Approximation Algorithms

NP-Complete and NP-Hard

- NP-complete problems are decision problems that cannot be solved in polynomial time if $P \neq NP$.
- NP-hard problems are problems that are at least as hard as NP-complete problems.
- Optimization versions of NP-complete decision problems are NP-hard.

How to Solve NP-Hard Problems?

- Yet many important problems are NP-hard and we need to solve them.
- One direction is to find approximation algorithms, which do not find optimal solutions but instead find approximately optimal ones.
- NOTE: the focus of approximation algorithms is theoretical, devoted not to getting the best results in practice, but to getting the best provable results.

Approximation Algorithms

- For any given optimization (minimization) problem and approximation algorithm $\mathcal{A}$ to solve it:
  - Let $\Pi =$ Set of all instances of the problem.
  - For all instances $I \in \Pi$ define $OPT(I) =$ cost of optimal solution for I
  - $A(I) =$ cost of solution produced by A on I.

- Then if
  - $\forall I \in \Pi, \frac{A(I)}{OPT(I)} \leq \rho$
  - A is a factor ρ approximation algorithm (ρ-approximation algorithm).
  - Also say the approximation ratio is ρ

Approximation Algorithms

- For any given optimization (minimization) problem and approximation algorithm $\mathcal{A}$ to solve it:
  - Let $\Pi =$ Set of all instances of the problem.
  - For all instances $I \in \Pi$ define $OPT(I) =$ cost of optimal solution for I
  - $A(I) =$ cost of solution produced by A on I.

- Then if
  - $\forall I \in \Pi, \frac{A(I)}{OPT(I)} \leq \rho(n)$
  - A is a factor $\rho(n)$ approximation algorithm ($\rho(n)$-approximation algorithm).
  - Also say the approximation ratio is $\rho(n)$

Strategies to Design Approximation Algorithms

- Greedy algorithms
- Randomized algorithms
- Linear programming and randomized rounding

Greedy Algorithms

Greedy Algorithms

- A greedy algorithm works in phases. At each phase:
  - You take the best you can get right now, without regard for future consequences
  - Need to ensure feasible solution
  - You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum

Vertex Cover

- Given an undirected graph $G = (V, E)$, $S \subseteq V$ is a vertex cover if every edge in G has an endpoint in S.
- Minimum Vertex Cover: Given a graph G, find the minimum vertex cover.
- We have proved that the decision version is NP-complete, so the optimization version is NP-hard.

Approximate Vertex Cover

- Consider the following algorithm:

Approximate Vertex Cover (AVC):
$S \leftarrow \emptyset$
$E' \leftarrow E[G]$
while E' is not empty
    let (u, v) be any edge from E'
    $S \leftarrow S \cup \{u, v\}$
    remove from E' every edge incident with u or v
return(S)

Time complexity: $O(V + E)$

Approximate Vertex Cover

- Theorem: AVC is a 2-approximation algorithm for the minimum vertex cover problem.

- For any graph G define

$$AVC(G) = \text{ size of (\# of vertices in) } S, |S|$$

$$OPT(G) = \text{ size of smallest vertex cover of } G.$$

Then, for all G

$$1 \leq \frac{AVC(G)}{OPT(G)} \leq 2$$

Approximate Vertex Cover

- Theorem: AVC is a 2-approximation algorithm for the minimum vertex cover problem.
- To prove the theorem, we need to show:
  1. For any graph, AVC produces a vertex cover
  2. The ratio between AVC and OPT is at most 2
  3. (optional) Give the tight example

