

On the Complexity of the Pancake Problem

Fuxiang Yu
Department of Computer Science
State University of New York
Stony Brook, NY, USA

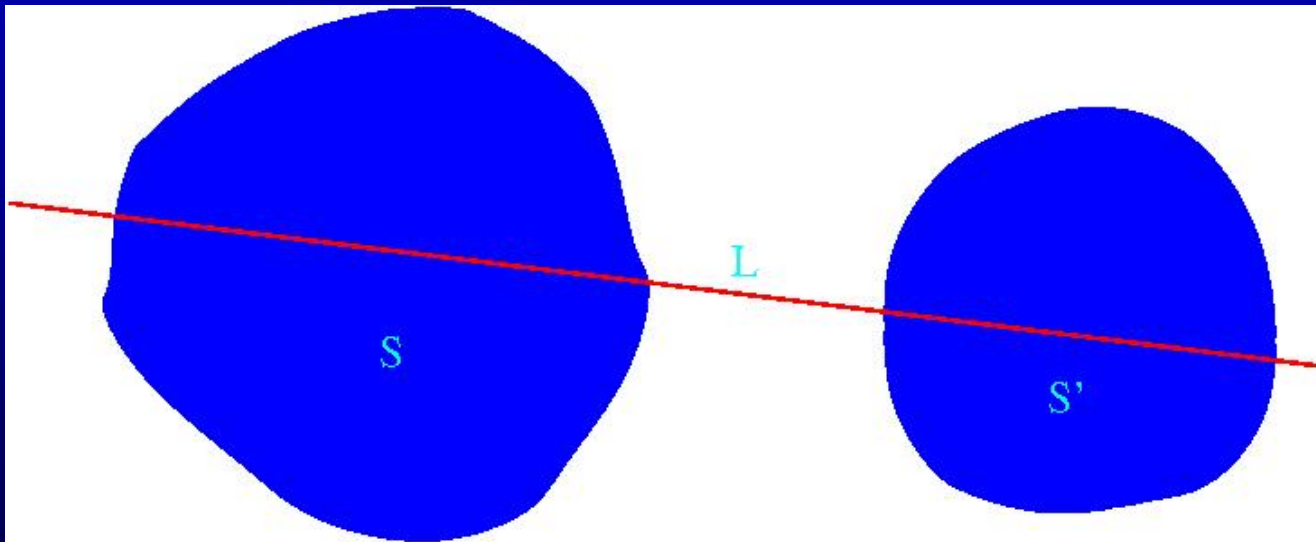
October 28, 2006

Outline of the talk

- The pancake theorem.
- Related work.
- The oracle Turing machine model.
- Our study:
 - ★ Our results.
 - ★ Proofs.
 - ★ The intermediate value theorem.

The pancake problem

Given two sets (pancakes) S and S' in \mathbb{R}^2 . There exists a line L that simultaneously bisects them.



Suppose that S and S' have some polynomial-time structures. What is the complexity of finding the line L ?

The pancake problem: related work

Two versions of the pancake problem have been studied in computational geometry.

- S and S' are two point sets in \mathbb{R}^2 , and the measure is the number of points. The complexity is $O(n)$. (Edlsbrunner et al. [1986], Lo et al. [1992])
- S and S' are polygons. The complexity is $O(n)$. (Stojmenovic [1991], Abbott et al. [2005])

Now our setting is more complicated: S and S' are polynomial-time approximable.

The oracle Turing machine model

- Discrete NP theory briefly.
- The oracle Turing machine model.

Discrete NP Theory

P : Class of sets $\subseteq \{0, 1\}^*$ accepted by polynomial-time *deterministic* Turing machines. E.g., PRIME, LP, ...

NP : Class of sets $\subseteq \{0, 1\}^*$ accepted by polynomial-time *nondeterministic* Turing machines. E.g., SAT, VC, IP, TSP, FACTORING, ...

One big question: $P = NP$?

The oracle Turing machine model of Ko and Friedman

We use the oracle Turing machine model of Ko and Friedman [1982], for two reasons:

- It's based on the Turing machine, so it is closely related to discrete complexity.
- The complexity is easily defined on this model.

In this model,

- A number or function is computable if approximations can be computed.
- The complexity measure is the number of bit operations (to output an approximation with a given precision)

Representations of real numbers

Let $\mathbb{D}_n := \{m/2^n : m \in \mathbb{Z}\}$ and $\mathbb{D} = \cup \mathbb{D}_n$. A number in \mathbb{D} is called a dyadic number.

