15451 Spring 2023

Approximation Algorithms

Elaine Shi

What can we do for NPC problems?

What can we do for NPC problems?

Design poly-time approximation algorithms

Consider the solution version (rather than the decision version)

m machines, n jobs, j-th job takes time p_j

m machines, n jobs, j-th job takes time p_j

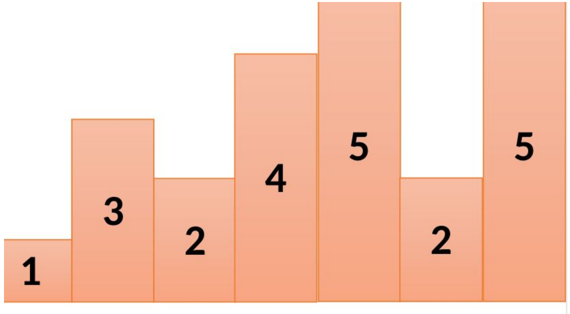
partition jobs to the machines to minimize the *makespan*, defined as

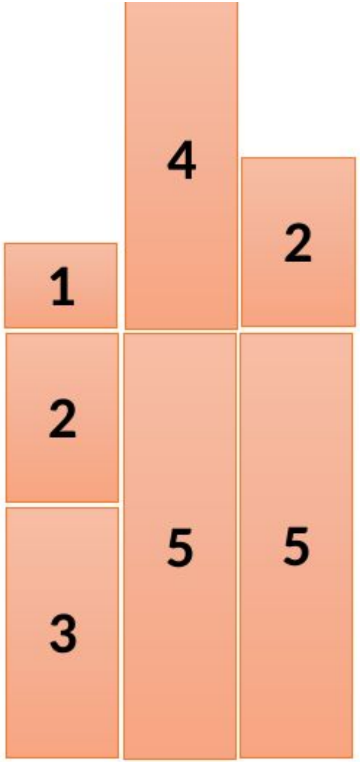
$$\max_{i \in [m]} \left(\sum_{j \in S_i} p_j \right)$$

where S_i is the set of jobs assigned to machine i

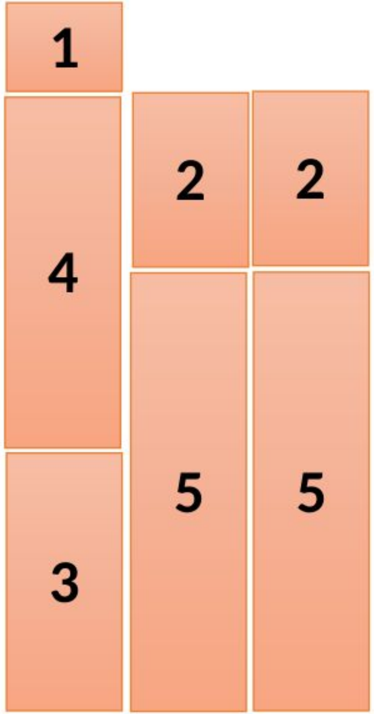
Example

m = 3





Makespan = 9



Makespan = 8

Claim:

The job assignment problem is NPC

Claim:

The job assignment problem is NPC

Proof: reduce subset sum to job assignment

Subset sum:

given $a_1, a_2, \dots a_n$, is there a subset that sum up to $\frac{1}{2} * (a_1 + a_2 + \dots + a_n)$?

Greedy:

Take any unassigned job, give it to the machine with current minimum load

Claim:

Greedy achieves **2**-approximation, i.e., achieves a makespan at most 2 times the optimal

Claim:

Greedy achieves 2-approximation, i.e., achieves a makespan at most 2 times the optimal

Let i^* be the most loaded machine, let $\mathbf{j^*}$ be the last job assigned to it, let $\mathbf{L}$ be the load on i^* before we assign j^* to it

Claim:

Greedy achieves 2-approximation, i.e., achieves a makespan at most 2 times the optimal

Let i^* be the most loaded machine, let $\mathbf{j^*}$ be the last job assigned to it, let $\mathbf{L}$ be the load on i^* before we assign j^* to it

$ALG = p_{j^*} + L$

Claim:

Greedy achieves 2-approximation, i.e., achieves a makespan at most 2 times the optimal

Let i^* be the most loaded machine, let $\mathbf{j^*}$ be the last job assigned to it, let $\mathbf{L}$ be the load on i^* before we assign j^* to it

$ALG = p_{j^*} + L$

Observe that $OPT \geq p_{j^*}$, $OPT \geq L$

Can we do better?

