Advanced algorithms

Exercise sheet #6 (Solutions) — Approximation algorithms

November 9, 2022

Exercice 1 (TSP with triangle inequality)**.** Let G be a complete graph with n vertices, labelled from 1 to n. To each of the $\frac{1}{2}n(n-1)$ edges (u,v) is associated a distance $d(u,v)$. The traveling salesman looks for a minimum-length tour that starts and ends on 1 and visits every vertex exactly once. The decision version of this problem is NP-complete.

We assume furthermore that the distance d satisfies the triangle inequality: $d(u,v) \leqslant d(u,w) + d(w,v)$ for any vertex u, v and w.

Let T be a minimum spanning tree, rooted at 1, and let H be the tour obtained by the pre-order depth first traversal of T.

(a) What is the complexity of computing H?

> *Solution — This is $O(n^2 \log n)$ with Kruskal or Prim algorithm. Can be lowered to $O(n^2)$ with more advances algorithms.*

(b) Let H^* be an optimal tour. Show that the length $c(T)$ of T (that is the sum of the distances of the edges in T) is at most the length $c(H^*)$ of H^*.

> *Solution — If we remove one edge from H^*, we obtain a spanning tree T^*. Therefore $c(H^*) > c(T^*) \geqslant c(T)$, by minimality of T.*

(c) Using the triangle inequality, show that $c(H) \leqslant 2c(T)$. Deduce an approximation algorithm, with approximation factor 2, for computing an optimal tour.

> *Solution — Consider the tour L (with repeated vertices) obtained from the depth-first traversal of T: each edge is taken once downward and one upward. It is clear that $c(L) = 2c(T)$. Moreover, H is obtained from L by deleting upward edges: A path of the form $b \to a \to b \to c$ is replaced by $b \to a \to c$ directly, so by the triangle inequality this change cannot increase the distance. Therefore,$c(H) \leqslant c(L) = 2c(T)$.*
>
> *So, by the previous question, $c(H) \leqslant 2c(H^*)$: We have a 2-approximation algorithm.*

Exercice 2 (Multiterminal cut (Pâle 2013))**.** Let $G = (V,E)$ be a connected graph endowed with a weight function $c(e) \geq 0$ for each edge $e \in E$ and with a distinguished subset S of vertices, called *terminals*.

A *multiterminal cut* of G is a set of edges $F \subseteq E$ whose removal would disconnect all terminals from each other.

The weight of a multiterminal cut is the sum of the weight of its elements. Given S, we aim at computing a minimum-weight multiterminal cut, or rather an approximation.

(a) Given a multiterminal cut F and $v \in S$, let $G_v[F]$ be the connected component of $G \setminus F$ containing v. Moreover, let F_v be the subset of F of all edges with exactly one end in $G_v[F]$. Show that any path in G from v to any other $w \in S$ has an edge in F_v.

> *Solution — Let $w \in S$, $w \neq v$. Let P be a path joining v and w. By definition of a multiterminal cut, P passes through an edge in F. Therefore, it must go out of the connected component $G_v[F]$: It contains an edge with only one end in $G_v[F]$,* i.e. *that belongs to F_v.*

(b) For $v \in S$, let E_v be a minimum-weight set of edges such that any path in G from v to any other $w \in S$ has an edge in E_v. Show that E_v can be computed in polynomial time. What is the complexity of your algorithm ?

> *Solution —*
>
> *The idea is to build a flow network from G and consider all vertices $S \setminus \{v\}$ as a single terminal t. The problem would now reduce to finding a $v-t$ cut of minimum capacity in this flow network.*
>
> *To build the flow network from G, we transform each edge e in two edges e_1, e_2 (to make them directed) both with capacity $c(e)$. Moreover, we add an edge (w, t) for each $w \in S \setminus \{v\}$.*
>
> *By the max-flow min-cut theorem, a $v - t$ cut can be computed with a flow algorithm such as Edmond–Karp algorithm, in complexity $O(n^2 m)$.*

(c) Deduce a 2-approximation algorithm for the problem of computing a minimum-weight multiterminal cut.

> *Solution — Let $U = \bigcup_{v \in S} E_v$. It is a multiterminal cut.*
>
> *Let F^* be a minimum-weight multiterminal cut. By minimality of each E_v, we have $c(F_v^*) \geqslant c(E_v)$. So that $c(U) \leqslant \sum_{v\in S} c(E_v) \leqslant \sum_{v \in S} c(F_v^*)$. But each edge of F^* can only belong to at most two different F_v^*. So $\sum_v c(F_v^*) \leqslant 2c(F^*)$.*

Exercice 3 (Vertex cover with linear programming)**.** Let $G = (V, E)$ be a graph with a weight function $c(v) \geq 0$ on the vertices. We aim at computing an approximate minimum-weight vertex cover of G. Recall that a vertex cover is a set $S \subset V$ so that each edge has at least one end in S.

Consider the following linear program:

$$\begin{aligned} \text{minimize } & \sum_{v \in V} c(v) x_v \\ \text{such that } & x_u + x_v \geqslant 1, \quad \forall \{u, v\} \in E \\ & 1 \geq x_v \geq 0, \quad \forall v \in V, \end{aligned}$$

with the optimal value λ^* and an optimal solution $(x_v^*)_{v \in V}$.

(a) Let S^* be a minimum-weight vertex cover of G. Show that $c(S^*) \geqslant \lambda^*$.

