Chapter 3: The Tournament Divide and Conquer Technique

Aditi Goswami Daphne Liu

December 5, 2006

Outline: Lecture 1

1. Introduction
2. Definitions
   P-sel
   P/poly
3. Tools
   Tournaments
   Superloser Set Theorem
4. Main Results
   P-sel $\subseteq$ P/poly
   P-sel $\subseteq$ P/quadratic

Chapter Objectives

- Divide and conquer
  Binary search
  Tournament theory result
- Tournament theory result is an upper bound on the nonuniform complexity of semi-feasible sets (Sec. 3.1)
- *NP* machines cannot find unique satisfying assignments to satisfiable formulas unless the polynomial hierarchy collapses (Sec. 3.3)

Semi-feasible Problems: Motivation

- P – the class of feasible problems; the class of sets with polytime membership algorithms
- Consider a language which may not have polytime algorithms, but for which one can efficiently decide which of the two strings given is more likely (or no less likely) to be in the language
- P-sel – the class of sets with polytime semi-membership algorithms

P-sel – Semi-feasible Computation

Def: A set L is **P-selective** if there exists a function $f : \Sigma^* x \Sigma^* \rightarrow \Sigma^*$ such that

1. f is polynomial-time computable
2. $\forall x, y \in \Sigma^*, f(x, y) \in \{x, y\}$
3. $\forall x, y \in \Sigma^*, \{x, y\} \cap L \neq \emptyset \Longrightarrow f(x, y) \in L$

f is a **selector function** for set L.

Given two inputs, f always chooses one that is no less likely to be in the set L.

P-sel – Semi-feasible Computation

Def: **P-sel** is the class of all P-selective sets.

Example: $L_\Omega = \{x \mid x \geq \Omega\}$, where

- Ω is a real number
- the first occurrence of x is a binary string
- the second occurrence of x is treated as a real number whose binary representation is the first occurrence of x

Why is L_Ω P-selective?
Answer: $f(x, y) = max(x, y)$

$P \subseteq$ P-sel

Theorem: $P \subseteq$ P-sel.
Proof:

- Let $L \in P$.
- Define the selector function f to be such that
  $f(x, y) = x$ if $x \in L$
  $f(x, y) = y$ otherwise

In fact, $P \subsetneq$ P-sel, as we will soon find out!

$P \subsetneq$ P-sel

Recall the halting problem, $HP = \{x \mid x \in L(M_x)\}$, is R.E. but not recursive.

Chaitin constant / halting probability (due to Gregory Chaitin): probability that a randomly generated program for a given model of computation or programming language will halt
$\Omega_{Chaitin} = \Sigma_{p \in HP} 2^{-|p|}$

Theorem: If $L_{\Omega_{Chaitin}}$ were recursive, then $R.E. = RECURSIVE$.
Proof: Homework.

NP-hard sets in P-sel?

As we have seen, P-sel contains some undecidable sets!

Does P-sel contain any NP-complete language?

....

Theorem: There exists an NP-hard and P-selective language if and only if P = NP.
Proof: Homework.

P/poly: Motivation

Consider a language which may not be decided in polynomial time, but with the "small, appropriate advice" for interesting lengths of input, may be decided in polynomial time.

P/poly is the class of all languages that can be decided by a polytime machine using a polynomial amount of advice. Such a polytime machine is called a **P/poly advice interpreter**, given polynomially many advice bits for every input string length.

Advice can be anything desired by the P/poly algorithm designer. It may be terribly hard to compute, but must depend only on the length of the input.

P/poly - Formal Definition

Def: Class **P/poly** denotes $\{L|(\exists A \in P)(\exists$ polynomial $f)[L \in A/f]\}$, where A/f denotes the class of all languages such that for some function h satisfying $(\forall n)[|h(n)| \leq f(n)]$, it holds that $L = \{x \mid \langle x, h(|x|)\rangle \in A\}$.

Notes:

- Language A is the advice interpreter
- Function f specifies the amount of advice available
- Function h gives the advice

TALLY $\subseteq$ P/poly

Theorem: TALLY $\subseteq$ P/poly.
Proof:

- Let $L \in$ TALLY.
- Select the advice function: only 1 advice bit needed so
  $f(n) = 1$
  $h(n) = 1$ if $1^n \in L$
  $h(n) = 0$ otherwise
- Define advice interpreter $A = \{\langle x, y \rangle \mid x \in 1^* \text{ and } y = 1\}$
- $A \in P$ and f is a polynomial $\Longrightarrow L \in$ P/poly.

SPARSE $\subseteq$ P/poly

Let SPARSE denote the set of all sparse languages.
Theorem: SPARSE $\subseteq$ P/poly.

