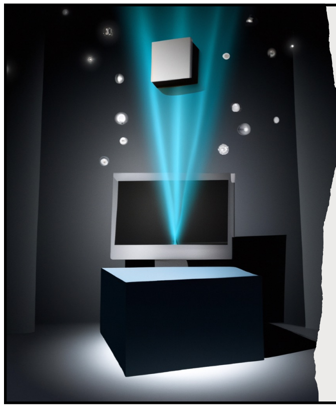


CS151 Complexity Theory

Lecture 15
May 23, 2023

1

Arthur-Merlin Games

- Delimiting # of rounds:
  - **AM[k]** = Arthur-Merlin game with k rounds, Arthur (verifier) goes first
  - **MA[k]** = Arthur-Merlin game with k rounds, Merlin (prover) goes first

Theorem: **AM[k]** (**MA[k]**) equals **AM[k]** (**MA[k]**) with perfect completeness.

- i.e., $x \in L$ implies accept with probability 1
- proof on problem set

May 23, 2023 CS151 Lecture 15

2

Arthur-Merlin Games

Theorem: for all constant $k \geq 2$

AM[k] = AM[2].

- Proof:
  - we show **MA[2]** ⊆ **AM[2]**
  - implies can move all of Arthur's messages to beginning of interaction:

AMAMAM…AM = AAMMAM…AM

… = AAA…AMMM…M

May 23, 2023 CS151 Lecture 15

3

Arthur-Merlin Games

- Proof (continued):
  - given $L \in$ **MA[2]**

$x \in L \Rightarrow \exists m\ \Pr_r[(x, m, r) \in R] = 1$

order reversed

$\Rightarrow \Pr_r[\exists m\ (x, m, r) \in R] = 1$

$x \notin L \Rightarrow \forall m\ \Pr_r[(x, m, r) \in R] \leq \varepsilon$

$\Rightarrow \Pr_r[\ \forall m\ (x, m, r) \in R] \leq 2^{|m|}\varepsilon$

  - by repeating t times with independent random strings r, can make error $\varepsilon < 2^{-t}$
  - set $t = m+1$ to get $2^{|m|}\varepsilon < ½$.

May 23, 2023 CS151 Lecture 15

4

MA and AM

- Two important classes:
  - **MA = MA[2]**
  - **AM = AM[2]**
- definitions without reference to interaction:
  - $L \in$ **MA** iff $\exists$ poly-time language R

$x \in L \Rightarrow \exists\ m\ \Pr_r[(x, m, r) \in R] = 1$

$x \notin L \Rightarrow \forall\ m\ \Pr_r[(x, m, r) \in R] \leq ½$

  - $L \in$ **AM** iff $\exists$ poly-time language R

$x \in L \Rightarrow \Pr_r[\ \exists\ m\ (x, m, r) \in R] = 1$

$x \notin L \Rightarrow \Pr_r[\ \exists\ m\ (x, m, r) \in R] \leq ½$

May 23, 2023 CS151 Lecture 15

5

MA and AM

- $L \in$ **AM** iff $\exists$ poly-time language R

$x \in L \Rightarrow \Pr_r[\ \exists\ m\ (x, m, r) \in R] = 1$

$x \notin L \Rightarrow \Pr_r[\ \exists\ m\ (x, m, r) \in R] \leq ½$

- Relation to other complexity classes:
  - both contain **NP** (can elect to not use randomness)
  - both contained in $\prod_2$. $L \in \prod_2$ iff $\exists\ R \in$ **P** for which:

$x \in L \Rightarrow \Pr_r[\ \exists\ m\ (x, m, r) \in R] = 1$

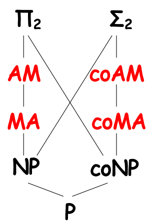
$x \notin L \Rightarrow \Pr_r[\ \exists\ m\ (x, m, r) \in R] < 1$

