Arthur-Merlin Games

- Clearly $\text{Arthur-Merlin} \subseteq \text{IP}$
  - "private coins are at least as powerful as public coins"

- Proof that $\text{IP} = \text{PSPACE}$ actually shows $\text{PSPACE} \subseteq \text{Arthur-Merlin} \subseteq \text{IP} \subseteq \text{PSPACE}$
  - "public coins are at least as powerful as private coins"!

- Delimiting # of rounds:
  - $\text{AM}[k] = \text{Arthur-Merlin game with } k \text{ rounds, Arthur (verifier) goes first}$
  - $\text{MA}[k] = \text{Arthur-Merlin game with } k \text{ rounds, Merlin (prover) goes first}$

**Theorem:** $\text{AM}[k]$ ($\text{MA}[k]$) equals $\text{AM}[k]$ ($\text{MA}[k]$) with perfect completeness.
  - i.e., $x \in L$ implies accept with probability 1
  - proof on problem set

**Theorem:** for all constant $k \geq 2$

$\text{AM}[k] = \text{AM}[2]$.

- Proof:
  - we show $\text{MA}[2] \subseteq \text{AM}[2]$
  - implies can move all of Arthur’s messages to beginning of interaction:
    $\text{AMAMAM...AM} = \text{AAMMAM...AM} = \text{AAA...AMMM...M}$

- Proof (continued):
  - given $L \in \text{MA}[2]$
    \[ x \in L \Rightarrow \exists m \Pr_r[(x, m, r) \in R] = 1 \]
  - implies can make error $\epsilon < 2^{-t}$
    \[ \Pr_r[\exists m \text{ such that } (x, m, r) \in R] \leq \epsilon \]
  - by repeating $t$ times with independent random strings $r$, can make error $\epsilon < 2^{-t}$
  - set $t = m + 1$ to get $2^m \epsilon < \frac{1}{2}$. 
**MA and AM**

- Two important classes:
  - **MA = MA[2]**
  - **AM = AM[2]**

- Definitions without reference to interaction:
  - \( L \in MA \iff \exists \text{poly-time language } R \)
    \( x \in L \Rightarrow \exists m \Pr_{[x, m, r]}[(x, m, r) \in R] = 1 \)
    \( x \notin L \Rightarrow \forall m \Pr_{[x, m, r]}[(x, m, r) \in R] \leq \frac{1}{2} \)
  - \( L \in AM \iff \exists \text{poly-time language } R \)
    \( x \in L \Rightarrow \Pr_{[x, m, r]}[(x, m, r) \in R] = 1 \)
    \( x \notin L \Rightarrow \Pr_{[x, m, r]}[(x, m, r) \in R] \leq \frac{1}{2} \)

**MA and AM**

- **Theorem**: \( coNP \subseteq AM \implies PH = AM \).

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

**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 \( \Rightarrow GNI \) \( coNP \)-complete \( \Rightarrow coNP \subseteq AM \Rightarrow PH = AM \)
Derandomization revisited

- \( L \in \text{MA} \iff \exists \text{poly-time language } R \)
  - \( x \in L \Rightarrow \exists m \Pr_r[(x, m, r) \in R] = 1 \)
  - \( x \notin L \Rightarrow \forall m \Pr_r[(x, m, r) \in R] \leq 1/2 \)
- Recall PRGs:
  - for all circuits \( C \) of size at most \( s \):
    - \( \Pr[y \mid C(y) = 1] - \Pr[z \mid C(G(z)) = 1] \leq \epsilon \)

Using PRGs for MA

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

Using PRGs for MA

- can compute \( \Pr_z[C_{x,m}(G(z)) = 1] \) exactly
  - evaluate \( C_x(G(z)) \) on every seed \( z \in \{0,1\}^t \)
  - running time \( O(|C_x| + (\text{time for } G))2^t \)
  - \( x \in L \Rightarrow \exists m \Pr_r[C_{x,m}(G(z)) = 1] = 1 \)
  - \( x \notin L \Rightarrow \forall m \Pr_r[C_{x,m}(G(z)) = 1] \leq 1/2 + \epsilon \)
- \( L \in \text{NP} \) if PRG with \( t = O(\log n) \), \( \epsilon < 1/4 \)

**Theorem:** \( E \) requires exponential size circuits \( \Rightarrow \text{MA} = \text{NP} \).

MA and AM

(under a hardness assumption)

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

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

What about AM, coAM?
Oracle circuits

- circuit C
  - directed acyclic graph
  - nodes: AND (\(\wedge\)); OR (\(\vee\)); NOT (\(\neg\)); variables \(x_1, x_2, x_3, \ldots, x_n\)

- A-oracle circuit C
  - also allow “A-oracle gates”

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 \(s(n)^{1/x}\)-size circuits. Contradiction.