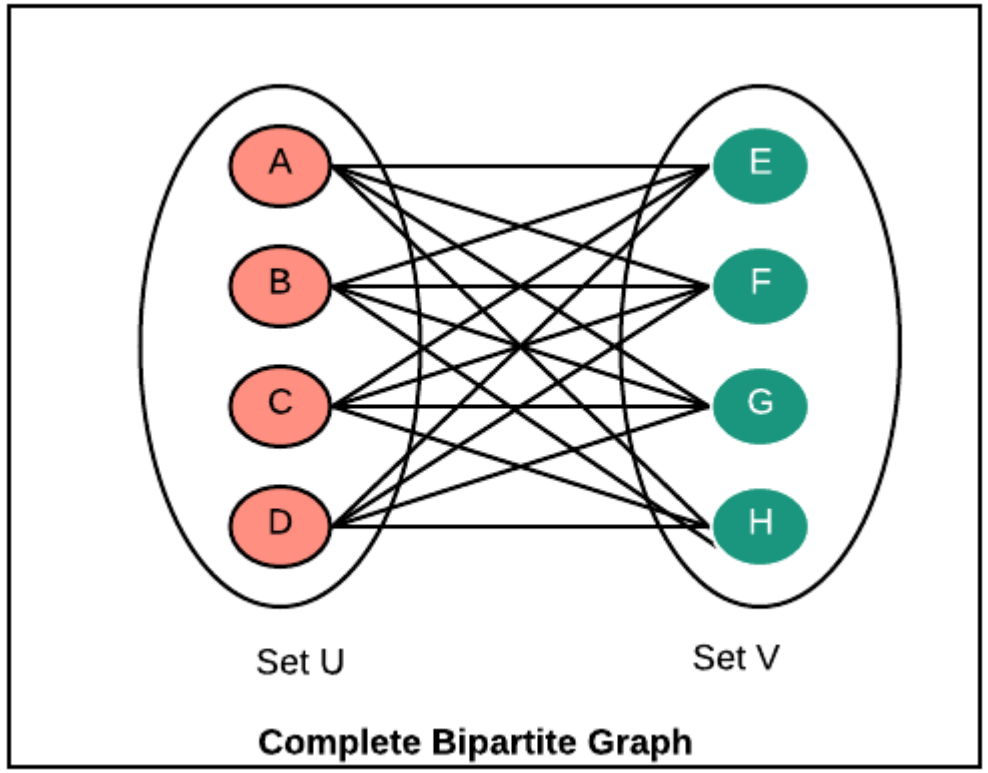
Approximate Vertex Cover: Proof

- Step 1: No edge will be left uncovered by the end of the algorithm. Otherwise, E' is not empty.

- Step 2: For each edge (u, v) that is considered in AVC, at least one of u, v should be added into the vertex set to cover this edge. Let n be the number of these edges, then $OPT \geq n$, and we have $AVC = 2n$, so $\frac{AVC}{OPT} \leq 2$

Approximate Vertex Cover: Proof

- Step 3: Consider the complete bipartite graph $K_{n,n}$



The Set Covering Problem

- Let X be a set and $\mathcal{F}$ a family of subsets of X such that $X = \cup_{F \in \mathcal{F}} F$.
- The problem is to find a minimum-size subset $C \subseteq \mathcal{F}$ that covers X, i.e., $X = \cup_{F \in \mathcal{C}} F$.
- For example, $X = \{1,2,3,4,5,6\}$ and $\mathcal{F}$ contains the subsets
  - $F_1 = \{1,5\}$
  - $F_2 = \{2,3,6\}$
  - $F_3 = \{2,5,6\}$
  - $F_4 = \{1,4\}$
  - $F_5 = \{2,3,4,6\}$
- $\{F_1, F_2, F_4\}$ covers X but is not a minimal size solution.
- $\{F_1, F_5\}$ is a minimal size solution.

The Set Covering Problem

- Finding a minimal size set cover is a NP-Hard problem.
- Its decision version is NP-complete.
- It can be shown that Vertex Cover $\leq_p$ Set Cover

A Greedy Set Cover Algorithm

- Consider following algorithm

Greedy Set Cover (GSC):
$U \leftarrow X$
$C \leftarrow \emptyset$
while $U \neq \emptyset$
    select a $F \in \mathcal{F}$ that maximizes $|F \cap U|$
    $U \leftarrow U - F$
    $C \leftarrow C \cup \{F\}$
return (C)

Time complexity: $O(|X| \cdot |\mathcal{F}| \cdot |X|)$

Greedy Set Cover: Proof

- Theorem: GSC is an H_n-approximation algorithm for the set covering problem, where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$.

