maintain the property as f is restricted. As long as the number of unqueried variables is at least 1, the size of the range of the restricted function is even, so the property ensures that the function is not constant.

[2] S_n denotes the symmetric group on n elements, also known as the set of permutations of size n

[3] A graph property is *monotone* if adding edges to the graph preserves the property.

[4] G is *transitive* if $\forall i,j \exists g \in G : g(i) = j$.

For the above choice of f, an inclusion/exclusion argument shows

$$\begin{aligned}\#\{G : G \text{ has no isolated vertex}\} &\equiv \sum_{k=0}^{n}(-1)^{k-1}\binom{n}{k}2^{\binom{n-k}{2}} \pmod 2\\ &\equiv (-1)^{n-1}n + (-1)^{n} \pmod 2.\end{aligned}$$

Thus provided n is even, an odd number of graphs have no isolated nodes, and f is evasive.

Note for later that we can generalize the above condition. In particular, an inductive argument in the same spirit shows:

Lemma 1.1

$$2^{n-D(f)} \mid \#\{x : f(x) = 1\}.$$

2 Evasiveness, continued

2.1 Connectivity is Evasive

If $f \equiv$ "G is connected", then f is evasive. To see this have the adversary answer "no" unless that answer would imply that the graph was disconnected, in which case she answers "yes". In this way the adversary maintains that a spanning tree exists among the "yes" and unqueried edges. If some edge (i, j) has not been queried, can the answer be known? If it is known, it must be "yes", and the "yes" edges must contain a spanning tree, so a path of "yes" edges connects i to j. Of the edges on this path, suppose the last edge queried is (u, v). At this point, we have a contradiction, because the the adversary could have answered "no" to the query of (u, v) while maintaining the possible connectedness of the graph through the other "yes" edges and edge (i, j).

Consideration shows this argument generalizes to any monotone f with the property that for any x such that $f(x) = 1$, and any $x_i = 1$, we can set $x_i = 0$, possibly setting some other $x_j = 1$, without changing the value of the function.

2.2 "Tree" Functions are Evasive

tree functions

A general class of simple but evasive functions are *tree functions* — those which have formulas using $\vee$ and $\wedge$ in which every variable occurs exactly once. The adversary has the following strategy. When asked for the value of x_i, if x_i occurs in a conjunction $(\cdots \wedge x_i \wedge \cdots)$ in the formula, the adversary claims $x_i = 1$. Otherwise x_i occurs in a disjunction and the adversary responds that $x_i = 0$. The adversary plugs the answered value into the formula, simplifies it, and continues. In this way, the adversary maintains that one variable is removed from the formula with each question, so the result can not be known unless every variable has been queried. (Clearly the same proof applies if the formula also contains negations.)

2.3 The AKR Conjecture is True for Prime n

Previously we proved that if the number of x with $f(x) = 1$ is odd, then f is evasive, and noted that this can be generalized to show that $2^{n-D(f)}$ divides this number. Here is an alternate extension: let $|x|$ denote the number of 1's in x. Define

$$\mu(f) = \sum_{f(x)=1} (-1)^{|x|}.$$

Then we can use the property $\mu(f) = \mu(f|_{x_i=0}) - \mu(f|_{x_i=1})$ to show that if $\mu(f) \neq 0$ then f is evasive. In particular, the adversary maintains that μ applied to the restricted function (i.e. f restricted by the partial assignment given by the adversary's responses so far) is non-zero, so that the restricted function is non-trivial unless all variables have been queried. (The reader may want to check the base case of this argument.)

More generally, define $p_f(t) = \sum_x f(x)t^{|x|}$. Then for a constant function c of k variables, $p_c(t) = (1+t)^k$, and so an inductive argument similar to the above shows

$$(t+1)^{n-D(f)} \,|\, p_f(t).$$

Next we use the μ criterion to prove the generalization of the AKR conjecture for prime n. A counter-example exists with $n = 14$ when n is not required to be prime.

Theorem 2.1 *If $f : \{0,1\}^n \to \{0,1\}$ is weakly symmetric, $f(\underline{0}) \neq f(\underline{1})$, and n is prime, then f is evasive.*

Proof: We will show $\mu(f) = \sum_x f(x)(-1)^{|x|} \neq 0$. The first part of the proof is to use the weak symmetry of f and the primality of n to show that there is a permutation consisting of a single cycle leaving f invariant. The second is to use this fact and the primality of n to group the inputs yielding $f(x) = 1$ except 0 or 1 into equivalence classes of size n, thus showing that $\mu(f) \equiv 1 (\text{mod } n)$, so that $\mu(f) \neq 0$.

Since f is weakly symmetric, there exists a transitive subgroup Γ of S_n leaving f invariant. Consider the partition of $\Gamma = U_1 \cup \cdots \cup U_n$ where $g \in U_i$ iff $g(1) = i$. The transitivity of Γ ensures that each U_i is of the same size, so n divides $|\Gamma|$. Since n is prime and $n \,|\, |\Gamma|$, Cauchy's theorem implies that Γ contains an element γ of order n. Since n is prime, such a permutation necessarily consists of a single cycle.

Now (assuming WLOG that $f(0) = 0$) we partition the inputs x into classes such that two elements are in the same class iff one is obtainable from the other by rotation (i.e. application of γ). Since γ leaves f invariant, and n is prime, it follows that unless every x_i is the same, each of the n possible rotations of x are distinct. Thus the values of x such that $f(x) = 1$ can be partitioned into classes of size n, except for $x = \underline{1}$. It follows that the number of such inputs modulo n is 1, so that the number of distinct non-zero terms in the expression for μ is 1 modulo n, and μ is not zero. $\square$

[Here is a sketch of how to generalize the theorem for $n = p^a$ a prime power. It is no longer necessarily true that Γ has a cyclic element, but now Γ has a transitive (sylow) subgroup Γ' of order p^b, with p^b but not p^{b+1} dividing $|\Gamma|$.

Again we group the terms of $\mu(f)$ so that two x's are in the same group if mapped by Γ' to each other. We look at the orbits of Γ' acting on $\{0,1\}^n$. The number of elements in an orbit divides $\Gamma' = p^b$ and is not equal to 1 unless $x = \underline{0}$ or $x = \underline{1}$, and all vectors in the same orbit give the same value of f.

Using this grouping we show $\mu(f) \equiv (-1)^n \bmod p$, so $\mu(f) \neq 0$.]

Before we observed that $(t+1)^{n-D(f)} \,|\, p_f(t) = \sum_x f(x)t^{|x|}$. We define $p_f(t_1, \ldots, t_n) = \sum_x f(x)t_1^{x_1} \cdots t_n^{x_n}$, and generalize this observation in the next lemma.

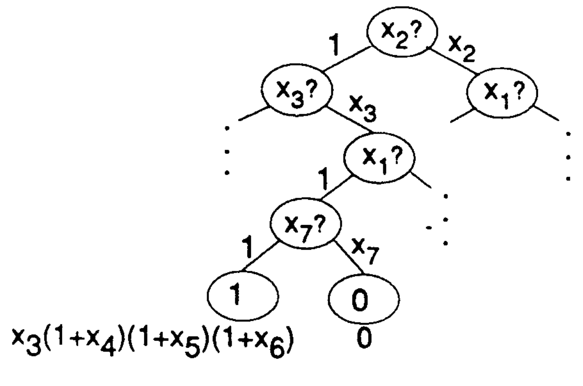

Figure 2: A term of P_f

Lemma 2.2 $p_f \in \langle (t_{i_1}+1)\cdots(t_{i_{n-D(f)}}+1) : 1 \le i_1 < \cdots < i_{n-D(f)} \le n \rangle$[5] ideal

Proof: First, if $f \equiv 0$, $p_f = 0$, and if $f \equiv 1$, $p_f = \sum_x t_1^{x_1}\cdots t_n^{x_n} = (t_1 + 1)\cdots(t_n+1)$. If f is not constant, fix a minimum depth decision tree for f and use $p_f = p_{f|_{x_1=0}} + t_1 p_{f|_{x_1=1}}$ to expand p_f into a sum of terms, each term corresponding to a "yes" leaf of the tree. Each such term is of the form $(\Pi_{i\in S_1} t_i) \times (\Pi_{i\in\overline{S}}(t_i+1))$, where S_1 is the set of indices of variables queried and found to be 1, and S is the set of variables queried. $\square$

Here is another way to look at this result. Let $z_i = t_i + 1$, and

$$\begin{aligned} Q_f(z_1,\ldots,z_n) &= p_f(z_1-1,\ldots,z_n-1) \\ &= \sum_x f(x)(z_1-1)^{x_1}\cdots(z_n-1)^{x_n} \\ &= \sum_x f(x) \sum_{y\le x} z_1^{y_1}\cdots z_n^{y_n} \times (-1)^{|x-y|} \end{aligned}$$

möb

[5] $\langle\cdots\rangle$ represents the ideal generated by $\cdots$ (the smallest set of polynomials closed under subtraction and under multiplication by any polynomial). An equivalent formulation of this lemma is

$$p_f = \sum_{1\le i_1 < \cdots < i_{n-D(f)} \le n} P_{i_1,\ldots,i_{n-D(f)}} \times (t_{i_1}+1)\cdots(t_{i_{n-D(f)}}+1),$$

where the $P_{\ldots}$ are integer coefficient polynomials of the t_i.

$$= \sum_{y} \left(\sum_{x \geq y} (-1)^{|x-y|} f(x) \right) z_1^{y_1} \cdots z_n^{y_n}$$

(Here the inequality $y \leq x$ means $\forall i, y_i \leq x_i$.)

ӧbius transform

By the previous lemma, the terms in Q_f have degree at least $n - D(f)$ in the z_i. Thus if $|y| < n - D(f)$, then $\sum_{x \geq y} (-1)^{|x-y|} f(x) = 0$. The left hand side of this equality is known as the ***möbius transform*** $\mathcal{M}f$ of f. Thus we have:

Corollary 2.3 *For* $|y| \leq n - D(f)$, $(\mathcal{M}f)(y) = 0$.

3 Non-Evasive Monotone Properties Give Contractable Complexes

In the previous lecture, we showed that every non-trivial weakly symmetric function on p^k variables for prime p is evasive.

In this lecture we continue our study of evasiveness, introducing some topological concepts related to simplicial complexes. We show that the simplicial complex associated with a non-evasive monotone function is contractable. This is the first part of a technique due to Kahn, Saks, and Sturtevant; our goal is to prove that all non-trivial, monotone, bipartite graph properties[6] and non-trivial, monotone graph properties of graphs with a prime power number of nodes are evasive.

3.1 Simplicial Complexes

simplicial complex

A *simplicial complex* is a finite collection $\mathcal{K}$ of sets such that

1. $\forall X \in \mathcal{K}, Y \subseteq X \Rightarrow Y \in \mathcal{K}$, and
2. $\mathcal{K} \neq \emptyset$.

$V(\mathcal{K})$

$V(\mathcal{K})$, the vertices of $\mathcal{K}$, consists of the elements of the sets in $\mathcal{K}$.

affine$\{S\}$, **conv**$\{S\}$

Corresponding to $\mathcal{K}$ one can construct a geometric realization $\widehat{\mathcal{K}} \subseteq \mathbb{R}^{V(\mathcal{K})}$. First, one defines the mapping $\widehat{\cdot} : V(\mathcal{K}) \rightarrow \mathbb{R}^{V(\mathcal{K})}$ so that no vertex is mapped into the affine hull[7] of any other subset of the vertices (for instance, one maps the vertices to the unit vectors). Then one extends $\widehat{\cdot}$ to any set X of vertices by $\widehat{X} = \mathrm{conv}\{\widehat{v} : v \in X\}$, and to any collection $\mathcal{C}$ (such as $\mathcal{K}$) of sets by $\widehat{\mathcal{C}} = \cup_{X \in \mathcal{C}} \widehat{X}$.

simplex

affinely independent

Note that for any $X \in \mathcal{K}$, $\widehat{X}$ is a *simplex* — the convex hull of a set of vectors none of which lies in the affine hull of any subset of the others. (Such a set of vectors is said to be ***affinely independent***.)

geometric simplicial complex

A collection $\mathcal{K} = \{S_1, \ldots, S_m\}$ of simplices in $\mathbb{R}^N$ is said to form a *geometric simplicial complex* if:

face

1. $\forall S_i \in \mathcal{K}$, T a face[8] of $S_i \Rightarrow T \in \mathcal{K}$, and
2. $\forall S_i, S_j \in \mathcal{K}, S_i \cap S_j \neq \emptyset \Rightarrow S_i \cap S_j$ is a face of both S_i and S_j.

polyhedron

A *polyhedron* is then defined as the union of the sets in any geometric simplicial complex. Note that such an entity is not necessarily convex, for instance the surface of an octahedron is a polyhedron formed by the geometric simplicial complex consisting of its faces, edges, and vertices.