Proof sketch:
Let $L \in$ SPARSE. Then a polynomial p exists such that $|L^{=n}| \leq p(n)$.
What advice? How much advice?

- There are at most polynomially many strings of length n
- If someone gave us all the strings in $L^{=n}$ as a piece of advice, we could decide in polynomial time whether a given input x of length n is in L
- The advice for each possible input length is polynomial in size

$P \subsetneq P/poly$

Theorem: $P \subsetneq P/poly$.

Proof idea:

- Show any P language is also a P/poly language
- Show P is a proper subset of P/poly as the latter contains some undecidable sets

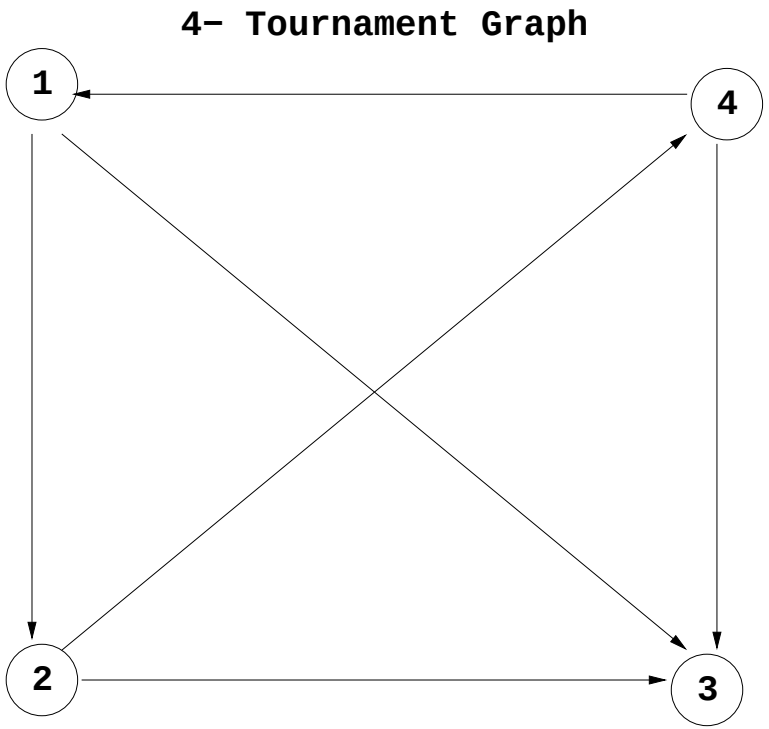
K-tournaments

Def: A **k-tournament** is a graph with k nodes, no self-loops, and exactly one directed edge between each distinct pair of nodes.

Intuitions:

- Each directed edge represents a game played between two players
- Each directed edge points towards the winner (no ties are allowed)

Example: 4-tournament



Comments:

- Arrows point towards the winners
- Each node is considered to defeat itself

Theorem 3.1

Let G be a k-tournament on nodes $V_G = \{1, 2, ..., k\}$ with edges E_G. There exists a set $H \subseteq V_G$ such that
1. $||H|| \leq \lfloor log(k+1) \rfloor$
2. for each $v \in (V_G - H)$, there exists some $g \in H$ such that $(g, v) \in E_G$

In other words, there is a **superloser subset** H (of V_G) whose cardinality is logarithmic in the number of nodes in G, and each node in V_G defeats at least one of the nodes in H.

Proof 3.1

Consider a tournament involving k players, each playing $(k-1)$ games. Some player must lose at least $\lceil \frac{k-1}{2} \rceil$ of its games. (Why?)

Take the player and add it to H, removing it and all other players defeating it from G. Now there remain at most $\lceil \frac{k}{2} \rceil - 1$ nodes in G. Repeat this on G until no nodes remain.

$||H||$ is bounded by the recurrence relation:
$S(0) = 0$
$S(k) \leq 1 + S(\lceil \frac{k}{2} \rceil - 1)$, for each $k \geq 1$

This implies that $S(k) \leq \lfloor log(k+1) \rfloor$.

Connection between P-sel and Tournament

Let L be a P-selective set.

We can construct a tournament for $L^{=n}$ from L's P-selector f.

- Represent each element in $L^{=n}$ as a node
- f is a 2-ary function defined for all possible inputs x and y.
- For each pair of distinct vertices x and y in the tournament, taking $f(x, y)$ as the winner, create a directed edge from the loser node of the two to the winner node $f(x, y)$.

We can construct such a tournament for $L^{=n} \subseteq L$ for each n.

Theorem 3.2

P-sel $\subseteq$ P/poly.

Proof idea:
At each length n in a P-selective set L, there will always exist, by Thm 3.1, a small set H of nodes in L such that every node in $L^{=n}$ defeats one of these nodes, and H will function as a "small advice set" for L.

