Communications and Transport Systems
Department of Science and Technology **Fall 2018**
Linköping University

Christiane Schmidt

Design and Analysis of Algorithms Part 2 - Approximation and online algorithms homework 3, 18.10.2018

Problem 1 (The Metric Traveling Salesman Problem):

(a) We considered an idea for constructing a tour via doubling an MST, and using short cuts if possible (using the triangle inequality).

Show that this procedure gives an approximation algorithm and determine its approximation factor.

(b) Give an example to show that your bound for the algorithm is tight, i.e., give an example in which your given approximation factor is achieved.

Problem 2 (The Traveling Salesman Problem):
We consider the general—not the geometric (or metric)—variant of the Traveling Salesman Problem. Consider the following greedy algorithm for the TSP in complete graphs: Let S denote the set of all visited vertices (at the start $S := \varnothing$). Start with an arbitrary vertx $v \in V$. Add v to V and choose an edge $e = \{v, w\}$ of minimal weight, with $w \in V \setminus S$. Proceed with w. In case $S = V$, move back to the start vertex.
Give an example in which the ratio of algorithm to optimum can be arbitrarily bad.

Problem 3 (Vertex Cover):
We consider two greedy algorithms for the Vertex Cover problem in a graph $G = (V, E)$:

Greedy 1:
$C := \varnothing$
while $E \neq \varnothing$ **do**
  Choose an edge $e \in E$ and choose a vertex v of e.
  $C := C \cup \{v\}$
  $E := E \setminus \{e \in E : v \in e\}$
end
return C

Greedy 2:
$C := \varnothing$
while $E \neq \varnothing$ **do**
  Choose a vertex with maximal degree in the *current* graph.
  $C := C \cup \{v\}$
  $E := E \setminus \{e \in E : v \in e\}$
end
return C

Show that for both algorithms a constant approximation factor cannot be guaranteed, not even in bipartite graphs.

Problem 4 (The Geometric Traveling Salesman Problem II):
We considered a TSP-approximation that doubles the edges of an MST of the given graph $G = (V, E)$.
Consider the output from the MST computation. This graph is not Eulerian, because any tree must have nodes of degree one. Let O be the set of odd-degree nodes in the MST. For any graph, the sum of its node degrees must be even, because each edge in the graph contributes 2 to this total. The total degree of the even-degree nodes must also be even, but then the total of degree of the odd-degree nodes must also be even. In other words, we must have an even number of odd degree nodes; $|O| = 2k$ for some positive integer k.
Suppose that we pair up the nodes in O: $(i_1, i_2), (i_3, i_4), \ldots, (i_{2k-1}, i_{2k})$. Such a collection of edges that contain each node in O exactly once is called a *perfect matching* (as known from Network Algorithms!!) of O. One of the classic results of combinatorial optimization is that given a complete graph (on an even number of nodes) with edge costs, it is possible to compute the perfect matching of minimum total cost in polynomial time.
Given the MST, we identify the set O of odd-degree nodes with even cardinality, and then compute a minimum-cost perfect matching on O. If we add this set of edges to our MST, we have constructed an Eulerian graph on our original set of cities: it is connected (because the spanning tree is connected) and has even degree (because we added a new edge incident to each node of odd degree in the spanning tree). We can now shortcut this graph to produce a tour of no greater cost.
Prove that the given algorithm for the metric Traveling Salesman Problem is a $\frac{3}{2}$-approximation algorithm.

Problem 5 (Weighted Vertex Cover):
For the Weighted Vertex Cover Problem, we are given a graph $G = (V, E)$, each vertex is given a weight: $w : V \to \mathbb{R}^+$. We ask for a vertex cover C, such that the total weight of the cover is minimized: $\min \sum_{v \in C} w(v)$.

Consider the following algorithm for the weighted vertex cover problem: For each vertex v, $t(v)$ is initialized to its weight, and when $t(v)$ drops to 0, v is picked in the cover. $c(e)$ is the amount charged to edge e.

Algorithm

1. Initialization:
   - $C = \varnothing$
   - $\forall v \in V : t(v) = w(v)$
   - $\forall e \in E : c(e) = 0$
2. While C is no a vertex cover, do:
   - Pick an uncovered edge, say (u, v). Let $m = \min(t(u), t(v))$
   - $t(u) = t(u) - m$
   - $t(v) = t(v) - m$
   - $c(u, v) = m$
   - Include in C all vertices having $t(v) = 0$.
3. Output C.

(a) Apply the algorithm to the example graph given in Figure 1; in case you can pick more than one uncovered edge, choose the edge with minimal index.

(b) Show that this is a 2-approximation algorithm. Hint: Show that the total amount charged to edges is a lower bound on OPT and that the weight of cover C is at most twice the total amount charged to edges.

Problem 6 (Input complexity): Give the input complexities for the problems TSP, Hamiltonian Circuit and Vertex Cover.

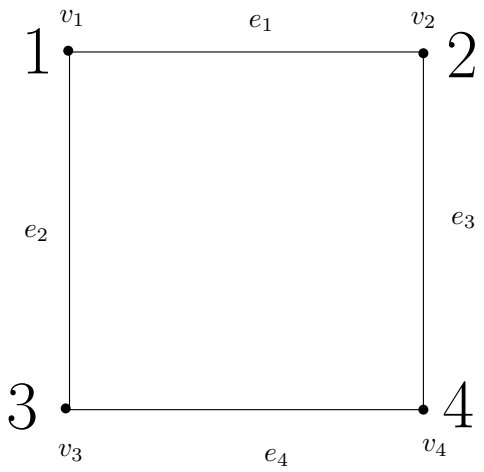

Figure 1: A graph for the Weighted Vertex Cover Problem. The large numbers define the weights, that is, $w(v_i) = i$.