  - so clear that **AM** $\subseteq \prod_2$
  - know that **MA** ⊆ **AM**

May 23, 2023 CS151 Lecture 15

6

MA and AM



7

MA and AM

Theorem: **coNP ⊆ AM ⇒ PH = AM.**

- Proof:
  - suffices to show $\Sigma_2 \subseteq$ **AM** (and use **AM** $\subseteq \Pi_2$)
  - L $\in \Sigma_2$ iff $\exists$ poly-time language R
    - $x \in L \Rightarrow \exists y \forall z\, (x, y, z) \in R$
    - $x \notin L \Rightarrow \forall y \exists z\, (x, y, z) \notin R$
  - Merlin sends y
  - 1 AM exchange decides **coNP** query: $\forall z\, (x, y, z) \in R$?
  - 3 rounds; in **AM**

8

MA and AM

- We know **Arthur-Merlin = IP**.
  - "public coins = private coins"

Theorem (GS): **IP[k] ⊆ AM[O(k)]**

- stronger result
- implies for all constant $k \geq 2$,

  IP[k] = AM[O(k)] = AM[2]

- So, GNI ∈ **IP[2] = AM**

9

Back to Graph Isomorphism

- The payoff:
  - not known if GI is **NP**-complete.
  - previous Theorems:

    if GI is **NP**-complete then **PH = AM**
  - unlikely!
  - Proof: GI **NP**-complete ⇒ GNI **coNP**-complete ⇒ **coNP ⊆ AM ⇒ PH = AM**

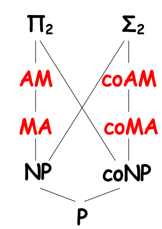
10

MA and AM

- Two important classes:
  - **MA = MA[2]**
  - **AM = AM[2]**
- definitions without reference to interaction:
  - L ∈ **MA** iff $\exists$ poly-time language R
    - $x \in L \Rightarrow \exists m\ \Pr_r[(x, m, r) \in R] = 1$
    - $x \notin L \Rightarrow \forall m\ \Pr_r[(x, m, r) \in R] \leq ½$
  - L ∈ **AM** iff $\exists$ poly-time language R
    - $x \in L \Rightarrow \Pr_r[\exists m\ (x, m, r) \in R] = 1$
    - $x \notin L \Rightarrow \Pr_r[\exists m\ (x, m, r) \in R] \leq ½$

11

MA and AM

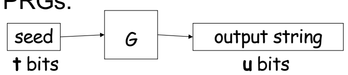


12

Derandomization revisited

- $L \in$ **MA** iff $\exists$ poly-time language R
  - $x \in L \Rightarrow \exists\, m\ \Pr_r[(x, m, r) \in R] = 1$
  - $x \notin L \Rightarrow \forall\, m\ \Pr_r[(x, m, r) \in R] \le ½$

- Recall PRGs:



  – for all circuits C of size at most **s**:

  $|\Pr_y[C(y) = 1] - \Pr_z[C(G(z)) = 1]| \le \boldsymbol{\varepsilon}$

Using PRGs for MA

- $L \in$ **MA** iff $\exists$ poly-time language R
  - $x \in L \Rightarrow \exists\, m\ \Pr_r[(x, m, r) \in R] = 1$
  - $x \notin L \Rightarrow \forall\, m\ \Pr_r[(x, m, r) \in R] \le ½$
- produce poly-size circuit C such that
  - $C(x, m, r) = 1 \Leftrightarrow (x,m,r) \in R$
- for each x, m can hardwire to get $C_{x,m}$
  - $\exists\, m\ \Pr_y[C_{x,m}(y) = 1] = 1$ (“yes”)
  - $\forall\, m\ \Pr_y[C_{x,m}(y) = 1] \le 1/2$ (“no”)

Using PRGs for MA

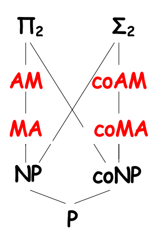
poly(n)

- can compute $\Pr_z[C_{x,m}(G(z)) = 1]$ exactly
  - evaluate $C_{x,m}(G(z))$ on every seed $z \in \{0,1\}^t$
  - running time $(O(|C_{x,m}|)+(\text{time for }G))\mathbf{2^t}$ poly-time
  - $x \in L \Rightarrow \exists\, m\ [\Pr_z[C_{x,m}(G(z)) = 1] = 1]$
  - $x \notin L \Rightarrow \forall\, m\ [\Pr_z[C_{x,m}(G(z)) = 1] \le ½ + \epsilon]$
  - $L \in$ NP if PRG with $t = O(\log n)$, $\epsilon < 1/2$

Theorem: **E** requires exponential size circuits $\Rightarrow$ **MA** = **NP**.