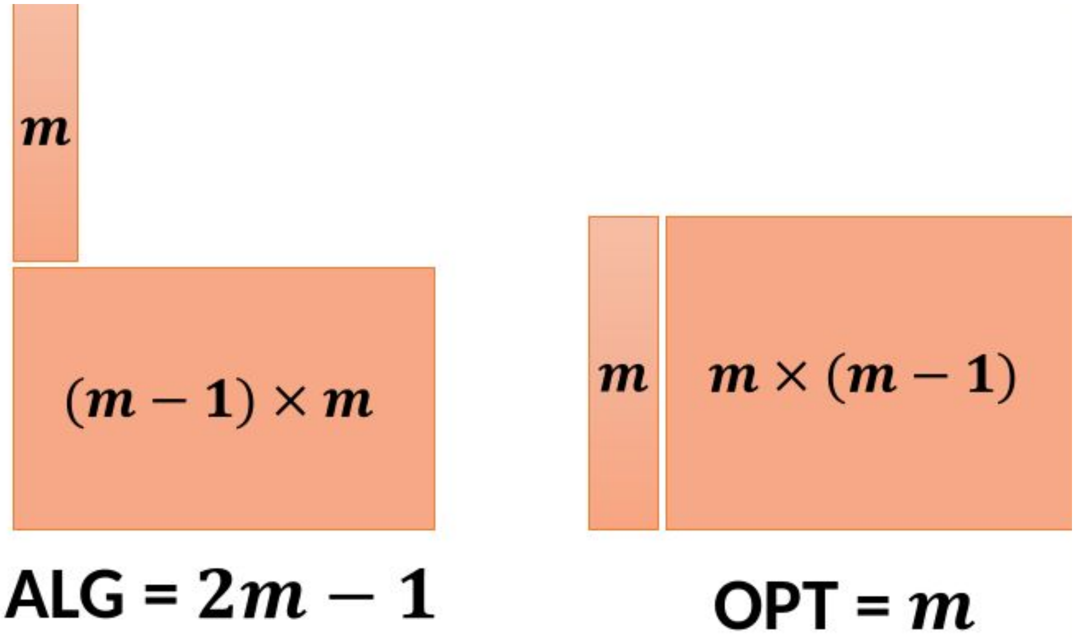
Can we do better?

What is a bad case?

Can we do better?

What is a bad case?

m(m-1) jobs of size 1,
1 job of size m



m(m-1) jobs of size 1,
1 job of size m

Sorted Greedy:

Run greedy largest unassigned job first

Let i* be the most loaded machine, let j* be the last job assigned to it

Want to show:

$$p_{j*} + L \leq 1.5 \cdot OPT$$

Suffices to show: $p_{j*} \leq OPT/2$

We may assume $L > 0$ (why?)

Every machine has at least one job assigned before, and its size is at least p_{j^*}

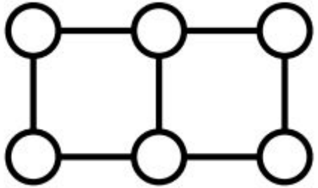
We may assume L > 0 (why?)

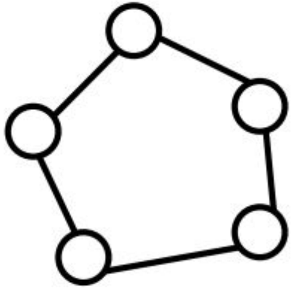
Every machine has at least one job assigned before, and its size is at least p_{j^*}

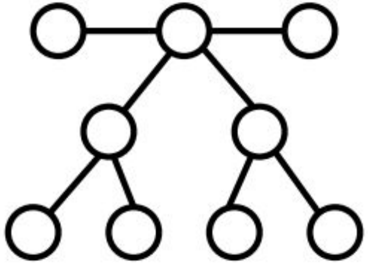
$$2p_{j^*} \leq OPT$$

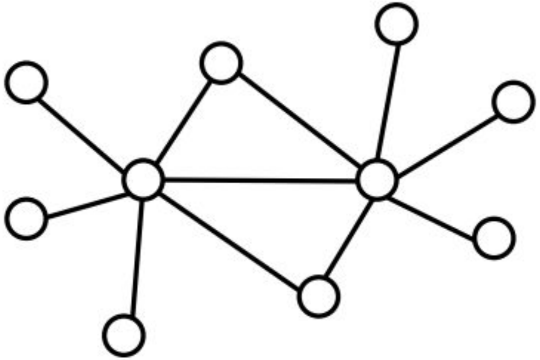
Vertex-Cover:
Given a graph G, find the smallest set of vertices such that every edge is incident to at least one of them.