[6]A bipartite graph property $f(x_{ij} : i \in V, j \in W)$ is a boolean function invariant under permutations of the edges induced by permutations of V and W. See "graph property".

[7] $\mathrm{affine}\{S\} = \left\{\sum_{v \in S} \alpha_v v : \sum \alpha_v = 1\right\}$; $\mathrm{conv}\{S\} = \left\{\sum_{v \in S} \alpha_v v : \sum \alpha_v = 1, \alpha_v \geq 0\right\}$.

[8]A *face* of a simplex $S = \mathrm{conv}\{V\}$ is a set $S' = \mathrm{conv}\{V'\} : V' \subseteq V$.

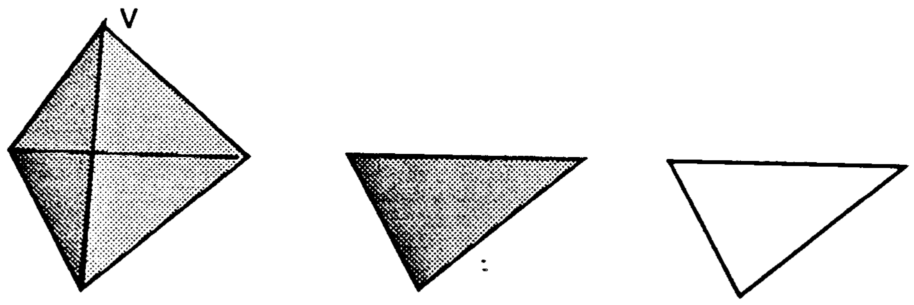

Figure 3: $\partial\Delta_3$,$\partial\Delta_3\backslash v$, and $\partial\Delta_3/v$.

3.2 Contractability

ontractable

Intuitively, a set $T \subseteq \mathbb{R}^N$ is ***contractable*** if it can be continuously shrunk to a single point, while never breaking through its original boundary. Technically, T is contractable if there exists a continuous mapping $\Phi : T \times [0,1] \rightarrow T$ with $\forall x \in T, \Phi(x,0) = x. \Phi(x,1) = p_0$ for some $p_0 \in T$. One can show that the choice of p_0 is immaterial.

If T consists of 2 distinct points in $\mathbb{R}^1$, no such mapping can exist because at some time the mapping would have to switch from mapping a point to itself to mapping the point to the other point.

If the underlying simplicial complex is a graph, if the graph is disconnected one can similarly show that the set is not contractable. Similarly, if the graph has a cycle, at any time the cycle will be in the image of the mapping, so a cyclic graph is not contractable.

Conversely, if the graph is a tree, then one can contract the graph by repeatedly contracting the edges leading to leaves.

(Note that we consider contractability of a simplicial complex synonymous with the contractability of its geometric realizations.)

omplex

Generalizing the contraction of a tree described above, we will obtain a useful sufficient condition for contractability. (Surprisingly, for a general simplicial complex $\mathcal{K}$, it is undecidable whether $\mathcal{K}$ is contractable.) For $v \in V(\mathcal{K})$, define

$$\begin{aligned} \mathcal{K}\backslash v &= \{X \in \mathcal{K} : v \notin X\} \\ \mathcal{K}/v &= \{X \in \mathcal{K} : v \notin X, X \cup \{v\} \in \mathcal{K}\}. \end{aligned}$$

$\mathcal{K}$ minus v

nk of v in $\mathcal{K}$

The first is called ***$\mathcal{K}$ minus v***; the second is called ***link of v in $\mathcal{K}$***.

Considering the boundary of the 3 dimensional simplex, $\partial\Delta_3$, which is not contractable, one sees that $\partial\Delta_3\backslash v \equiv \Delta_2$ is contractable but $\partial\Delta_3/v$, a three node cycle, is not.

Lemma 3.1 *If for some v, $\mathcal{K}/v$ and $\mathcal{K}\backslash v$ are contractable, then $\mathcal{K}$ is contractable.*

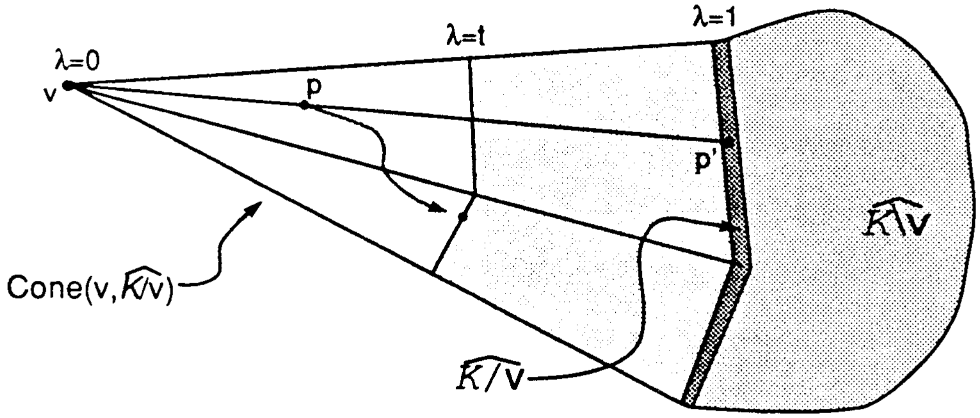
λ=0
v
p
λ=t
λ=1
p'
$\widehat{\mathcal{K}\backslash v}$
Cone(v, $\widehat{\mathcal{K}\backslash v}$)
$\widehat{\mathcal{K}/v}$

Figure 4: Contraction of $\widehat{\mathcal{C}}$ onto $\widehat{\mathcal{K}/v}$.

Proof: Let $\mathcal{C}$ denote $\{X \in \mathcal{K} : v \in X\}$, so that $\widehat{\mathcal{K}} = \widehat{\mathcal{C}} \cup \widehat{\mathcal{K}\backslash v}$.

The first step in contracting $\widehat{\mathcal{K}}$ is to use the contractability of $\mathcal{K}/v$ to construct a mapping Ψ which contracts[9] $\widehat{\mathcal{C}}$ onto $\widehat{\mathcal{K}/v}$, leaving $\widehat{\mathcal{K}\backslash v}$ fixed. Once this is accomplished, all of the points have been contracted into $\widehat{\mathcal{K}\backslash v}$, so applying the contraction of $\widehat{\mathcal{K}\backslash v}$ completes the contraction of $\widehat{\mathcal{K}}$.

Suppose $\widehat{\mathcal{K}/v}$ is contracted by Φ to p_0. Denote a point p in $\widehat{\mathcal{C}}$ by (p', λ), where $p' \in \widehat{\mathcal{K}/v}$ and $\lambda \in [0,1]$ such that $p = \lambda p' + (1-\lambda)v$. (Note that this denotation is continuous and invertible except at v.)

Let $\widehat{\mathcal{C}}_\lambda = \{(p', \lambda) : p' \in \mathcal{K}/v\}$, so $\widehat{\mathcal{C}}_1 = \mathcal{K}/v$ and $\widehat{\mathcal{C}}_0 = \{v\}$.

The contraction can be envisioned as flattening $\widehat{\mathcal{C}}$. At time t, each $\widehat{\mathcal{C}}_\lambda$ for $\lambda < t$ will have been flattened into $\widehat{\mathcal{C}}_t$, until all of $\widehat{\mathcal{C}}$ is flattened into $\widehat{\mathcal{K}/v}$. Once $\widehat{\mathcal{C}}_\lambda$ is mapped onto $\widehat{\mathcal{C}}_t$, as t grows, instead of letting the image of $\widehat{\mathcal{C}}_\lambda$ grow with $\widehat{\mathcal{C}}_t$ (which would lead to a discontinuity at v), we contract it using Φ to counteract the growth.

$$\Psi(p,t) = \begin{cases} (\Phi(p', 1-\lambda/t), t) & \text{if } t \geq \lambda, \text{ and} \\ p & \text{if } t \leq \lambda. \end{cases}$$

If $t = 0$ or $\lambda = 1$ then Ψ is the identity. If $t = 1$ all points are mapped into $\widehat{\mathcal{K}/v}$. We leave it to the reader to verify the continuity of Ψ, remarking only that as $p \to v$, $\lambda \to 0$, so $\Phi(p', \lambda) \to p_0$, independent of p'. □

[9]We generalize the notion of contraction to a point in the natural way to allow contraction to arbitrary contractable subsets.

3.3 Monotone Functions

A monotone boolean function $f \not\equiv 1$ gives a simplicial complex

$$\mathcal{K}_f = \{S \subseteq \{1,\ldots,n\} : f(x^S) = 0\}$$

in a natural way, and vice versa.[10] Also,

$$\begin{aligned}
\mathcal{K}_{f|_{x_i=0}} &= \{S \subseteq \{1,\ldots,i-1,i+1,\ldots,n\} : S \in \mathcal{K}_f\} = \mathcal{K}_f \backslash i, \\
\mathcal{K}_{f|_{x_i=1}} &= \{S \subseteq \{1,\ldots,i-1,i+1,\ldots,n\} : S \cup \{i\} \in \mathcal{K}_f\} = \mathcal{K}_f / i.
\end{aligned}$$

By now we may begin to suspect a relation between non-evasiveness and contractability. We prove such a relation in the next lemma.

Lemma 3.2 (Kahn-Saks-Sturtevant) *If $f \not\equiv 1$ is non-evasive, then $\mathcal{K}_f$ is contractable.*

Proof: Assume $f : \{0,1\}^n \to \{0,1\}$ is non-evasive, with $f \not\equiv 1$.

If $n > 1$, then there exists an i such that $f|_{x_i=0}$ and $f|_{x_i=1}$ are non-evasive. Provided $f|_{x_i=1} \not\equiv 1$, we can assume by induction that $\mathcal{K}_{f|_{x_i=0}}$ and $\mathcal{K}_{f|_{x_i=1}}$ are contractable. By the preceding lemma and remarks, it follows that $\mathcal{K}_f$ is contractable.

If $n > 1$ and $f|_{x_i=1} \equiv 1$, then $\mathcal{K}_f = \mathcal{K}_{f|_{x_i=0}}$, and $f|_{x_i=0}$ is non-evasive, so again by induction $\mathcal{K}_f$ is contractible.

Otherwise $n = 1$, so $f \equiv 0$ and $\mathcal{K}_f = \{\emptyset, \{1\}\}$, which is contractible.

$\square$

We have now established a link between evasiveness of monotone functions and the contractability of the associated simplicial complex. In the next lecture, we will use the symmetry properties of monotone graph properties and some more topology to show in some cases that the associated complexes are not contractable, and thus that the original functions are evasive.

[10] $(x^S)_i = \begin{cases} 1 & i \in S, \\ 0 & i \notin S. \end{cases}$

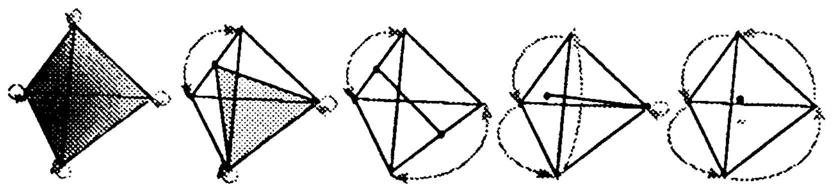

Figure 5: Fixed Points of Simplicial Maps of Δ_3

4 Fixed Points of Simplicial Maps Show Evasiveness

In the previous lecture we showed that the simplicial complex associated with a monotone non-evasive function is contractable. In this lecture we present the following argument.

Standard fixed point theorems in topology tell us that a continuous function mapping a contractable polyhedron into itself has a fixed point. On the other hand, the invariance of a monotone function f under a permutation π of the inputs implies that the geometric realization of the permutation maps $\widehat{\mathcal{K}_f}$ into itself, and thus has a fixed point if f is non-evasive. For f a monotone bipartite graph property or a monotone graph property on graphs with a prime power number of nodes, we characterize the possible fixed point sets of such mappings to show that if f is non-trivial, no fixed point can exist, so that f is evasive.

