

How do we deal with NP-complete (or other hard) problems?

Heuristics - but can we guarantee we'll at least get close to the right answer

Euclidean traveling salesman problem

↳ compute MST + traverse in pre-order (like DFS) - Lec 7

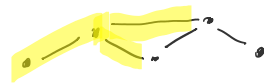
Recall: Vertex cover is NP-complete.

3SAT → Independent Set → Vertex Cover

Given undirected graph G , find smallest set S of vertices so every edge has an endpoint in S

Greedy Attempt 1: Repeatedly pick a node that covers a new edge.

Fails: You might pick all but the last node if you choose badly.

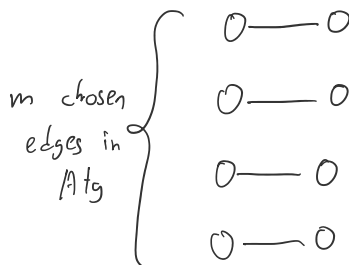


Better greedy alg 2: Repeatedly pick an uncovered edge (u, v) , + add both u + v to S . (and cover edges incident to either u, v)

Thm. Alg 2 is a 2-approximation.

i.e. it returns a vertex cover of size \leq twice the optimal.

proof.



Lemma: If Alg 2 chooses m edges, then the optimal set cover S^* must have $|S^*| \geq m$. (lower bound)

pf. The m chosen edges share no endpoints, so to cover just these edges, we need at least m nodes. \Rightarrow any set cover must have $\geq m$ nodes. $\Rightarrow |S^*| \geq m$. □

Notice that Alg 2 returns a set S with $|S| = 2m$ by construction.

$\Rightarrow |S| \leq 2|S^*|$.

\Rightarrow Alg 2 is a 2-approximation □

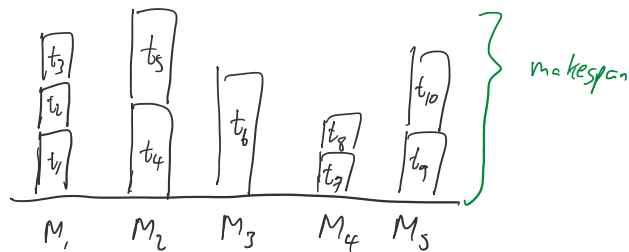
→ Alg C is a C approximation



Note that we don't know what size the actual vertex cover is, just that our greedy edge selection alg gets close.

Problem (Minimum Makespan) Given n jobs of length t_1, t_2, \dots, t_n , and m machines M_1, \dots, M_m , schedule the jobs on the machines to minimize the makespan.

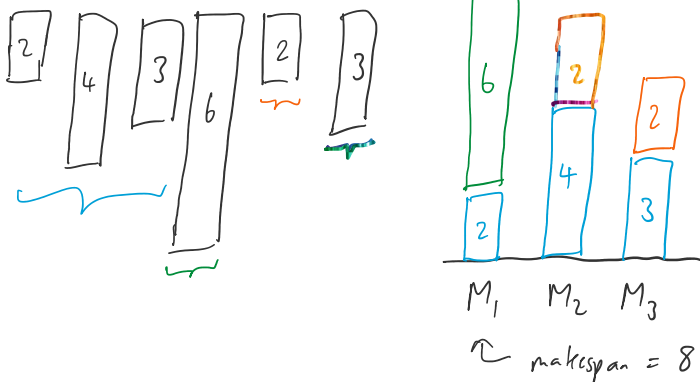
Def. The makespan is the total time to complete all jobs, running in parallel.



Minimizing Makespan is NP-complete. (proof left to student)

Greedy Alg 1: Given items t_1, \dots, t_n , schedule items in list order on least used machine, so far.

Example:



Initially all machines are unused.

Then schedule by whichever machine is least occupied

Let $G(I)$ be the makespan obtained using Alg 1 for instance I .

Let $OPT(I)$ be smallest possible makespan.

can't be better than optimal

Goal: Prove \forall instance I , $G(I) \leq \alpha OPT(I)$ for some $\alpha \geq 1$.

Goal: Prove \forall instance I , $G(I) \leq \alpha \text{OPT}(I)$ for some $\alpha \geq 1$.
 α -approximation

Lower bounds:

- Some machine must do more work than $t_j \forall j$
 $\Rightarrow \text{OPT}(I) \geq \max_j t_j$
- Some machine must do more work than average
 $\Rightarrow \text{OPT}(I) \geq \frac{1}{m} \sum_j t_j$

Thm: $G(I) \leq 2 \text{OPT}(I)$.

pf. Let M be the machine with max load in greedy sol.
 Let t_j be last job assigned to M .
 Let L_i be the load the greedy alg puts on machine i
 ($G(I) = L_M$)

When t_j was placed on M , M had lightest load. Why?

Every machine had load $\geq G(I) - t_j$

$$\Rightarrow m(G(I) - t_j) \leq \sum_k L_k \leq \sum_j t_j$$

less than load at item j being placed
load on machines at end
total load from items

not counting the last item placed on M , it had less than average load.

$$\Rightarrow G(I) - t_j \leq \frac{1}{m} \sum_j t_j \leq \text{OPT}(I)$$

Also $t_j \leq \text{OPT}(I)$. (other lower bound)

$$\Rightarrow G(I) \leq \text{OPT}(I) + t_j \leq 2\text{OPT}(I)$$



This 2-approx proof was more involved, since we had to split up the solution to use two different lower bounds

It turns out our analysis is close to tight in that greedy will sometimes give sol that are nearly twice the optimal.

^(sorted)
Greedy Alg 2: First sort jobs by decreasing length, then apply greedy alg 1.

Intuition: place big jobs first, then fit in little ones where-ever.

Thm. Sorted greedy alg 2 is a $\frac{3}{2}$ -approximation alg.
i.e. it gives $G(I) \leq \frac{3}{2} \text{OPT}(I)$

proof. Let t_1, \dots, t_n be the jobs sorted by length in decreasing order.

Notice: When t_{m+1} is placed on a machine, there is already another longer job on that machine.

$$\Rightarrow \text{OPT}(I) \geq 2t_{m+1}$$

Now if $n \geq m+1$, then the last item t_j placed on M has $t_j \leq t_{m+1} \leq \frac{1}{2} \text{OPT}(I)$.

$$\Leftrightarrow G(I) \leq \text{OPT}(I) + t_j \leq \frac{3}{2} \text{OPT}(I) \text{ by new lower bound.}$$

If you are placing tasks, important to schedule bigger ones first.