• For a minimization problem, an \(\alpha\)-approx algo
A takes any instance \(I\) of the problem and
returns a solution \(S\) s.t.
\[
\text{cost}(S) \leq \alpha \cdot \text{opt}(S)
\]
\(\text{opt}\) is cost of opt soln
for instance \(I\)

Note: \(\alpha > 1\), and closer it's to 1, the
better is quality of the algorithm.

• For a maximization problem, we have a similar
defn except
\[
\text{cost}(S) \geq \frac{\text{opt}(S)}{\alpha}
\]
Again \(\alpha > 1\).

Sometimes one says a \(p\)-approx with \(p < 1\)
for max. problems -- in that case one means
\[
\text{cost}(S) \geq p \cdot \text{opt}(D)
\]

Examples

1. Traveling Salesman Problem (TSP)

   Input: \(n\) points on a metric space \((X, d)\)
   \((u, v, w) \in X,\)
   \(d(u, v) \leq d(u, w) + d(w, v)\)

   Output: A tour ordering of vertices in \(X\)
   \((\sigma_1, \sigma_2, \ldots, \sigma_n) \leftarrow \text{permutation.}\)

   Objective: Minimize
   \[
   \sum_{i=1}^{n-1} d(\sigma_i, \sigma_{i+1}) + d(\sigma_n, \sigma_1)
   \]
2. **Matching**

**Input**: Undirected graph \( G = (V, E) \)

**Output**: \( M \subseteq E \) s.t. \( \deg_M(v) \leq 1 \)

degree in \( G(V, M) \)

**Example**:

![Graph Example](image)

Both the red edges and the blue edges are valid matchings.

**Objective**: Maximize \( |M| \) \( \text{in P} \)

---

**Algorithm for Matching Problem**

a. Initially \( M = \emptyset \), empty set.

b. Consider edges of \( G \) in any order.

c. While considering edge \((u, v)\)

\[
\text{if} \quad \deg_M(u) = \deg_M(v) = 0,
\]

\[
\text{then} \quad M = M + (u, v)
\]

Simple algorithm, fast, how good is it?
Claim: The above algorithm is a 2-approx algo.

Proof: Fix a graph $G$ and let $M^*$ be the maximum matching. Let $M$ be the matching returned by the above algorithm.

We wish to show $|M| \geq \frac{1}{2} |M^*|.$

In order to do so, we define a many-to-1 map $\phi: M^* \rightarrow M$ s.t.

For all $e \in M^*$, there are at most two edges $e_1, e_2 \in M^*$ with

$\phi(e_1) = e \neq \phi(e_2) = e$

This will prove $|M| \geq \frac{1}{2} |M^*|.$

For all $(u, v) \in M^* \cap M$, $\phi(u, v) = (u, v)$

For all $(u, v) \in M^* \setminus M$, since we haven’t picked it yet $M$

$\Rightarrow \exists (v, w) \text{ or } (u, x) \text{ or both in } M.$

Arbitrarily map $\phi(u, v)$ to one of them.

Pick an edge $(u, v) \in M,$

If $e \in M^*$ has $\phi(e) = (u, v)$

then $e \sim (u, v)$, i.e., $e$ and $(u, v)$ must share a common end point.
then $e \sim (uv)$, i.e. $e$ and $(uv)$ must share a common end point.

No two $e, f \in M^*$ can share the same endpt. $\therefore M^*$ is a matching.

Since $(uv)$ has only two endpts, at most 2 edges in $M^*$ map to $(uv)$.

---

**Algorithm for TSP**

**Preliminaries**

**Eulerian Tour:** A **walk** in $G$ is an Eulerian walk if every edge is visited exactly once. It's called an Eulerian tour if start and end points are same.

![Graph 1](image1.png)

![Graph 2](image2.png)

1 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 5

**Theorem:** $G$ has an Eulerian tour iff $G$ is connected and $\deg(v)$ is even for all $v$.

Such graphs are called **Eulerian**.
Checking if there is a tour visiting every edge of a $G$ (exactly once) is easy.

**Eulerian tours vs metric TSP**

Given a metric $(X,d)$, let $G$ be the complete graph with $wt(u,v) = d(u,v)$. Let $F$ be any Eulerian subgraph of $G$.

Claim: There is a tour of cost $\leq wt(F)$

$$wt(F) = \sum_{e \in F} wt(e)$$

**Proof:** Let $W$ be the Eulerian tour of $F$.

$\sigma = \text{Shortest}(W)$

whenever a vertex is repeated we just skip it.

**eg:** in the example above $W$ is $5 \to 1 \to 2 \to 5 \to 4 \to 3 \to 2 \to 1 \to 4 \to 3 \to 5$

$\text{Shortest}(W) = 5 \to 1 \to 2 \to 4 \to 3 \to 5$

$\text{cost}(\sigma) \leq \text{cost}(W)$ **of Δ-ineq**.

**eg:** $2 \to 5 \to 4 \to \text{shortest}$
\[ \rightarrow 2 \rightarrow 4 \rightarrow \]

but \[ d(2,4) \leq d(2,5) + d(5,4) \]

This is where metric prop is crucially used.

\[ : \text{Finding "small" tours in } (X,d) \text{ boils down to finding "small cost" Eulerian subgraphs of } G. \]

---

**Algo 1**

1. Let \( T \) be the MST of \( G \)
2. Let \( 2T \) be the graph on \( X \)
   obtained by taking two parallel copies of each edge of \( T \).
3. Let \( W \) be the Eulerian tour of \( 2T \)
4. Return \( \text{Shortcut}(W) \)

**Thm:** Algo 1 is a 2-approximation algorithm for TSP.

**Pf:**

1. \( 2T \) is Eulerian by defn.
2. \[ \text{cost}(2T) \leq 2 \text{cost}(T) \leq 2 \text{OPT} \]
   since the opt tour contains a cycle.
In the previous algorithm, we ensured that every degree is even by taking two copies of every edge in $T$. But we can do something better.

Suppose $T$ is the MST. The “problematic” vertices are the odd-degree vertices.

Obs: # of odd-degree nodes in any tree is even.

Idea: “Pair these nodes up.”

How? In the cheapest possible way. By adding a minimum wt perfect matching.
Algorithm 2

1. Find T: MST of G
2. O be the set of odd-degree vertices in T
3. M be the min wt perfect matching connecting O in G
4. TUM is an Eulerian graph.

W: Eulerian tour in TUM
Return: Shortcut (W)

Thm: The above algo is $\frac{3}{2}$-approximate.

Pf: Suffices to show $wt(M) \leq \frac{1}{2} \cdot opt$

Since $w(T) \leq opt$.

Again look at the opt. tour and consider the O-vertices in this tour.

Note:
Note:

\[ \text{OPT} \geq "\text{total blue length}" \geq 2 \cdot (\text{min wt matching}) \]

since the blue lines partition into two matchings.