4.1 Fixed Points of Simplicial Mappings

Suppose $\mathcal{K}$ and $\mathcal{K}'$ are (abstract) simplicial complexes. Then $\varphi : V(\mathcal{K}) \rightarrow V(\mathcal{K}')$ is a *simplicial map* provided $\forall X \in \mathcal{K} \Rightarrow \varphi(X) \in \mathcal{K}'$, that is, provided φ preserves the property of being in the complex when applied to sets. Such a map yields a continuous linear map $\widehat{\varphi} : \widehat{\mathcal{K}} \rightarrow \widehat{\mathcal{K}'}$ by mapping the vertices of $\widehat{\mathcal{K}}$ in correspondence with φ, and mapping convex combinations of the vertices to the corresponding convex combinations of their images:

simplicial map

$$\widehat{\varphi}\left(\sum_{v \in \mathcal{K}} \alpha_v \widehat{v}\right) = \sum_{v \in \mathcal{K}} \alpha_v \widehat{\varphi(v)}.$$

Note that for $x \in \widehat{\mathcal{K}}$, the representation $x = \sum_{v \in \mathcal{K}} \alpha_v \widehat{v} : \sum_v \alpha_v = 1$ is unique. The set $\Delta_x = \{v : \alpha_v \neq 0\}$, a simplex of $\mathcal{K}$, is called the *support simplex* of x, and is, of course, also unique.

support simplex

So given a simplicial map $\varphi : \mathcal{K} \rightarrow \mathcal{K}$, which for our purposes we will assume is one-to-one, what are the fixed points? Suppose x is a fixed point with support simplex Δ. Then $\widehat{\varphi(\Delta)}$ contains x, and hence contains $\widehat{\Delta}$. Since $\varphi(\Delta)$ and Δ

enter of gravity

rbit

are the same size, it follows that $\varphi(\Delta) = \Delta$, that is φ permutes the vertices in Δ. This in turn implies that the center of gravity[11] of $\widehat{\Delta}$ is also fixed by φ.

What are the other fixed points in $\widehat{\Delta}$? If the orbits[12] of the permutation induced by φ on Δ are $H_1, \ldots, H_k$, then $\widehat{H_i}$ is a face of $\widehat{\Delta}$, with the center of gravity a fixed point. Also, any convex combinations of these centers of gravity is a fixed point.

Conversely, if $x = \sum_{v\in\Delta} \alpha_v \widehat{v}$ is a fixed point, then $x = \widehat{\varphi}(x) = \sum_{v\in\Delta} \alpha_v \widehat{\varphi(v)}$ is also a representation of x, and because the representation of x in this way is unique, there exists a permutation π of the vertices in Δ such that $\alpha_{\pi(v)}\widehat{\pi(v)} = \alpha_v\widehat{\varphi(v)}$, that is, $\varphi(v) = \pi(v)$ and $\alpha_v = \alpha_{\pi(v)}$. It follows that $\alpha_v = \alpha_{\varphi(v)}$, so that for each orbit H_i of φ on Δ, we can choose β_i so that $u \in H_i \Rightarrow \alpha_u = \beta_i$. Thus we have

$$x = \sum_{v\in\Delta} \alpha_v \widehat{v} = \sum_i \sum_{v\in H_i} \beta_i \widehat{v} = \sum_i \beta_i |H_i| w_i.$$

In other words, x is a convex combination of the centers of mass of the faces corresponding to the orbits.

To view this from a more combinatorial perspective, suppose the orbits of φ on the vertices of $\mathcal{K}$ are $H_1, \ldots, H_N$, and assume that the first t of these are those which are also simplices of $\mathcal{K}$. Let $w_i : 1 \le i \le t$ denote the center of gravity of $\widehat{H_i}$. Then each w_i is a fixed point, and any proper convex combination x of a subset $\{w_{i_1}, \ldots, w_{i_r}\} \subseteq \{w_1, \ldots, w_t\}$ is also a fixed point, provided only that the x is in fact in $\widehat{\mathcal{K}}$, that is, provided $H_{i_1} \cup \cdots \cup H_{i_r} \in \mathcal{K}$.

In sum, if fix(φ) denotes the fixed points of $\widehat{\varphi}$, then fix(φ) $= \widehat{\mathcal{H}}$, where

$$\mathcal{H} = \{\{i_1, \ldots, i_r\} : H_{i_1} \cup \cdots \cup H_{i_r} \in \mathcal{K}\},$$

and the vertices $V(\mathcal{H})$ of the simplicial complex $\mathcal{H}$ are $\{1, \ldots, t\}$, with $\widehat{i}$ taken to be the center of gravity of the face $\widehat{H_i}$.

4.2 Fixed Point Theorems

Next we give some theorems which give sufficient conditions for the existence of fixed points, and which characterize some useful properties of the fixed point sets. The first is Brouwer's fixed point theorem.

Theorem 4.1 (Brouwer) *Any continuous map of a simplex to itself has a fixed point.*

An alternate formulation of this theorem follows.

[11] The center of gravity of a face $\widehat{H}$ is $\sum_{v\in H} v/|H|$.

[12] The orbits, or cycles, of a permutation are the minimal sets of elements such that the permutation takes no element out of its set.

Theorem 4.2 *There does not exist a continuous map from a simplex to its boundary leaving the boundary fixed.*

If there were a continuous function $f : S \to S$ with no fixed point, we could construct a function $g : S \to \partial S$ leaving ∂S fixed as follows. Given $x \in S$, obtain $g(x)$ by projecting a ray from the point $f(x)$ through the point x to the boundary ∂S.

Note that for any polyhedron S, if ∂S is not contractable, then there can not exist a continuous map from S to its boundary leaving the boundary fixed.

[Also recall our earlier claim that a set is contractable iff the cone[13] formed by the set with an affinely independent point v can be mapped continuously to the set.]

cone

Theorem 4.3 (Lefshetz) *If $\widehat{\mathcal{K}}$ is contractable, then any continuous map from $\widehat{\mathcal{K}}$ to itself has a fixed point.*

4.3 Application to Graph Properties

We are finally in a position to apply these techniques to show evasiveness. Recall the previous result that a function $f : \{0,1\}^n \to \{0,1\}$ invariant under some cyclic permutation of its inputs and with $f(0) \neq f(1)$, is evasive, provided the number of inputs is prime. To start, we show that any non-trivial monotone function invariant under a cyclic permutation of the inputs is evasive.

Lemma 4.4 *Suppose $f : \{0,1\}^n \to \{0,1\}$ is a monotone, non-trivial function invariant under a cyclic permutation of its inputs. Then f is evasive.*

Proof: Assume without loss of generality that f is invariant under the permutation $\varphi(i) = i + 1 \bmod n$, and assume f is non-evasive. Consider $\mathcal{K}_f$. As shown in the previous lecture, $\widehat{\mathcal{K}_f}$ is contractable. Since f is invariant under φ, φ is a simplicial map of $\mathcal{K}_f$, so that $\widehat{\varphi}$ has a fixed point (by Lefshetz' theorem).

As discussed in the beginning of this lecture, the fixed points correspond to orbits of φ contained in $\mathcal{K}_f$. Since the only orbit of φ is $\{1, 2, \ldots, n\}$, this set must be in K_f. Thus $f(1, 1, \ldots, 1) = 0$, a contradiction. $\square$

The success of this technique hinges on our being able to characterize the orbits, and hence the fixed point set, of a permutation under which the function is invariant. The proof of the next theorem is essentially the same as the previous, except that the characterization of the orbits is trickier. Before we give the theorem, we give the Hopf index formula, an extension of Lefshetz' fixed point theorem which we need for the proof.

[13]The cone of a set with a vertex consists of the convex combinations of v with points in the set.

uler characteristic

Theorem 4.5 (Hopf Index Formula) *For φ a simplicial one-to-one mapping of $\mathcal{K}$ a contractable simplicial complex, the Euler characteristic*[14] *of the fixed point set $\widehat{\mathcal{H}}$ of $\widehat{\varphi}$ is -1.*

Theorem 4.6 (Yao) *Non-trivial monotone bipartite graph properties are evasive.*

Proof: Let $f(x_{ij} : i \in U, j \in W)$ be a non-evasive monotone bipartite graph property. Let φ be a permutation of the edges corresponding to a cyclic permutation of the vertices of W while leaving U fixed.

The fixed point set of $\widehat{\varphi}$ on $\widehat{\mathcal{K}_f}$ is characterized by:

$$\text{fix}(\varphi) = \widehat{\mathcal{H}}, \{u_{i_1}, \ldots, u_{i_r}\} \in \mathcal{H} \Leftrightarrow f(x^{\{u_{i_1}, \ldots, u_{i_r}\} \times W}) = 0$$

with the vertices of $\widehat{\mathcal{H}}$ being the centers of gravity of the faces corresponding to the orbits of φ.

The orbits of φ correspond to the nodes of U: each orbit contains all of the edges touching a single node of U, so we identify each vertex of $\mathcal{H}$ with a node of U. Since $\mathcal{K}_f$ is contractable, there are fixed points, so some edge set $\{u\} \times W$ has $f(x^{\{u\} \times W}) = 0$.[15] By the symmetry of f, therefore, any choice of u yields $f(x^{\{u\} \times W}) = 0$. The sets in $\mathcal{H}$ correspond to edge sets $\{u_{i_1}, \ldots, u_{i_r}\} \times W$, and again by symmetry either all or none of these sets for any given r are in $\mathcal{H}$. By monotonicity, then, $\mathcal{H}$ is characterized by

$$\{u_{i_1}, \ldots, u_{i_r}\} \in \mathcal{H} \Leftrightarrow r \le r_0$$

for some r_0. The Euler characteristic of $\mathcal{H}$ is thus

$$\begin{aligned} & -\binom{|U|}{1} + \binom{|U|}{2} - \cdots + (-1)^{r_0}\binom{|U|}{r_0} \\ = & -\binom{|U|-1}{0} - \binom{|U|-1}{1} + \binom{|U|-1}{1} + \binom{|U|-1}{2} - \cdots \\ = & -\binom{|U|-1}{0} + (-1)^{r_0}\binom{|U|-1}{r_0}. \end{aligned}$$

By the Hopf formula, this equals -1, which implies that $r_0 = |U|$, i.e. $U \in \mathcal{H}$, so $U \times W \in \mathcal{K}_f$, and $f(x^{U \times W}) = 0$. Thus $f \equiv 0$. $\square$

One might expect the proof to be simpler for a general graph property, which is invariant under a larger class of permutations. Unfortunately, since a general

[14] The Euler characteristic $\chi(\widehat{\mathcal{K}})$ of a polyhedron $\widehat{\mathcal{K}}$ is defined by $\chi(\widehat{\mathcal{K}}) = \sum_{x \in \mathcal{K}, x \neq \emptyset} (-1)^{|x|}$. (Recall $\mu(f)$.) The Euler characteristic is invariant under topological deformation, and is thus useful for classifying topological types. For instance, a contractable set has Euler characteristic -1.

[15] Considered as acting on the complete simplex $\Delta_{|U| \times |W|}$, $\widehat{\varphi}$ necessarily has fixed points. The question is whether any of these fixed points are in $\widehat{\mathcal{K}_f}$.

graph has more edges, the orbits of any given permutation are generally more complicated. However, when the number of nodes is a prime power, we can still characterize the orbits, and thus show evasiveness. Before we show this, we give a more general fixed point theorem.

Theorem 4.7 *Let Γ be a group of mappings of a contractable geometric complex $\widehat{\mathcal{K}}$ onto itself. Let Γ_1 be a normal subgroup*[16] *of Γ with $|\Gamma_1| = p^k$, for a prime p, and with Γ/Γ_1 cyclic. Then there exists an $x \in \widehat{\mathcal{K}}$ such that $\forall \varphi \in \Gamma, x = \widehat{\varphi}(x)$.*

normal subgroup

Note that we are no longer talking about the fixed points of a single simplicial mapping, but rather the fixed points of a group of simplicial mappings. If the orbits[17] of Γ are $H_1, \ldots, H_N$, with the first t of these in $\mathcal{K}$, and $w_i : 1 \leq i \leq t$ is the center of gravity of $\widehat{\mathcal{H}_i}$, then fix$(\Gamma) = \widehat{\mathcal{H}}$, where:

$$\{i_1, \ldots, i_r\} \in \mathcal{H} \Leftrightarrow H_{i_1} \cup \cdots \cup H_{i_r} \in \mathcal{K},$$

and $V(\mathcal{H}) = \{1, \ldots, t\}$, with $\widehat{i} = w_i$.