Proof 3.2: Objective

Consider any P-sel set L.

To show $L \in \mathrm{P/poly}$, it suffices to show:
1. $(\forall x)[x \in L \text{ iff } \langle x, g(|x|)\rangle \in A]$, where A is the advice interpreter set from the definition of P/poly
2. $(\exists q)(\forall n)[|g(n)| \leq q(n)]$, where q is the polynomial bound on the advice size

Proof 3.2: K-tournament at $L^{=n}$

Consider the length-n strings in L, $L^{=n}$.

Construct a k-tournament from $L^{=n}$.
Taking each string in $L^{=n}$ to be a node, we need a way to decide which string is the "winner" for each pair of distinct strings in $L^{=n}$.

Proof 3.2: Selector Function

The P-sel set L has a selector function f.

Define f' to be such that $f'(x, y) = f(min(x, y), max(x, y))$.

- f' is commutative; i.e. $f'(x, y) = f'(y, x)$
- f' is also a selector function for L

The commutativity of f' induces a tournament on $L^{=n}$. We construct a simple graph G such that for any $a, b \in L^{=n}, a \neq b$, it holds that $(a, b) \in E_G$ iff $f'(a, b) = b$.

Proof 3.2: Properties of $L^{=n}$

G corresponding to $L^{=n}$ is a tournament.

By Thm 3.1, we have that there is set $H_n \subseteq L^{=n}$ such that
1. $||H|| \leq \lfloor log(||L^{=n}|| + 1) \rfloor \leq (n + 1)$
2. for each $v \in L^{=n}$, there exists some $h \in H_n$ such that $f'(h, v) = v$

Proof 3.2: Advice

To show $L \in \mathrm{P/poly}$, we need to specify an **advice function** g and an **advice interpreter set** A.

The **advice function** $g(n)$ outputs H_n.
The **advice interpreter set** is
$A = \{\langle x, y\rangle \mid y$ is a (possibly empty) list of elements $v_1, ..., v_z$, and for some j it holds that $f'(v_j, x) = x\}$.

Proof of $A \in P$:
On input $\langle x, y\rangle$,
FOR $j \in [1, z]$
  ACCEPT IF $f'(v_j, x) == x$.
REJECT.

Proof 3.2: Correctness

On the previous slide, we provided a deterministic polytime algorithm deciding A.

We need to verify $(\forall x)[x \in L \text{ iff } \langle x, g(|x|)\rangle \in A]$.

1. Suppose $x \in L$ where $|x| = n$. Then $\langle x, g(|x|)\rangle \in A$ by our choice of g and A.
2. Suppose $\langle x, g(|x|)\rangle \in A$ where $|x| = n$. Then for some $h \in H_n$, it must be that $f'(h, x) = x$. But since $H_n \subseteq L^{=n}$ and f' is a selector function for L, the fact that x defeats an element of L implies that $x \in L$.

Remarks

We have proved that P-sel $\subseteq$ P/poly.

Could we be a bit more precise about the number of advice bits? Do $O(n^2)$ bits suffice?

- $||H_n|| \leq \lfloor log(||L^{=n}|| + 1) \rfloor \leq (n + 1)$
- $|h| = n$ for each each $h \in H_n$
- Thanks to the precise definition we used for advice, for each $n \geq 1$, it holds that the total number of advice bits in H_n is upper-bounded by $n(n + 1)$

In fact, P-sel $\subseteq$ P/quadratic.

Outline: Lecture 2

1. Introduction
2. Definitions
   Deterministic functions
   Nondeterministic functions
3. Aids
   Cook's Theorem
   F-selective
4. Main Result
   Lemma 3.25: NPSV-sel $\cap$ NP $\subseteq$ (NP $\cap$ coNP)/poly
   Theorem 3.20: If all NPMV functions have NPSV refinements, then PH $=$ ZPP$^{\mathrm{NP}}$.

Unique Solutions Collapse the Polynomial Hierarchy : Motivation

- Consider an NPTM N deciding SAT.
- N can nondeterministically find all satisfying assignments of the input formula ϕ.
- Is it easy to find the lexicographically smallest assignment of a given formula ϕ?
- An individual path of the computation tree of N has no information about other paths.
- Thus, culling a single solution may be harder than finding all solutions. We will try to give evidence for this.