- The Cauchy function representation. A function $\phi : \mathbb{N} \rightarrow \mathbb{D}$ represents a real number x if
 - (1) For all n , $\phi(n) \in \mathbb{D}_n$.
 - (2) For all n , $|\phi(n) - x| < 2^{-n}$.
- The general left cut representation. For any Cauchy function representation ϕ of x , the left cut L_ϕ associated with ϕ is $\{d \in \mathbb{D}_n : d \leq \phi(n), n \in \mathbb{N}\}$.

These two representations are equivalent (on the P level):

$$\phi(n) = \max(\mathbb{D}_n \cap L_\phi).$$

Computable real numbers

- A real number x is computable if a Cauchy function representation ϕ of x is a computable function (or equivalently, if a left cut L_ϕ is a recursive language).
- A real number x is polynomial-time computable if a Cauchy function representation ϕ of x is polynomial-time computable (or equivalently, if L_ϕ is polynomial-time computable).

Computable real numbers with other complexity can be defined similarly.

Example: the number π is polynomial-time computable.

- Many “ π ” formulas can be used to show this result.
- We use the Bailey-Borwein-Plouffe formula:

$$\pi = \sum_{k=0}^{\infty} 16^{-k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right).$$

Open question: Is π *real time* computable? That is, is π linear time computable?

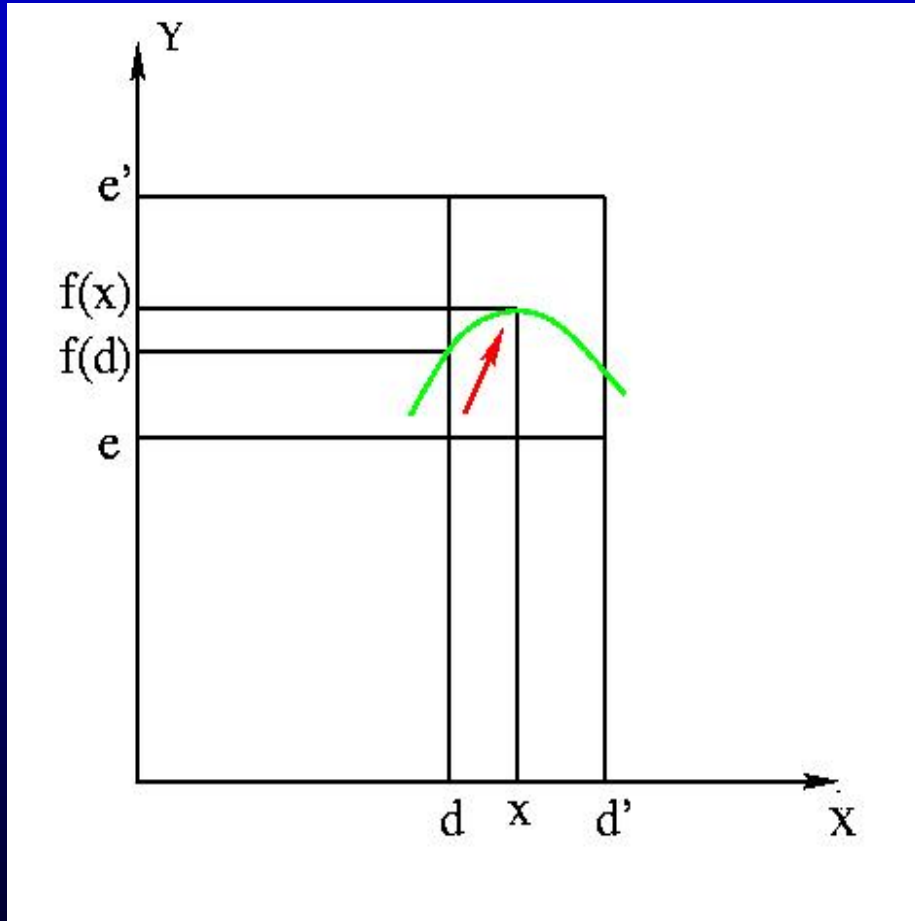
Computable real functions: general rules

- Functions have *polynomial modulus of continuity*.
 $\exists k(|x - y| < 2^{-n^k} \Rightarrow |f(x) - f(y)| < 2^{-n})$.

