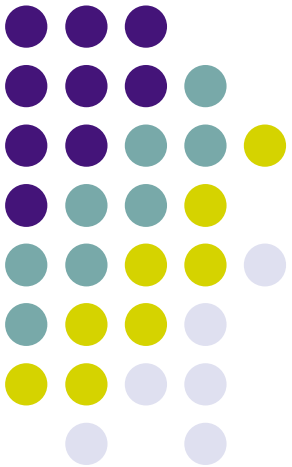
Complexity Classes V

More PCPs

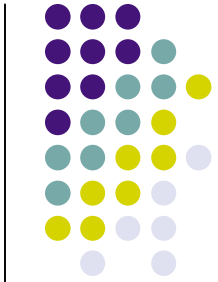
Eric Rachlin

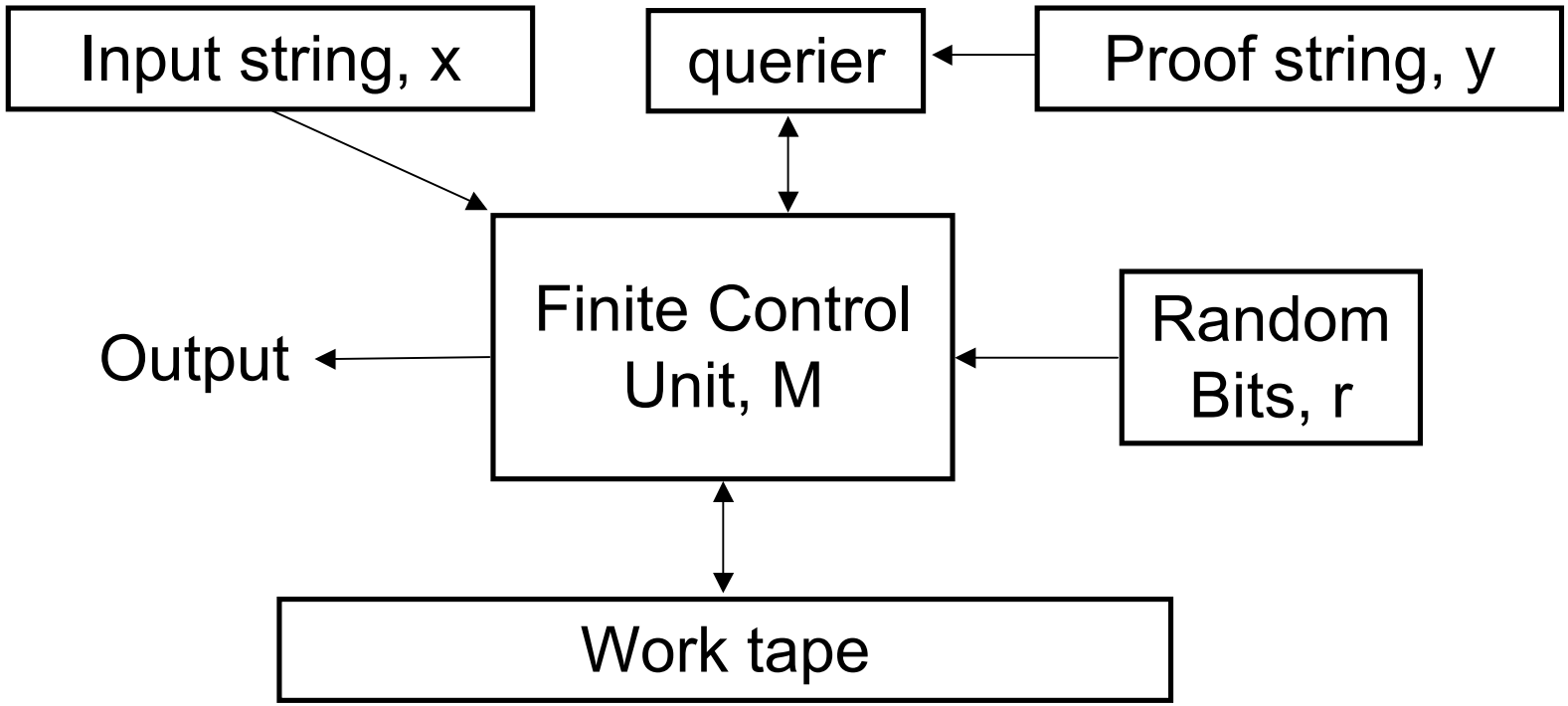


Recall from last time

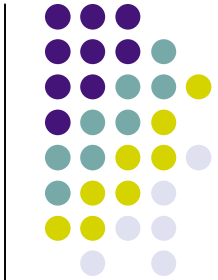
- Nondeterminism is equivalent to having access to a “certificate”.
  - If a valid certificate exists, the machine accepts.
  - We see that problems in NP, which appear hard to solve, are easy to check.
- For PCPs, machines also have access to a certificate (called a proof).
  - The proof is selectively queried using random bits.
  - A valid proof causes the machine to accept, an invalid proof will be rejected with high probability.