Deterministic Functions

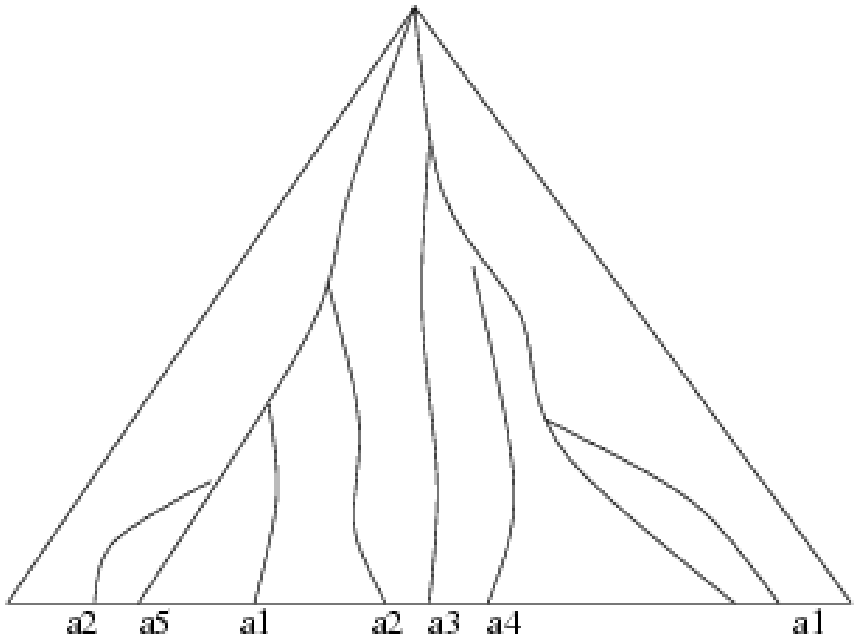
Def: FP denotes the class of all functions computed by deterministic polynomial-time Turing machines.

- FP functions are single-valued.
- These functions can be partial.

Nondeterministic Functions

Def: **NPMV** denotes the class of all multi-valued functions computed by nondeterministic polynomial-time Turing machines.

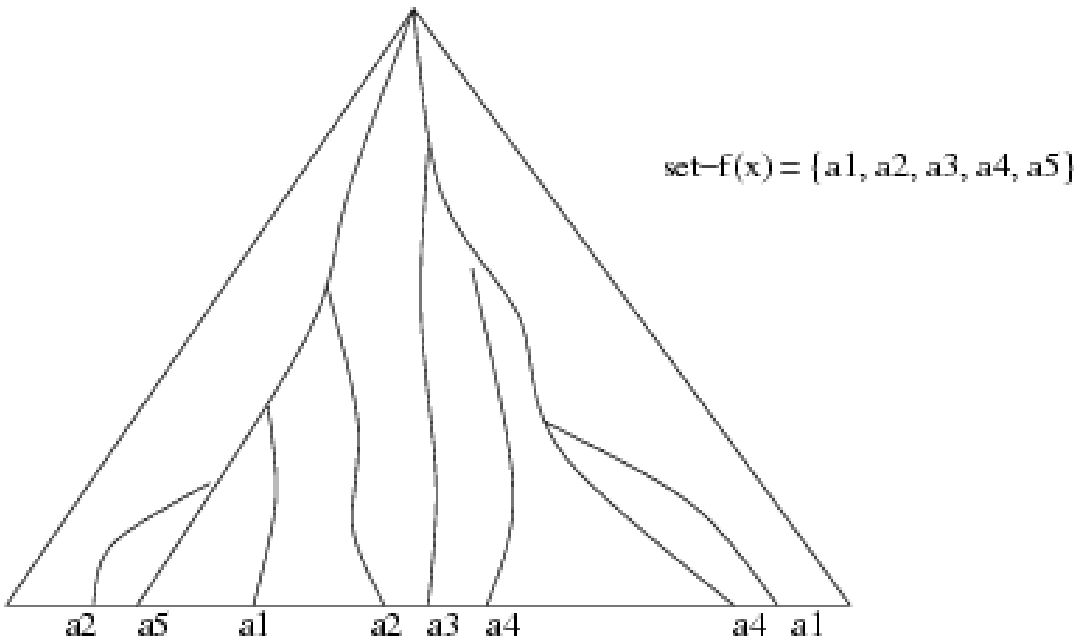
- A function f is in NPMV if there is an NPTM N such that on every input x, f's outputs are exactly the outputs of N.



Nondeterministic Functions

Def: **set-f(x)** is defined as the set of all outputs of an NPMV function f on input x.

