Advanced Algorithms

Lecture by Shay Mozes
Typeset by Steven Karas

2016-11-20

Homework **must** be submitted as PDF. Proofs must be complete, and not leave out "trivial" details.

Question 3, we needed to translate back to the original weights.

Also note that "justify" means "prove".

1 Solvable problems

All problems that we can solve. Counterexample: Halting problem.

2 P

Complexity class of problems solvable in polynomial time.

Formal Definition Set of all problems solvable with some algorithm with complexity $O(n^k)$ for some $k \in \mathbb{R}$

3 NP

Complexity class somewhere between P and PSPACE.

History Discovered in early 1970s by Karp, Cook, Levin. Discussed the problems directly, rather than the algorithms or theoretical machine model. As of 2016, no one has proven either that P $\subset$ NP, nor P = NP

Hamiltonian Cycle A Hamiltonian cycle in an undirected graph is a sequence of edges/vertices that visits every vertex exactly once.

Formal Definition Lacking...

3.1 NP-Complete

Complexity class of problems in NP that are at least as hard as all other problems in NP, and in NP. If any of the NP-Complete problems can be solved in polynomial time, then all problems in NP-Complete can be solved in polynomial time.

If a problem is NP-Complete, it is recommended to not attempt to solve it, but rather use an approximation, or circumvent the problem some other way.

Reduction S is reducible in polynomial-time to Q ($S \leq_P Q$) if given a black box that solves Q in polynomial time, it is possible to solve S in polynomial time.

Formal Definition S is NP-Complete if:

- $S \in$ NP
- Every problem R in NP is reducible to P: $R \leq_P S, \forall R \in$ NP

Practice: prove that if $S \leq_P Q$ and S is NP-Hard, then Q is NP-Hard

3.2 Optimization vs Decision

P, NP are formally defined in terms of decision problems (problems with a yes/no answer). Many of our opt/search problems can be recast in terms as a decision problem.

Example: Hamiltonian cycle

3.3 SAT problem

The first problem to be proven as NP-Complete. Given a boolean formula of n variables, can we assign values such that the formula is TRUE?

Example

$$((x_1 \wedge x_2) \vee \neg((\neg x_1 \wedge x_3) \vee x_4) \wedge \neg x_2$$

TODO: fill in from slide 15

Cook-Levin Theorem SAT is NP-Complete. Note: Argued from first principles, not reduction. Proof not given here.

3.4 Graph Coloring

A coloring of a graph $G(V, E)$ is a function $c : V \to \mathbb{N}$ such that for any edge $(u, v) \in E$, $c(v) \neq c(u)$. Given a graph G and integer k, is there a coloring of G with k colors?

The chromatic number is the minimum number of colors needed

3.5 k-Clique Problem

A clique in a graph G is a subset of vertices fully connected to each other (i.e. a complete subgraph of G). How large is the largest clique in the graph?

NOTE that k-Clique is a polynomial problem for a given k.

3.6 Vertex Cover

A vertex cover of a graph G is a set of vertices incident to every edge in G. What is the minimal cover size?

As a decision problem Does a vertex cover of size k exist?

3.7 Traveling Salesman Problem

Optimization variant: a salesman must travel to n cities, visiting each city exactly once and finishing back where he started. Minimize the travel time.

Modeled as a complete graph with cost $c(i, j)$ to go from city i to city j.

Decision version Ask if there exists a path p with cost $c_p \leq k$

3.8 Partition

Given a set of integers, whose total sum is 2S, can be partition them into two sets, each of which adds up to S?

3.9 Subset-Sum

3.10 Independent Set

Given a graph $G = (V, E)$ and k; Is there a subset S of at least k vertices in V such that no pair of vertices in S has an edge between them.

Optimization problem: find a maximum size independent set of

NOTE: maximal is local, maximum is global[1].

3.11 Steiner Tree

Given a graph $G = (V, E)$, a set of vertices $T \subset V$, and a bound B. Is there a tree connecting all the vertices of T of total weight at most B?

Note that $T = V$ is polynomial as it is equivalent to the MST problem.

3.12 Exact Cover

Given a set $U = \{u_1, ..., u_n\}$ and subsets $S_1, S_2, ..., S_m \subseteq U$. Determine if there is a set of disjoint sets whose union is U.

3.13 Bin Packing

Given a set of numbers $A = \{a_1, ..., a_n\}$, a capacity B, and the number of bins K. Determine if A can be partitioned into $S_1, ..., S_K$ such that for all i: $\sum_{j \in S_i} a_j \leq B$

[1] This may appear in the homework

3.14 Dealing with NP-Hardness

Stop looking for efficient algorithms. Either trade off optimal solutions for efficient approximations, or sidestep the problem.