To see this, note first that for any $\varphi \in \Gamma$, an orbit of Γ is expressible as a disjoint union of orbits of φ, so any w_i is expressible as a convex combination of centers of mass of the faces corresponding to orbits of φ, so w_i is fixed by φ. Thus any convex combination of the w_i is fixed by each φ.

Conversely, if $x = \sum_{v \in \Delta} \alpha_v \widehat{v}$ is a fixed point of Γ, for any u, w in an orbit H_i of Γ there is a φ such that $\varphi(u) = w$ and $x = \widehat{\varphi}(x) = \sum_{v \in \Delta} \alpha_v \widehat{\varphi(v)}$, so the uniqueness of the representation of x implies $\alpha_u = \alpha_w$. Consequently we can choose $\beta_1, \ldots, \beta_t$ such that $\forall i, u \in H_i \Rightarrow \alpha_u = \beta_i$, and

$$x = \sum_{v \in \Delta} \alpha_v \widehat{v} = \sum_{H_i \subseteq \Delta} \sum_{v \in H_i} \beta_i \widehat{v} = \sum_{H_i \subseteq \Delta} \beta_i |H_i| w_i.$$

Thus the previous techniques continue to apply when we have a group of simplicial mappings. The previous theorem is exactly what we need for the proof of the next theorem:

Theorem 4.8 *Suppose f is a non-trivial monotone graph property on graphs with a prime power p^k number of nodes. Then f is evasive.*

Proof: Think of the nodes of our graph G as identified with GF(p^k).[18] Consider the linear mappings $x \mapsto ax + b$: GF$(p^k) \rightarrow$ GF(p^k) as a group Γ. Let Γ_1 be the mappings $x \mapsto x + b$, so $|\Gamma_1| = p^k$. The normality of Γ_1 follows from $(((ax+b)+b')-b)/a = x+b'/a$. The factor group Γ/Γ_1 is isomorphic to the group ~~of mappings $x \mapsto ax : a \neq 0$, i~~.e. the multiplicative group[19] of GF$(p^k) - \{0\}$,

GF(p^k)

multiplicative group

[16] A subgroup Γ_1 of Γ is normal if $\forall x \in \Gamma, x\Gamma_1 x^{-1} = \Gamma_1$.

[17] The orbits of a collection of permutations are the minimal sets invariant under every permutation.

[18] GF(p^k) is the Galois field of order p^k. For $k = 1$ this is the field of integers $0, 1, \ldots, p-1$ under arithmetic modulo p. For larger k, it is the field of polynomials over GF(p) modulo an irreducible polynomial of order k.

[19] The multiplicative group of a field is the group formed on the elements other than 0 under multiplication.

which is known to be cyclic. Thus the preceding theorem applies, and every action of Γ has a fixed point on $\widehat{\mathcal{K}_f}$. Since Γ is transitive on the edges, the only orbit of Γ consists of all of the edges. Thus if f is non-evasive, so that $\mathcal{K}_f$ is contractable and Γ has a fixed point, then $\mathcal{K}_f$ must have as an element the set of all edges, and $f \equiv 0$. $\square$

5 Non-Deterministic and Randomized Decision Trees

In this lecture we use previous results to show a $\Theta(n^2)$ lower bound on the decision tree complexity of a general non-trivial monotone graph property, and we begin discussion of decision trees for probabilistic algorithms.

5.1 Near Evasiveness of Monotone Graph Properties

The key lemma in our study so far of the decision tree complexity of monotone Boolean functions has been :

If $f : \{0,1\}^n \rightarrow \{0,1\}$ is a non-evasive, monotone function, with $f(\underline{0}) = 0$, then $\mathcal{K}_f$ is contractible.

We also noted that :

If $\mathcal{K}_f$ is contractible then $\chi(\mathcal{K}_f) = -1$ (i.e. $\sum_{S:f(S)=0}(-1)^{|S|} = 0$).

We would like to extend this theory to general (non-monotone) functions as well. However, $\mathcal{K}_f$, as defined, is not a simplicial complex in the case of a non-monotone function f. Although the AKR conjecture is known to be false for non-monotone functions, if f is weakly symmetric, $f(\underline{0}) \neq f(\underline{1}) = 1$, and the number of variables of f is prime, we have shown that f is evasive. (Theorem 2.1).

We have seen that monotone graph properties on graphs with a prime power number of nodes (theorem 4.8), or on bipartite graphs (theorem 4.6), are evasive. Next we show that for any non-trivial monotone graph property, by restricting the property to some $\Omega(n)$ size subgraph, we can obtain one of these two kinds of properties. Since $D(f)$ is at least $D(f|_R)$ for any restriction R, this will imply that any non-trivial monotone property f $D(f) = \Omega(n)$.

Theorem 5.1 *Let f be a monotone, non-trivial graph property, then $D(f) \geq cn^2$ for some positive constant c.*

Proof: Let G be a graph on n nodes. Choose a prime p such that $n/2 < p < 2n/3$ (it follows from number theoretic arguments that such a prime exists). Let S be a subset of the nodes of G such that $|S| = p$. Let K_S denote the complete graph on S, with all other $(n-p)$ nodes being isolated. Since f is monotone, and non-trivial, $f(0) = 0$. Now, there are several cases:

Case 1, $f(K_S) = 1$. [20] Let R be the restriction $x_{i,j} = 0 : (i,j) \notin S$. Then $f|_R$ is a monotone, non-trivial graph property on a graph with a prime number of nodes, and $f(G)$

$$D(f) \geq D(f|_R) = \binom{p}{2} \geq (n^2 - n)/8.$$

[20] $f(G)$ for a graph $G = (E, V)$ is shorthand for $f(X^E)$.

Case 2, $f(K_S) = 0$. In this case $f(K_{V\setminus S}) = 0$, since $K_{V\setminus S}$ is a complete graph that is smaller than K_S $(n/2 \leq p)$, and f is monotone.

Let $H = K_{V\setminus S} \cup \{S \times (V\setminus S)\}$ (Note the abuse of notation here, as we are really interested in unordered pairs).

Case 2.1, $f(H) = 1$. Let $R = \{x_{i,j} = 0 : i,j \in S\} \cup \{x_{i,j} = 1 : (i,j) \in S\setminus V\}$. Then $f|_R(0) = f(V\setminus S) = 0$, and $f|_R(1) = f(H) = 1$, and $f|_R$ is a monotone bipartite graph property on a graph with $p(n-p)$ edges. Thus $D(f) \geq D(f|_R) = p(n-p) \geq 2n^2/9$.

Case 2.2, $f(H) = 0$. Let $R = \{x_{i,j} = 1 : (i,j) \in H\}$. Then $f|_R(0) = f(H) = 0$, $f|_R 1 = f(1)$, and $f|_R$ is a monotone graph property on the subgraph induced by the vertices in S, so as in case 1 $D(f) \geq (n^2 - n)/8$.

□

5.2 Non-Deterministic Decision Trees

Recall that we defined $D_0(f)$, and $D_1(f)$, and showed that $D(f) \leq D_0(f)D_1(f)$.

Theorem 5.2 (Babai or Nisan?) *Suppose f is weakly symmetric (invariant under a transitive group Γ), then $D_0(f)D_1(f) \geq n$.*

Proof: Recall $D_0(f) = \min\{k : f = E_1 \wedge E_2 \ldots \wedge E_N, E_i = x_{i_1}^{\epsilon_{i_1}} \vee \ldots \vee x_{i_k}^{\epsilon_{i_k}}\}$. Similarly, $D_1(f) = \min\{k : f = F_1 \vee F_2 \ldots \vee F_M, F_i = x_{i_1}^{\epsilon_{i_1}} \wedge \ldots \wedge x_{i_k}^{\epsilon_{i_k}}\}$. Recall; there must be a variable, x_i, that occurs in both E_1 and F_1 (otherwise, we can force the function value to be 0, and 1, at the same time). Let $\gamma \in \Gamma$. Let E_i^γ be E_i after the action of γ. Since f is invariant under Γ, we can rewrite$f = E_1^\gamma \ldots E_N^\gamma$. Therefore E_1^γ must have a variable in common with F_1.

The crucial observation at this point is that for a transitive group Γ of mappings on a set S, the quantity $q = \#\{\gamma \in \Gamma : \gamma(x) = y\}$ is independent of x and y. This is because for any x, y, and y' we can map $\{\gamma \in \Gamma : \gamma(x) = y\}$ 1-1 into $\{\gamma \in \Gamma : \gamma(x) = y'\}$ by composing any fixed map $\gamma' : \gamma(y) = y'$ with the maps in the first set. This shows independence of y and a similar argument shows independence of x.

In fact, we can determine q by the equation (for fixed x_0)

$$qn = \sum_y \#\{\gamma \in \Gamma : \gamma(x_0) = y\} = |\Gamma|,$$

so $q = |\Gamma|/n$.

Returning to the original argument, that E_1^γ has a variable in common with F_1 for every γ means that every γ maps something from E_1 to something in

F_1, i.e. there are $|\Gamma|$ γ mapping some x from E_1 to some y from F_1. Since any given pair $x \in E_1$ and $y \in F_1$ (again abusing notation) has at most $q = |\Gamma|/n$ γ's mapping x to y, it follows that there are at least $|\Gamma|/q = n$ pairs (x, y) with $x \in E_1$ and $y \in F_1$, i.e. $|E_1||F_1| \geq n$.

Recalling that $|E_1| \leq D_0(f)$ and $|F_1| \leq D_1(f)$ finishes the argument. □

5.3 Randomized Decision Trees

The general question in complexity of randomized algorithms is "Does the ability to flip a coin add computational power?" Over the past twenty years we have learned that the answer is a definitive yes. Generally, randomization may give an algorithm the ability to avoid a few bad computation paths, and thus better its worst case behavior.

From the decision tree model, there are a number of ways to model randomized algorithms. One is that at each node, rather than definitely querying some variable, we choose which variable to query randomly according to a probability distribution dependent on the node and the previous results of random choices. For a given input the number of input bits queried is then a random variable, and the decision tree complexity is the maximum over all inputs of the expected value of the number of input bits queried.

An alternate model is that the algorithm makes all random choices in advance, and from that point on is deterministic. In this model an algorithm for deciding a property is specified as a probability distribution over all possible decision trees for the property. For a given tree T, if $\delta(x, T)$ denotes the number of input bits queried for a given input $x \in \{0,1\}^n$, and p_T denotes the probability the algorithm choosing T, then the decision tree complexity of the algorithm is

$$\max_x \sum_T p_T \delta(x, T).$$

This will be made more concrete by the following example. The example is due to Saks, Snir, and Wigderson, and gives a tree formula which has randomized decision tree complexity $o(n)$. (We have seen previously that all tree formulae have (deterministic) decision tree complexity n.)

The function $f_k : \{0,1\}^n \to \{0,1\}$, $n = 2^k$, is a tree formula. (I.e. it is defined by a formula in which each variable occurs exactly once.) We may define f_k inductively by

$$\begin{aligned} f_0(x_1) &= x_1 \\ f_{k+1}(x_1, \dots, x_n) &= \begin{cases} f_k(x_1, \dots, x_{n/2}) \wedge f_k(x_{n/2+1}, \dots, x_n) & k \text{ odd}, \\ f_k(x_1, \dots, x_{n/2}) \vee f_k(x_{n/2+1}, \dots, x_n) & k \text{ even}. \end{cases} \end{aligned}$$

That is, we take a balanced binary tree and construct a formula by labeling the leaves with the variables and labeling internal nodes of the tree alternately with "and" and "or" gates as we go up the tree.

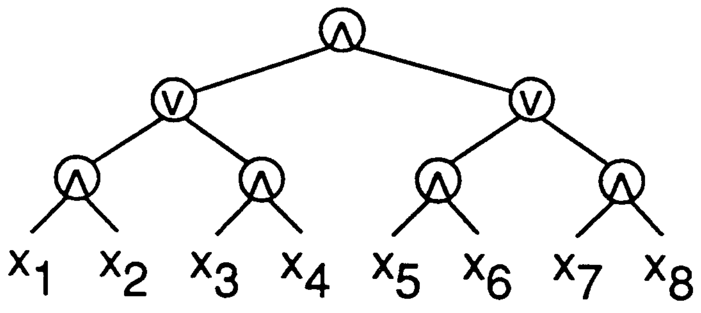

Figure 6: A Function with Randomized Complexity $o(n)$.

