Lecture 23: May 15, 2001

Today: Approximation algorithms

Coping with NP-hardness

- Exponential time algorithms for small inputs (e.g., $(100/99)^n$ time is not bad for $n \leq 1000$)
- Polynomial time algorithms for some (e.g., random) inputs
- Polynomial time algorithms for all inputs, but *incorrect* (popular approach, but not advised)
- Polynomial time algorithms for all inputs, but *approximate*

Approximation algorithms

An algorithm A is *ρ-approximate*, if for any input, the cost C_A of the solution it produces is at most ρ times the cost C^* of the optimal solution.

E.g., an algorithm which produces TSP tour at most 10% longer than the optimal one.

Similar definition for the maximization problems.

We are only interested in polynomial time approximation algorithms.

Example of approximation algorithms

Are there any approximation algorithms ? Yes, almost every NP-hard problem has a corresponding approximation algorithm. In particular:

- TSP: has 2-approximation algorithm (in CLR). Can be improved to 1.5.
- Set Cover: has $O(\log n)$-approximation algorithm.

Set cover

- Given: a collection of sets $S_1 \dots S_m \subset X$, $\cup_i S_i = X$, $|X| = n$
- Goal: find $C \subset \{1 \dots n\}$ such that $\cup_{i \in C} S_i = X$, minimizing $|C|$

Very natural problem, e.g.:

- X- the set of tasks
- S_i - the set of tasks which can be performed by person i
- Want to find the minimal team which can perform all tasks

Unfortunately, problem is NP-complete (e.g., by reduction from Vertex Cover)

Greedy algorithm

Repeat until all elements are covered:

- choose a new set S_i which contains largest number of yet uncovered elements
- add i to C
- marked all elements from S_i as covered

Natural algorithm, but can be fooled (e.g., consider sets $\{1, 2\}, \{3, 4\}, \{5, 6\}, \{1, 3, 5\}$).

Nevertheless, it is an $O(\log n)$-aproximation algorithm

Proof

Let k be the size of the minimal cover.

Fact: At any point, there is at least one set S_i which covers $\geq 1/k$ fraction of yet uncovered elements.

Proof: If each set covered $< 1/k$ fraction of (yet uncovered) elements, then there would be no way to cover X using only k sets.

$O(\log n)$-approximation

Let u_i be the number of uncovered elements after the ith step. We know that

$$u_{i+1} \leq (1 - 1/k) u_i$$

Initially, $u_0 = n$, thus

$$u_i \leq (1 - 1/k)^i n$$

Therefore, for $i = k \ln n$ we have

$$u_i \leq e^{-\ln n} n = 1$$

and thus after $k \ln n + 1$ steps the algorithm covers all elements and stops.