# Approximation Algorithms Seminar 1

## *Set Cover, Steiner Tree and TSP*

Siert Wieringa

siert.wieringa@tkk.fi

# Contents

Approximation algorithms for:

- Set Cover
- Steiner Tree
- TSP

# Set Cover

Given:

- A universe $U$ of $n$ elements.
- A collection of subsets of $U$, $S = \{S_1, \ldots, S_k\}$.
- A cost function $c : S \rightarrow Q^+$.

Find a minimum cost subcollection of $S$ that covers all elements of $U$.

# Set Cover - Example

$$U = \{1, 2, 3, 4, 5\}$$

$$S = \{S_1, S_2, S_3\}$$

$$S_1 = \{4, 1, 3\}$$

$$S_2 = \{2, 5\}$$

$$S_3 = \{1, 4, 3, 2\}$$

$$c : S \longrightarrow Q^+$$

$$c(S_1) = 5$$

$$c(S_2) = 10$$

$$c(S_3) = 3$$

# Set Cover - Example

$$U = \{1, 2, 3, 4, 5\}$$

$$S = \{S_1, S_2, S_3\}$$

$$S_1 = \{4, 1, 3\}$$

$$S_2 = \{2, 5\}$$

$$S_3 = \{1, 4, 3, 2\}$$

$$c : S \rightarrow Q^+$$

$$c(S_1) = 5$$

$$c(S_2) = 10$$

$$c(S_3) = 3$$

$$S_1 \cup S_2 \subseteq U$$

$$S_2 \cup S_3 \subseteq U$$

# Set Cover - Example

$$U = \{1, 2, 3, 4, 5\}$$

$$S = \{S_1, S_2, S_3\}$$

$$S_1 = \{4, 1, 3\}$$

$$S_2 = \{2, 5\}$$

$$S_3 = \{1, 4, 3, 2\}$$

$$c : S \rightarrow Q^+$$

$$c(S_1) = 5$$

$$c(S_2) = 10$$

$$c(S_3) = 3$$

$$S_1 \cup S_2 \subseteq U$$

$$S_2 \cup S_3 \subseteq U$$

$$c(S_1) + c(S_2) = 15$$

$$c(S_2) + c(S_3) = 13$$

# Set Cover - Example

$U = \{1,2,3,4,5\}$

$S = \{S_1, S_2, S_3\}$

$S_1 = \{4,1,3\}$

$S_2 = \{2,5\}$

$S_3 = \{1,4,3,2\}$

$c : S \to Q^+$

$c(S_1) = 5$

$c(S_2) = 10$

$c(S_3) = 3$

$$S_1 \cup S_2 \subseteq U$$

$$S_2 \cup S_3 \subseteq U$$

$$c(S_1) + c(S_2) = 15$$

$$c(S_2) + c(S_3) = 13$$

So $S_2 \cup S_3$ is a set cover for $U$

# Set Cover - Greedy algorithm 1/4

$\frac{cost(s)}{|S-C|}$ is the *cost-effectiveness* of a set S.

1 $C \leftarrow \emptyset$

2 While $C \neq U$ do

Find the set $S$ with the highest $\alpha = \frac{cost(s)}{|S-C|}$

For all $e \in S - C$, set $price(e) = \alpha$.

$C \leftarrow C \cup S$.

3 Output the picked sets.

Number the elements $e$ of $U$ in the order in which they where covered, $e_1, \ldots, e_k$.

# Set Cover - Greedy algorithm 2/4

**Lemma 2.3** *For each* $k \in \{1, \ldots, n\}$, $price(e_k) \leq \frac{OPT}{n-k+1}$.

**Proof** In every iteration the leftover sets of the optimal solution *can* cover the remaining elements at a cost of at most $OPT$. Therefore, amongst those sets there must be an element with cost at most $\frac{OPT}{|\overline{C}|}$ with $\overline{C}$ the set of uncovered elements. $\overline{C}$ contains at least $n-k+1$ elements.

$$price(e_k) \leq \frac{OPT}{|\overline{C}|} \leq \frac{OPT}{n-k+1}$$

□

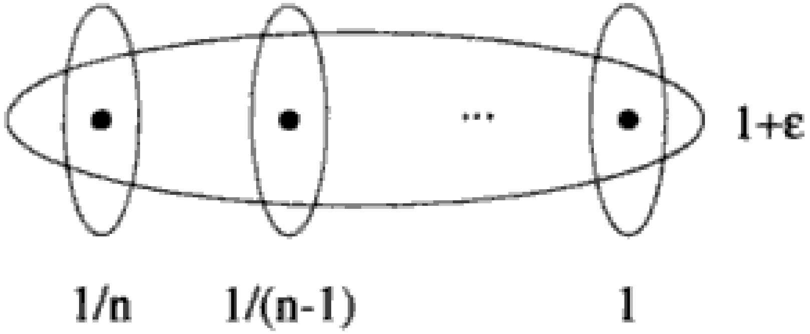
# Set Cover - Greedy algorithm 3/4

**Theorem 2.4** The greedy algorithm is an $H_n$ factor approximation algorithm for the minimum set cover problem, where $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$.