We saw previously that tree formulae are evasive. The evasiveness of f also follows by the weak symmetry of f and the fact that the number of variables is a prime power. Our task is to construct a randomized algorithm for f with expected decision tree complexity $o(n)$.

First, for convenience, we get rid of the asymmetry at different levels by replacing the and-gates and or-gates by nand-gates (negated and-gates). Specifically, we use $(f_1 \vee f_2) \wedge (f_3 \vee f_4) = (f_1 \overline{\wedge} f_2) \overline{\wedge} (f_3 \overline{\wedge} f_4)$ to replace all gates except possibly the gate at the root with nand-gates. If the gate at the root does not become a nand-gate, it is an and-gate, and we simply negate it. This complements f, but doesn't change the complexity of the function.

Now that we have nand-gates at all the nodes, consider the evaluation of the function. If for some nand-gate, we know one input is 0, we know the gate outputs 1, independent of the other input. Thus our randomized strategy will be to start at the root, choose one of the two inputs uniformly at random to evaluate, and recursively evaluate it. If it returns 0 we return 1, otherwise we evaluate the other input recursively, returning 1 if the other input returns 0, and 1 otherwise.

Let a_k denote the expected number of variables checked to compute f if $f(x) = 0$, and let b_k denote the expected number if $f(x) = 1$.

If $f(x) = 0$, then both inputs must be evaluated and are 1. If $f(x) = 1$, then either both inputs are 0, in which case we definitely only evaluate one input, or one input is 0, in which case we have at least a one in two chance to evaluate only one input. This yields

$$\begin{aligned} b_k &= 2a_{k-1}, \\ a_k &\leq \max\{b_{k-1}, \frac{1}{2}b_{k-1} + \frac{1}{2}(a_{k-1} + b_{k-1})\} \\ &= \frac{1}{2}b_{k-1} + \frac{1}{2}(a_{k-1} + b_{k-1}). \end{aligned}$$

We can write this as

$$\begin{aligned}\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} &= \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \\ \begin{pmatrix} a_k \\ b_k \end{pmatrix} &= \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}\begin{pmatrix} a_{k-1} \\ b_{k-1} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}^k \begin{pmatrix} 1 \\ 1 \end{pmatrix}.\end{aligned}$$

To estimate a_k and b_k from such a recurrence relation, we can examine the eigenvalues of the matrix. The action of the matrix on an eigenvector (by definition) is just to stretch the vector by the corresponding eigenvalue. Thus on repeated application of the matrix, the norm of the eigenvector with largest eigenvalue will grow most rapidly. For a vector which is not an eigenvector, we can consider it as a convex combination of eigenvectors and similarly show that with repeated application of the matrix, its norm grows no faster than that of the eigenvector with largest eigenvalue. These considerations show that if there is a single largest eigenvector λ, then after k applications of the matrix the resulting vector has norm $\lambda^k + o(\lambda^k)$ times the norm of the original vector. Thus a_k and b_k are approximately λ^k. For the above matrix, the eigenvalues are $\frac{1}{4}(1 \pm \sqrt{33})$, so that $a_k, b_k \approx \left(\frac{(1\pm\sqrt{33})}{4}\right)^k$.

For our randomized algorithm, $k = \log_2 n$, so we have

$$a_k, b_k \approx \left(\frac{(1+\sqrt{33})}{4}\right)^{\log_2 n} = n^{\log_2\left(\frac{(1+\sqrt{33})}{4}\right)} \approx n^{0.754} = o(n),$$

and we are done. (See the end of this lecture for details of the calculation.)

One might suspect that one could improve this algorithm by sampling the inputs randomly and using the result to bias the choice of which input to evaluate first towards the input with more 1's in the subtree. It turns out this doesn't help; in fact the above algorithm is essentially optimal. (Although we don't give the proof.)

In the next lecture we will begin to study the probabilistic version of the AKR-conjecture, which is that for a graph of n nodes the expected decision tree complexity of a randomized algorithm for a non-trivial graph property is $\Theta(n^2)$. A lower bound of n follows from $D_R(f) \geq D_0(f)D_1(f) \geq n$. Yao improved this lower bound to $n \log n$ and introduced some techniques which we will study. Valerie King improved the bound to $\Theta(n^{5/4})$ in her thesis, and subsequently Hajnal improved the bound to $\Theta(n^{4/3})$.

One question which arises is the complexity of probabilistic algorithms which are allowed a small probability of error. (Called a *Las Vegas algorithm*.) One can gain a little bit, for instance consider the majority function, which is 1 provided a majority of its inputs are 1. By random sampling half the inputs,

Las Vegas algorithm

say, one can compute the majority function with a small probability of error. On the other hand, there is a lower bound on the complexity of $\sqrt{D(f)}$, which leaves a large gap.

5.3.1 Details of Calculating a_k and b_k

eigenvalues

eigenvector

Following are the calculations of a_k and b_k in more detail; they may be skipped by anyone familiar with linear algebra. Let $M = \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}$. The ***eigenvalues*** of M are the values $\lambda \neq 0$ such that there exists a corresponding ***eigenvector*** x with $Mx = \lambda x$. Rewriting this as $(M - \lambda I)x = 0$, we see that the eigenvalues are those values for which $M - \lambda I$ is singular, i.e. has determinant $|M - \lambda I| = 0$.

$$\begin{aligned} \begin{vmatrix} \frac{1}{2} - \lambda & 1 \\ 2 & -\lambda \end{vmatrix} &= 0, \\ \lambda^2 - \frac{1}{2}\lambda - 2 &= 0, \\ \lambda &= \frac{1}{4}(1 \pm \sqrt{33}). \end{aligned}$$

Once we have the eigenvalues λ_1 and λ_2, let x_1 and x_2 be corresponding eigenvectors, and define the matrices

$$\begin{aligned} D &= \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \\ X &= \begin{pmatrix} x_1 & x_2 \end{pmatrix}. \end{aligned}$$

It is easy to verify that $MX = XD$, so $M = XDX^{-1}$, $M^2 = XD^2X^{-1}$, ..., and $M^k = XD^kX^{-1}$. Since

$$D^k = \begin{pmatrix} \lambda_1^k & 0 \\ 0 & \lambda_2^k \end{pmatrix},$$

we have

$$\begin{pmatrix} a_k \\ b_k \end{pmatrix} = M^k \begin{pmatrix} 1 \\ 1 \end{pmatrix} = XD^kX^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

After one determines the eigenvectors x_1 and x_2, it is an easy matter to complete this and give a closed form for a_k and b_k.

(Notes by Sigal Ar and Neal Young.)

6 Lower Bounds on Randomized Decision Trees

We begin this lecture with a proof of a basic result, Farkas' lemma. The lemma gives a necessary and sufficient condition for the existence of a solution to a set of linear inequalities. We then discuss (and prove) von Neumann's min-max theorem. The theorem gives some insight into the advantages of randomization. We then present some techniques developed by Yao applying the min-max theorem to give lower bounds on randomized decision tree complexity.

6.1 Farkas' Lemma

For a system of linear equalities, everyone knows necessary and sufficient conditions for solvability. Farkas' lemma is the analogue for systems of linear inequalities. Consider the problem "Does a system of linear inequalities

$$\sum_{j=1}^{m} a_{ij}x_j \quad \leq \quad b_i \quad (i = 1, \ldots, n) \tag{1}$$

have a solution?" Roughly, we expect this problem to be in NP, since we can exhibit an easy proof (an assignment of x) if it is. Farkas' lemma implies that it is also in co-NP, that is, if it is not solvable there is also an easy proof that it isn't.

If the system is not solvable, we will prove it by exhibiting λ_i satisfying $\sum_i \lambda_i a_{ij} = 0$ $(j = 1, \ldots, m)$, $\sum_i \lambda_i b_i < 0$, and $\lambda_i \geq 0$ $(i = 1, \ldots, n)$. This provides a proof that no x satisfies (1), because if such an x existed we would have

$$0 = \sum_j x_j \sum_i \lambda_i a_{ij} = \sum_i \lambda_i \sum_j a_{ij} x_j \leq \sum_i \lambda_i b_i < 0.$$

Lemma 6.1 (Farkas) *For any a_{ij} and b_i, $(j = 1, \ldots, m,\ i = 1, \ldots, n)$*

$$\sum_j a_{ij}x_j \leq b_i \quad (i = 1, \ldots, n) \tag{2}$$

has a solution x if and only if

$$\begin{aligned} \sum_i \lambda_i a_{ij} &= 0 \quad (j = 1, \ldots, m) \\ \sum_i \lambda_i b_i &< 0 \\ \lambda_i &\geq 0 \quad (i = 1, \ldots, n) \end{aligned} \tag{3}$$

has no solution λ.

Proof: Consider the vectors $(a_{i,1}, ..., a_{i,m}, b_i), (i = 1, ..., n)$. Consider the cone of these vectors. (3) says exactly that $y^* = (0, ..., 0, -1)$ is in this cone. Suppose that (3) has no solution, so y^* is not in the cone. Then there exists a separating hyperplane $H = \{y : \sum_j y_j h_j = 0\}$ such that y^* is on one side of H and the cone is on the other, i.e. $\sum_j y_j^* h_j < 0$ and $a_{i1}h_1 + \cdots + a_{in}h_n + b_i h_{n+1} \geq 0 \quad (i = 1, ..., m)$. The first condition says that $h_{n+1} \geq 0$, and the second says that $\sum_j a_{ij}h_j/h_{n+1} \geq -b_i \quad (i = 1, ..., n)$. Thus letting $x_j = -\frac{h_j}{h_{n+1}}$, we have a solution to 2. □

6.2 Von Neumann's Min-Max Theorem

Recall our second characterization of a randomized algorithm via decision trees. For a function $f : \{0,1\}^n \to \{0,1\}$, on an input x, an algorithm $\mathcal{A}$ chooses a deterministic algorithm for f with decision tree T according to some probability distribution p (independent of x!), and then runs the deterministic algorithm on x.

On a given input x, having chosen a particular tree T, $\mathcal{A}$ (deterministically) takes some complexity $\delta(T, x)$ to compute $f(x)$. The expected complexity for $\mathcal{A}$ on x is given by $\sum_T p_T \delta(T, x)$. We are interested in the worst case expected complexity for $\mathcal{A}$ (i.e. the adversary chooses x to maximize the complexity): $\max_x \sum_T p_T \delta(T, x)$. Finally, if $\mathcal{A}$ is optimal, it minimizes this worst-case complexity, and thus takes complexity

$$D_R(f) = \min_p \max_x \sum_T p_T \delta(T, x).$$

We can view this process as a game, in which we choose p (determining $\mathcal{A}$), and then the adversary, knowing our choice, chooses x. To play the game, we run $\mathcal{A}$ on x. Our goal is to minimize the expected complexity; the adversary's goal is to maximize it.

zero-sum game

For this situation (called a ***zero-sum game***, since we lose exactly what our opponent gains), there is a general theorem, von Neumann's min-max theorem. We have a game, defined by a matrix M, for two players — the row player and the column player. The game is played as follows. The column player chooses a column c and the row player chooses a row r. The column player then pays M_{rc} to the row player.

The column player wants to minimize M_{rc}, and the row player wishes to maximize it.

To make this concrete, assume we are playing a game (see figure 6.2) where each player chooses either 1 or 2. If we choose the same as the adversary, we pay her 1. Otherwise we pay nothing. What should our strategy be? Suppose our strategy is to choose 1. After playing the game several times, with the adversary beating us every time, we begin to suspect that the adversary knows

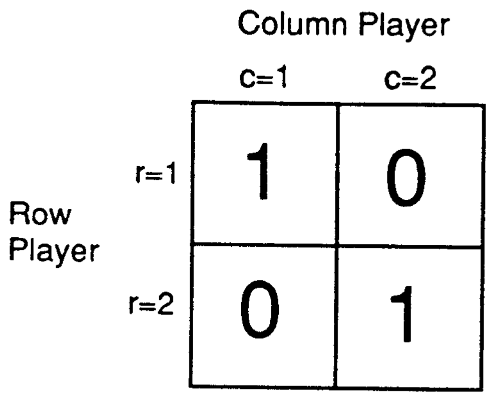

Figure 7: A Simple Zero-Sum Game

our strategy and has used that knowledge to beat us. What can we do, given that the adversary may know our strategy and use that information to try to beat us?