- Step 1: The output of GSC is a set cover of X.

Greedy Set Cover: Proof

- Step 2: Prove the approximation ratio H_n.
- Number the elements of U in the order in which they were covered by GSC, resolving ties arbitrarily. Let $e_1, e_2, \ldots, e_n$ be this numbering.
- Define the price of an element to be the average cost at which it is covered, i.e.,

$$price(e_k) = \frac{1}{\#\text{ of new elements covered in this iteration}}$$

- So GSC select the set that minimizes the price at each iteration.
- Lemma:

$$|C| = \sum_{k=1}^{n} price(e_k)$$

Greedy Set Cover: Proof

- Lemma: For each $k \in \{1, \dots, n\}$,

$$price(e_k) \leq \frac{OPT}{n - k + 1}$$

- Proof:
  - In any iteration, the left-over sets of the optimal solution can cover the remaining elements at a cost of at most OPT.
  - If we select these sets in any order, the average price of each remaining element would be $\frac{OPT}{|U|}$. This means the set selected by GSC should have price at most $\frac{OPT}{|U|}$.
  - In the iteration in which e_k was covered, $|U| \geq n - k + 1$.
  - As a result, $price(e_k) \leq \frac{OPT}{|U|} \leq \frac{OPT}{n-k+1}$.

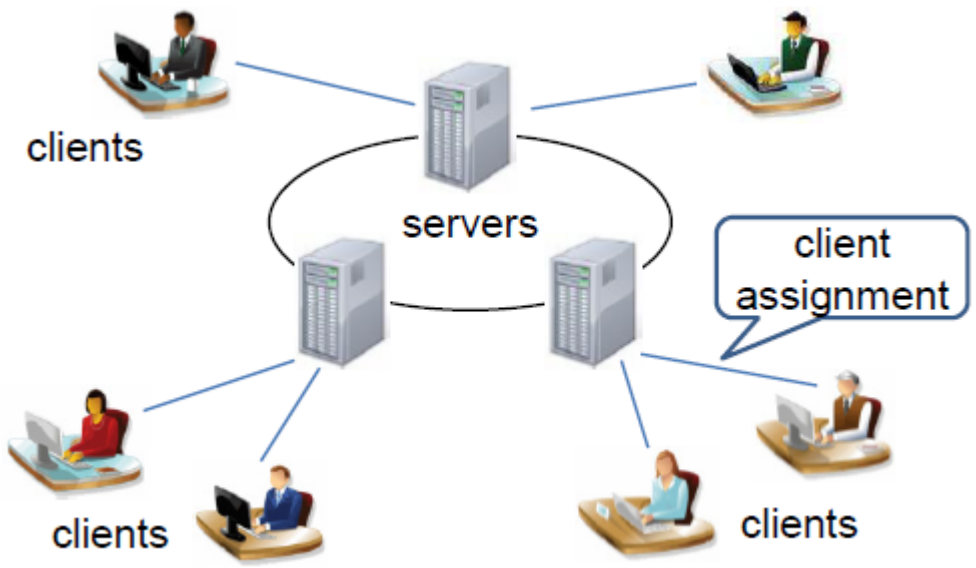
Greedy Set Cover: Proof

$$|C| = \sum_{k=1}^{n} price(e_k) \leq \sum_{k=1}^{n} \frac{OPT}{n-k+1} = \left(1 + \frac{1}{2} + \cdots + \frac{1}{n}\right) \cdot OPT$$

Greedy Set Cover: Proof

- Step 3: The approximation ratio is tight (example skipped).

Real-World Example



Randomized Algorithms

Quick Review of Probability Theory

- A random variable is a real number that is the outcome of a random event.
  - For example, X can be the number of spots showing after throwing a dice.
- The expectation of X is

$$E[X] = \sum_i i \cdot \Pr(X = i) = \int \alpha f_X(\alpha) d\alpha$$

- Some basic facts:
  - If X and Y are any two random variables then $E[X + Y] = E[X] + E[Y]$.
  - If c is any number then $E[cX] = cE[X]$.
  - If X and Y are independent then $E[XY] = E[X] \cdot E[Y]$.