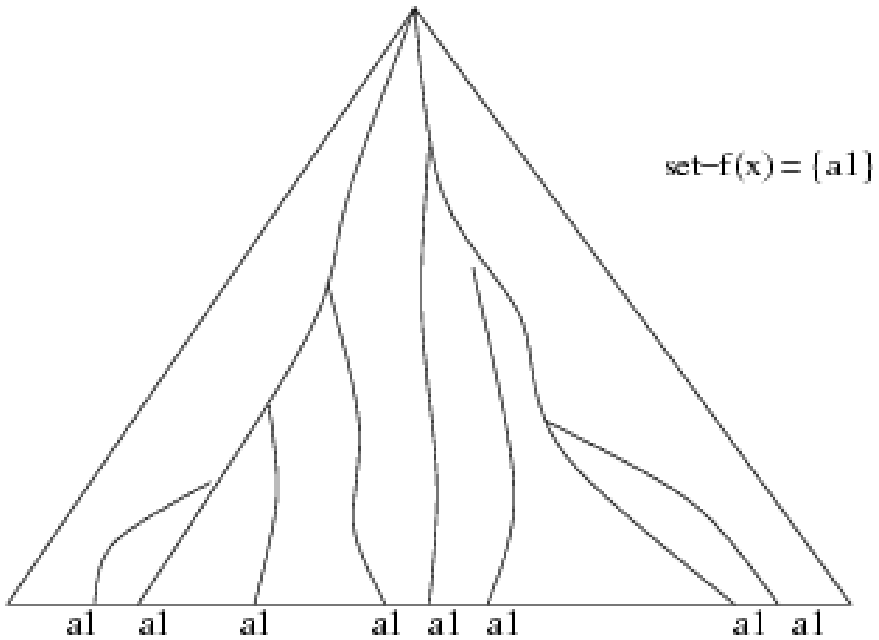
- set-f(x)= {a | a is an output of f(x)}
- If the function f is undefined on x, set-f(x)= $\emptyset$.



Nondeterministic Functions

Def: **NPSV** denotes the class of all single-valued functions computed by nondeterministic polynomial-time Turing machines.

- A function f $\in$ NPSV if $(\forall x)$ [||set-f(x)|| $\leq$ 1].
- NPSV $\subset$ NPMV



Refinement

- Given multivalued functions f and g, g is a refinement of f if
  1. $(\forall x)$ [set-g(x) $= \emptyset \iff$ set-f(x) $= \emptyset$], and
  2. $(\forall x)$ [set-g(x) $\subseteq$ set-f(x)]
- Function g is defined on precisely the same inputs that f is defined on.
- Function g has at most as many outputs as f, and no new ones.
- Refinement captures the notion of "thinning" a multi-valued function.

Cook's Theorem

Theorem: Let N_i be a standard enumeration of NPTMs such that N_i runs in time $n^i + i$. There is a function $f_{cook} \in FP$ mapping from strings to encoding of boolean formulas, such that

1. $(\forall i)(\forall x)$ [$N_i(x)$ accepts $\iff f_{cook}(N_i, x, 0^{|x|^i+i}) \in SAT$]
2. $(\exists g_{cook} \in FP)(\forall i)(\forall x)[g_{cook}(f_{cook}(N_i, x, 0^{|x|^i+i})) = \langle N_i, x \rangle]$
3. $(\exists h_{cook} \in FP)(\forall i)(\forall x)(\forall a)$[if a is a satisfying assignment of $f_{cook}(N_i, x, 0^{|x|^i+i})$, then $h_{cook}(N_i, x, a, 0^{|x|^i+i})$ outputs an accepting computation path of $N_i(x)$]

Cook's Theorem

We now prove an important lemma which connects the notion of refinement of NPMV functions to that of NPTMs that can generate unique solutions.

Lemma 3.23

Lemma: Every NPMV function has an NPSV refinement iff $(\exists f \in NPSV)(\forall F \in SAT)$ [f(F) is a satisfying assignment of F]

Consider an NPMV function h defined by set-h(F) $= \{$a $\mid$ a is a satisfying assignment of F$\}$.
Clearly, the only-if direction of proof holds.

Lemma 3.23 Proof

Lemma: Every NPMV function has an NPSV refinement iff $(\exists f \in NPSV)(\forall F \in SAT)$ [f(F) is a satisfying assignment of F]

Now, we show the if direction.
Let $\hat{f}$ be a function in NPSV such that $(\forall F \in SAT)$ [$\hat{f}$(F) is a satisfying assignment of F].
Let g be a function in NPMV computed via a function-computing NPTM N_i. Functions f_{cook} and g_{cook} are the same as described earlier.

Lemma 3.23 Proof Cont'd.

We give an NPTM N which computes an NPSV refinement of g.

On input x,

- N nondeterministically computes $f_{cook}(N_i, x, 0^{|x|^i+i})$ and then nondeterministically guesses a path of NPSV function $\hat{f}$.
- If along our guessed path, $\hat{f}$ has no output then we will have no output along the current path.
- Otherwise we go along the current path and check whether the output (say, α) is a satisfying assignment of $f_{cook}(N_i, x, 0^{|x|^i+i})$.