Decision problem: Given G and k, does G contain a vertex cover of size ≤ k?









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

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

Does this give a good approximation?

Pick the vertex that covers the most uncovered edges

Pick the vertex that covers the most uncovered edges

Achieves only O(log n) approximation

A 2-approximation algorithm

Pick an arbitrary edge. Add both endpoints.
Throw out all edges covered and repeat.
Continue until no uncovered edges left.

Proof:

Alg finds a matching M, with |M| edges

Observe that

$$|M| \leq OPT$$
$$Alg = 2\,|M|$$

Another 2-approximation algorithm

- **Find an LP solution (possibly fractional)**

  $0 <= x_i <= 1$, for each edge (i, j), $x_i + x_j >= 1$

  Minimize $\Sigma_i\, x_i$

- **Round the LP solution:**

  If $x_i >= ½$, set to 1, else set to 0

Example

Proof:

LP ≤ OPT

Rounded solution ≤ 2 LP

Known Results

Best known approximation:

$$2 - \Theta(1/\sqrt{\log n})$$

7/6 approximation is NP hard [Hastad]

Improved to 1.361 [Dinur and Safra]

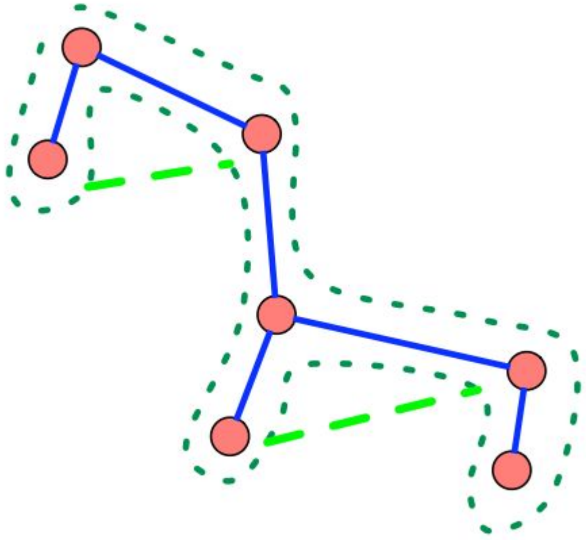
Metric Traveling Salesman Problem

Find the shortest path to visit n cities, each exactly once, returning to where you started.

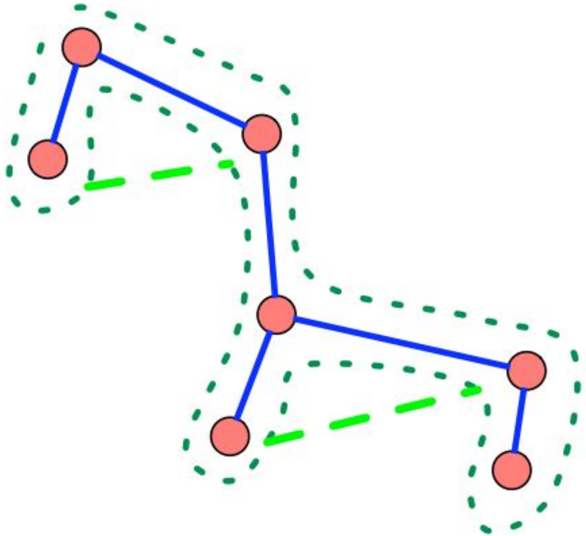
Metric: distances are symmetric and obey triangle inequality

Compute MST

Output a pre-order traversal of the MST



This algorithm achieves 2 approximation.

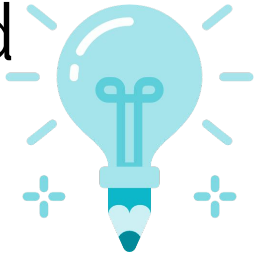


Christofide's algorithm

Compute the MST T.

Compute a minimum weight perfect matching M between the vertices of odd degree in T.

G = T ∪ M. Return the TSP tour constructed from shortcutting an Eulerian tour on G.



Karlin, Klein, and Oveis Gharan improves it to 1.49999999999999999999999999999999999 approximation.

(Not an April fools joke.
See https://arxiv.org/abs/2007.01409.)