Example

Throw two dice. Let X and Y be the respective number of spots showing on each of them.

$$E[X] = E[Y] = \sum_{i=1}^{6} i \cdot \Pr(X = i) = \sum_{i=1}^{6} \frac{i}{6} = \frac{7}{2}.$$

Therefore the expected sum of the two dice's spots is

$$E[X + Y] = E[X] + E[Y] = \frac{7}{2} + \frac{7}{2} = 7.$$

Since X and Y are independent

$$E[XY] = E[X] \cdot E[Y] = \frac{7}{2} \cdot \frac{7}{2} = \frac{49}{4}$$

Example

Now define

$$A = \begin{cases} 1 & X \text{ is even} \\ 0 & X \text{ is odd} \end{cases}, \; B = \begin{cases} 0 & X \text{ is even} \\ 1 & X \text{ is odd} \end{cases}$$

Calculation shows that $E[A] = E[B] = \frac{1}{2}$, $A + B = 1$ and $AB = 0$.

Notice

$$E[A + B] = 1 = \frac{1}{2} + \frac{1}{2} = E[A] + E[B]$$

but

$$E[AB] = 0 \neq \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2} = E[A] \cdot E[B].$$

This is because A and B are not independent.

Indicator Random Variables

- Let W be some event. The indicator random variable of W is the function

$$I_W = \begin{cases} 1 & \text{if } W \text{ happens} \\ 0 & \text{if } W \text{ does not happen} \end{cases}$$

- For example. Suppose we throw a dice and let X be the number of spots that show. W could be the event "X is even". Then

$$I_W = \begin{cases} 1 & X \text{ is even} \\ 0 & X \text{ is odd} \end{cases}$$

- The important fact about indicator random variables is

$$E[I_W] = \Pr(W \text{ happens }).$$

Example

- Let $G = (V, E)$ be a complete graph with $V = \{1, \dots, n\}$. Now pick a subset $S \subseteq V$ at random using the following procedure.

> $S = \emptyset$.
> For $i = 1$ to n do
> Flip a fair coin. If "heads", set $S = S \cup \{i\}$.

- What is the expected number of edges in the cut $\delta(S) = (S, V - S) = \{(u, v) : u \in S, v \in V - S\}$?

Example

Define

$$I_{i,j} = \begin{cases} 1 & \text{if } i \in S \text{ and } j \notin S \text{ or } j \in S \text{ and } i \notin S \\ 0 & \text{otherwise} \end{cases}.$$

Then

$$\delta(S) = \sum_{1 \le i < j \le n} I_{i,j}$$

Note that

$$E\left[I_{i,j}\right] = \Pr(i \in S \text{ and } j \notin S) + \Pr(j \in S \text{ and } i \notin S) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

so

$$E[\delta(S)] = \sum_{1 \le i < j \le n} E\left[I_{i,j}\right] = \sum_{1 \le i < j \le n} \frac{1}{2} = \frac{n(n-1)}{4}$$

Max Cut

- Max Cut: Given a weighted complete graph $G = (V, E)$ and find a cut $\delta(S), S \subseteq V$ with maximum value. The value of a cut $\delta(S)$ is defined to be

$$\delta(S) = \sum_{(u,v):u \in S, v \in V-S} w(u, v).$$

A Randomized Approximation Algorithm for Max Cut

- Consider the previous procedure as a randomized approximation algorithm

Random Max Cut
$S = \emptyset$.
For $i = 1$ to n do
    Flip a fair coin. If "heads", set $S = S \cup \{i\}$.