Von Neumman's key observation is that we can use randomization to negate the adversary's advantage in knowing our strategy. For our simple game, if our strategy is to choose 1 or 2 randomly, each with probability 1/2, then no matter what strategy the adversary picks, we have an expected loss of at most 1/2.

More generally, in any zero-sum game we have a randomized (also called a *mixed*) strategy which negates the adversary's advantage in knowing our strategy. To make precise the notion of "negating the advantage," we consider turning the tables, so that she chooses her strategy first, and then we choose ours. Then provided we choose the randomized strategy that negates her advantage in the first situation, and she chooses the randomized strategy which negates our advantage in the second situation, we will expect to do just as well in the first situation as the second. Formally,

Theorem 6.2 (Von Neumann's min-max theorem) *For any zero-sum game M,*

$$\min_p \max_q q^{\mathrm{T}} M p = \max_q \min_p q^{\mathrm{T}} M p.$$

(Note that p and q range over all probability distributions of columns and rows, respectively.)

We start with some observations. In the min-max theorem, in the inner max and min, randomization is not important. That is, (if e_j denotes the jth unit vector in a vector space implicit in the context)

$$\forall p, \max_q p^{\mathrm{T}} M p = \max_j e_j^{\mathrm{T}} M p,$$

and similarly for the inner term on the right hand side. This is because, for a fixed p, $q^{\mathrm{T}}Mp$ is a linear function of q, and thus is maximized at one of the vertices e_j of the probability space.
Proof: Obviously,

$$\forall q_0, p_0, \max_q q^{\mathrm{T}} M p_0 \geq q_0^{\mathrm{T}} M p_0 \geq \min_p q_0^{\mathrm{T}} M p.$$

Thus

$$\min_{p_0} \max_q q^{\mathrm{T}} M p_0 \geq \max_{q_0} \min_p q_0^{\mathrm{T}} M p.$$

The other direction is not so easy. We suppose that $\exists t$:

$$\forall p : (p \geq 0, \sum_j p_j = 1) \ \max_j e_j^{\mathrm{T}} M p \geq t$$

and we want to show that

$$\exists q : q \geq 0, \sum_j q_j = 1, \forall i, q^{\mathrm{T}} M e_i \geq t.$$

This will follow almost immediately from Farkas' lemma.

Suppose that $\nexists q : q \geq 0, \sum_j q_j = 1, \forall i, q^{\mathrm{T}} M e_i \geq t$. Farkas' lemma implies that there exists $\lambda = (\lambda_1, .., \lambda_n)$, $\mu = (\mu_1, ..., \mu_m)$, and α such that:

$$\begin{aligned} \lambda^{\mathrm{T}} M + \mu + \alpha \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} &= \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, && (4) \\ \sum_i \lambda_i t + \alpha &> 0, && (5) \\ \lambda_i, \mu_j &\geq 0. \end{aligned}$$

Note that α is not constrained to be non-negative and that in (5) the expression is constrained to be positive, rather than negative. These are essentially trivial variations from the standard form of Farkas' lemma. The first is because the inequality corresponding to α is in fact an equality, and the second is because the unsatisfiable constraints are of the form $\cdots \geq t$, rather than $\cdots \leq t$.

Letting $p = \lambda / \sum_i \lambda_i$, (4) and (5) imply

$$p^{\mathrm{T}} M \leq -\frac{\alpha}{\sum_i \lambda_i} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \leq t \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix},$$

which contradicts our assumption. □

6.3 Lower Bounds

The discussion before the proof, in our original context, means that once we choose p, determining $\mathcal{A}$, the adversary can choose a specific input x, rather than a distribution of inputs, to give the worst case expected behavior for $\mathcal{A}$. Alternatively, if the adversary chooses an input distribution first, then we can do our best by subsequently choosing the best deterministic algorithm, rather than a randomized algorithm, for this input distribution. Thus the min-max theorem implies that if the adversary can choose a distribution for which she can force any deterministic algorithm to have an expected complexity of at least t, then for every randomized algorithm there is an input such that the expected complexity of the algorithm on that input is at least t.

To show a lower bound on the decision tree complexity of any randomized algorithm, then, it suffices to show the same bound on the complexity of all deterministic algorithms for some fixed input distribution. This is the technique we will use.

As a starting point, recall that $D_R(f)$ denotes the minimum expected decision tree complexity of a randomized algorithm for f. An almost trivial observation is

$$D_R(f) \geq \max\{D_0(f), D_1(f)\}. \tag{6}$$

This follows because when $f(x) = i$, *any* algorithm must look at at least $D_i(f)$ bits of x before it can be sure $f(x) = i$. Last lecture we showed that for (non-trivial) weakly symmetric f, $D_0(f)D_1(f) \geq n$. Thus $D_R(f) \geq \sqrt{n}$.

Next we give a lemma which uses the min-max theorem to give a better lower bound for certain types of functions.

Lemma 6.3 (Yao) *Let $f : \{0,1\}^n \to \{0,1\}$. Suppose we can partition $\{1,...,n\}$ into $S_1,...,S_r$ so that $|S_i| \geq t$ and*

$$f(x) = \begin{cases} 0 & \text{if } \forall i \, |\{j : x_j = 0\} \cap S_i| \geq t, \\ 1 & \text{if } \exists i \, |\{j : x_j = 0\} \cap S_i| = 0. \end{cases}$$

Then $D_R(f) \geq \Omega\left(\frac{n}{t}\right)$.

(Note that the condition on f leaves some values of f unspecified. Also note that this bound doesn't follow immediately from (6).)

Proof: We give an input distribution on x and give a lower bound on the expected complexity of any deterministic algorithm on this input distribution.

To generate x, choose t indices j uniformly at random from each S_i and set $x_j = 0$. Set the remaining $x_{j'} = 1$. Then $f(x) = 0$, and for an algorithm to verify this it must query an x_j with the value 0 from each S_i. How many queries must the algorithm expect to make before finding such an x_j in a given S_i? Since the t x_j with value 0 were chosen randomly, we can view the algorithm as sampling randomly (without replacement) from a set of size k until it finds one of the t x_j chosen to be 0. The expected time for this is $\Omega(k/t)$. By linearity of expectation, the expected time to find a 0 in each S_i is thus at least $\frac{n}{k}\Omega\left(\frac{k}{t}\right) = \Omega(n/k)$. □

We will apply this lemma to lower bound the randomized complexity of monotone bipartite graph properties on $2n$ nodes (n in each part). The complexity of such properties is not quite solved. The trivial bound $D_R(f) \geq \sqrt{\#\text{inputs}}$ gives a lower bound of n. This was first improved to $n \log n$ by Yao, who introduced the general techniques for the problem. Valerie King improved the bound to $n^{5/4}$, and subsequently Hajnal improved the bound to $n^{4/3}$.

6.3.1 Graph Packing

graph packing

Our problem is closely related to the problem of *graph packing*. Two graphs G_1 and G_2 on n nodes can be packed if (possibly after relabeling the vertices) the edge sets of the graphs don't overlap. If G_1 and G_2 pack, then $G_1 \subseteq \overline{G_2}$.

Suppose G_1 is a minimal G in a graph property $\mathcal{P}_f = \{G : f(G) = 1\}$, and G_2 is a minimal G such that $\overline{G} \in \overline{\mathcal{P}}$. Observe that G_1 and G_2 don't pack. Furthermore, if any two G_1 and G_2 don't pack, we have $|E(G_1)|\,|E(G_2)| \geq n^2$ by essentially the same argument that showed $D(f) \geq D_0(f)D_1(f)$. More interestingly, one can show

$$d_{\max}(G_1)d_{\max}(G_2) \geq n/2.$$

An instance of this is that if a graph G has $d_{\min}(G) > n/2$, then there exists a hamiltonian cycle C. That is, if $d_{\max}(\overline{G}) < n/2$, then $\overline{G}$ and any cycle C pack.

6.3.2 Yao's $d_{\max}/\overline{d}$ Lemma

Next we will in some sense specialize lemma 6.3 to the case of a monotone bipartite graph property. We will choose a minimal graph G for the property, and show how to construct a containing graph G' such that we can partition a large subset of the edges of G' into disjoint sets such that if we remove edges from the subsets, the property holds if we leave any set untouched, but fails if we remove at least a small number from each set. Lemma 6.3 will apply to show essentially that any randomized algorithm has to check many edges in each set.

Lemma 6.4 (Yao) *Let f be a (non-trivial) monotone bipartite graph property on bipartite graphs with $2n$ nodes, the two parts U and W each having n nodes, and let $\mathcal{P} = \{G : f(G) = 1\}$.*

If G is a graph in $\mathcal{P}$ with minimum $d^U_{\max}(G)$ (i.e. minimum maximum degree over vertices in U) and $\overline{d}^U(G)$ is the average degree of vertices of G in U, then

$$D_R(f) \geq \Omega(1)\frac{d^U_{\max}(G)}{\overline{d}^U(G)}.$$

Proof: Of the $G \in \mathcal{P}$ with minimum $d_{\max}$, choose one with the fewest number of maximum degree vertices, so that if any graph has fewer vertices of degree $d_{\max}$ (and no higher degree vertices) it is not in $\mathcal{P}$.

Assume $d_{\max} \geq 4\overline{d}$, otherwise the bound is trivial. Assume also that the vertices are labeled so that vertex 0 is a maximal degree vertex in U, and vertices $1, ..., n/2$ are in U and have degree at most $h = 2\overline{d}$.

We form the containing graph G' by adding edges from vertex i to the neighbors of vertex 0 and vertex $i+1$. G' has two essential properties. First, if we delete any $2h+1$ edges out of each vertex $0, 1, ..., n/2$, we destroy the property $\mathcal{P}$, because we reduce the number of vertices in U with degree $d_{\max}$ by 1.

Second, if for some i (possibly 0), for each vertex $j = 0, 1, ..., i-1, i+1, ..., n/2$, we delete the edges from j into $\Gamma_0 - \Gamma_i - \Gamma_{i+1}$, we preserve $\mathcal{P}$. This is because we can permute the vertices of U so that the permuted graph contains the original graph G. We do this as follows: shift vertex 0 to vertex 1, vertex 1 to vertex 2, ..., and vertex i to vertex 0.

To apply lemma 6.3, we define $S_i = \{i\} \times (\Gamma_0 - \Gamma_i - \Gamma_{i+1})$ (for $i = 1, ..., n/2$), $S_0 = \{0\} \times (\Gamma_0 - \Gamma_1)$, and $S = \cup_i S_i$.

Then if we obtain f' by restricting the domain of f to graphs which agree with G' on edges not in S, f' is a function of $|S| \geq n\,(d_{\max} - 2h)$ variables. If we start with G' and within each partition S_i delete $2h+1$ edges, f' becomes 0, but if we start with G' and delete edges within S leaving at least one partition complete the function stays 1. Thus lemma 6.3 implies that

$$D_R(f) \geq D_R(f') \geq \Omega\left(\frac{n\,(d_{\max} - 2h)}{2h+1}\right) \geq \Omega\left(n\frac{d_{\max}}{\overline{d}}\right).$$

□

7 Randomized Decision Tree Complexity, continued

In this lecture we continue giving lower bounds on randomized decision tree complexity, combining the various bounds we have developed to show a lower bound of $\Omega(n^{5/4})$ on the randomized decision tree complexity of any non-trivial, monotone, bipartite graph property.

The general method is to choose both a minimal graph with the property and a graph whose complement is minimal in the complementary property, and then to use the fact that the two graphs don't pack to get constraints on the maximum and average degrees of the two graphs, and finally to apply Yao's lemma from last lecture to bound the randomized complexity via the constraints on the degrees.

7.1 More Graph Packing

Recall that graphs G_1 and G_2 pack if after some relabeling of the nodes of G_1 the two graphs have no common edges. That is, $\exists G_1', G_1' \cong G_1, G_1' \subseteq \overline{G_2}$.

Recall that $d^U_{\max}(G)$, for a bipartite graph $G = (U, W, E)$, denotes the maximum degree of a vertex in U in G, and $\bar{d}(G)$ denotes $|E|/n$, the average degree of a vertex in G. For this lecture, we will restrict our attention to bipartite graphs $G_1 = (U, W, E_1)$ and $G_2 = (U, W, E_2)$ with equal size parts, i.e. $|U| = |W| = n$, so that the average degree in each part is the same.