Computable real functions: general rules

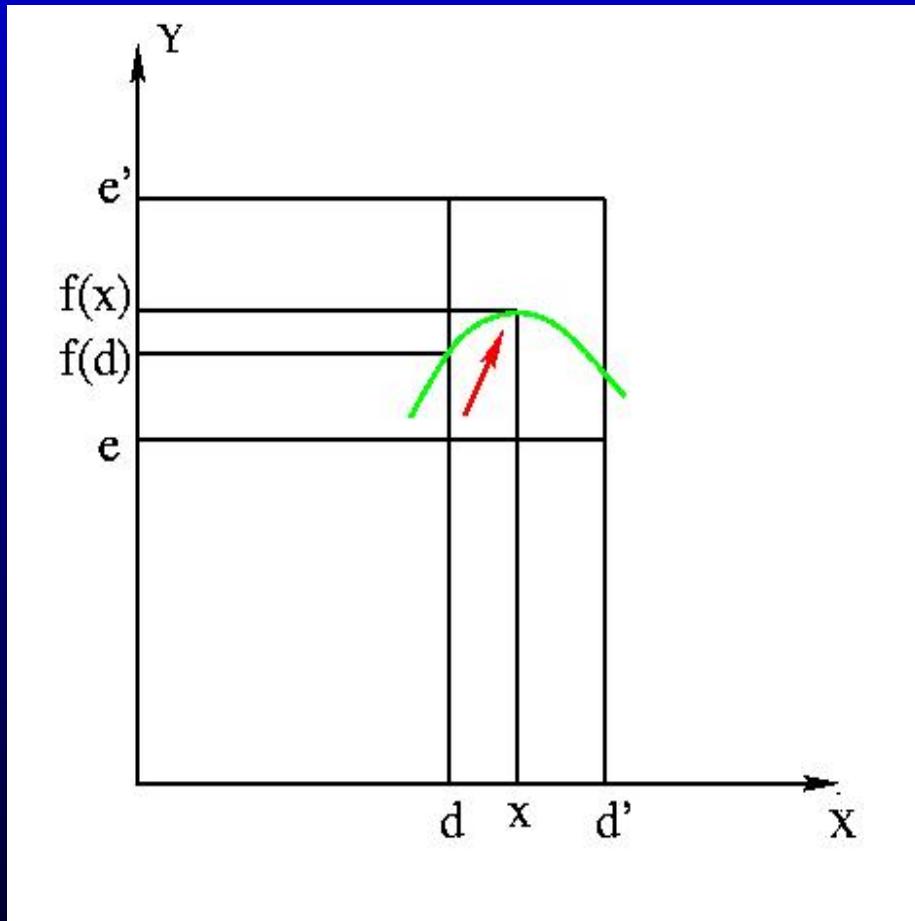
- Functions have *polynomial modulus of continuity*.
 $\exists k(|x - y| < 2^{-n^k} \Rightarrow |f(x) - f(y)| < 2^{-n})$.
- Discrete computational devices can compute approximations to values of functions at dyadic points.

How to compute $f(x)$



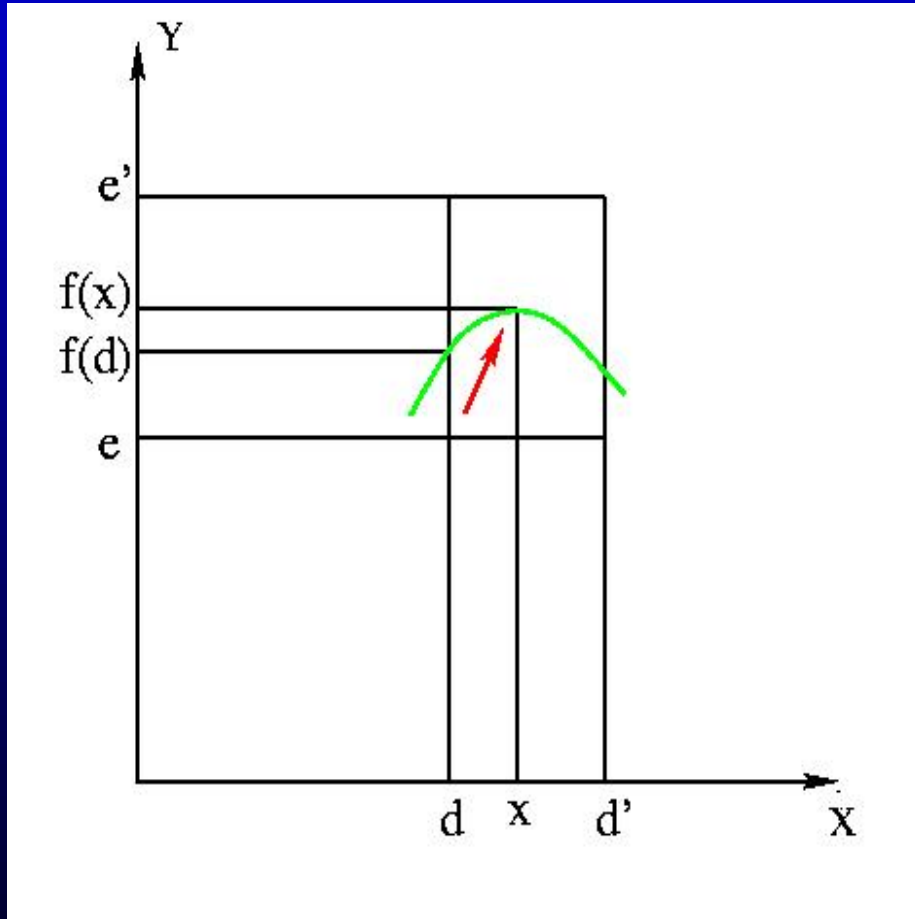
- Oracle gives a dyadic $d \approx x$;

