Approximating Solutions to Hard Problems

Approximation algorithms,
Probabilistically Checkable Proof Systems,
and Hardness of Approximation

Approximation algorithms

Motivating example:

- Max SAT: Given a Boolean formula φ in conjunctive normal form, find the maximum number of clauses that can be simultaneously satisfied
- This is an optimization version of the classical SAT decision problem

*S*uggest simple algorithms that aim to satisfy as many clauses as possible

Approximation algorithms for Max SAT

- Greedy algorithm: assign a truth value to the variables in turn, choosing a value for variable xi that satisfies at least half of the not-yet-satisfied clauses in which xi appears

Approximation algorithms for Max SAT

- Even simpler: either the all-true or all-false assignment satisfies at least half of the clauses (why?)

Approximation algorithms for Min Vertex Cover

- Given an undirected graph $G = (V,E)$, find a minimum vertex cover for G. A vertex cover is a set of nodes that are incident on all edges of G

Suggest simple algorithms that aim to find the smallest possible vertex cover

Approximation algorithms for Min Vertex Cover

Greedy algorithm:

- Start with $S = \emptyset$
- Repeat until the graph has no edges:
  - Pick the vertex v that is incident on the most edges (breaking ties arbitrarily), add v to S and remove its incident edges from the graph

Approximation algorithms for Min Vertex Cover

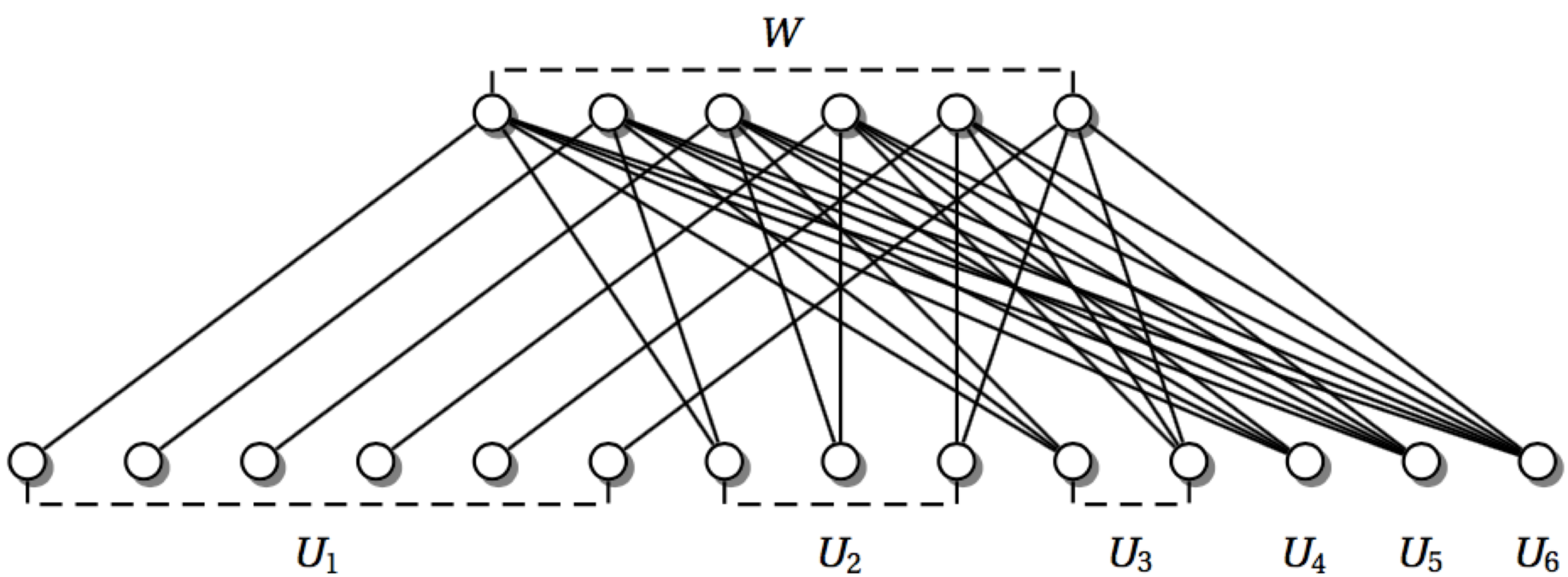

FIGURE 9.3: The bipartite graph B_t for $t = 6$. The smallest vertex cover is W, but the greedy algorithm covers all the vertices in $U = \bigcup_{i=1}^{t} U_i$, and $|U|/|W| = \Theta(\log n)$.

From The Nature of Computation by Chris Moore

Approximation algorithms for Min Vertex Cover

Conservative algorithm:

- Start with $S = \emptyset$
- Repeat until the graph has no edges:
  - Pick any edge of E, and add *both* of its endpoints to S. Delete these two vertices from the graph, as well as all incident edges

This algorithm finds a vertex cover of size at most twice the minimum – why?