> *Solution — If $x_v^* \in \{0, 1\}$, then it is easy to build an optimal solution. Let $S = \{v \in V \mid x_v^*\}$. The constraint $x_u + x_v \geq 1$ for $(u, v) \in E$ ensures that each edge has an end in S,* i.e. *that S is indeed a vertex cover and λ^* is exactly the weight of a S.*
>
> *Therefore, vertex covers of G correspond to integer solutions. When we allow x_v to take arbitrary real number values, the minimum cannot be larger ! Hence, $\lambda^* \leq c(S^*)$.*

Remark. *Adding the extra constraint $x_v \in \{0,1\}$ (or more generally $x_v \in \mathbb{Z}$) is called* Integer Programming *and is significantly harder than* Linear Programming.

(b) From the optimal solution $(x_v^*)_{v\in V}$, construct a vertex cover S of G such that $c(S) \leqslant 2\lambda^*$.

Solution — Let $S = \{v \in V \mid x_i^* \geq \frac{1}{2}\}$.

- *For any edge $(u,v) \in E$, we have $x_u^* + x_v^* \geq 1$, so either $x_u^* \geq \frac{1}{2}$ or $x_v^* \geq \frac{1}{2}$. Thus, at least one of them will be selected in S which is indeed a vertex cover.*
- *The set S has only vertices with $x_v^* \geq \frac{1}{2}$. Therefore, the following inequalities hold:*
$$\frac{1}{2}c(S) = \sum_{v\in S} c(v)\frac{1}{2} \leq \sum_{v\in S} c(v)x_v^* \leq \sum_{v\in V} c(v)x_v^* = \lambda^*$$

For further readings, see [?, §11.6].

Exercice 4 (The center selection problem)**.** Let V be a finite set endowed with a distance function $d : V \times V \to [0,\infty)$ which satisfies the usual properties of distance functions:

- Separation: $d(u,v) = 0 \Leftrightarrow u = v$ for all $u, v \in S$,
- Symmetry: $d(u,v) = d(v,u)$ for all $u, v \in S$,
- Triangle inequality: $d(u,v) \leq d(u,w) + d(w,v)$ for all $u, v, w \in S$.

For any subset $S \subset V$, we define its covering radius

$$\operatorname{rad}(S) = \max_{v\in V} \min_{s\in S} d(v,s).$$

It is the maximal distance of an element of V to the closest element of S. Given an integer k, a subset S of size $\leq k$ of minimal covering radius is called a set of *centers*.

(a) Let $r \geqslant 0$ and assume that there exists a subset $S^* \subseteq V$ of k centers such that $\operatorname{rad}(S^*) \leqslant r$. Design a greedy algorithm to compute a $S \subseteq V$ with $\#S \leqslant k$ and $\operatorname{rad}(S) \leqslant 2r$.

Solution —

$S \leftarrow \varnothing$
while $\exists v \in V, d(v,S) > 2r$ ***do***
  $S \leftarrow S \cup \{v\}$
end while

By design, $\operatorname{rad}(S) \leqslant 2r$*, it only remains to show that $\#S \leqslant k$. For each v that is added to S there is some $c_v \in S^*$ such that $d(c_v, v) \leqslant r$, by hypothesis. We want to prove that this c_v is unique for each v.*

For any distinct $v, w \in S$, $d(v,w) > 2r$, by design, and, $2r < d(v,w) \leqslant d(v,c_v) + d(c_v,c_w) + d(c_w,w) \leq 2r + d(c_v,c_w)$, by the triangle inequality. It follows that $c_v \neq c_w$.

Therefore, the map $v \mapsto c_v$ defines an injection from S to S^*, so $\#S \leqslant \#S^*$.

(b) Let r^* be the minimum value of $\operatorname{rad}(S^*)$, for $S^* \subseteq V$ and $\#S^* = k$. Design an algorithm to compute in polynomial time a $S \subseteq V$ with $\#S \leqslant k$ and $\operatorname{rad}(S) \leqslant 2r^*$.

Solution — *We can try to guess r^* and apply the previous algorithm. Note that r^* belongs to the finite set $\{d(v, w) \mid v, w \in V\}$. This gives the following algorithm (which we can refine using dichotomy).*

$D \leftarrow \{d(v, w) \mid v, w \in V\}$
for $r \in D$, *by increasing order* ***do***
  $S \leftarrow \varnothing$
  while $\exists v \in V, d(v, S) > 2r$ ***do***
    $S \leftarrow S \cup \{v\}$
  end while
  if $\#S \leqslant k$ ***then***
    return S
  end if
end for

We can also guess r^* on the fly.

$S \leftarrow \varnothing$
while $\#S \leqslant k$ ***do***
  $v \leftarrow \mathrm{argmax}_{v \in V}\, d(v, S)$
  $S \leftarrow S \cup \{v\}$
end while

We now prove the correctness of this last algorithm. Let S be the output of this algorithm and let $r = \mathrm{rad}(S)$.

Let $p \in V$ such that $d(p, S) = r$ and let $S' = S \cup \{p\}$. We first claim that for any $v, w \in S$, if $v \neq w$ then $d(v, w) \geqslant r$. Indeed, at each iteration of the algorithm, we pick the point that is the furthest to the previously selected centers. Since p was not selected, and that $d(p, v) \geqslant r$ for any $v \in S$, it follows that all centers have distance at least r to the previous ones.

Now, S' is covered by the k balls of radius r^* whose centers are the points in S^*. So there are two points in S' that are covered by the same center. In particular, their distance is at most $2r^*$. It follows that $r \leqslant 2r^*$.

*See [**?**, §11.2] for more details.*