PCP Verifiers



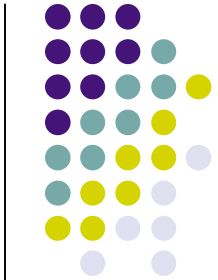


PCP Complexity Classes



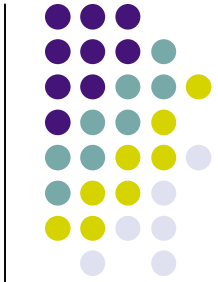
- **$PCP_{c,\,s}[q(n), r(n)]$** is the class of languages that can be recognized with by some Turing machine (which we call a "verifier") with soundness s (or less) and completeness c (or more) using O(r(n)) random bits and O(q(n)) queries to a proof.
  - Completeness c means that there exists a proof such that strings in L are accepted with probability c.
  - Soundness s means that for all proofs the TM accepts strings not in L with probability s.

The PCP Theorem



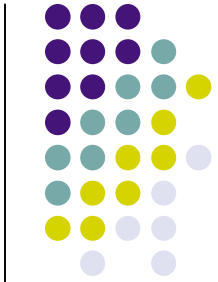
- PCP Theorem (Arora, Lund, Motwani, Sudan, and Szegedy): NP = $PCP_{1,\ 1/2}$ [1, log(n)].
- Recently, a simpler proof was given by Dinur.
  - An NP-complete problem is reduced to a problem in $PCP_{1,\ 1/2}$ [1, log(n)]
- The theorem gives us our first hard to approximate problem.
- Other hardness results can then be proven though gap preserving reductions.

Why $PCP_{1, 1/2}$ [1, log(n)]



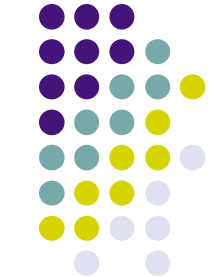
- If NP = $PCP_{1, 1/2}$ [1, log(n)], then every language in NP can be recognized by a machine that makes a constant number of random queries to a polynomial-sized proof.
- In the spirit of Cook's Theorem, the behavior of these machines can be captured as an instance of SAT.
- Now instances of SAT will have a gap.

Cook's Theorem



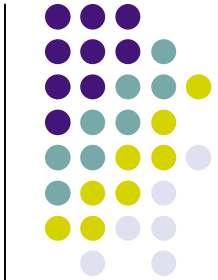
- It is easy to show that the following language, ACCEPT, is NP-complete:
  - Let <M, x, 1^t> be a triple consisting of a deterministic Turing machine, a binary input to M, and a string of t 1's.
  - <M, x, 1^t> is in the language if M accepts some string of the form <x, y> in at most t steps. (y represents a certificate).
- To prove SAT is NP complete, give a polynomial time algorithm designing a circuit (written as a boolean formula) outputting 1 if and only if M accepts <x, y> after t steps.
- Once one language is shown to be NP-complete, others are shown to be NP-complete through reductions.

An Inapproximability Result



- The following language, PROB, is NP-complete:
  - Let <M, x, 1^t> be a triple consisting of a Turing machine with access to log(t) random bits, a binary input x, and a string of t 1's.
  - <M, x, 1^t> is in the language if M accepts some input <x, y> in t steps with probability p = 1.
  - If M ignores its random bits, PROB is the same as ACCEPT.
- Since PROB is NP-complete, any language in NP can be reduced to PROB through some polynomial time reduction, R.
- The PCP Theorem implies R exists such that:
  - M's behavior on <x, y>, when given a particular sequence of random bits, is only a function of O(1) bits of y.
  - OPT = p_{max} cannot be approximated to within a factor of 2.
- The PCP Theorem gives us a gap introducing reduction!

Getting a Result for SAT

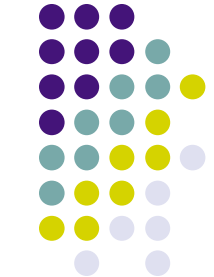


- Any language, L, in NP = $PCP_{1,\ 1/2}$ [1, log(n)], can be reduced to an instance of PROB where MAXPROB(x) = 1 iff L(x) = 1, and MAXPROB(x) ≤ 1/2 iff L(x) = 0.
- The PCP theorem implies a gap introducing reduction to PROB for any L in NP!
- A gap preserving reduction from PROB to SAT shows SAT is also hard to approximate.