- Backtracking - build every solution in turn
- Branch and bound - trim off unreasonable branches from tree search
- Dynamic programming
- Change the problem - parameterized complexity, solving a more structured problem (MST instead of Steiner Tree)
- Guaranteed Approximations - performance guarantees in return for results that are within some bound of the global optimum
- Heuristic Approximation - using heuristics in return for decent average results

4 Approximation Algorithms

Just because a problem is NP-Complete, doesn't mean we can't find an approximate solution efficiently. Even more so, we can get some guarantees on performance and optimality.

4.1 Additive Error

For a few NP-Hard problems, there are approximation algorithms that produce an **almost optimal** solution. One that is only off from the global optimum by some additive constant.

Minimization problems: $Alg(I) \leq Opt(I) + c$

Maximization problems: $Alg(I) \geq Opt(I) - c$

Edge Coloring An edge coloring of the graph $G = (V, E)$ is the assignment of colors to edge such that if e_1 and e_2 share an endpoint, then $c(e_1) \neq c(e_2)$.

Let Δ be the maximal degree of some vertex in G. There is a proof that for any graph G, the minimal color set is Δ or $\Delta + 1$. However, to tell which is NP-Hard.

However, there is a polynomial algorithm that colors any graph G using $\Delta + 1$ colors. In this case, $Alg(I) \leq Opt(I) + 1$

4.2 r-Approximation

All the approximation algorithms we will cover are factor r:

Vertex Cover,

Minimization Definition Alg is a r-approximation if $Alg(I) \leq r \cdot Opt(I)$ for any instance I.

Example: TSP TSP is a minimization problem. If we have an algorithm A that finds, for any graph, a tour

Example: MST This is trivially exact....

Maximization Definition Alg is a r-approximation if $Alg(I) \geq (1/r) \cdot Opt(I)$ for any instance I. Sometimes an alternative notation is used.

Example: Maximum Clique If we have an algorithm A that finds for any graph a clique

4.3 Matching

A matching in a graph G is a subset M of E such that the degree of each vertex in $G' = (V', M)$ is 0 or 1. Basically, matching vertices to at most 1 other vertex.

Example From slide 8.

4.4 Example: Vertex Cover

Given $G = (V, E)$, find a minimum sized subset W of V such that for every (v, u) in E, at least one of v or u is in W.

We are willing to end up with a vertex cover W that is not the minimum vertex cover. But, we don't want it to be too large, and we want to be able to find it in polynomial time.

```
While E is not empty:
1. select an arbitrary edge (u,v)
2. add both u and v to the cover
3. delete all edges incident to either u or v
```

Proof of Correctness Verbal proof in class. See video around 8:19pm

Proof of Approximation This is a 2-approximation. Let c be the number of iterations. The VC has size $2c$. the edges selected in step 1 form a matching of size c. If we consider a graph G' in which only these edges appear, then if we want to cover these edges, then any minimal cover must include at least c vertices.[2]

We want to show that Opt has a lower bound, and that Alg is approximate to within some upper bound wrt Opt.

Complexity Analysis Verbal proof $O(n^2)$. So long as the complexity is polynomial, we don't care that much.

Greedy Variant Sort vertices by degree, and select the highest degree. However, approximation ratio of this approach is not bound by r. For any r, there exists some graph for which the VC chosen by the algorithm is r-times larger than the optimal VC.

[2] Good verbal formal proof at 8:31pm in video

Proof For some parameter m, the first layer of our constructed graph has $m!$ vertices. This first layer is the optimal vertex cover. We then build a second layer with $\frac{m!}{2}$ vertices. Each vertex in the second layer is connected to two vertices in the first layer. As such, each vertex in the second layer has degree 2. Each layer i has $\frac{m!}{i}$ vertices connected to i vertices from the first layer. There are m such layers, with the last one having $\frac{m!}{m} = (m-1)!$ vertices, of degree m. Trivially, the optimal vertex cover is $Opt(I) \leq m!$. In this case[3]:

$$Alg(I) = \sum_{i=2}^{m} \frac{m!}{i} = m! \sum_{i=2}^{m} \frac{1}{i} = \Theta(m! \log m)$$

4.5 Example: Euclidean TSP

Given n points on a Euclidean plane, find the shortest tour. For each pair of vertices a,b; we are given the distance dist(a,b) from a to b.

Triangle Inequality Distances in the plane satisfy the triangle inequality: $dist(a,b) \leq dist(a,c) + dist(c,b)$. This means that indirect paths are at best as good as a direct path.

NOTE: the weight of a minimum spanning tree is always less than the weight of the optimal tour.[4]. Our approximation is the DFS of the MST.

Proof of Correctness The DFS tour defines a spanning cycle of the graph, which includes each edge in its cycle exactly twice. Therefore, the length of the the DFS tour is exactly twice the length of the MST. However, we may visit some cities more than once.

To get a legal TSP tour, we must jump from one leaf of the MST to the next. Because of the triangle inequality, these jumps can only reduce the total length of the tour.

[3]Proof with drawings on whiteboard starting at 8:37pm
[4]Verbal proof at 8:53pm