The traveling salesman problem

Theorem 1. For any polynomial time computable function \( \alpha(n) \), TSP cannot be approximated within a factor of \( \alpha(n) \), unless \( P = NP \).

Proof:

- Suppose there is an approximation algorithm \( A \) such that
  \[
  A(I) \leq \alpha(n) \cdot \text{OPT}(I) \text{ for all instances } I \text{ of TSP.}
  \]

- We will show that \( A \) can be used to decide whether a graph contains a Hamiltonian cycle (which is NP-hard), implying \( P = NP \).

- Let \( G \) be an undirected graph. We define a complete graph \( G' \) on the same vertices as follows:
  - Edges that appear in \( G \) are assigned a weight of 1.
  - Edges that do not exist in \( G \) get a weight of \( \alpha(n) \cdot n \).

- If \( G \) has a Hamiltonian cycle, the corresponding tour in \( G' \) has a cost of \( n \).

- If \( G \) has no Hamiltonian cycle, any tour in \( G \) has cost at least \( \alpha(n) \cdot n + 1 \).

- Hence, if we run \( A \) on \( G' \) it has to return a solution of cost \( \leq \alpha(n) \cdot n \) in the first case, and a solution of cost \( > \alpha(n) \cdot n \) in the second case.

- Thus, \( A \) can be used to decide whether \( G \) contains a Hamiltonian cycle.

The metric traveling salesman problem

A 2-approximation algorithm for \( \Delta \text{TSP} \):

1. Find a minimum spanning tree \( T \) of \( G \).
2. Double every edge of \( T \) to obtain a Eulerian graph.
3. Find a Eulerian tour \( T \) on this graph.
4. Output the tour that visits the vertices of \( G \) in the order of their first appearance in \( T \). Let \( \mathcal{C} \) be this tour.

Proof:
• Note that \( \text{cost}(T) \leq \text{OPT} \) because deleting an edge from an optimal tour yields a spanning tree.

• Moreover, \( \text{cost}(T) = 2 \cdot \text{cost}(T) \).

• Because of the triangle inequality, \( \text{cost}(C) \leq \text{cost}(T) \).

• Hence,

\[
\text{cost}(C) \leq 2 \cdot \text{OPT}.
\]

A 3/2-approximation algorithm for \( \Delta \text{TSP} \):

1. Find a minimum spanning tree \( T \) of \( G \).

2. Compute a min-cost perfect matching \( M \) on the set of odd-degree vertices of \( T \).

3. Add \( M \) to \( T \) to obtain a Eulerian graph.

4. Find a Eulerian tour \( T \) on this graph.

5. Output the tour that visits the vertices of \( G \) in the order of their first appearance in \( T \). Let \( C \) be this tour.

Proof:

• Let \( \tau \) be an optimal tour, i.e., \( \text{cost}(\tau) = \text{OPT} \).

• Let \( \tau' \) be the tour on the odd-degree nodes of \( T \), obtained by short-cutting \( \tau \).

• By triangle inequality, \( \text{cost}(\tau') \leq \text{cost}(\tau) \).

• Note that \( \tau' \) is the union of two perfect matchings.

• The cheaper of these two matchings has cost at most \( \text{cost}(\tau')/2 \).

• Hence,

\[
\text{cost}(C) \leq \text{cost}(T) \leq \text{cost}(T) + \text{cost}(M) \leq \text{OPT} + \frac{1}{2} \text{OPT}.
\]

The set cover problem

Input: \( U = \{1, \ldots, n\}, \ S = \{S_1, \ldots, S_k\} \subseteq 2^U, \ c : S \to \mathbb{Z}_+ \).

Output: \( J \subseteq \{1, \ldots, k\} \) such that \( \bigcup_{i \in J} S_i = U \) and \( \sum_{i \in J} c(S_i) \) is minimal.
• Special case: vertex cover problem.

A greedy algorithm:
1. \( C := \emptyset \).
2. \( \text{WHILE } C \neq U \text{ DO} \)
3. \( \text{Let } S := \arg \min \left\{ \frac{c(S)}{|S \setminus C|} : S \in \mathcal{S} \right\} \).
4. \( \text{Let } \alpha := \frac{c(S)}{|S \setminus C|} \).
5. \( \text{Pick } S, \text{ and for each } e \in S \setminus C, \text{ set } \text{price}(e) = \alpha. \)
6. \( C := C \cup S. \)
7. Output the picked sets.

• Let \( e_1, \ldots, e_n \) be the order in which the elements of \( U \) are covered by the greedy algorithm.

**Lemma 2.** For each \( k \in \{1, \ldots, n\} \), \( \text{price}(e_k) \leq \text{OPT} / (n - k + 1) \).

Proof:
• Let \( i(k) \) be the iteration in which \( e_k \) is covered.
• Let \( \mathcal{O} \subseteq \mathcal{S} \) be the sets chosen by an optimal solution.
• Let \( \mathcal{O}_{i(k)} \subseteq \mathcal{O} \) be the sets in \( \mathcal{O} \) not (yet) chosen by the greedy algorithm in iterations \( 1, \ldots, i(k) \).

• Note that \( \{e_k, \ldots, e_n\} \subseteq \bigcup_{S \in \mathcal{O}_{i(k)}} S \) and \( \sum_{S \in \mathcal{O}_{i(k)}} c(S) \leq \text{OPT} \).

• Hence, there exists a set \( S \in \mathcal{O}_k \) of average cost \( \frac{c(S)}{|S \setminus C|} \) at most \( \frac{\text{OPT}}{n - k + 1} \).