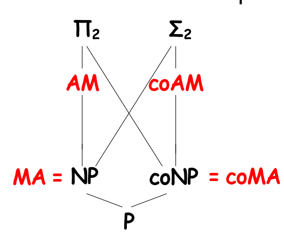
MA and AM

(under a hardness assumption)



MA and AM

(under a hardness assumption)



What about AM, coAM?

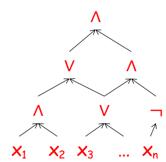
Derandomization revisited

Theorem (IW, STV): If **E** contains functions that require size $2^{\Omega(n)}$ circuits, then **E** contains functions that are $2^{\Omega(n)}$–un-approximable by circuits.

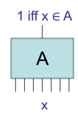
Theorem (NW): if **E** contains $2^{\Omega(n)}$-unapp-roximable functions there are poly-time PRGs fooling poly(n)-size circuits, with seed length $t = O(\log n)$, and error $\epsilon < 1/4$.

Oracle circuits

- **circuit** C
  - directed acyclic graph
  - nodes: AND (∧); OR (∨); NOT (¬); variables x_i



- **A-oracle circuit C**
  - also allow "A-oracle gates"



19

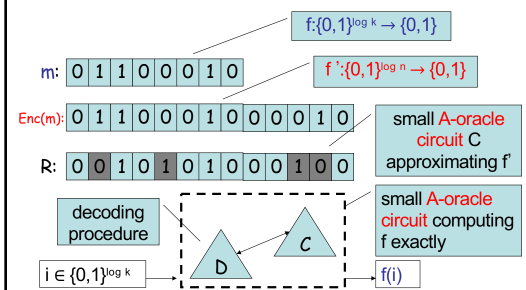
Relativized versions

Theorem: If **E** contains functions that require size $2^{\Omega(n)}$ A-oracle circuits, then **E** contains functions that are $2^{\Omega(n)}$-unapproximable by A-oracle circuits.

- Recall proof:
  - encode truth table to get hard function
  - if approximable by $s(n)$-size circuits, then use those circuits to compute original function by size $s(n)^{O(1)}$-size circuits. Contradiction.

20

Relativized versions



21

Relativized versions

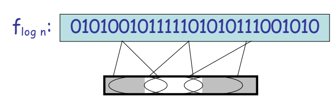
Theorem: if **E** contains $2^{\Omega(n)}$-unapproximable fns., there are poly-time PRGs fooling poly(n)-size A-oracle circuits, with seed length $t = O(\log n)$, and error $\epsilon < 1/4$.

- Recall proof:
  - PRG from hard function on $O(\log n)$ bits
  - if doesn't fool s-size circuits, then use those circuits to compute hard function by size $s \cdot n^{\delta}$-size circuits. Contradiction.

22

Relativized versions

$G_n(y)=f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m})$

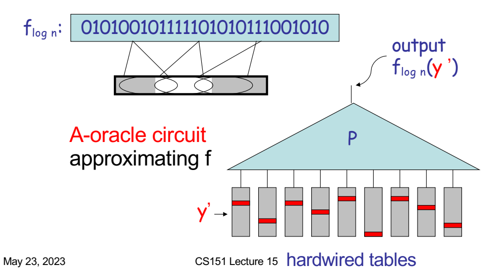


- doesn't fool A-oracle circuit of size s:
  $|\Pr_x[C(x) = 1] - \Pr_y[C(G_n(y)) = 1]| > \varepsilon$
- implies A-oracle circuit P of size s' = s + O(m):
  $\Pr_y[P(G_n(y)_{1\ldots i-1}) = G_n(y)_i] > ½ + \varepsilon/m$

23

Relativized versions

$G_n(y)=f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m})$



24

Using PRGs for AM

L ∈ **AM** iff ∃ poly-time language R

x ∈ L ⇒ $\Pr_r$[∃ m (x, m, r) ∈ R] = 1

x ∉ L ⇒ $\Pr_r$[∃ m (x, m, r) ∈ R] ≤ ½

- produce poly-size SAT-oracle circuit C such that 1 SAT query, accepts iff answer is "yes"