How to compute $f(x)$



- Oracle gives a dyadic $d \approx x$;
- Discrete computational device outputs $e \approx f(d)$;

How to compute $f(x)$



- Oracle gives a dyadic $d \approx x$;
- Discrete computational device outputs $e \approx f(d)$;
- $e \approx f(x)$.

Complexity of continuous functions

The complexity of f depends on the discrete computational device M that computes f .

- If $M \in P$, then $f \in P$.
- If $M \in LOGSPACE$, then $f \in LOGSPACE$.
- If $M \in NC$, then $f \in NC$.

Example: $f(x) = e^x$ on $[0, 1]$ is polynomial-time computable

Idea of proof:

- $e^x = \sum_{k=0}^{\infty} \frac{1}{k!} x^k$.
- Cut the tail: $e^x \approx \sum_{k=0}^n \frac{1}{k!} x^k$.
- Approximate each term and sum up.

Definition of polynomial-time approximable sets

This concept was introduced by Chou and Ko [1995].

- We use a sequence $\{S_n\}$ of sets to approximate a set S .
- There exist a polynomial p and an polynomial-time oracle Turing machine M such that M decide whether $\mathbf{d} \in \mathbb{D}_{p(n)}^2$ is in S_n . S_n is the union of squares of side $2^{-p(n)}$ whose centers are those $\mathbf{d} \in \mathbb{D}_{p(n)}^2 \cap S_n$.
- The Lebesgue measure of the error set $E_n(M) = (S_n - S) \cup (S - S_n)$ is no more than 2^{-n} (i.e., M only makes mistakes in a small set).

For short, we use P -approximable sets instead.

Definition of polynomial-time recognizable sets

This concept was also introduced by Chou and Ko [1995].

- We use a sequence $\{S_n\}$ of sets to approximate a set S .
- There exist a polynomial p and an polynomial-time oracle Turing machine M such that M decide whether $\mathbf{d} \in \mathbb{D}_{p(n)}^2$ is in S_n . S_n is the union of squares of side $2^{-p(n)}$ whose centers are those $\mathbf{d} \in \mathbb{D}_{p(n)}^2 \cap S_n$.
- $E_n(M) = (S_n - S) \cup (S - S_n) \subseteq \{\mathbf{z} : \text{dist}(\mathbf{z}, \partial S) < 2^{-n}\}$ (i.e., M only makes mistakes around the boundary).

For short, we use P -recognizable sets instead.