• Since \( e_k \) is covered by the set with the smallest average cost,

\[
\text{price}(e_k) \leq \frac{\text{OPT}}{n - k + 1}.
\]

**Theorem 3.** The greedy algorithm is an \((\ln n + 1)\)-approximation algorithm.

Proof:
• Since the cost of each set picked is distributed among the new elements covered, the total cost of the set cover returned by the greedy algorithm is equal to \( \sum_{k=1}^{n} \text{price}(e_k) \).

• By the previous lemma,

\[
\sum_{k=1}^{n} \text{price}(e_k) \leq \left( 1 + \frac{1}{2} + \cdots + \frac{1}{n} \right) \cdot \text{OPT} = H_n \cdot \text{OPT}.
\]

An integer programming formulation:

\[
\begin{align*}
\text{min} & \quad \sum_{S \in \mathcal{S}} c(S)x_S \\
\text{s.t.} & \quad \sum_{S \ni e} x_S \geq 1 \quad e \in \mathcal{U} \\
& \quad x_S \in \{0, 1\} \quad S \in \mathcal{S}
\end{align*}
\]

And its linear programming relaxation:

\[
\begin{align*}
\text{min} & \quad \sum_{S \in \mathcal{S}} c(S)x_S \\
\text{s.t.} & \quad \sum_{S \ni e} x_S \geq 1 \quad e \in \mathcal{U} \\
& \quad x_S \geq 0 \quad S \in \mathcal{S}
\end{align*}
\]

And its dual:

\[
\begin{align*}
\text{max} & \quad \sum_{e \in \mathcal{U}} y_e \\
\text{s.t.} & \quad \sum_{e \in S} y_e \leq c(S) \quad S \in \mathcal{S} \\
& \quad y_e \geq 0 \quad e \in \mathcal{U}
\end{align*}
\]

“Dual Fitting.”

**Lemma 4.** The vector \( y \) defined by \( y_e := \frac{\text{price}(e)}{H_n} \) is a feasible solution to the dual linear program.

Proof:

• Consider a set \( S \in \mathcal{S} \) consisting of \( k \) elements.

• Number the elements in the order in which they are covered by the greedy algorithm, say \( e_1, \ldots, e_k \).
• Consider the iteration in which the algorithm covers $e_i$.

• At this point, $S$ contains at least $k - i + 1$ uncovered elements.

• $S$ itself can cover $e_i$ at an average cost of at most $\frac{c(S)}{k - i + 1}$.

• Hence, $\text{price}(e_i) \leq \frac{c(S)}{k - i + 1}$ and $y_{e_i} \leq \frac{1}{H_n} \cdot \frac{c(S)}{k - i + 1}$.

• Overall, $\sum_{i=1}^{k} y_{e_i} \leq \frac{c(S)}{H_n} \cdot \left( \frac{1}{k} + \frac{1}{k - 1} + \cdots + \frac{1}{1} \right) = \frac{H_k}{H_n} \cdot c(S)$. 

\[ \square \]

**Theorem 5.** The greedy algorithm is an $H_n$-approximation algorithm.

Proof:
\[
\sum_{e \in U} \text{price}(e) = H_n \cdot \sum_{e \in U} y_e \leq H_n \cdot \text{LP} \leq H_n \cdot \text{OPT}.
\]

\[ \square \]

“LP rounding:”

1. Find an optimal solution to the LP relaxation.

2. Pick all sets $S$ for which $x_S \geq 1/f$ in this solution.

Here, $f$ is the frequency of the most frequent element.

**Theorem 6.** The LP rounding algorithm achieves an approximation factor of $f$.

Proof:

• Let $C$ be the collection of picked sets.

• Consider an arbitrary element $e \in U$.

• Since $e$ is in at most $f$ sets, one of them must be picked to the extent of at least $1/f$ in the fractional cover.

• So $C$ is a feasible set cover.

• The rounding process increases $x_S$, for each $S \in C$, by a factor of at most $f$.

\[ \square \]

A tight example:

• Consider a hypergraph: vertices correspond to sets, and hyperedges correspond to elements.
Let $V = V_1 \cup \ldots \cup V_k$, where each $V_i$ has cardinality $k$.

- There are $n^k$ hyperedges: each picks one element from each $V_i$.
- Each set (i.e., vertex) has cost 1.
- Picking each set to the extent of $1/k$ gives an optimal fractional cover of cost $n$.
- Given this fractional solution, the rounding algorithm will pick all $nk$ sets.
- On the other hand, picking all sets (vertices) in $V_1$ gives a set cover of cost $n$.

“The primal-dual method:”

- Start with a primal infeasible and a dual feasible solution (usually $x = 0$ and $y = 0$).
- Iteratively improve the feasibility of the primal solution and the optimality of the dual solution.
- The primal solution is always extended integrally.
- The current primal solution is used to determine the improvement to the dual, and vice versa.
- The cost of the dual solution is used as a lower bound.

(Relaxed) complementary slackness:

- Primal condition:
  \[ x_S \neq 0 \implies \sum_{e \in S} y_e = c(S). \]

- Dual condition:
  \[ y_e \neq 0 \implies \sum_{S \ni e} x_S \leq f. \]
  \[ \text{— Trivially satisfied!} \]

A factor $f$ approximation algorithm:

1. $x := 0$, $y := 0$.
2. REPEAT
3. Pick an uncovered element $e$ and raise $y_e$ until some set becomes tight.
4. Include all tight sets in the cover and update $x$.
5. UNTIL all elements are covered
6. RETURN $x$.

Proof:

$$\sum_{S \in \mathcal{C}} c(S)x_S = \sum_{S \in \mathcal{C}} \left( \sum_{e \in S} y_e \right) x_S \leq \sum_{e \in U} y_e \sum_{S \ni e} x_S \leq f \cdot \sum_{e \in U} y_e \leq f \cdot \text{OPT}$$