  C(x, r) = 1 ⇔ ∃ m (x,m,r) ∈ R
- for each x, can hardwire to get C_x

  $\Pr_y[C_x(y) = 1] = 1$ ("yes")

  $\Pr_y[C_x(y) = 1] \le ½$ ("no")

25

Using PRGs for AM

x ∈ L ⇒ [$\Pr_z[C_x(G(z)) = 1] = 1$]

x ∉ L ⇒ [$\Pr_z[C_x(G(z)) = 1] \le ½ + \epsilon$]

- C_x makes a *single* SAT query, accepts iff answer is "yes"
- if G is a PRG with t = O(log n), ϵ < ¼, can check in NP:
  - does $C_x(G(z)) = 1$ for all z?

26

Relativized versions

Theorem: If **E** contains functions that require size $2^{\Omega(n)}$ A-oracle circuits, then **E** contains functions that are $2^{\Omega(n)}$ -unapproximable by A-oracle circuits.

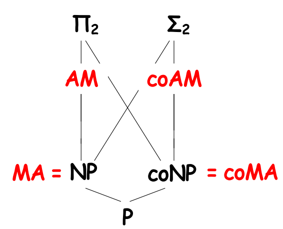
Theorem: if **E** contains $2^{\Omega(n)}$-unapproximable functions there are PRGs fooling poly(n)-size A-oracle circuits, with seed length t = O(log n), and error ϵ < ½.

Theorem: **E** requires exponential size SAT-oracle circuits ⇒ **AM** = **NP**.

27

MA and **AM**

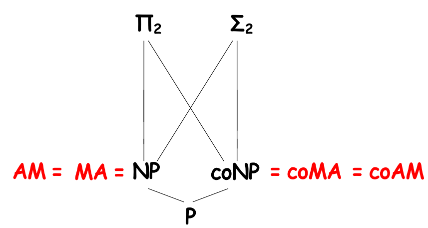
(under a hardness assumption)



28

MA and **AM**

(under a hardness assumption)



29

New topic(s)

Optimization problems,

Approximation Algorithms,

and

Probabilistically Checkable Proofs

30

Optimization Problems

- many hard problems (especially **NP**-hard) are optimization problems
  - e.g. find *shortest* TSP tour
  - e.g. find *smallest* vertex cover
  - e.g. find *largest* clique

  - may be minimization or maximization problem
  - "opt" = value of optimal solution

31

Approximation Algorithms

- often happy with approximately optimal solution
  - warning: lots of heuristics
  - we want approximation algorithm with guaranteed approximation ratio of r
  - meaning: on every input x, output is guaranteed to have value

    at most r*opt for minimization

    at least opt/r for maximization

32

Approximation Algorithms

- Example approximation algorithm:
  - Recall:

  Vertex Cover (VC): given a graph G, what is the *smallest* subset of vertices that touch every edge?

  - **NP**-complete

33

Approximation Algorithms

- Approximation algorithm for VC:
  - pick an edge (x, y), add vertices x and y to VC
  - discard edges incident to x or y; repeat.
- Claim: approximation ratio is 2.
- Proof:
  - an optimal VC must include at least one endpoint of each edge considered
  - therefore 2*opt $\geq$ actual

34

Approximation Algorithms

- diverse array of ratios achievable
- some examples:
  - (min) Vertex Cover: 2
  - MAX-3-SAT (find assignment satisfying largest # clauses): 8/7
  - (min) Set Cover: ln n
  - (max) Clique: $n/\log^2 n$
  - (max) Knapsack: $(1 + \varepsilon)$ for any $\varepsilon > 0$

35

Approximation Algorithms

(max) Knapsack: $(1 + \varepsilon)$ for any $\varepsilon > 0$

- called Polynomial Time Approximation Scheme (PTAS)
  - algorithm runs in poly time for every fixed $\varepsilon>0$
  - poor dependence on ε allowed
- If all **NP** optimization problems had a PTAS, almost like **P** = **NP** (!)