Reducing PROB to SAT

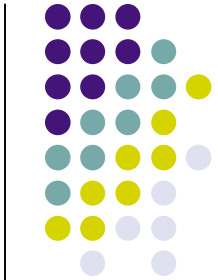
- We need to give a PTIME gap preserving reduction of an instance of PROB, <M, x, 1^t>, to an instance of SAT, <C_1, C_2, …, C_t>
  - Here each C_i denotes some constant number of clauses.
- Let $P_x(y)$ denote the probability M accepts x given y. This probability can computed in PTIME.
- Let $M_{r,\,x}(y)$ denote whether or not M accepts <x, y> on a random sequence r.
  - $P_x(y) = \sum_r M_{r,\,x}(y)/t$ (there are log(t) random bits)
- Recall that $M_{r,\,x}(y)$ is a function of only O(1) bits of y.

Finish the reduction



- Let $y_1, y_2, \ldots, y_t$ be the bits of the proof, y .
- Each $M_{r,\,x}(y)$ is a function of only O(1) y_i .
- Let the i^{th} clause, C_i, the conjunctive normal form (CNF) of $M_{r,\,x}(y)$.
- For an assignment of values to $y_1, y_2, \ldots, y_t$, all clause in each C_i for which $M_{r,\,x}(y) = 1$ are satisfied. At least one clause in each C_i for which $M_{r,\,x}(y) = 0$ is unsatisfied.
- Note that each C_i has constant length.

What's the Gap?

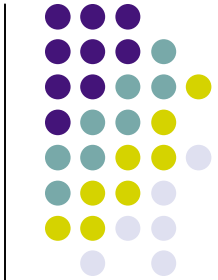


- The gap introducing reduction to PROB, reduced problems in NP to strings of the form <M, x, 1^t>,
- Since M recognizes languages in $PCP_{1, 1/2}$ [1, log(n)] our approximation gap was 2. Also, each $M_{r, x}(y)$ is a function of at most q = O(1) bits of y.
- In our gap preserving reduction, an instance of SAT is produced with at most $t2^q$ clauses of q literals.
  - All clauses can be satisfied if <M, x, 1^t> is in PROB
  - If <M, x, 1^t> is not in PROB, any proof, y, results in M rejecting for at least half of all r.
  - For each r that rejects, at least one clause is unsatisfied.

The gap 1/2 becomes a gap of $1 - (t/2)/t2^q = \mathbf{1 - 1/2^{q+1}}$

.

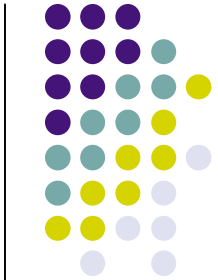
From SAT to 3SAT



- 3SAT is SAT where each clause has at most three variables.
- 3SAT was one of the first problems shown to be NP-complete by a reduction from SAT.
- The standard reduction is already gap preserving!
- Clause of literals, $(lit_1, lit_2, lit_3, \ldots, lit_k)$ becomes:

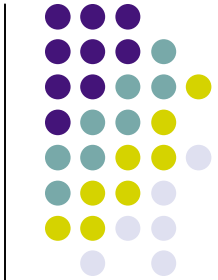
$$(lit_1, lit_2, x_1)\ (\overline{x}_1, lit_3, x_2),\ \ldots (\overline{x}_{k-3}, lit_{k-1}, lit_k)$$

Now what's the gap?



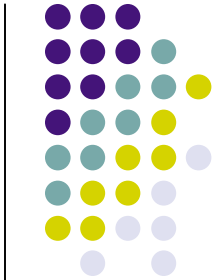
- The instances of SAT we reduced to 3SAT have clauses with at most q literals, and a gap of at least $1 - 1/2^{q+1}$.
  - either all N clauses can be satisfied
  - or at least $N/2^{q+1}$ clauses cannot be satisfied.
- To reduce SAT to 3SAT, each clause of q literals becomes q - 2 clauses of 3 literals.
  - Whenever the original clause is unsatisfied, at least one of the q - 2 new clauses is unsatisfied.
- The gap is now $1 - (N/2^{q+1})/N(q-2) = \mathbf{1 - 1/(q - 2)2^{q+1}}$

And we could keep going…



- See Approximation Algorithms by Vazirani for:
  - 3SAT to 3SAT(k)
  - 3SAT(k) to vertex cover
  - Vertex cover(k) to Steiner tree
- One more quick example: 3SAT to 3Coloring
  - Represent each clause as a triangle.
  - Create an additional “master” node for each literal and connect it to every node representing its compliment.
  - Connect all master nodes to an additional vertex.
- Point of interest: Planar graph coloring is still NP-complete, but all planar graphs are 4-colorable.

Can we do better?



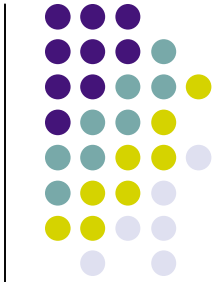
- For a number of problems, we have shown there exists a constant beyond which no polynomial approximation exists (if P ≠ NP).
- Though interesting in theory, the actual value of the constant depends on q, the number of queries. We used the PCP theorem as a black box, and don't know q.
- Also, for many problems, no constant factor approximation is known. Can we show none exists?