Lemma 3.23 Proof Cont'd.

- If α is not a satisfying assignment of $f_{cook}(N_i, x, 0^{|x|^i+i})$ then we have no output along the current path.
- Otherwise we deterministically compute $h_{cook}(N_i, x, \alpha, 0^{|x|^i+i})$ (call the output **p**). We then deterministically compute what value is output along the computation path **p** of $N_i(x)$ and output that value along our current path.

F-Selective

Def: A set L is **F-selective** if there is a multi-valued selector function $f \in$ F such that

1. $(\forall x, y)$ [set-f(x,y) $\subseteq$ {x,y}], and
2. $(\forall x, y)$ [(x $\in$ L $\vee$ y $\in$ L) $\Longrightarrow \emptyset \neq$ set-f(x,y) $\subseteq$ L].

This definition extends the notion of semi-feasible computation to the class of partial nondeterministic functions.

Theorem: NPSV-sel is closed under $\leq_m^p$ reduction

Proof:

Given: A $\leq_m^p$ B

B $\in$ NPSV-sel

f_B - NPSV selector for B

g - Function $\in$ FP many-one reducing A to B

$$\text{set-}f_A(x,y) = \begin{cases} \{x\} & \text{if } f_B(g(x),g(y)) = g(x) \\ \{y\} & \text{else if } f_B(g(x),g(y)) = g(y) \\ \emptyset & \text{otherwise} \end{cases}$$

- f_A is the **NPSV-selector** function for A
- A $\in$ NPSV-sel

ZPP

Def: **ZPP** is defined as the class of all problems for which a probabilistic Turing machine exists with the following properties:

1. It always returns the correct YES or NO answer.
2. The running time is unbounded, but is polynomial on average for any input.

ZPP Cont'd.

Notes:

- ZPP - zero error, probablistic, polynomial time.
- The algorithm is allowed to flip a truly random coin while it is running and it only returns correct answers.
- Such algorithm are called as Las Vegas algorithms.
  Example: Randomized quicksort.
- ZPP $\subseteq$ NP.

Important Lemmas

Lemma 3.25: $\text{NPSV-sel} \cap \text{NP} \subseteq (\text{NP} \cap \text{coNP})/\text{poly}$

We will formally prove lemma 3.25 in the next lecture.

Lemma 3.26: $\text{NP} \subseteq (\text{NP} \cap \text{coNP})/\text{poly} \Longrightarrow \text{PH} = \text{NP}^{\text{NP}}$

Lemma 3.27: $\text{NP} \subseteq (\text{NP} \cap \text{coNP})/\text{poly} \Longrightarrow \text{PH} = \text{ZPP}^{\text{NP}}$

Collapse Of Polynomial Hierarchy

Theorem 3.20: If all NPMV functions have NPSV refinements, then PH $=$ ZPP$^{\mathrm{NP}}$.

Proof Idea: We construct an NPMV-selector function for SAT and show that its NPSV refinement is the NPSV-selector function for SAT. And, hence we show that NP $\subseteq$ NPSV-sel $\cap$ NP. By Lemma 3.25 NP $\subseteq$ (NP $\cap$ coNP)/poly. And by Lemma 3.27, PH $=$ZPP$^{\mathrm{NP}}$.

Proof 3.20

Theorem 3.20: If all NPMV functions have NPSV refinements, then PH $=$ ZPP$^{\mathrm{NP}}$.

Consider a total, multi-valued function f_{SAT} defined as set-f_{SAT}(x,y) $= \{x,y\} \cap$ SAT.
Clearly, $f_{SAT} \in$ NPMV.

Proof:
On input (x,y),
Nondeterministically choose x or y.
Guess a satisfying assignment for the chosen formula.
If the guessed assignment satisfies the formula, output the satisfying assignment.

Proof 3.20: NPSV-selector Function for SAT

Theorem 3.20: If all NPMV functions have NPSV refinements, then PH $=$ZPP$^{\mathrm{NP}}$.

Let g_{SAT} be the NPSV refinement of the function f_{SAT}.

- g_{SAT} is defined if at least one of its arguments is in SAT.
- if at least one of the g_{SAT} arguments is in SAT then g_{SAT} outputs some argument $\in$ SAT.

g_{SAT} is an **NPSV-selector** function for SAT.
SAT $\in$ **NPSV-sel**.

Proof 3.20 Cont'd.

Recall the theorem: NPSV-sel Is Closed Under $\leq_m^p$ Reduction.
As each NP set $\leq_m^p$ reduces to SAT, NP $\subseteq$ NPSV-sel $\cap$ NP.
But we know that NPSV-sel $\cap$ NP $\subseteq$ (NP $\cap$ coNP)/poly by Lemma 3.25.
Thus, **NP $\subseteq$ (NP $\cap$ coNP)/poly**.