36

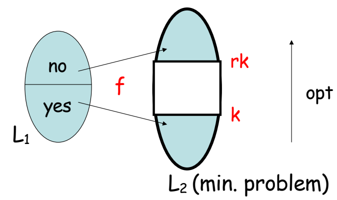
Approximation Algorithms

- A job for complexity: How to explain failure to do better than ratios on previous slide?
  - just like: how to explain failure to find poly-time algorithm for SAT...
  - first guess: probably NP-hard
  - what is needed to show this?
- "gap-producing" reduction from **NP**-complete problem L_1 to L_2

37

Approximation Algorithms

- "gap-producing" reduction from **NP**-complete problem L_1 to L_2



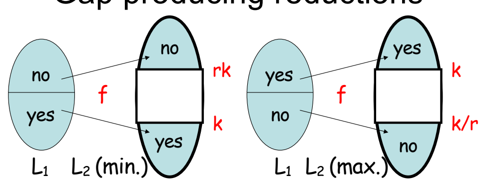
38

Gap producing reductions

- r-gap-producing reduction:
  - f computable in poly time
  - $x \in L_1 \Rightarrow \text{opt}(f(x)) \le k$
  - $x \notin L_1 \Rightarrow \text{opt}(f(x)) > rk$
  - for max. problems use "$\ge k$" and "$< k/r$"
- Note: target problem is not a language
  - **promise problem** (yes ∪ no *not* all strings)
  - "promise": instances always from (yes ∪ no)

39

Gap producing reductions



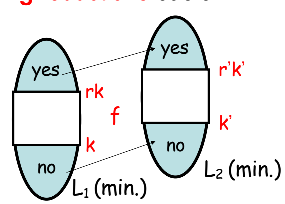
- Main purpose:
  - r-approximation algorithm for L_2 distinguishes between f(yes) and f(no); can use to decide L_1
  - "**NP**-hard to approximate to within r"

40

Gap preserving reductions

- gap-producing reduction difficult (more later)
- but **gap-preserving** reductions easier

Warning: many reductions **not** gap-preserving



41

Gap preserving reductions

- Example **gap-preserving** reduction:
  - reduce MAX-k-SAT with gap ε
  - to MAX-3-SAT with gap ε'
  - constants
  - "MAX-k-SAT is NP-hard to approx. within ε ⇒ MAX-3-SAT is NP-hard to approx. within ε' "
- **MAXSNP** (PY) – a class of problems reducible to each other in this way
  - PTAS for **MAXSNP**-complete problem iff PTAS for all problems in **MAXSNP**

42

MAX-k-SAT

- Missing link: first gap-producing reduction
  - history's guide
    it should have something to do with SAT
- Definition: MAX-k-SAT with gap ε
  - instance: k-CNF φ
  - YES: some assignment satisfies **all** clauses
  - NO: no assignment satisfies more than (1 – ε) fraction of clauses

Proof systems viewpoint

- k-SAT **NP**-hard ⇒ for any language L∈**NP** proof system of form:
  - given x, compute reduction to k-SAT: ϕ_x
  - expected proof is satisfying assignment for ϕ_x
  - verifier picks random clause ("local test") and checks that it is satisfied by the assignment

    x ∈ L ⇒ Pr[verifier accepts] = 1

    x ∉ L ⇒ Pr[verifier accepts] < 1

Proof systems viewpoint

- MAX-k-SAT with gap ε **NP**-hard ⇒ for any language L ∈ **NP** proof system of form:
  - given x, compute reduction to MAX-k-SAT: ϕ_x
  - expected proof is satisfying assignment for ϕ_x
  - verifier picks random clause ("local test") and checks that it is satisfied by the assignment

    x ∈ L ⇒ Pr[verifier accepts] = 1

    x ∉ L ⇒ Pr[verifier accepts] ≤ (1 – ε)
  - can repeat O(1/ε) times for error < ½

Proof systems viewpoint

- can think of reduction showing k-SAT NP-hard as designing a proof system for **NP** in which:
  - verifier only performs local tests
- can think of reduction showing "MAX-k-SAT with gap ε" NP-hard as designing a proof system for **NP** in which:
  - verifier only performs local tests
  - invalidity of proof* evident all over: "holographic proof" and an ϵ fraction of tests notice such invalidity