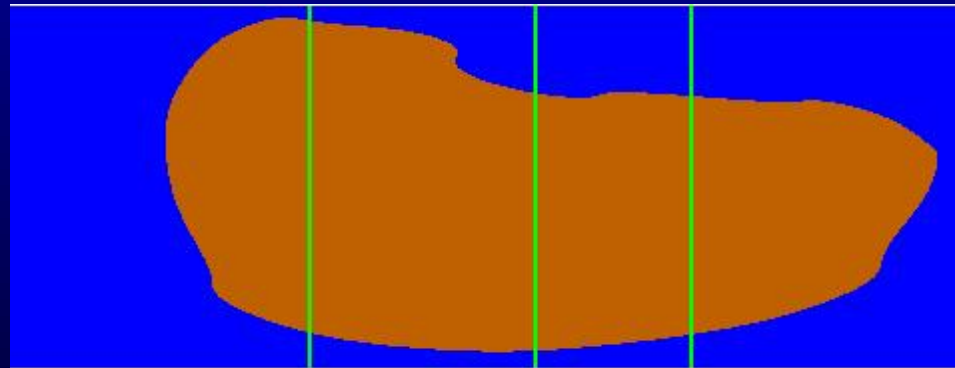
The pancake problem

We study the following sub problems.

- Bisecting one set at a given direction.
- Bisecting simultaneously two sets.

Bisecting one set S at a given direction

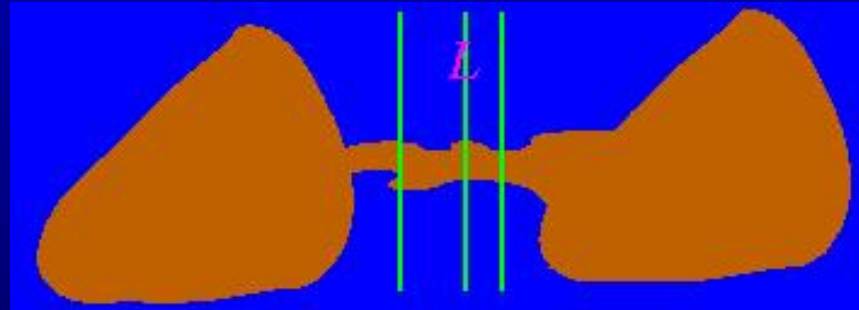
- At a given direction, there exists a line L (called a **bisector**) that cuts S into two parts of equal area.
- One method to find L is **binary search**.



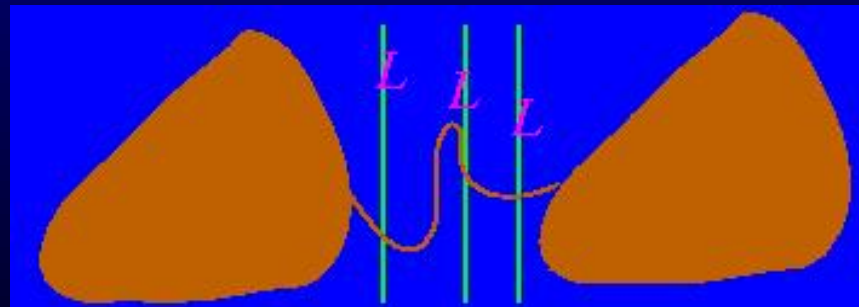
Thickness

How quick to find the bisector L depends on how **thick** S is around L .

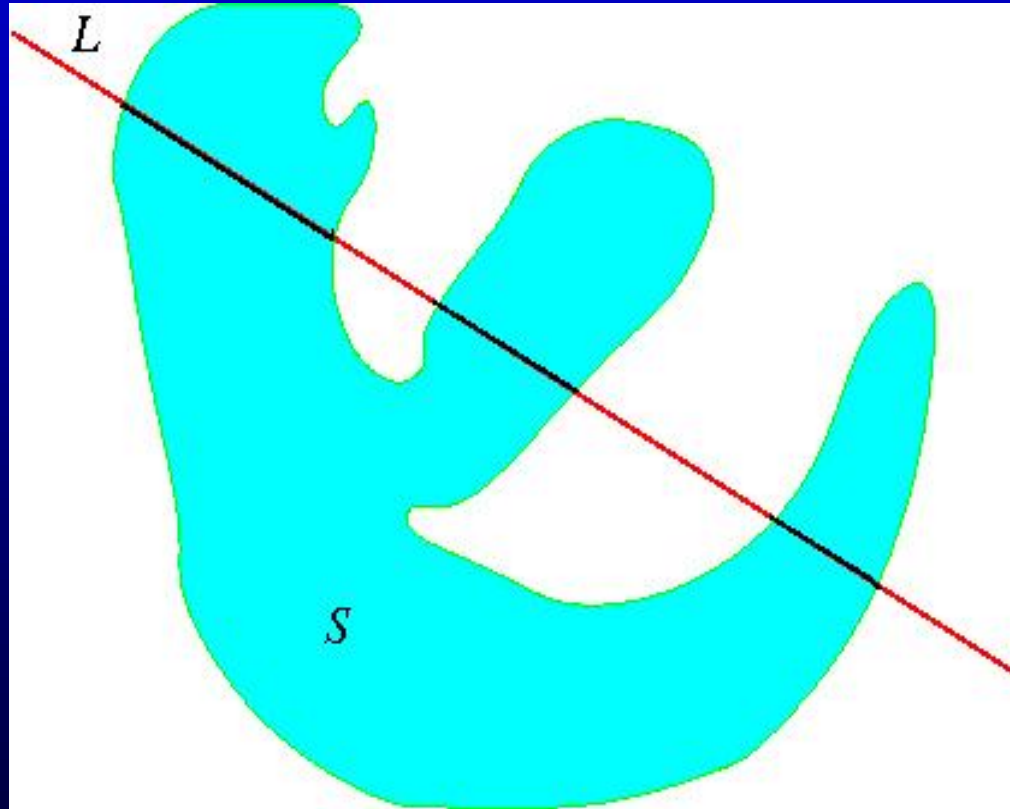
- A small positive thickness makes it harder to find L .