We will show the following lemma.

Lemma 7.1 (Bollobas-Eldridge) *If*

$$d^U_{\max}(G_1)d^W_{\max}(G_2) + d^U_{\max}(G_2)d^W_{\max}(G_1) \le n$$

then G_1 and G_2 pack.

There is also a non-bipartite version of this lemma:

Lemma 7.2 *For two graphs G_1 and G_2, if*

$$d_{\max}(G_1)d_{\max}(G_2) \le n/2$$

then G_1 and G_2 pack.

The proof of the second lemma, which we omit, is similar to that of the first lemma, which we give. While there is no gap in the $n/2$ term; the conjecture is that $(d_{\max}(G_1) + 1)(d_{\max}(G_2) + 1) < n$ also gaurantees packing.

Proof: (Lemma 7.1.) Suppose G_1 and G_2 don't pack, and we have relabeled the vertices of G_1 so as to minimize the number of overlapping edges. There is some overlapping edge (u, w). Consider swapping the labels of u and a vertex

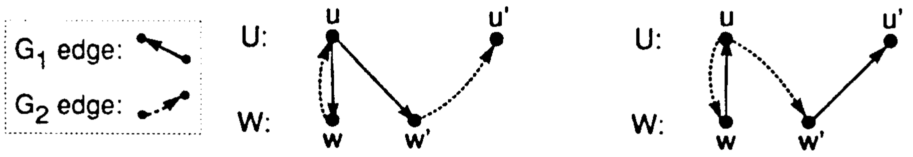

Figure 8: Swapping the Labels of u and u': Two Cases.

$u' \in U$ in G_1. With the current labeling, there is at least one overlapping edge out of u and u', so after the swap this must also be the case.

This entails that either an edge $(u, w') \in E_1$ will be mapped on to an edge $(u', w') \in E_2$ by the swap, or that an edge $(u', w') \in E_1$ will be mapped onto an edge $(u, w') \in E_2$ by the swap. This means that every $u' \neq u$ is reachable from u either by following an edge in G_1 and then an edge in G_2 or by following an edge in G_2 and then an edge in G_1.

There are at most $d^U_{\max}(G_1)d^W_{\max}(G_2) - 1$ paths of the first kind, (one of the candidates leads back to u), and similarly at most $d^U_{\max}(G_2)d^W_{\max}(G_1) - 1$ of the second kind. Thus

$$d^U_{\max}(G_1)d^W_{\max}(G_2) + d^U_{\max}(G_2)d^W_{\max}(G_1) \geq n + 1.$$

□

Graph packing captures many graph theoretic notions, for instance if G_1 is a K_d (a graph with a d-clique and $n - d$ isolated nodes), and G_2 consists of d n/d cliques, then G_1 and G_2 pack, and this is equivalent to saying the vertices of G_1 are d-colorable[21] with each color coloring n/d nodes. In fact it can be proved[22] that if G_1 is of maximal degree $d - 1$, then G_1 and G_2 pack.

d-colorable

On the other hand, if G_1 instead consisted of a $d + 1$-clique and $n - d - 1$ isolated nodes, then G_1 and G_2 would not pack, since G_1 would not be d-colorable. Since the product of the maximum degrees for this pair of graphs is $d(n/d - 1) = n - d$, which for $d = n/2$ is $n/2$, this gives a tight lower bound for lemma 7.1 and the conjecture.

7.2 Application of Packing Lemma

So we have this packing lemma, which is fairly straightforward; how can we use it?

Given a monotone bipartite graph property $\mathcal{P}_f$ on bipartite graphs $G = (U, W, E)$ with $|U| = |W| = n$, choose G_1 to have the lexicographically smallest

[21] The vertices of a graph are d-colorable if one can color them with d colors so no edge touches two vertices of the same color. The edges of a graph are d-colorable if one can color them with d-colors so no vertex touches two edges of the same color.

[22] Hajnal and Steverédi?

degree sequence[23] of vertices in U, so that G_1 is minimal, no graph in $\mathcal{P}$ has lesser $d^U_{\max}$, and of those with equal $d^U_{\max}$, none has fewer vertices of this degree. Similarly, choose G_2 lexicographically smallest with $\overline{G_2} \notin \mathcal{P}$.

degree sequence

We know G_1 and G_2 don't pack, otherwise $\overline{G_2} \subseteq G_1 \in \mathcal{P}$.

We have several bounds on $D_R(f)$:

$$\begin{aligned} D_R(f) &\geq \bar{d}(G_1)n, & (7)\\ D_R(f) &\geq c\frac{d^U_{\max}(G_1)}{\bar{d}(G_1)}n. & (8)\end{aligned}$$

Bound (7) says that D_R is at least the number of edges in G_1, which is trivial since G_1 is minimal. Bound (8) is Yao's lemma, shown in the previous lecture. It also holds if G_2 replaces G_1 and/or W replaces U, a fact we shall use.

Bound (7) implies that $\bar{d}(G_1) \leq D_R/n$, and thus (8) implies

$$d^U_{\max}(G_1) \leq \frac{D_R}{cn}\bar{d}(G_1) \leq \frac{1}{c}\left(\frac{D_R}{n}\right)^2,$$

and similarly for W and G_2 possible replacing U and G_1, respectively. Since G_1 and G_2 don't pack, lemma 7.1 implies that

$$\frac{2}{c^2}\left(\frac{D_R}{n}\right)^4 \geq d^U_{\max}(G_1)d^W_{\max}(G_2) + d^U_{\max}(G_2)d^W_{\max}(G_1) > n,$$

which in turn implies that $D_R \geq \left(\frac{c^2n^5}{2}\right)^{1/4} = \Omega(n^{5/4})$.

Thus any non-trivial bipartite graph property has $D_R = \Omega(n^{5/4})$.

7.3 An Improved Packing Lemma

To improve this result, we need an improved packing lemma:

Lemma 7.3 *Let G_1, G_2 be bipartite graphs. If*

$$\begin{aligned} d^U_{\max}(G_1)\bar{d}(G_2) &< \frac{n}{100}, & (9)\\ d^U_{\max}(G_2)\bar{d}(G_1) &< \frac{n}{100}, & (10)\end{aligned}$$

and

$$d^W_{\max}(G_1), d^W_{\max}(G_2) < \frac{n}{1000\log n}, \qquad (11)$$

then G_1 and G_2 pack.

[23]The degree sequence of a graph is the list of degrees of the vertices of the graph, from largest to smallest.

(The condition (11) is a technical condition, needed for the proof but not truly a restriction. In particular, if (11) is violated, we will see that Yao's lemma gives an immediate lower bound of $\Omega(n^{3/2}/\sqrt{\log n})$ on D_R.)
Proof: This proof is somewhat more complicated, we sketch the proof. In particular, we omit some final computations.

We have $G_1 = (U_1, W_1, E_1)$ and $G_2 = (U_2, W_2, E_2)$. We will assume the above conditions, and show that if we fix a random relabeling f of W_1, with non-zero probability there is a relabeling g of U_1 so that G_1 relabeled by f and g shares no edges with G_2.

In spirit the idea is initially similar to the previous packing lemma. There we showed that from the standpoint of a given vertex u, if after ruling out neighbors of neighbors of u there was a vertex left, we could swap the labels of u and the vertex and possible reduce the number of edge overlaps. Thus we showed roughly that the product of the maximum degree in U and maximum degree in W was at least n.

Here, since the vertices of W_1 have been randomly mapped onto the vertices of W_2, the neighbors of u in G_1 are mapped onto an essentially random set of size at most $d_{\max}(G_1)$ in W_2. Since the set in W_2 is essentially random, we will be able to show (using the technical condition) that the size of its neighbor set is

$$c|W_2|\bar{d}(G_2) = cd_{\max}(G_1)\bar{d}(G_2) = cn/100$$

with probability at most $\frac{1}{2n}$. Thus with probability at least 1/2, for each u there will be less than $n/2$ u' ruled out as possible images under g. We will also show that the existence of g is equivalent to the existence of a perfect matching connecting each u with a possible image u', and thus the existence of $n/2$ possible images of each u is sufficient to guarantee the existence of g.

So fix a relabeling f of W_1 uniformly at random. When will there be a g relabeling U_1 so that no edges are shared? The constraint is that if an edge (u, v) is in E_1, then the edge $(g(u), f(v))$ is not in E_2. Thus for each $u \in U_1$ we must find a $u' \in U_1$ such that $N_{G_2}(u') \cap f(N_{G_1}(u))$ is empty. ($N_G(v)$ denotes the neighboring vertices of v in G.) The only additional constraint is that each u have a unique such u'.

In other words, if we define a bipartite graph $H = (U_1, U_1, F)$, where

$$F = \{(u, u') : N_{G_2}(u') \cap f(N_{G_1}(u)) = \emptyset\},$$

then g exists iff F has a perfect matching.

The Frobenius-Konig-Hall theorem states that a bipartite graph $G = (U, W, E)$ has a perfect matching iff for every vertex set $X \subseteq U$ we have $|N(X)| \geq |X|$. We don't use this in full generality, rather we use a consequence. Namely, if $d_{\min}(G) \geq n/2$, then G has a perfect matching. This follows from the FKH theorem as follows: any set X violating is of size at greater than $n/2$. But then any vertex in W has some edge into X, since $U - X$ is not big enough to contain all the edges out of any vertex in W. Thus all vertices in W are neighbors of X.

(Just for fun, note that we can also use the previous lemma to show this. Namely if $d_{\text{min}}(G) \geq n/2$, then $\overline{G}$ (with $d_{\text{max}}(\overline{G}) \leq n/2$) and a perfect matching (with $d_{\text{max}}(G) = 1$) pack.)

Now F is a random graph, but not in the usual sense. We will argue that with non-zero probability the minimum degree of F is at least $n/2$, so that it has a perfect matching, and g exists.

To bound the minimum degree of H from below; we ask how many edges from a vertex u can be excluded. An edge (u, u') is excluded if there is a $(u, w) \in E_1$ with $(u', f(w)) \in E_2$. The idea is that the the number of such (u, w) is bounded by $d^U_{\text{max}}(G_1)$, while for a given w the number of such (u', w) is around $\overline{d}(G_2)$, so that for a given u, the number of excluded u' (which must be at least n) is at most around the product of these two. (By reversing the roles of G_1 and G_2, we can bound the number of edges (u, u') into a given u' which are excluded, thus ensuring $d_H(u')$ is also at least $n/2$ for vertices in the second part of H.)

By the definition of F,

$$\begin{aligned} \#\{u' : (u, u') \notin H\} &\leq |N_{G_2}(f(N_{G_1}(u)))| \\ &\leq \sum_{u' \in f(N_{G_1}(u))} d_{G_2}(u'). \end{aligned} \tag{12}$$

Thus the probability that $d_H(u) < n/2$ (for u in the first part of the bipartite graph) is bounded by the probability that (12) is greater than $n/2$. (The case for the second part is similar, and we omit it.) The point is that $f(N_{G_1}(u))$ is essentially a random subset of W_2 of size at most $d^U_{\text{max}}(G_1)$, so the sum of the degrees of vertices in W_2 should be bounded by $O(\overline{d}(G_2) d^U_{\text{max}})$ with high probability. In the full proof one shows that for a given u the probability of $D_H(u) \geq n/2$ is at most $\frac{1}{2n}$, so that the probability of all u having degree less than $n/2$, and thus of a matching, and the consequent g, existing, is at least $1/2$.

Here we show exactly what computations we are leaving out: If we define

$$\omega_i = \frac{d_{G_2}(w_i)}{n\overline{d}(G_2)} \quad (w_i \in W_2)$$

then $\sum_i \omega_i = 1$, $\omega_i \geq 0$, and we want to bound the probability that

$$\sum_{i \in S} \omega_i > \frac{1}{2\delta},$$

where $\delta = \overline{d}(G_2)$ and S is a set chosen uniformly at random from sets of some size at most $d^U_{\text{max}}(G_1)$, which is bounded by $\frac{n}{100\delta}$ by hypothesis.

