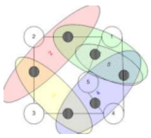
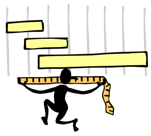
Algorithm Design and Analysis
Victor Adamchik — CS 15-451 — Spring 2014
Lecture 38 — Apr 23, 2014 — Carnegie Mellon University

Approximation Algorithms - II

P ≠ NP

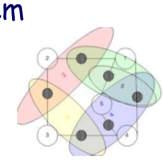




Plan:

Set Cover
MAX-SAT

Set Covering Problem



Given a collection of subsets

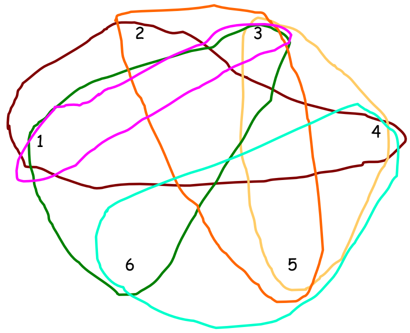
$\mathfrak{U} = \{S_1, \ldots, S_m\}$

Find a min-size subset C such that C covers $\mathfrak{U}$.

Famous NP-complete problem

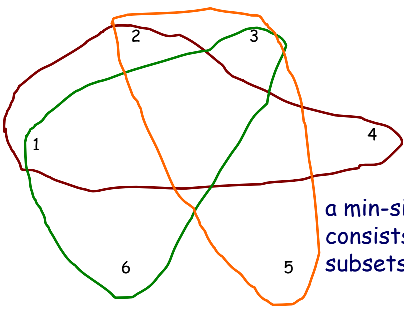
Visualizing Set Cover

$S = \{1, \ldots, 6\}$, $S_1 = \{1,2,4\}$, $S_2 = \{3,4,5\}$, $S_3 = \{1,3,6\}$, $S_4 = \{2,3,5\}$, $S_5 = \{4,5,6\}$, $S_6 = \{1,3\}$



Visualizing Set Cover

$S = \{1, \ldots, 6\}$, $S_1 = \{1,2,4\}$, $S_2 = \{3,4,5\}$, $S_3 = \{1,3,6\}$, $S_4 = \{2,3,5\}$, $S_5 = \{4,5,6\}$, $S_6 = \{1,3\}$



a min-size cover consists of three subsets

Greedy Algorithm

$\mathfrak{U}$ = C [empty cover]
C = {}
While there is uncovered element
    Find the subset S_k covers the most elems
    $\mathfrak{U} \longleftarrow \mathfrak{U} - S_k$
    $C \longleftarrow C \cup S_k$
Return C

How Good of an Approximation?

We'd like to compare the number of subsets returned by the greedy algorithm to the optimal

The optimal is unknown, however, if it consists of k subsets, then any part of the universe can be covered by k subsets!

How Good of an Approximation?

Theorem. If the optimal solution uses k sets, the greedy algorithm finds a solution with at most $k \ln n$ sets.

Proof. Since the optimal solution uses k sets, there must some set that covers at least a $1/k$ fraction of it.

Therefore, after the first iteration $n - n/k$ elems left.

Theorem. If the optimal solution uses k sets, the greedy algorithm finds a solution with at most $k \ln n$ sets.

Proof. (contd)

The algo chooses the most of the elems left

which is $(n - n/k)/k$

Thus, after the second iteration there are

$n - n/k - (n - n/k)/k$ elems left

Observe, $n - n/k - (n - n/k)/k = n (1-1/k)^2$

Theorem. If the optimal solution uses k sets, the greedy algorithm finds a solution with at most $k \ln n$ sets.

Proof. (contd)

More generally, after y rounds, there are at most

$$n (1-1/k)^y$$

elems left.

Choosing $y = k \ln n$, we get

$$n (1-1/k)^y = n (1-1/k)^{k\, y/k} \le n\, e^{-y/k} = n\, e^{-\ln n} = 1$$

MAX-SAT

Given a CNF formula (like in SAT), try to maximize the number of clauses satisfied.

CNF is a conjunction of clauses, where each clause is a disjunction of literals $(X_1 \lor X_2 \lor \dots \lor X_k)$.

Famous NP-complete problem.

Exactly-3-SAT Approximation

Theorem. If every clause has size exactly 3, then there is a simple randomized algorithm that can satisfy at least a 7/8 fraction of clauses.

Proof. Try a random assignment to the variables.

Pr[clause is false] = ?

Since there is only one out of 8 combinations that can make it false, the probability of the clause being false is 1/8.

Exactly-3-SAT Approximation

Theorem. If every clause has size exactly 3, then there is a simple randomized algorithm that can satisfy at least a 7/8 fraction of clauses.

Proof. (cont)

So if there are m clauses total, the expected number satisfied is (7/8) m.

If the assignment satisfies less, just repeat.

With high probability it won't take too many tries before you do at least as well as the expectation.

Exactly-3-SAT Approximation

With high probability it won't take too many tries before you do at least as well as the expectation.

Proof. (cont)

Let Z be the random variable denoting the number of clauses satisfied by a random assignment.

Let $p_k = \Pr[Z = k]$

$$\le (\tfrac{7}{8}m - \tfrac{1}{8}) \sum_{0 \le k < 7/8m} p_k \qquad \le m \sum_{7/8m \le k \le m} p_k = x\,m$$

$$E[Z] = \frac{7}{8}m = \sum_{0 \le k \le m} k\,p_k = \sum_{0 \le k < 7/8m} k\,p_k + \sum_{7/8m \le k \le m} k\,p_k \le (\tfrac{7}{8}m - \tfrac{1}{8}) + x\,m$$

It follows, $x \ge \frac{1}{8m}$

Exactly-3-SAT Approximation

Theorem. If every clause has size exactly 3, then there is a simple randomized algorithm that can satisfy at least a 7/8 fraction of clauses.