- A zero thickness makes L not unique.



Thickness



$$thk_S(L) = \text{length}(L \cap S) = \sum \text{length}(\text{black line segments}).$$

Thickness: mathematical definition

- Let S be a bounded set in \mathbb{R}^2 . A line L defined by $x = a$ will divide S into two parts, with the left part denoted $S_{x \leq a}$ and the right part $S_{x \geq a}$. We define the *thickness* of S at L , denoted $thk_S(L)$, as follows:

$$thk_S(L) = \liminf_{\delta \rightarrow 0} \frac{area(S_{x \leq a + \delta}) - area(S_{x \leq a})}{\delta}.$$

- For a line L at an angle $\alpha \in [0, \pi)$, we can also define $thk_S(L)$ in a similar way. More precisely, we rotate S and L about the origin by an angle $\pi/2 - \alpha$ to obtain S' and L' , respectively, and define $thk_S(L) = thk_{S'}(L')$.

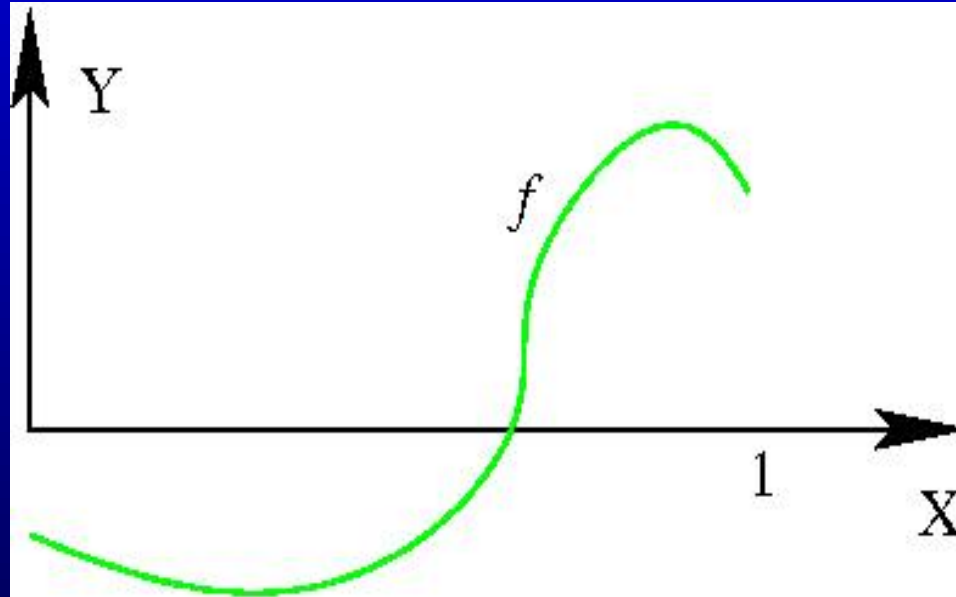
Bisecting a set at a given direction

It can be arbitrarily hard to find a line L that bisects a P -approximable set S at a given direction (e.g., parallel to the y -axis) if we do not require a positive thickness around L .

- Let $b \in (0, 1)$ be a computable real number. There exists a P -approximable/recognizable set $S \in [0, 1]^2$, such that the line defined by $x = b$ is the unique bisector of S at angle $\pi/2$; furthermore, for any line L' defined by $x = b'$, where $b' \in (0, 1) - \{b\}$, $thk_S(L') > 0$.

