Universität Bremen

FB3: Mathematik/Informatik

Dr. Martin Böhm
Dr. Ruben Hoeksma
Prof. Dr. Nicole Megow

Summer 2019

Approximation Algorithms

Exercise Sheet 1 (09. 04. 2019)

Exercise 1. We will first practice showing that a problem is NP-complete by reducing it from another one. Consider the classical KNAPSACK problem, which can be defined as follows:

Input: n items, each having associated a value $v_i \geq 0$, a weight $w_i \geq 0$, a knapsack capacity $b \geq 0$ and a lower bound on profit $p \geq 0$.

Decide: Does there exist a subset of items S with $\sum_{i \in S} w_i \leq b$ of total value $\sum_{i \in S} v_i$ at least p?

a) Show that the KNAPSACK problem lies in NP.

b) Show that it is NP-complete by giving a simple reduction from SUBSET SUM:

Input: n non-negative numbers $A = \{a_1, a_2, \ldots, a_n\}$, a number s.

Decide: Is there a subset of A that sums up to exactly s?

Exercise 2. At the last lecture we have learned of the problem DOMINATING SET:

Input: An undirected, unweighted graph $G = (V, E)$, an integer k.

Decide: Is there a set of at most k vertices $D \subseteq V$, such that for every vertex $v \in V$ it holds that (a) $v \in D$ or (b) a neighbor of v is in D?

Our goal will be to use this problem to show that K-CENTER is a problem that is hard to approximate. The definition would be as follows:

Definition 1. *An optimization problem is* NP-hard to approximate *within factor α if an α-approximation algorithm can be used to solve some NP-complete problem.*

And our goal is to prove:

Theorem 1. K-CENTER *is NP-hard to approximate within factor less than* 2.

Our proof will proceed as follows:

a) Find a reduction from DOMINATING SET to K-CENTER such that `yes`-instances of DS are mapped to instances of K-CENTER with optimal radius 1 ...

b) ... and `no`-instances are only mapped to instances with optimal radius ≥ 2.

c) If we find this reduction, then it is immediately clear that any 1.99-approximation algorithm is able to distinguish `yes`-instances from `no`-instances.

Try to find the right reduction from DOMINATING SET to K-CENTER.

Exercise 3. Consider SCHEDULING WITH DEPENDENCIES: we schedule jobs of different lengths on m computers (m is a part of the input), but we also have a *dependence graph* on the jobs, and we can schedule a job only when all its dependencies are completed.

a) Prove the following lower bound on the optimum:

"$OPT \geq$ length of any chain in the input. A *chain* is a sequence of jobs where each one depends on the previous one. Its length is then the total processing time of all the jobs in the chain."

b) Design a greedy 2-approximation algorithm for this problem.

Exercise 4. In the metric k-SUPPLIER PROBLEM we get $m + n$ points on input, where m of them are (in advance) marked as *suppliers* and the rest are *consumers*. Between all those points is a metric (with a triangle inequality as usual). Our task in this case is to select k suppliers so that we minimize the longest distance between a customer and its closest supplier.

Suggest and analyze a 3-approximation algorithm for the k-SUPPLIER PROBLEM.

Solution. We will proceed in an analytical fashion to show the process of figuring out the solution.

Recalling the k-CENTER PROBLEM, we consider the optimum placement of the k suppliers and balls of radius OPT around each optimal supplier s_{OPT} (by *ball* of radius r we just mean the set of all suppliers and consumers within distance r, as defined by the metric of the problem).

As in k-CENTER, we know that (by optimality) all consumers are located within these k balls. In the analysis of k-center, we claimed that all centers selected by our algorithm are also within these k balls, and in fact every ball contains at least one of those.

We will use to our advantage that we have a different approximation ratio to prove – here we can use 3-approximation instead of just 2. This means we can increase the balls. How much can we enlarge them?

Setting them to $2OPT$ is a natural choice, and a right one – if we can prove that every ball of radius $2OPT$ contains at least one supply center s_{ALG} selected by the algorithm, then $ALG \leq 3OPT$. This will be true because the distance from s_{ALG} to s_{OPT} is at most $2OPT$, and the distance from s_{OPT} to any consumer point c covered by s_{OPT} is OPT by the assumption of optimality. In total, we get $d(s_{ALG}, c) \leq 3OPT$.

A snag in the k-SUPPLIER PROBLEM is that there *can* be possible supplier centers completely outside every ball, and the algorithm could have erroneously selected those instead of the suppliers within the balls. You can imagine one (very bad) supply center being $100cm$ away while OPT is just $1cm$.

To eliminate the bad supply centers, we look at one possible consumer (located in at least one ball of radius $2OPT$). We observe that all points within distance OPT to this consumer (including the optimal supply center and possibly other supply centers) are within the ball of radius $2OPT$ as well.

We cannot teach the algorithm to ignore all supply centers further than OPT from any customer – the algorithm does not know OPT in advance. We need to use a lower bound on OPT and ignore all supply centers further away than this lower bound.

A natural lower bound for one consumer c is just $d(c, s)$ for the *nearest* supply point s. Any other supply point chosen by the algorithm or optimum has to be in distance at least $d(c, s)$.

We therefore separate supply points into two groups: A supply point s is *interesting* if there exists a consumer which has s as the closest supply point to it. The rest are *uninteresting*.

Looking back, we see that our choice is a good one – for any supply point s_{OPT} of the optimum, the ball of radius $2OPT$ contains all interesting points for every consumer c with $d(s_{OPT}, c) \leq OPT$. In other words, if our algorithm chooses only interesting points, it will never select a point that is outside every ball.

What remains is to give an algorithm and show that indeed, our algorithm does not select two points from the same ball while leaving one ball completely empty.

A possible algorithm is the following:

a) Choose one interesting point as a supply center arbitrarily.

b) Then, choose every next interesting point that minimizes the current maximum the most. Proceed until you place all k points.

The rest of the analysis of this algorithm is the same as for the k-CENTER PROBLEM.