The Approximation Hierarchy

- An approximation algorithm for x produces a value OPT' such that OPT'/OPT ≤ f(x)OPT.

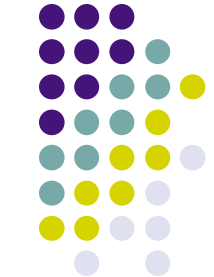
- Some known hardness results for f(x), assuming P ≠ NP:
  - f(x) = Ω(1): SAT, 3SAT, Vertex Cover
  - f(x) = $\Omega(\log(|x|)^{-1})$: Set Cover
  - f(x) = $|x|^{\Omega(-c)}$: Clique, Independent Set

Proving Hardness for Clique



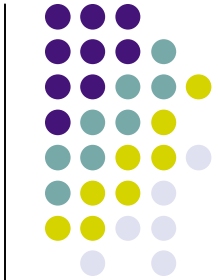
- Reducing PROB, <M, x, 1^t>, to SAT:
  - Let $M_{r,\,x}(y)$ denote the function "Does M accept input x on random sequence r, given proof y?"
  - <M, x, 1^t> becomes t sets of clauses representing the CNF of each $M_{r,\,x}(y)$. Each requires at most q = O(1) bits of y.
- Reducing PROB to CLIQUE
  - <M, x, 1^t> becomes t sets of vertices, one for each $M_{r,\,x}(y)$.
  - In each set, each vertex, v_i, represents one of the (at most) 2^q possible inputs, i, to that $M_{r,\,x}(y)$.
  - v_i "accepts" if $M_{r,\,x}(i) = 1$.
  - v_i and v_j share an edge if they both accept, and i and j are "consistent" inputs. They agree on any bits of y in common.

So What's the Gap?



- The PCP theorem says that if x is in the language, some y causes all $M_{r,\,x}(y)$ to accept.
  - For this y, one vertex per set accepts. Since these vertices are consistent they form a clique of size t.
- If x is not in the language, every y causes at least half of the $M_{r,\,x}(y)$ to reject.
  - A clique of k vertices yields a y causing k $M_{r,\,x}(y)$ to accept.
  - There is no clique of size t/2.
- NP = $PCP_{1,\,1/2}$ [1, log(n)] implies a gap of 1/2.
- What if NP = $PCP_{1,\,1/n}$ [log(n), log(n)]?

Increasing the Gap

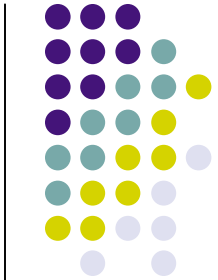


- If NP = $PCP_{1,\ 1/n}$ [log(n), log(n)], each set of 2^q vertices is now of size $O(n^d)$ for some d.
  - Here n denotes the length of x. Note t is polynomial in n.
- As before, if x is in L all $M_{r,\ x}(y) = 1$ for some y.
  - The graph has a clique of size t.
- If x isn't in L, less than 1/n of $M_{r,\ x}(y) = 1$ for all y.
  - The graph has no clique of size 1/n.
- If NP = $PCP_{1,\ 1/n}$ [log(n), log(n)], we reduce PROB to CLIQUE with a gap of $(1/n)/t = 1/n^c$ for some c.
- The total number of vertices is $N = O(tn^d) = O(n^{d+c-1})$, so the gap is at least $N^{-O(c/(d+c))}$.

NP = $PCP_{1,\ 1/n}$ [log(n), log(n)],

- PCP theorem: NP = $PCP_{1,\ 1/2}$ [1, log(n)]
- If a PCP verifier's protocol is repeated log(n) times, soundness goes from 1/2 to 1/n.
  - This implies NP = $PCP_{1,\ 1/n}$ [log(n), $\log^2(n)$].
- Unfortunately our gap introducing reduction to CLIQUE relies on O(log(n)) random bits.
- We must simulate log(n) independent runs of of verifier's protocol using a single seed sequence of O(log(n)) bits.
  - We do this next time using expander graphs.
  - Expanders are used in the proof of the PCP theorem, as well as many other areas of theoretical computer science.

Conclusion



- The PCP theorem implies that PROB was NP-complete and hard to approximate if P ≠ NP.
- The PCP theorem implies o(1) hardness of approximation results for SAT, 3SAT, etc…
- A modified version of the PCP theorem gives a much stronger $n^{o(1)}$ result for clique.
  - We'll prove NP = $PCP_{1,\ 1/n}$ [log(n), log(n)] using expanders.
- For our big-O hardness results to become numerical values, we must know the actual number of queries used by the verifier for languages in NP.