This is related to the **intermediate value theorem**.

The intermediate value theorem

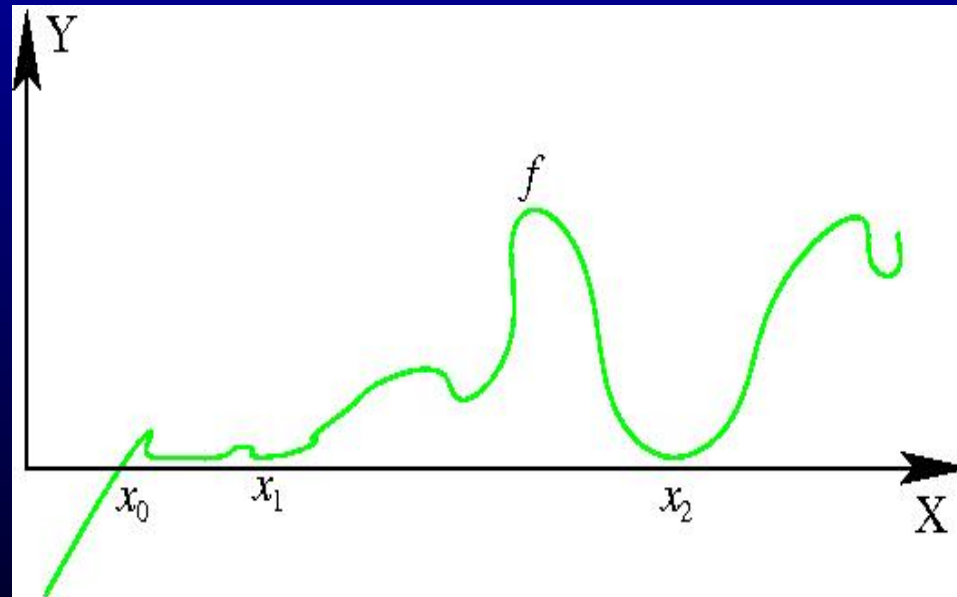


Math form: If $f : [0, 1] \rightarrow \mathbb{R}$ is a continuous function such that $f(0)f(1) < 0$, then there exists $x \in (0, 1)$ such that $f(x) = 0$. (We call x a zero of f .)

The intermediate value theorem (cont.)

Computability: f has at least one computable zero.

Complexity: This zero, even if it is unique and f is polynomial-time computable, can be arbitrarily hard to compute. (Ko [1991])



The intermediate value theorem (cont.)

Complexity: This zero is polynomial-time computable if f is polynomial-time computable and has a **polynomial inverse modulus**. (Ko [1991])

The intermediate value theorem (cont.)

Complexity: This zero is polynomial-time computable if f is polynomial-time computable and has a **polynomial inverse modulus**. (Ko [1991])

Relation with our problem: a positive thickness implies a polynomial inverse modulus.

Bisecting a set at a given direction: a positive thickness around the bisector

In the following, $(a) \Rightarrow (b) \Rightarrow (c)$:

(a) $FP = \#P$.

(b) For any P -approximable set $S \subseteq [0, 1]^2$ that has a positive thickness $W > 0$ around the unique vertical bisector L defined by $x = b$, the real number b is polynomial-time computable.

(c) $FP_1 = \#P_1$.

Upper bound: $\#P$. Lower bound: $\#P_1$.

$\#P$: Counting classes

$\#P$ is an important class introduced by Valiant [1979].

- $\#P$ is the class of functions that count the number of accepting paths of nondeterministic polynomial-time Turing machines.
- Discrete examples: $\#SAT$, $Perm$, $\#VC$. (Valiant [1979] et al.)
- Continuous example: Integration. (Friedman [1984], Ko [1986])

$FP_1, \#P_1, P_1, \dots$: **Unary classes**

- P and NP are classes of sets $\subseteq \{0, 1\}^*$; and FP and $\#P$ are classes of functions defined on $\{0, 1\}^*$.
- P_1 and NP_1 are classes of sets $\subseteq \{0\}^*$; and FP_1 and $\#P_1$ are classes of functions defined on $\{0\}^*$.
- Unary classes are closely related to real numbers. (Ko [1991])

Bisecting simultaneously two sets: the Borsuk-Ulam Theorem

The pancake theorem is also related to the Borsuk-Ulam Theorem (besides the intermediate value theorem).