- Let OPT be the value of an optimal cut
- Theorem: $E[\delta(S)] \geq \frac{OPT}{2}$

A Randomized Approximation Algorithm for Max Cut

Proof: Let $I_{i,j}$ be defined in the previous example. Then

$$\delta(S) = \sum_{1 \le i < j \le n} w(i,j) I_{i,j}.$$

Therefore

$$\begin{aligned} E[\delta(S)] &= E\left[\sum_{1 \le i < j \le n} w(i,j) I_{i,j}\right] \\ &= \sum_{1 \le i < j \le n} w(i,j) E\left[I_{i,j}\right] \\ &= \frac{1}{2} \sum_{1 \le i < j \le n} w(i,j) \\ &\ge \frac{1}{2} OPT. \end{aligned}$$

The MAX SAT problem

- Let $x_1, x_2, \ldots, x_n$ be BOOLEAN variables. These variables are set to be either TRUE (T) or FALSE (F). A variable x_i is T if and only if its negation $\bar{x}_i$ is F and vice-versa.
- A clause is the disjunction of boolean variables and their negations, e.g., $x_1 \vee \bar{x}_3 \vee x_4$.
- Given a truth assignment for the $x_1, x_2, \ldots, x_n$ a clause is satisfied if at least one of its elements is T. For example, $x_1 \vee \bar{x}_3 \vee x_4$ is satisfied if $x_1 = T, x_3 = F$ or $x_4 = T$.
- Given n boolean variables, m clauses $C_i, i = 1, \ldots, m$ over those variables and a weight $w_i \geq 0$ for each clause the $MAX\ SAT$ problem is to find a truth assignment for the variables that maximizes the total weight of the clauses satisfied. This problem is NP hard.

Randomized Algorithm for MAX SAT

- Consider following algorithm

Random MAX SAT
For $i = 1$ to n do
    Flip a fair coin.
        If "heads", set $x_i = T$.
        else set $x_i = F$.

- Theorem: Let OPT be the weight of the optimal assignment and W the weight of the random assignment. Then

$$E[W] \geq \frac{OPT}{2}.$$

Randomized Algorithm for MAX SAT

Proof: Set

$$I_j = \begin{cases} 1 & \text{if } C_j \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases}.$$

Let l_j be the number of variables in C_j. Then

$$\begin{aligned} E[W] &= E\left[\sum_j w_j I_j\right] \\ &= \sum_j w_j E[I_j] \\ &= \sum_j w_j \Pr(C_j \text{ is satisfied}) \\ &= \sum_j w_j \left(1 - 2^{-l_j}\right) \\ &\geq \frac{1}{2} \sum_j w_j \\ &\geq \frac{1}{2} OPT \end{aligned}$$

An Aside

- MAX 3SAT is the version of MAX SAT in which every clause C_j has exactly three variables in it, i.e. $\forall j, l_j = 3$.
- A theorem says that if there is an approximation algorithm that always returns a solution to the MAX 3SAT that is $> \frac{7}{8} OPT$ then $P = NP$.
- Note that the simple algorithm on the previous page actually returns an assignment whose expectation is $\geq \frac{7}{8} OPT$ when $\forall j, l_j = 3$. Thus, in some sense, it is a best possible approximation algorithm for MAX 3SAT.

Derandomization

Review of Conditional Probabilities and Expectation

- The Probability of X conditioned on A is

$$\Pr(X \mid A) = \frac{\Pr(X \cap A)}{\Pr(A)}.$$

- Example: Roll two dice and let X and Y be the respective number of dots they show. Then

$$\Pr(X + Y = 9 \mid Y \text{ is even }) = \frac{\Pr(X + Y = 9 \text{ and } Y \in \{2,4,6\})}{\Pr(Y \in \{2,4,6\})}$$

$$= \frac{\Pr((X,Y) \in \{(5,4),(3,6)\})}{1/2}$$

$$= \frac{2/36}{1/2} = \frac{1}{9}.$$

Review of Conditional Probabilities and Expectation

- Some basic formulas are

$$E(X \mid A) = \sum_i i \cdot \Pr(X = i \mid A)$$

- and

$$E(X) = E(X \mid A) \cdot \Pr(A) + E(X \mid \bar{A}) \cdot \Pr(\bar{A})$$

where $\bar{A}$ is the event " A does not happen" which has $\Pr(\bar{A}) = 1 - \Pr(A)$.