We also know that if NP $\subseteq$ (NP $\cap$ coNP)/poly, PH $=$ ZPP$^{\mathrm{NP}}$ by Lemma 3.27.
Hence, if all NPMV functions have NPSV refinements, then PH $=$ ZPP$^{\mathrm{NP}}$.

Remarks

We proved that if all NPMV functions have NPSV refinements, then PH $=$ZPP$^{\mathrm{NP}}$.

Corollary 3.21: If all NPMV functions have NPSV refinements, then PH $=$ NP$^{\mathrm{NP}}$.

What does it mean for a NPMV funtion to have an NPSV refinement?

We can cull out unique solutions of the nondeterministic functions. Thus, unique solutions collapse the polynomial hierarchy!!!

Outline: Lecture 3

1. Review
   K-tournaments
   Theorem 3.1
2. Main Result
   Lemma 3.25: NPSV-sel $\cap$ NP $\subseteq$ (NP $\cap$ coNP)/poly.
3. Conclusion

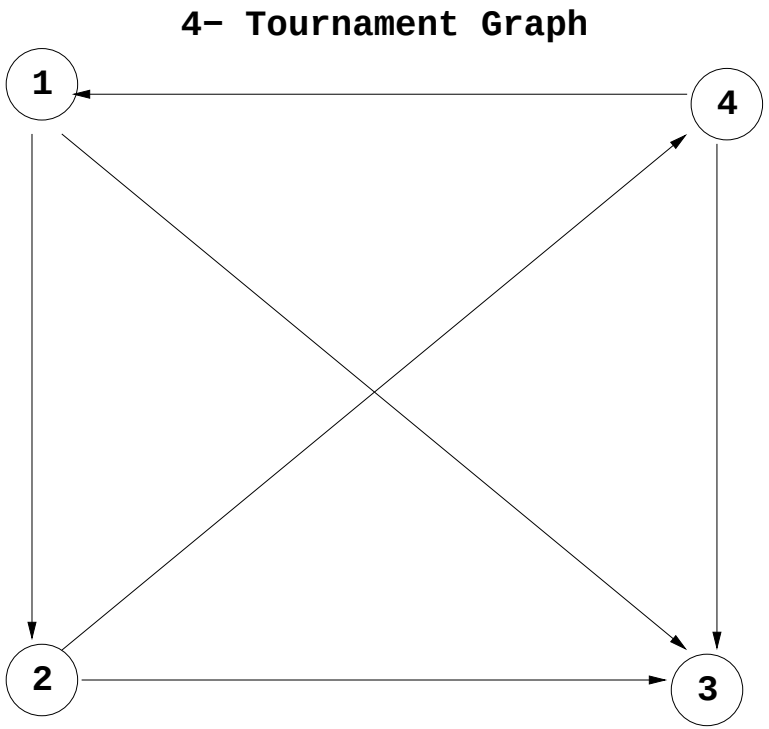
K-tournaments

Def: A **k-tournament** is a graph with k nodes, no self-loops, and exactly one directed edge between each distinct pair of nodes.

Intuitions:

- Each directed edge represents a game played between two players
- Each directed edge points towards the winner (no ties are allowed)

Example: 4-tournament



Comments:

- Arrows point towards the winners
- Each node is considered to defeat itself

Theorem 3.1

Theorem: Let G be a k-tournament on nodes $V_G = \{1, 2, ..., k\}$ with edges E_G. There exists a set $H \subseteq V_G$ such that
1. $||H|| \leq \lfloor log(k+1) \rfloor$,
2. for each $v \in (V_G \setminus H)$, there exists at least one $g \in H$ such that $(g, v) \in E_G$.

In other words, there is a **superloser subset** H (of V_G) whose cardinality is logarithmic in the number of nodes in G, and each node in V_G defeats at least one of the nodes in H.

Lemma 3.25

Lemma: NPSV-sel $\cap$ NP $\subseteq$ (NP $\cap$ coNP)/poly.

Proof idea:
For a set $L \in$ NPSV-sel $\cap$ NP, at each length n, there will always exist, by Thm 3.1, a small set H of nodes in L such that every node in $L^{=n}$ defeats one of these nodes, and H will be a part of our "small advice set" for L.

Proof 3.25: Objective

Consider any set $L \in$ NPSV-sel $\cap$ NP.

To show $L \in (NP \cap coNP)/poly$, it suffices to show:
1. $(\forall x)[x \in L$ iff $\langle x, g(|x|)\rangle \in A]$, where A is the NP $\cap$ coNP advice interpreter set for L.
2. $(\exists q)(\forall n)[|g(n)| \leq q(n)]$, where q is the polynomial bound on the advice size.