Math form: If f is a continuous function from a circle C to \mathbb{R} , then there exists a pair of antipodal points x and x^* (i.e., $x = -x^*$) such that $f(x) = f(x^*)$.

Computability: at least one of such x is computable.

Complexity: It is arbitrarily hard to compute such an x even if f is polynomial-time computable and x is the unique one that satisfies the condition.

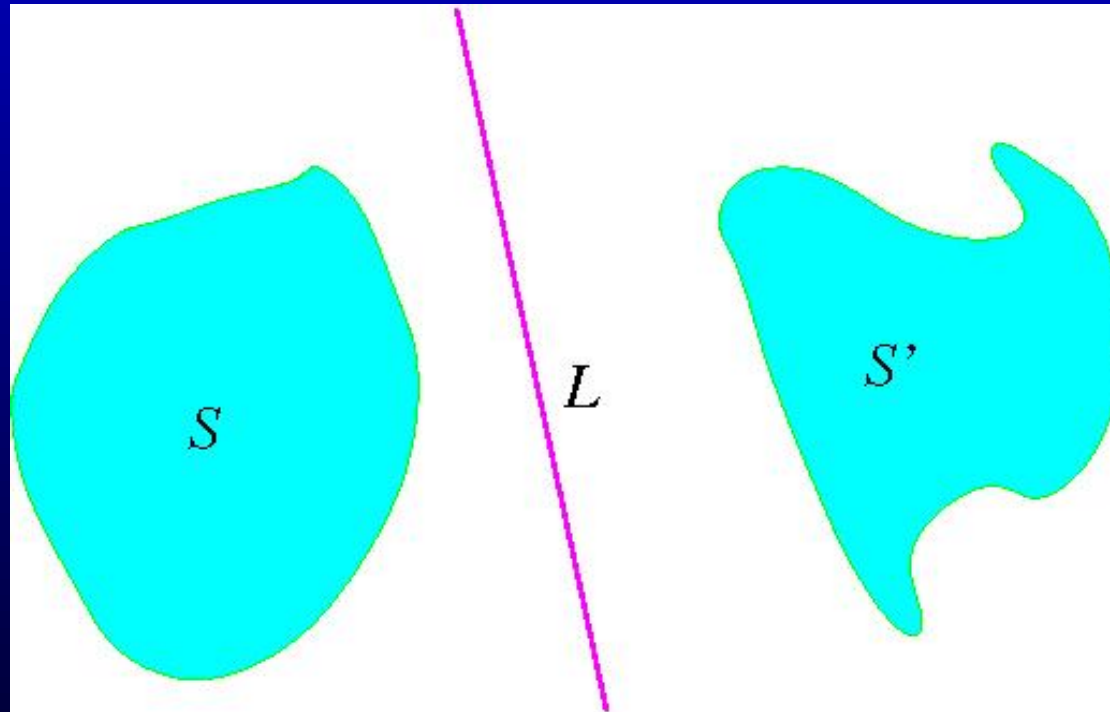
Bisecting simultaneously two sets

It is arbitrarily hard to compute the common bisector L of two P -approximable sets S and S' , even if L is unique.

- Let $\alpha \in [0, \pi)$ be a computable real number. Then there exist two convex P -approximable/recognizable Jordan domains S and S' that have only one common bisector L , and the angle from the positive x -axis to L is α .

Bisecting simultaneously two sets

We say two sets S and S' **linearly separable** if there exists a line L such that S and S' are on the different sides of L .



Bisecting simultaneously two sets

The linearly separable condition of two sets will reduce the complexity. In the following, $(a) \Rightarrow (b) \Rightarrow (c)$:

(a) $FP = \#P$.

(b) For any two linearly separable bounded P -approximable sets S and S' with positive thickness around all bisectors, the unique line L that bisects simultaneously the two sets is polynomial-time computable.

(c) $FP_1 = \#P_1$.

Ideas of proof of the last theorem

- For any angle α , there exists a line L_α that bisects S .
- Line L_α divides S' into two parts. We denote the area difference of these two parts $g(\alpha)$.
- The linearly separable condition assures that $g(\alpha)$ has a polynomial inverse modulus around its zero.
- The condition $FP = \#P$ assures that $g(\alpha)$ is polynomial-time computable. Together with the above condition that $g(\alpha)$ has a polynomial inverse modulus around its zero α_0 , α_0 is polynomial-time computable.