Theorem (Hastad, 1997).
If there is an *c-approximation* with c >7/8, then P = NP.

What about MAX-SAT in general?

Suppose we have a CNF formula of m clauses, with m_1 clauses of size 1, m_2 of size 2, etc., and $m = m_1 + m_2 + \ldots$

Theorem. *Then a random assignment satisfies*

$$\sum_j m_j \left(1 - \frac{1}{2^j}\right)$$

clauses in expectation.

Note

$$\sum_j m_j \left(1 - \frac{1}{2^j}\right) \ge \frac{1}{2} \sum_j m_j \ge \frac{1}{2} OPT$$

Deterministic SAT Approximation

Suppose we have a CNF formula of m clauses, with m_1 clauses of size 1, m_2 of size 2, etc., and $m = m_1 + m_2 + \ldots$

Pick X_1, for each of the two possible settings we then calculate the expected number of clauses satisfied if we were to go with that setting, and then set the rest of the variables randomly.

$$E[\,] = \frac{1}{2} E[X_1 = \text{true}] + \frac{1}{2} E[X_1 = \text{false}]$$

$$E[X_1 = \text{true/false}] \ge E[\,] \ge \frac{1}{2} OPT$$

Deterministic SAT Approximation

$$\overline{X_1} \wedge (X_1 \vee \overline{X_2}) \wedge (X_1 \vee X_2 \vee \overline{X_3}) \wedge (\overline{X_1} \vee \overline{X_2} \vee \overline{X_3}) \wedge (X_1 \vee X_2 \vee X_3)$$

If we set X_1=false, we get in expectation

$$1 + \frac{1}{2} + \frac{3}{4} + 1 + \frac{3}{4} = 4$$

If we set X_1=true, we get in expectation

$$0 + 1 + 1 + \frac{3}{4} + 1 = 3\frac{3}{4}$$

So, we choose X_1=false

Deterministic SAT Approximation

So, we choose X_1=false

$$\overline{X}_1 \wedge (X_1 \vee \overline{X}_2) \wedge (X_1 \vee X_2 \vee \overline{X}_3) \wedge (\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3) \wedge (X_1 \vee X_2 \vee X_3)$$

$$\overline{X}_2 \wedge (X_2 \vee \overline{X}_3) \wedge (X_2 \vee X_3)$$

If we set X_2=false, we get in expectation

$$1 + \frac{1}{2} + \frac{1}{2} = 2$$

If we set X_2=true, we get in expectation

$$0 + 1 + 1 = 2$$

Deterministic SAT Approximation

$$E[] = \frac{1}{2}E[X_1 = \text{true}] + \frac{1}{2}E[X_1 = \text{false}]$$

Fix X_1 to the setting that gives us a larger expectation.

Now go on to X_2 and do the same thing, setting it to the value with the highest expectation, and then X_3 and so on.

Since we always pick the setting whose expectation is larger, this expectation never decreases

Using LP

We can set this problem as an integer programming

We define $X_k \in \{0,1\}$ for each variable and each $Z_k \in \{0,1\}$ for each clause C_k.

The goal $\max \sum_j m_j Z_j$

Subject to $Z_j \le \sum_{i \in C_j^+} X_i + \sum_{i \in C_j^-} (1 - X_i)$

where C_j^+ are the variables that appear in C_j without negation.

Relaxation

Since ILP is NP-complete, we solve a relaxation of the problem.

We define $0 \le X_k \le 1$ and $0 \le Z_k \le 1$

After solving LP we get $X_k = p_k$ and do a probabilistic rounding of the result.

$$\Pr[C_j = \text{false}] = \prod_{i \in C_j^+} (1 - p_i) \prod_{i \in C_j^-} p_i$$

Next we use the arithmetic and geometric mean

$$\prod_i p_i \le \left(\frac{1}{n} \sum_i p_i \right)^n$$

Relaxation

$$\Pr[C_j = \text{false}] = \prod_{i \in C_j^+} (1 - p_i) \prod_{i \in C_j^-} p_i \le \frac{1}{\ell_j^{\ell_j}} \left(\sum_{i \in C_j^+} (1 - p_i) + \sum_{i \in C_j^-} p_i \right)^{\ell_j}$$

ℓ_j - # of literals. We simplify this using constrains on Z_j

$$Z_j \le \sum_{i \in C_j^+} X_i + \sum_{i \in C_j^-} (1 - X_i)$$

to get $\Pr[C_j = \text{false}] \le \left(\frac{\ell_j - Z_j}{\ell_j} \right)^{\ell_j}$

and

$$\Pr[C_j = \text{true}] \ge 1 - \left(1 - \frac{Z_j}{\ell_j} \right)^{\ell_j} = \left(1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right) Z_j$$

Relaxation

$$\Pr[C_j = \text{true}] \ge \left(1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right) Z_j \ge \beta Z_j$$

$$\beta = 1 - \frac{1}{e} \approx 0.63$$

From here we get a β-*approximation.*

$$E[] = \sum_j m_j \Pr[C_j = \text{true}] \ge \beta \sum_j m_j Z_j \ge \beta\, OPT_{relaxed} \ge \beta\, OPT$$

Theorem (1994). Randomized rounding is a 0.63-*approximation* algorithm.

Non-linear rounding

We don't have to take the output of the linear program as the probabilities for X_k.

We could use some function to generate probabilities for X_k.

$$\Pr[C_j = \text{false}] = \prod_{i \in C_j^+} (1 - f(X_i)) \prod_{i \in C_j^-} f(X_i)$$

This will result in a $\frac{3}{4}$-*approximation* algorithm

The best result is a 0.78-*approximation* algorithm