451/651 Lecture 16 – Approximation Algorithms

Today we examine several classical **NP**-complete problems from the point of view of obtaining approximate solutions that are "close" to optimum.

To do so we'll switch back to the "solution" version of problems instead of the "decision" version.

Job Scheduling to Minimize Load

- You have m identical machines
- You want to schedule n jobs. Each job $j \in \{1, 2, \ldots, n\}$ has a processing time $p_j > 0$.
- You want to partition the jobs among the machines to minimize the load of the most-loaded machine.
- In other words, if S_i is the set of jobs assigned to machine i, define the *makespan* of the solution to be $\max_{\text{machines } i}(\sum_{j \in S_i} p_j)$.
- You want to minimize the makespan of the solution you output.

Alternative Model. Blocks

- You have n blocks. Block j has height p_j.
- You have to stack the blocks into m separate stacks so as to minimize the height of the tallest stack.
- The *makespan* is the height of the tallest stack.

Wait, first this is **NP**-complete

The SUBSET SUM problem is this: Given integers $w_1, \ldots, w_n$ and T, does there exist a subset of the w_i integers that sums exactly to T?

(See the last lecture for a proof that this is **NP**-complete.)
(Also recall that we saw a DP algorithm for this that ran in $O(nT)$ time.)

Easy exercise to prove that we can reduce the SUBSET SUM problem to the decision version of the MAKESPAN problem.

Algorithm 1: Greedy

Pick any block. Place it in the currently lowest stack.

Theorem: Greedy is at most 2 times optimum.

Proof:

Algorithm 2: Sorted Greedy

Same as Greedy, except place the blocks in decreasing order of size.

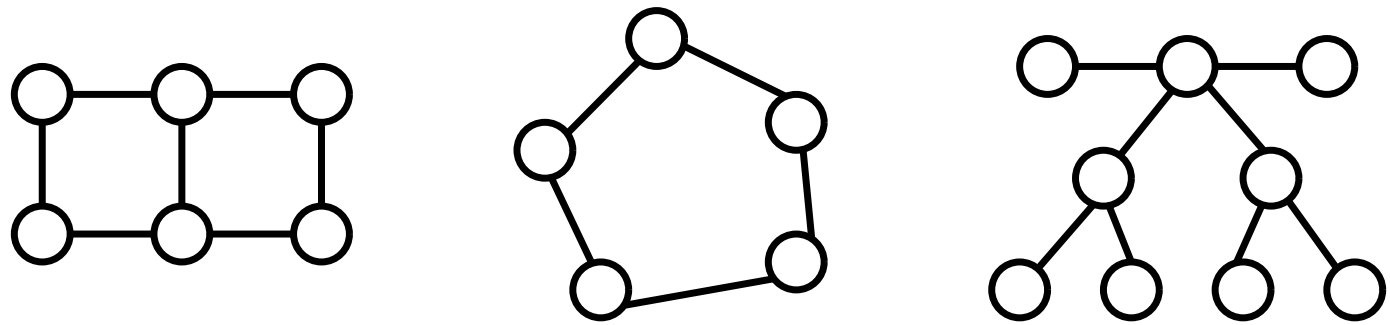
Theorem: Greedy is at most 1.5 times optimum.

Proof:

Vertex-Cover

Vertex-Cover Solution version: Given a graph G, find the smallest set of vertices such that every edge is incident to at least one of them.

Vertex-Cover Decision version: Given G and integer k, does G contain a vertex cover of size $\leq k$?



Exercise: Find a vertex cover in the graphs above of size 3. Show that there is no vertex cover of size 2 in them.

Approximation Algorithms for Vertex-Cover

If the optimal algorithm uses k^* vertices, our goal is to achieve some constant ck^* vertices, for as small a c as possible.

In fact we'll see how to achieve $c = 2$. Nobody knows if it's possible to achieve a factor of $c = 1.99$.

Greed worked before, so let's try it again:

Greedy 1: Pick an arbitrary vertex with at least one uncovered edge incident to it, put it into the cover, and repeat.

What would be a bad example for this algorithm?

Greedy Algorithms for Vertex-Cover, Contd.

Greedy 2: Picking the vertex that covers the *most* uncovered edges.

Can produce a solution $\Omega(\log n)$ times larger than optimal.

(See lecture notes for the construction.)

A 2-Approximation for Vertex-Cover

VC Alg 1: Pick an arbitrary edge. We know any vertex cover must have at least 1 endpoint of it, so let's take *both* endpoints. Then, throw out all edges covered and repeat. Keep going until there are no uncovered edges left.

Theorem: VC Alg 1 gives a 2-approximation for Vertex-Cover

Proof:
The key is to characterize (in the right way) the set of edges that VC Alg 1 finds.

Another 2-Approximation for Vertex-Cover

VC Alg 2: Set up an LP. For each node i define a variable $0 \leq x_i \leq 1$. For each edge (i, j) write a constraint $x_i + x_j \geq 1$. The objective function is $\min(x_1 + \cdots + x_n)$.

Solve the LP. Now for each variable $x_i \geq 1/2$ round it to 1. Round the others to 0.

Claim 1 The algorithm finds a valid vertex cover.

Claim 2 The one it finds is a 2-approximation.

Hardness of Approximating Vertex-Cover

Interesting fact: nobody knows any algorithm with approximation ratio $2 - \epsilon$, for constant $\epsilon > 0$ Best known is $2 - O(1/\sqrt{\log n})$, which is $2 - o(1)$.

There are results showing that a good-enough approximation algorithm will end up showing that P=NP. Clearly, a 1-approximation would find the exact vertex cover, and show this. Håstad showed that if you get a $7/6$-approximation, you would prove P=NP. This $7/6$ was improved to 1.361 by Dinur and Safra.

Beating $2 - \varepsilon$ has been related to some other open problems (it is "unique games hard"), but is not known to be NP-hard.

Metric Traveling Salesperson Problem

METRIC TSP is: Input n points with distances d_{ij} between any pair (which is a *metric* satisfying identity, symmetry, and triangle-inequality). This is a matrix D.

Solution version: Find a permutation of $1, 2, \ldots, n$ that such that visiting the cities in that order has minimum length.

MST-Based Algorithm: Think of the input as a complete graph G with a distances defined by d_{ij}.

1. Find a MST of G.
2. Use the "right hand rule" to walk around the tree.
3. Delete duplicates, giving a permutation.

MST-Based Algorithm for Metric TSP Contd.

Draw an example here:

MST-Based Algorithm for Metric TSP Contd.

Theorem: The MST-Based Algorithm is a 2-approximation for Metric TSP.

Proof:

Observe OPT $\geq$ MST.

The tour constructed is at most 2$*$MST.

Christofides' Algorithm for Metric TSP

We can improve this to a 1.5-approximation.

1. Find an MST T of G.
2. Let $S =$ the set of vertices of odd degree in T.
3. Find the minimum cost matching M of S.
4. Combine the edges of the T and M together.
5. All vertex are now of even degree.
6. Find an Euler tour of this set of edges.
7. Use shortcutting as before to construct a TSP tour.

Christofides' Algorithm for Metric TSP Contd.

Theorem: The algorithm is a 1.5-approximation for Metric TSP.

Proof:

Improved Approximation Algorithm for Metric TSP

A paper from 2020 by Karlin, Klein, and Oveis Gharan presents an improved algorithm, which gives a 1.4999999999999999999999999999999999 approximation. This is not an April fools joke. See `https://arxiv.org/abs/2007.01409`.