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 \(s' n^t\)-size circuits. Contradiction.
Using PRGs for AM

- \( L \in \text{AM} \) iff \( \exists \) poly-time language \( R \)
  - \( x \in L \Rightarrow Pr[\exists m (x, m, r) \in R] = 1 \)
  - \( x \notin L \Rightarrow Pr[\exists m (x, m, r) \in R] \leq \frac{1}{2} \)
- produce poly-size SAT-oracle circuit \( C \)
  such that \( C(x, r) = 1 \iff \exists m (x,m,r) \in R \)
  - for each \( x \), can hardwire to get \( C_x \)
    - \( Pr[C_x(y) = 1] = 1 \) ("yes")
    - \( Pr[C_x(y) = 1] \leq \frac{1}{2} \) ("no")

Relativized versions

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

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

**Theorem:** \( E \) requires exponential size SAT-oracle circuits \( \Rightarrow \text{AM} = \text{NP} \).

**MA and AM**

(under a hardness assumption)

\[ \text{AM} = \text{coAM} \]

\[ \text{MA} = \text{NP} = \text{coNP} = \text{coMA} = \text{coAM} \]

New topic(s)

Optimization problems, Approximation Algorithms, and Probabilistically Checkable Proofs
Optimization Problems

- many hard problems (especially \textbf{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

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 \( r \cdot \text{opt} \) for minimization
    at least \( \text{opt} / r \) for maximization

Approximation Algorithms

- Example approximation algorithm:
  - Recall:
    Vertex Cover (VC): given a graph \( G \), what is the smallest subset of vertices that touch every edge?
  - \textbf{NP}-complete

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: \( (1 + \epsilon) \) for any \( \epsilon > 0 \)

Approximation Algorithms

- called \textbf{Polynomial Time Approximation Scheme} (PTAS)
  - algorithm runs in poly time for every fixed \( \epsilon > 0 \)
  - poor dependence on \( \epsilon \) allowed
  - If all \textbf{NP} optimization problems had a PTAS, almost like \( \mathbf{P} = \mathbf{NP} \) (!)
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 $L_2$

May 13, 2015

37

Gap producing reductions

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

May 13, 2015

39

Gap preserving reductions

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

Warning: many reductions not gap-preserving

May 13, 2015

41

Gap preserving reductions

- Example gap-preserving reduction:
  - reduce MAX-k-SAT with gap $\epsilon$ to MAX-3-SAT with gap $\epsilon'$
  - “MAX-k-SAT is NP-hard to approx. within $\epsilon$ \( \Rightarrow \) MAX-3-SAT is NP-hard to approx. within $\epsilon'$ ”
- 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

May 13, 2015

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 \( \epsilon \)
  – instance: k-CNF \( \varphi \)
  – YES: some assignment satisfies all clauses
  – NO: no assignment satisfies more than \((1 - \epsilon)\) fraction of clauses

Proof systems viewpoint

• k-SAT NP-hard \( \Rightarrow \) for any language \( L \in \text{NP} \) proof system of form:
  – given \( x \), compute reduction to k-SAT: \( \varphi_x \)
  – expected proof is satisfying assignment for \( \varphi_x \)
  – verifier picks random clause ("local test") and checks that it is satisfied by the assignment
    \( x \in L \Rightarrow \Pr[\text{verifier accepts}] = 1 \)
    \( x \notin L \Rightarrow \Pr[\text{verifier accepts}] < 1 \)

• MAX-k-SAT with gap \( \epsilon \) NP-hard \( \Rightarrow \) for any language \( L \in \text{NP} \) proof system of form:
  – given \( x \), compute reduction to MAX-k-SAT: \( \varphi_x \)
  – expected proof is satisfying assignment for \( \varphi_x \)
  – verifier picks random clause ("local test") and checks that it is satisfied by the assignment
    \( x \in L \Rightarrow \Pr[\text{verifier accepts}] = 1 \)
    \( x \notin L \Rightarrow \Pr[\text{verifier accepts}] \leq (1 - \epsilon) \)
  – can repeat \( O(1/\epsilon) \) times for error < \( 1/2 \)

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 \( \epsilon \) 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 \( \epsilon \) fraction of tests notice such invalidity

PCP

• Probabilistically Checkable Proof (PCP) permits novel way of verifying proof:
  – pick random local test
  – query proof in specified k locations
  – accept iff passes test

• fancy name for a NP-hardness reduction

PCP

• \( \text{PCP}[r(n), q(n)] \): set of languages \( L \) with p.p.t. verifier \( V \) that has \((r, q)\)-restricted access to a string "proof"
  – \( V \) tosses \( O(r(n)) \) coins
  – \( V \) accesses proof in \( O(q(n)) \) locations
  – (completeness) \( x \in L \Rightarrow \exists \text{ proof such that} \)
    \( \Pr[V(x, \text{ proof}) \text{ accepts}] = 1 \)
  – (soundness) \( x \notin L \Rightarrow \forall \text{ proof*} \)
    \( \Pr[V(x, \text{ proof*}) \text{ accepts}] \leq 1/2 \)
PCP

- Two observations:
  - $\text{PCP}[1, \text{poly} \ n] = \text{NP}$
  - $\text{PCP}[\log \ n, 1] \subseteq \text{NP}$

**The PCP Theorem** (AS, ALMSS):

$$\text{PCP}[\log \ n, 1] = \text{NP}.$$  

---

**Corollary**: MAX-k-SAT is \textbf{NP}-hard to approximate to within some constant $\varepsilon$.

- using PCP[log n, 1] protocol for, say, VC
- enumerate all $2^{O(\log n)} = \text{poly}(n)$ sets of queries
- construct a $k$-CNF $\phi_i$ for verifier’s test on each
  - note: $k$-CNF since function on only $k$ bits
- “YES” VC instance $\Rightarrow$ all clauses satisfiable
- “NO” VC instance $\Rightarrow$ every assignment fails to satisfy at least $\frac{1}{2}$ of the $\phi_i$ $\Rightarrow$ fails to satisfy an $\varepsilon = (\frac{1}{2})2^{-k}$ fraction of clauses.