- Example: For the dice example above

$$E(X + Y \mid Y \text{ is even }) = \sum_i i \cdot \Pr(x + y = i \mid Y \text{ is even })$$

$$E(X + Y) = E(X + Y \mid Y \text{ is even}) \cdot \Pr(Y \text{ is even}) + E(X + Y \mid Y \text{ is odd}) \cdot \Pr(Y \text{ is odd})$$

Review of Conditional Probabilities and Expectation

- Another Example: $C = x_1 \vee x_2 \vee \bar{x}_3$ and S a "random" truth assignment for the x_i as described before.

$$\Pr(C \text{ satisfied} \mid x_1 = F) = \frac{\Pr(x_1 \vee x_2 \vee \bar{x}_3 \text{ satisfied and } x_1 = F)}{\Pr(x_1 = F)}$$
$$= \frac{\Pr(x_2 \vee \bar{x}_3 \text{ satisfied and } x_1 = F)}{\Pr(x_1 = F)}$$
$$= \frac{\Pr(x_2 \vee \bar{x}_3 \text{ satisfied }) \cdot \Pr(x_1 = F)}{\Pr(x_1 = F)}$$
$$= \Pr(x_2 \vee \bar{x}_3 \text{ satisfied })$$
$$= 1 - \left(\frac{1}{2}\right)^2$$
$$= \frac{3}{4}.$$

Review of Conditional Probabilities and Expectation

But

$$\Pr(C \text{ satisfied} \mid x_1 = T) = \frac{\Pr(x_1 \vee x_2 \vee \bar{x}_3 \text{ satisfied and } x_1 = T)}{\Pr(x_1 = T)}$$

$$= \frac{\Pr(x_1 = T)}{\Pr(x_1 = T)}$$

$$= 1.$$

Derandomization of the MAX SAT Algorithm

- Let I be an instance of MAX SAT, S a "random assignment" and W the weight of this random assignment.
- Recall that

$$E(W) = \sum_j w_j Pr\big(C_j \text{ satisfied by } S\big)$$

- The notation $E(W \mid x_1, x_2, \ldots x_k)$ will mean the expected value of W conditioned on the truth assignment of $x_1, x_2, \ldots x_k$ being given.
- By linearity of expectation we have

$$E(W \mid x_1, x_2, \ldots x_k) = \sum_j w_j \Pr\big(C_j \text{ satisfied by } S \mid x_1, x_2, \ldots x_k\big)$$

- where the probabilities are conditioned on the truth assignment of $x_1, x_2, \ldots x_k$ being given.

Derandomization of the MAX SAT Algorithm

- A deterministic algorithm for constructing a truth assignment S such that the weight associated with S satisfies $W \geq \frac{OPT}{2}$.

DeRandomized MAX SAT

For $k = 1$ to n do

  Calculate $W_T = E(W \mid x_1, x_2, \ldots x_{k-1}, x_k = T)$

  Calculate $W_F = E(W \mid x_1, x_2, \ldots x_{k-1}, x_k = F)$

  if $W_T \geq W_F$

    set $x_k = T$

    else set $x_k = F$

Return variable setting S.

Derandomization of the MAX SAT Algorithm

- Proof: For all $0 < k \leq n$

$$E(W \mid x_1, x_2, \ldots x_{k-1})$$
$$= \frac{1}{2} E(W \mid x_1, x_2, \ldots x_{k-1}, x_k = T) + \frac{1}{2} E(W \mid x_1, x_2, \ldots x_{k-1}, x_k = F)$$

- Notice that after setting x_k we must have (why?)

$$E(W \mid x_1, x_2, \ldots x_{k-1}) \leq E(W \mid x_1, x_2, \ldots x_k).$$

- This implies that

$$\frac{OPT}{2} \leq E(W) \leq E(W \mid x_1, x_2, \ldots x_n)$$

- So the truth assignment given by this deterministic algorithm is also a $\frac{1}{2}$ approximation of OPT.