**Proof** The total cost is equal to $\sum_{k=1}^{n} price(e_k)$. By Lemma 2.3, this is at most $(1 + \frac{1}{2} + \cdots + \frac{1}{n}) \cdot OPT$.

□

# Set Cover - Greedy algorithm 4/4




`books.google.com` (first 20 pages)

# Vertex Cover

- The vertex cover problem is a special case of set cover with the highest element occurence frequency $f = 2$.
- For vertex cover there is a factor 2 approximation.
- Set cover approximation algorithms, either factor $O(log\ n)$ or $f$.

# Vertex Cover as Set Cover

Consider a graph $G = (V, E)$ with:

- $V = \{ V_1, V_2, V_3 \}$
- $E = \{ (1,2), (2,3), (1,3) \}$

We might define each vertex by the set of edges connected to it. Now we have a set cover problem with:

- $U = \{ (1,2), (2,3), (1,3) \}$
- $S = \{ \{ (1,2), (1,3) \}, \{ (1,2), (2,3) \}, \{ (2,3), (1,3) \} \}$

# Set Cover - Layering algorithm

- Factor $f$ approximation algorithm for set cover.
- Let $w : V \to Q^+$ be the function assigning weights to the vertices of a graph $G = (V, E)$.
- A weight function is *degree-weighted* if there is a constant $c > 0$ such that the weight of each vertex $v \in V$ is $c \cdot deg(v)$.

# Set Cover - Layering algorithm

**Lemma 2.6** Let $w : V \longrightarrow Q^+$ be a degree-weighted function. Then the cost of selecting all vertices $w(V) \leq 2 \cdot OPT$.

**Proof** Let $c$ be the constant such that $w(v) = c \cdot deg(v)$, and let $U$ be an optimal vertex cover in $G$.

$$\sum_{v \in U} deg(v) \geq |E| \qquad w(U) \geq c|E|$$

The sum of the degree of all vertices of a graph is $2|E|$ so $w(V) = 2c|E| \leq 2 \cdot OPT$.

$\square$

# Vertex Cover - Layering algorithm

$G_0 = G, \; k = 0$

**while** $G_k = (V, E)$ has vertices $v \in V$ with $deg(v) > 0$

$\quad c = min\left(\frac{w(v)}{deg(v)}\right)$ over all $v \in V$ with $deg(v) > 0$

$\quad D_k = \{ \, v \mid v \in V \; and \; deg(v) = 0 \, \}$

$\quad W_k = \{ \, v \mid v \in V \; and \; w(v) = c \cdot deg(v) \, \}$

$\quad G_{k+1} =$ the graph induced on $V - (D_k \cup W_k)$

$\quad k = k + 1$

**return** $C = W_0 \cup \cdots \cup W_{k-1}$

# Vertex Cover - Layering algorithm

1 $G_0 = G,\ k = 0$

2 **while** $G_k = (V, E)$ has vertices $v \in V$ with $deg(v) > 0$

3 $\quad c = min\left(\frac{w(v)}{deg(v)}\right)$ over all $v \in V$ with $deg(v) > 0$

4 $\quad D_k = \{\ v \mid v \in V \text{ and } deg(v) = 0\ \}$

5 $\quad W_k = \{\ v \mid v \in V \text{ and } w(v) = c \cdot deg(v)\ \}$

6a $\quad V_{k+1} = V - (D_k \cup W_k)$

6b $\quad E_{k+1} = E - \{\ (i, j) \mid i \in (D_k \cup W_k) \text{ or } j \in (D_k \cup W_k)\ \}$

6c $\quad G_{k+1} = (V_{k+1}, E_{k+1})$

7 $\quad k = k + 1$

8 **return** $C = W_0 \cup \cdots \cup W_{k-1}$

# Vertex Cover - Layering algorithm

$G_0 = G, \; k = 0$

**while** $G_k = (V, E)$ has vertices $v \in V$ with $deg(v) > 0$

  $c = min\left(\frac{w(v)}{deg(v)}\right)$ over all $v \in V$ with $deg(v) > 0$ $t_k(v) = c \cdot deg(v)$

  $D_k = \{ \; v \mid v \in V \; and \; deg(v) = 0 \; \}$

  $W_k = \{ \; v \mid v \in V \; and \; w(v) = c \cdot deg(v) \; \}$

  $V_{k+1} = V - (D_k \cup W_k)$

  $E_{k+1} = E - \{ \; (i, j) \mid i \in (D_k \cup W_k) \; or \; j \in (D_k \cup W_k) \; \}$

  $G_{k+1} = (V_{k+1}, E_{k+1})$

  $k = k + 1$

**return** $C = W_0 \cup \cdots \cup W_{k-1}$

# Layering algorithm - Proof ?

Consider a vertex $v \in C$. If $v \in W_j$, its weight can be decomposed as:

$$w(v) = \sum_{i \leq j} t_i(v) \quad \text{????}$$

# Set Cover - Layering algorithm

1 $U_0 = U, S_0 = S, \ k = 0$

2 **while** $S_k$ has elements $s$ with $|s| > 0$

3 $\quad c = min\left(\frac{w(s)}{|s|}\right)$ over all $s \in S_k$ with $|s| > 0$