Optimization problems

An *optimization problem* ∏ has the following properties: Corresponding to an instance I of the problem is a set of solutions. Corresponding to each solution is a value, which is a positive rational number.

∏ is either a *maximization* problem, in which case we want to find the solution with *maximum* value, or a *minimization* problem. Let Opt(I) be the value of the optimal solution to I.

Optimization problems

An algorithm A is an *approximation algorithm* for ∏ if given an instance I of ∏, A computes a solution of I. Let A(I) denote the value of the solution computed by A on instance I. Let

$$R_A(I) = \max\{\ A(I)/Opt(I),\ Opt(I)/A(I)\ \}.$$

Note that $1 \le R_A(I)$ and the closer $R_A(I)$ is to 1, the better A performs on input I. Algorithm A has *approximation ratio* R_A if

$$R_A \ge R_A(I) \text{ for all instances I of } \prod.$$

Optimization problems

Max 3SAT has an approximation algorithm with approximation ratio 2.

Vertex Cover has an approximation algorithm with approximation ratio log n.

Are there algorithms with better approximation ratios? Is there a limit to how good the approximation ratios can be for these and other problems?

Next we'll introduce tools to help us answer the second question here.

The Gap Lemma

Gap Lemma: Let L be NP-complete. Suppose that there is a poly-time mapping from any instance x of L to instance x′ of maximization problem ∏ such that

$x \in L \Rightarrow \mathrm{Opt}(x') = g(x)$ and

$x \notin L \Rightarrow \mathrm{Opt}(x') < (1-c)\, g(x)$

where $g(x) \in \mathbb{N}$, g is poly-time computable, and $0<c<1$.

If ∏ has a poly-time approximation algorithm with approximation ratio $1 + c/(1-c)$, then NP = P.

Proof in handout.

Probabilistically Checkable Proof Systems (PCPs)

Consider a poly-time coin-flipping verifier V which receives an input x and a proof π, and outputs either 1 (yes) or 0 (no)

Let $V(x,\pi)$ denote V's output on x, π
(Note that $V(\pi, x)$ is a random variable)

V is a *probabilistically checkable proof system* (PCP) for language L if

$$x \in L \Rightarrow \exists\, \pi \in \{0,1\}^* \quad \Pr[V(x,\pi) = 1] = 1$$

$$x \notin L \Rightarrow \forall\, \pi \in \{0,1\}^* \quad \Pr[V(x,\pi) = 1] \leq ½$$

PCPs

We say that language L is in PCP(r(n), q(n)) if there is a PCP V for L such that, on all inputs x

- the verifier uses O(r(|x|)) random bits
- the verifier queries O(q(|x|)) bits of the proof
- the bits must be queried *non-adaptively*, i.e. the verifier decides which bits to query before seeing any of these bits

PCPs

PCP Theorem: NP = PCP(log n, 1)

We'll prove a weak version of this in the next class, let's look at its applications first

Max 3SAT is hard to approximate

Max 3SAT: Given a Boolean formula ϕ in 3-conjunctive normal form (i.e., each clause has at most three literals), find the maximum number of clauses that can be simultaneously satisfied

Max 3SAT is hard to approximate

Max 3SAT: Given a Boolean formula ϕ in 3-conjunctive normal form (i.e., each clause has at most three literals), find the maximum number of clauses that can be simultaneously satisfied

Theorem: For some constant $c > 1$, if there is a polynomial time approximation algorithm for Max 3SAT with approximation ratio c, then P=NP

Max 3SAT is hard to approximate

Proof: Let $L \in NP$, let V be a PCP for L that uses q queries, r(n) random bits, and gets a proof of length l(n). Fix instance x of L. For each string τ of length r(|x|) let $b_{\tau,1}$, $b_{\tau,2}$, ... $b_{\tau,q}$ be the positions of the proof that V queries on coin flip sequence τ.

Using V, we'll describe a mapping $x \rightarrow \Phi_X$ from instances of L to instances of Max 3SAT, and apply the Gap Lemma to conclude that Max 3SAT is hard to approximate.

Max 3SAT is hard to approximate

Useful facts (proofs will be provided in handout):

- For any Boolean function F of q variables there is an equivalent q-CNF formula (i.e., each clause has at most q literals) with at most 2^q clauses.
- For any q-CNF formula ϕ', there is a 3CNF formula ϕ such that ϕ' is satisfiable if and only if ϕ is. Moreover, the number of clauses in ϕ is at most q times the number of clauses in ϕ'.

Stronger results for Max 3SAT

- Hastad showed that if there is a ($8/7-\varepsilon$)-approximation algorithm for Max 3SAT, then NP=P.
- Karloff and Zwick provided an algorithm for Max 3SAT that seems to have approximation ratio 8/7.

Next Class

- More on proving hardness of approximation, e.g., for the Clique problem
- Proof of a weak version of the PCP theorem