MAX-SAT and p-Biased Coins

- A p-biased coin is one that, for $\frac{1}{2} \leq p \leq 1$, has
- $\Pr(\text{ Heads }) = p,\ \Pr(\text{ Tails }) = 1 - p.$

p-biased random MAX SAT
For $i = 1$ to n do
    Flip a p-biased coin.
    If "heads", set $x_i = T$
        else set $x_i = F$
Return variable setting S.

- Theorem: Let OPT be the optimal solution and W the solution returned by ϕ-biased random MAX SAT, then

$$E[W] \geq \phi \cdot OPT$$

where ϕ is equal to $\min(p, 1 - p^2)$ (with maximum ≈ 0.618 achieved at $p = 1 - p^2$)

MAX-SAT and p-Biased Coins

- Lemma: If $p \geq \frac{1}{2}$ and C is a clause that is not of the form "single variable negated" $(\bar{x}_i)$ then
$$\Pr(C \text{ is satisfied by } S) \geq \min(p, 1 - p^2)$$
- Proof: Let l be the number of variables in C.
- If $l = 1$ then $C = x_i$ for some i and
$$\Pr(C \text{ is satisfied by } S) = \Pr(x_i = T) = p.$$
- If $l > 1$ let n be the number of variables that appear negated in C. Then $l - n$ is the number that do not appear negated and
$$\Pr(C \text{ is satisfied by } S) = 1 - p^n(1 - p)^{l-n} \geq 1 - p^l \geq 1 - p^2.$$

MAX-SAT and p-Biased Coins

- Lemma: Given an instance of MAX-SAT in which every length 1 clause is not negated. Then
$$E[W] \geq \phi \cdot OPT$$
- Proof: Set
$$I_j = \begin{cases} 1 \text{ if } C_j \text{ is satisfied by } S \\ 0 \text{ otherwise} \end{cases}$$
- Then
$$\begin{aligned} E[W] &= E\left[\sum_j w_j I_j\right] \\ &= \sum_j w_j E\left[I_j\right] \\ &= \sum_j w_j \Pr\left(C_j \text{ is satisfied by } S\right) \\ &\geq \sum_j w_j \min(p, 1-p^2) \\ &= \phi \sum_j w_j \\ &\geq \phi \cdot OPT. \end{aligned}$$

MAX-SAT and p-Biased Coins

- The result on the previous page can be improved to work for all instances of MAX-SAT, even those that contain length one clauses that are negated variables.
- Let I be our instance of MAX-SAT. First suppose that variable x_i appears in some length one negated clause, $C_j = \bar{x}_i$, but does not appear in any length one non-negated clause as $C_{j'} = x_i$. We then introduce a new variable u_i and replace every $\bar{x}_i$ in I by u_i and every instance of x_i in I by $\bar{u}_i$. We can then solve this new instance of MAX-SAT without worrying about the negated length one clause C_j (solutions to the original problem are in one-to-one correspondence to solutions to the new problem).

MAX-SAT and p-Biased Coins

- The only hard case to deal with is when x_i appears both as negated clause $C_{j'} = \bar{x}_i$ and non-negated clause $C_j = x_i$.
- We may assume that $w_j \geq w_{j'}$ (why?)
- For all i, if $\bar{x}_i$ exists as its own clause let $v_i =$ wt of $\bar{x}_i$. Otherwise $v_i = 0$.
- Since a truth assignment can only satisfy one of x_i and $\bar{x}_i$ we have

$$OPT \leq \sum_j w_j - \sum_i v_i.$$

- Then

$$\begin{aligned} E[W] &= E\left[\sum_j w_j I_j\right] \\ &= \sum_j w_j \Pr\big(C_j \text{ is satisfied by } S\big) \\ &\geq \sum_{j,\forall i, C_j \neq \bar{x}_i} w_j \Pr\big(C_j \text{ is satisfied by } S\big) \\ &\geq \sum_{j,\forall i, C_j \neq \bar{x}_i} \phi w_j \\ &= \phi \cdot \left(\sum_j w_j - \sum_i v_i\right) \\ &\geq \phi \cdot OPT. \end{aligned}$$