4 $\quad D_k = \{\ s \mid s \in S_k \ and \ |s| = 0\ \}$

5 $\quad W_k = \{\ s \mid s \in S_k \ and \ w(s) = c|s|\ \}$

6a $\quad U_{k+1} = U - (D_k \cup W_k)$

6b $\quad S_{k+1} = \{\ s' \mid s \in S_k,\ s' = s - (D_k \cup W_k)\ \}$

7 $\quad k = k + 1$

8 **return** $C = W_0 \cup \cdots \cup W_{k-1}$

# Steiner Tree

Given:

- An undirected graph $G = (V, E)$ with nonnegative edge cost.
- A partitioning of the vertices $V$ into *required*, and Steiner edges.

Find a minimum cost tree in $G$ that contains all the required vertices and any subset of Steiner vertices.

# Metric Steiner Tree

A restriction of the Steiner Tree problem to those graphs that satisfy the *triangle inequality*. That is, $G$ has to be a complete undirected graph, and for any three vertices $u$, $v$ and $w$, $cost(u,v) \leq cost(u,w) + cost(v,w)$.

**Theorem 3.2** There is an approximation factor preserving reduction from the Steiner tree problem to the metric Steiner tree problem.

# Metric Steiner Tree $\leftrightarrow$ MST

**Theorem 3.3** The cost of a Minimal Spanning Tree on the required vertices is within $2 \cdot OPT$.

# Traveling salesman problem (TSP)

Given a complete graph with nonnegative edge costs, find a minimum cost cycle visiting evey vertex exactly once.

**Theorem 3.6** For any polynomial computable function $\alpha(n)$, TSP can not be approximated within a factor of $\alpha(n)$, unless $P = NP$.

# Traveling salesman problem (TSP)

**Proof** Using a polynomial factor $\alpha(n)$ approximation algorithm for TSP we can decide the Hamiltonian cycle problem which is NP-Hard in polynomial time. The existence of such an algorithm would therefore imply that $P = NP$.

(continued)

# Traveling salesman problem (TSP)

Reduction of Hamiltonian Cycle to polynomial factor approximation of TSP.

Assign a weight of 1 to edges of $G$. Extend $G$ to the complete graph $G'$ and give all added "nonedges" weight $\alpha(n) \cdot n$. If $G$ has a Hamiltonian cycle, then the corresponding tour in $G'$ has cost $n$.

If $G$ has no Hamiltonian cycle, any tour in $G'$ must use an edge of cost $\alpha(n) \cdot n$ and it therefore has cost $> \alpha(n) \cdot n$.

$\square$

# Metric TSP - Factor $2$ approx.

- The proof on the previous slide used edge weights that did not satisfy the triangle inequality.
- *Metric TSP* is also NP-Complete, but not hard to approximate.
- Cost of a MST is $\leq OPT$.
- Factor 2 approximation algorithm by using similar approach as in proof of Steiner Tree algorithm approximation factor.

# Metric TSP - Factor $\frac{3}{2}$ approx.

- For Eulerian path to exists all vertices must have even number of edges.
- Can be forced by doubling edges, smarter approach only concerns vertices with odd degree, $V'$.

1 Add maximum matching of $V'$ to the graph.

2 Find Euler tour in this graph.

3 Output "short-cutted" Euler tour.

- Note: $|V'|$ must be even since sum of the degree of all vertices is even ($2|E|$).

# Metric TSP - Factor $\frac{3}{2}$ approx.

**Lemma 3.11** Let $V' \subseteq V$, such that $|V'|$ is even, and let $M$ be a minimum cost perfect matching on $V'$. Then, $cost(M) \leq \frac{OPT}{2}$

**Proof** Consider an optimal TSP tour $\tau$ of $G$. Let $\tau'$ be the tour on $V'$ obtained by short-cutting $\tau$. By the triangle inequality, $cost(\tau') \leq cost(\tau)$. The tour $\tau'$ can be seen as the union of two perfect matchings on $V'$. The cheapest of those two matchings has cost $\leq \frac{cost(\tau')}{2} \leq \frac{OPT}{2}$. So, the optimal matching must also be of cost $\leq \frac{OPT}{2}$.

$\square$

# Metric TSP - Factor $\frac{3}{2}$ approx.

**Lemma 3.12** The presented algorithm achieves an approximation guarantee of $\frac{3}{2}$ for metric TSP.

**Proof** The cost of the Euler tour is $\leq cost(T) + cost(M) \leq OPT + \frac{1}{2}OPT = \frac{3}{2}OPT$. By the triangle inequality the cost of the path is also smaller than $\frac{3}{2}OPT$.

□

# Summary

We have:

- Seen approximation algorithms for a number of problems.
- Studied the approximation factors of those algorithms.
- Seen tight examples for the algorithms.

# Questions?