Proof 3.25: K-tournament at $L^{=n}$

Consider the length-n strings in L, $L^{=n}$.

Construct a k-tournament from $L^{=n}$.
Taking each string in $L^{=n}$ to be a node, we need a way to decide which string is the "winner" for each pair of distinct strings in $L^{=n}$.

Proof 3.25: Selector Function

The function f is the NPSV-selector for set L.

WLOG, $(\forall x, y)$[set-f(x,y) = set-f(y,x)].

The NPSV-selector function f induces a tournament on $L^{=n}$. We construct a simple graph G such that for any $a, b \in L^{=n}, a \neq b$, it holds that $(a, b) \in E_G$ iff set-f(a, b) = {b}.

Proof 3.25: Properties of $L^{=n}$

G corresponding to $L^{=n}$ is a tournament.
By Thm 3.1, we have that there is set $H_n \subseteq L^{=n}$ such that
1. $||H_n|| \leq \lfloor log(||L^{=n}|| + 1) \rfloor \leq n$
2. for each $v \in L^{=n}$, there exists at least one $h \in H_n$ such that set-f(v, h) $= \{v\}$

Proof 3.25: Advice

To show $L \in (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$, we need to specify an **advice function** g and an **advice interpreter set** A.

Let NPTM N_L accepts the set L.
The **advice function** $g(n)$ outputs
$\langle\langle h_1, h_2, ..., h_z\rangle, \langle w_1, w_2, ..., w_z\rangle\rangle$, where $\langle h_1, h_2, ..., h_z\rangle = H$ and each w_i is an accepting path of $N_L(h_i)$.
The advice function is polynomially length-bounded.

The membership proofs are required to be a part of advice.

Proof 3.25: Advice interpreter

The **advice interpreter set** is
$A = \{\langle x, \langle\langle a_1, a_2, ..., a_z\rangle, \langle w_1, w_2, ..., w_{z'}\rangle\rangle\rangle \mid$
$z = z'$ and
$(\forall i : 1 \leq i \leq z)[w_i$ is an accepting path of $N_L(a_i)]$ and
$(\exists i : 1 \leq i \leq z)$ [x $\in$ set-f(x,a_i)]$\}$.

We now show that $A \in \mathrm{NP} \cap \mathrm{coNP}$.
Clearly, $A \in \mathrm{NP}$.

Proof 3.25: The Set $\overline{A}$

The set $\overline{A}$ is defined as:
$\overline{A} = \{\langle x, \langle\langle a_1, a_2, ..., a_z\rangle, \langle w_1, w_2, ..., w_{z'}\rangle\rangle\rangle \mid$
$z \neq z'$ or
$(\exists i : 1 \leq i \leq z)[w_i$ is not an accepting path of $N_L(a_i)]$ or
$(\forall i : 1 \leq i \leq z)$ [x $\notin$ set-f(x,a_i)]$\}$.

We now prove that $\overline{A} \in \mathrm{NP}$.

Proof 3.25: A $\in$ coNP

We give an NPTM N which accepts $\overline{A}$

- If the input is syntactically ill-formed or $z \neq z'$, then N accepts immediately.
- Otherwise, N deterministically checks if $(\forall i : 1 \leq i \leq z)$ [w_i is an accepting path of $N_L(a_i)$].
- If the check fails, N accepts. Otherwise, N rejects if $x \in \{a_1, a_2, ..., a_z\}$.
- Else, N nondeterministically guesses and checks the unique value of set-$f(a_i$,x) for all i. The nondeterministic path(s) that correctly guess and check for all i which of x and a_i is the unique element in set-$f(a_i$,x) accept iff $(\forall i)$ set-$f(a_i$,x)$= \{a_i\}$.

Lemma 3.25: Correctness

On the previous slides, we proved that $A \in \mathrm{NP} \cap \mathrm{coNP}$.

We need to verify $(\forall x)[x \in L$ iff $\langle x, g(|x|)\rangle \in A]$.

1. Suppose $x \in L$. Then $\langle x, g(|x|)\rangle \in A$ by our choice of g and A.
2. Suppose $\langle x, g(|x|)\rangle \in A$ where $|x| = n$. Interpreter A will, when its second argument is the true advice accept only those strings that are in H_n or that defeat a string in H_n, which is exactly $L^{=n}$ by Thm 3.1 and the properties of NPSV-selector functions.

Conclusion

- Tournament theory result is an upper bound on the nonuniform complexity of semi-feasible sets.
- NP cannot find unique satisfying assignments to satisfiable formula unless the polynomial hierarchy collapses.