The average value of the ω_i is $1/n$, so the expected value of the sum is $\frac{1}{100\delta}$. Thus unless the ω_i are highly concentrated, which the technical condition

$d^W_{\max}(G_1) < \frac{n}{1000 \log n}$ prevents, the condition will hold. We omit the details of the computation. □

With this improved packing lemma, and the conditions (as before)

$$\frac{D_R}{n} \geq \overline{d}(G_1), \tag{13}$$

$$\frac{D_R}{n} \geq c\frac{d^U_{\max}(G_1)}{\overline{d}(G_1)} \tag{14}$$

(and the corresponding conditions with W and G_2 possibly replacing U and G_1, respectively), we can show an improved bound. (Recall that the first condition is essentially the trivial lower bound on D_R, while the second is Yao's lemma.)

Specifically, if the technical condition $d^W_{\max}(G_1) < \frac{n}{1000 \log n}$ (or any of the equivalent technical conditions) of the improved packing lemma are violated, then by conditions (13) and (14) $\frac{D_R^2}{n^2} > \Omega\left(\frac{n}{\log n}\right)$, so $D_R = \Omega(n^{3/2}/\sqrt{\log n})$.

Otherwise (as G_1 and G_2 don't pack) one of the other conditions is violated. We assume without loss of generality that it is (9): $d^U_{\max}(G_1)\overline{d}(G_2) < \frac{n}{100}$. Together with the above two conditions, this gives

$$\left(\frac{D_R}{n}\right)^3 \geq c\overline{d}(G_2)d^U_{\max}(G_1) \geq \frac{cn}{100}.$$

Thus $D_R = \Omega(n^{4/3})$.

It seems possible that this bound could be pushed a bit higher, perhaps to $n^{3/2}$. Currently this is the best lower bound known, and the best upper bounds known are $\Omega(n^2)$.

8 Randomized Complexity of Tree Functions — Lower Bounds

For any non-trivial monotone graph property $\mathcal{P}_f$ on graphs with n nodes we have seen that $D(f) = \Omega(n^2)$, $D_R(f) = \Omega(n^{3/2})$.

In this lecture we discuss tree functions — functions with formulas in which each variable occurs exactly once. We already know that for any tree function f, $D(f) = n$, and we previously saw a tree function f_0 (represented by a complete binary tree with alternating and and or-gates) with $D_R(f_0) = O(n^{0.75\ldots})$.

We show a lower bound on $D_R(f)$ for tree functions, which we use to deduce that

- $D_R(f_0) = \Omega(n^{0.75\ldots})$, and
- $D_R(f) = \Omega(n^{0.51})$ for any tree fn. f.

8.1 Generalized Costs

The most natural thing to consider for proving a lower bound on the complexity of a tree function $f(x,y) = g(x) \circ h(y)$ (with $x \in \{0,1\}^{n-i}$, $y \in \{0,1\}^i$, and $\circ \in \{\wedge, \vee\}$) is a top-down induction. Unfortunately we don't know how to get this to work.

Instead, Saks and Wigderson have looked at a bottom-up induction, in which a gate with two immediate inputs is replaced by a single input. First f is expressed as $f(x,y,w) = f'(x \circ y, w)$ (with $x \in \{0,1\}$, $y \in \{0,1\}$, $w \in \{0,1\}^{n-2}$, and $\circ \in \{\wedge, \vee\}$), and then a lower bound on f is given by a corresponding lower bound on $f'(v,w)$.

For this technique to work, we need to keep track of the fact that discovering v, which represents $x \circ y$, is somehow more expensive than just querying a bit. To do this, we generalize our notion of cost. We associate two costs $c_0(x_i)$ and $c_1(x_i)$ with each variable x_i which represents a bit of the input to f. The cost c_0 represents the cost to discover that x_i is 0, while the cost c_1 represents the cost to discover that x_i is 1. With such a cost function c, we define

$$D_R(f,c) = \min_p \max_x \sum_T p_T \delta(T,x,c),$$

where $\delta(T,x,c) = \sum_{i \in S, x_i=0} c_0(x_i) + \sum_{i \in S, x_i = 1} c_1(x_i)$, with S the set of variables queried by T on input x.

So we are given a function f with a set of costs c. To show a lower bound on $D_R(f,c)$ we choose the function f' so $f(x,y,w) = f'(x \circ y, w)$, and we choose a set of costs c' for the inputs of f' such that we can show $D_R(f,c) \geq D_R(f',c')$, thus inductively generating a lower bound for $D_R(f,c)$. We will assume that $\circ = \wedge$; the case $\circ = \vee$ is symmetric.

So, leaving the choice of c' unspecified as yet, we have f, c, f', and c'. We want to show $D_R(f,c) \geq D_R(f',c')$. The min-max theorem says $D_R(f,c)$ is also equal to

$$\max_q \min_T \sum_x q_x \delta(T,x,c),$$

that is, we can also obtain $D_R(f,c)$ by choosing the worst input distribution, and then the best deterministic algorithm for that distribution. Thus to show $D_R(f',c')$ is at most $D_R(f,c)$, we will assume a worst case distribution q^* of inputs to f', and we show that there exists a T' for f' such that $\sum_x q^*_x \delta(T',x,c') \leq D_R(f,c)$.

So we have the worst case distribution q^* for f', and we want to show the existence of a T' that does well on q^*. We will map q^* to a distribution q on the inputs to f, so that there exists an algorithm T^* for f such that $\sum_x q_x \delta(T^*,x,c) \leq D_R(f,c)$. We know such a T^* exists because q is at worst the worst-case distribution for f, in which case there still exists an algorithm with expected cost exactly $D_R(f,c)$. We will then construct T' based on T^*, so that their expected costs on their respective distributions can be correlated.

The basic idea will be that $T'(v,w)$ will mimic $T^*(x,y,w)$ for some choice of x and y such that $x \wedge y = v$. When T^* checks a variable in w, T' will do the same. When T^* checks x or y, T' may or may not check v.

What should the costs c' be? For variables other than v, c' will agree with c. We will wait to determine $c'_0(v)$ and $c'_1(v)$, choosing them as large as our proof techniques will allow.

What about the distribution q? We define $q(1,1,w) = q^*(1,w)$, $q(0,1,w) = p_x q^*(0,w)$, and $q(1,0,w) = p_y q^*(0,w)$, where p_y and $p_x = 1 - p_y$ will be determined later.

What about T'? Define T'_y on input (v,w) to mimic T^* on $(1,v,w)$ and T'_x on input (v,w) to mimic T^* on $(v,1,w)$. Define T' on input (v,w) to run T'_y with probability p_y and T'_x with probability p_x. That T' is a randomized strategy is no problem; since q^* is fixed one of the two deterministic strategies T'_y or T'_x will be at least as good.

We need to find the constraints on $c'_1(v)$ and $c'_0(v)$ which will allow us to show that the expected cost of T^* is at least that of T'. For this it suffices to show that the cost of $T^*(v,w)$ for any v and w is at most p_y times the cost of $T^*(1,v,w)$ plus p_x times the cost of $T^*(v,1,w)$.

8.2 The Saks-Wigderson Lower Bound

Now that we have the form of our argument, the rest is essentially a matter of checking cases. We should note that although most of the choices we have made above are straightforward, there is one choice which in fact anticipates in a clever way what we will need in the remaining part of the proof. In particular, we have chosen T' to run T'_x or T'_y with probability p_x or p_y, respectively, and, not coincidentally, we have chosen the distribution q to map $(0, w)$ to $(1, 0, w)$ or $(0, 1, w)$ with probability p_x or p_y, respectively, as well. This choice bears some consideration.

Returning to our argument, we want to find the conditions under which the cost of $T'(v, w)$ is at most p_y times the cost of $T^*(1, v, w)$ plus p_x times the cost of $T^*(v, 1, w)$. We consider the various cases for T^*, v, x, and y.

If $v = 1$, this reduces to the cost of $T'(1, w)$ being at most the cost of $T^*(x = 1, y = 1, w)$. If T^* queries neither x nor y, then this is clear, since T'_x, T'_y, and T^* query exactly the same variables. Otherwise, if T^* queries only x, then T^* pays $c_1(x)$ while T' pays an expected cost of $p_x c'_1(v)$ (the costs to query variables in w are again the same); if T^* queries only y, then T^* pays $c_1(y)$ while T' pays an expected cost of $p_y c'_1(v)$; if T^* queries both x and y then T^* pays $c_1(x) + c_1(y)$ while T' pays $c'_1(v)$. Thus T^* will pay at least what T' pays provided

- $p_x c'_1(v) \leq c_1(x)$, and
- $p_y c'_1(v) \leq c_1(y)$.

The case $v = 0$ is a bit more complicated. In this case, we want the cost of $T'(0, w)$ to be at most p_y times the cost of $T^*(1, 0, w)$ plus p_x times the cost of $T^*(0, 1, w)$. If neither queries x or y for this w then this is clear. Otherwise, both query x first or both query y first. We consider the case when x is queried first, the other case being symmetric.

There are then two cases, depending on whether or not $T^*(1, 0, w)$ queries y as well as x. ($T^*(0, 1, w)$ will not query y, since T^* is optimal and knows the value of $x \wedge y$ after querying x.) First assume that only x is queried by $T^*(1, 0, w)$. Then with probability p_y the cost to T' is the cost to $T^*(1, 0, w)$ minus $c_1(x)$, and with probability p_x the cost to T' is the cost to $T^*(0, 1, w)$ minus $c_0(x)$ plus $c'_0(v)$. Thus we are fine provided

- $-p_y c_1(x) - p_x c_0(x) + p_x c'_0(v) \leq 0$.

Next assume that x and y are queried by $T^*(1, 0, w)$. Then with probability p_y the cost to T' is the cost to $T^*(1, 0, w)$ minus $c_1(x)$ minus $c_0(y)$ plus $c'_0(v)$, and with probability p_x the cost to T' is the cost to $T^*(0, 1, w)$ minus $c_0(x)$ plus $c'_0(v)$. Then we are fine provided

- $p_y(-c_1(x) - c_0(y) + c'_0(v)) + p_x(-c_0(x) + c'_0(v)) \leq 0$.

Collecting all of these inequalities, and the symmetric inequalities for y queried first, we have that $D_R(f, c) \geq D_R(f', c')$ provided that $c'_0(v)$ and $c'_1(v)$ satisfy the constraints:

$$\begin{aligned} c'_1(v) &\leq c_1(x)/p_x, \\ c'_1(v) &\leq c_1(y)/p_y, \end{aligned}$$

$$\begin{aligned}
c_0'(v) &\leq p_y c_1(x)/p_x + c_0(x), \\
c_0'(v) &\leq p_x c_1(y)/p_y + c_0(y), \\
c_0'(v) &\leq p_y(c_1(x) + c_0(y)) + p_x c_0(x), \\
c_0'(v) &\leq p_x(c_1(y) + c_0(x)) + p_y c_0(y).
\end{aligned}$$

Choosing $c_1'(v) = c_1(x) + c_1(y)$ forces $p_x = \frac{c_1(x)}{c_1(x)+c_1(y)}$ and $p_y = \frac{c_1(y)}{c_1(x)+c_1(y)}$ and yields

Theorem 8.1 (Saks-Wigderson) *Let f be a tree function with binary $\wedge$ and $\vee$ gates. Then $D_R(f) \geq \max\{l^0(f), l^1(f)\}$, where*

$$\begin{aligned}
l^0(x_i) = l^1(x_i) &= 1, \\
l^1(g \wedge h) &= l^1(g) + l^1(h), \\
l^0(g \wedge h) &= \min\{l^0(g) + l^1(h), l^1(g) + l^0(h), \frac{l^1(g)l^0(g) + l^0(h)l^1(h) + l^1(g)l^1(h)}{l^1(g) + l^1(h)}\}, \\
l^0(g \vee h) &= l^0(g) + l^0(h), \\
l^1(g \vee h) &= \min\{l^1(g) + l^0(h), l^0(g) + l^1(h), \frac{l^0(g)l^1(g) + l^1(h)l^0(h) + l^0(g)l^0(h)}{l^0(g) + l^0(h)}\}.
\end{aligned}$$

Applying this to the function f_0 (the alternating and/or function mentioned at the beginning of the lecture) shows that the upper bound for that function is in fact tight.

Rafi Heiman and Avi Wigderson generalize this theorem to tree functions with arbitrary fan-in gates to show a lower bound of $\Omega(n^{0.51})$ for the randomized decision tree complexity for an arbitrary tree function. See *Randomized vs. Deterministic Decision Trees — Complexity for Read Once Boolean Functions* by Rafi Heiman and Avi Wigderson.

(Lecture by Rafi Heiman.)

Index