Approximation Algorithms (Part III)

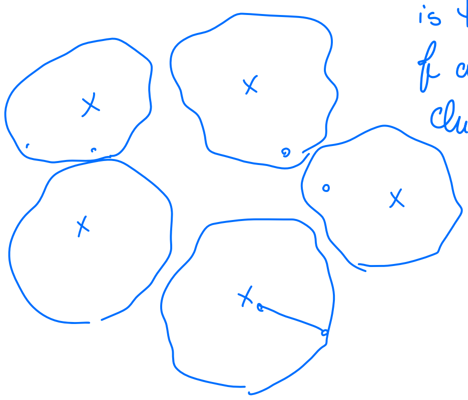
k-center clustering

We are given a complete weighted graph $G=(V, \binom{V}{2}, w)$ and parameter k.

The goal is to select a subset $U \subseteq V$ of size $|U|=k$ such that

$$\max_{v \in V} \min_{u \in U} w(v,u)$$

is minimized.



Radius of a cluster is the largest dist of a vertex in that cluster to the center

Remark: The problem is NP-hard to apx within a factor of ~ 1.8 even in metric spaces.

Thm [Ganzalez & Teofila '85]: metric k-center can be 2-apximated in poly time.

Alg: start with an arbitrary center $u_1 \in V$.

For $i=2$ to k:

$$u_i \leftarrow \arg\max_{u \in V} \min_{u_i} w(u, u_i)$$

Approximation guarantee

Centers $u_1, u_2, \dots, u_k, u_{k+1}$ (if we were to pick one extra center)

r_i: The i'th clustering radius

Obs: $r_1 \geq r_2 \geq \dots \geq r_k$.

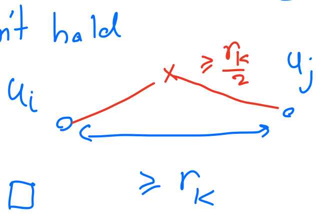
pf: Take any vertex u, the closest center among $\{u_1, \dots, u_i\}$ to u is no further away to u than the closest center in $\{u_1, \dots, u_{i-1}\}$. This means $r_i \leq r_{i-1}$. $\square$

Obs2: For any two distinct centers u_i, u_j, $i,j \in [k+1]$
$w(u_i, u_j) \geq r_k$.

pf: Suppose $j > i$. At the time of picking center u_j, we have $w(u_j, u_i) \geq r_{j-1}$.
This is because the distance between u_j & its closest center in $\{u_1, \dots, u_{j-1}\}$ is exactly r_{j-1}, which means $w(u_j, u_i) \geq r_{j-1}$.
By Obs1, $r_{j-1} \geq r_j \geq \dots \geq r_k$. $\square$
$j \leq k+1$

Obs 3: $OPT \geq r_k / 2$.

By Obs2, we have $k+1$ vertices $u_1, \dots, u_{k+1}$ with pairwise distance $\geq r_k$. In the optimal clustering, at least two of these must belong to the same cluster. This means that the radius of this cluster must be at least $\frac{r_k}{2}$.
(o/w triangle inequality won't hold between u_i, u_j and their center)



This means $OPT \geq \frac{r_k}{2}$. $\square$

Finally, note that r_k by definition is the radius of the clustering defined by $\{u_1, \dots, u_k\}$. Therefore Gonzalez returns a 2-apx. $\square$

Approximation Schemes

An algorithm that given an instance and any parameter $\varepsilon > 0$ obtains a $(1+\varepsilon)$ apx.

PTAS: An approx scheme that runs in poly nomial time for any fixed $\varepsilon > 0$.

(e.g. runs in $O(n^{1/\varepsilon})$.)

FPTAS: An approx scheme that runs in poly (# input size) $\times$ poly ($1/\varepsilon$).

Subset Sum

Given n numbers $x_1, \dots, x_n$ and a target number t. The goal is to see if there exists a subset of $x_1, \dots, x_n$ that sums up to t exactly.

There is an algorithm solving this in time $O(nt)$ using DP, but this is not considered polynomial in the input size which is at most $O(n \cdot \lg t)$.

The subset sum problem in fact turns out to be NP-hard, so we have no hope of designing a polynomial time alg (unless P=NP).

Consider the following optimization variant:

Find a subset S of $x_1, \dots, x_n$ s.t.

1) $\sum_{x_i \in S} x_i \leq t$.

2) $\sum_{x_i \in S} x_i$ is as large as possible.

An α-apx for subset sum ensures (1), and that $\sum_{x_i \in S} x_i \geq \frac{OPT}{\alpha}$.

Thm: For any $\varepsilon > 0$, there is an algorithm that $(1+\varepsilon)$-approximates subset sum in $O(\frac{n^3 \lg t}{\varepsilon})$ time. FPTAS

Consider the following algorithm

Subset Sum (X [1...n], t):

$S_0 \leftarrow \{\emptyset\}$

for $i \leftarrow 1$ to n:

$S_i \leftarrow S_{i-1} \cup (S_{i-1} + X[i])$

remove all elements of S_i bigger than t

return max S_n.

Note that for each i,

$$|S_i| \leq \min\{2^i, t\}.$$

The algorithm runs in $O(\min\{2^n, nt\})$ time.

Intuition: If two values in S_i are very close, we don't need to necessarily keep them both in S_i as far as approximations are concerned. So the goal is to reduce the size of each S_i to poly(n) and still argue we obtain a good apx.

ApproxSubsetSum (X [1...n], ε):

Sort (X)

$R_0 \leftarrow \{\emptyset\}$

for $i \leftarrow 1$ to n:

$R_i \leftarrow R_{i-1} \cup (R_{i-1} + X[i])$

$R_i \leftarrow \text{Filter}(R_i, \varepsilon/2n)$

remove all elements of R_i bigger than t

return max R_n

Filter (Z [1... k], δ):

Sort (Z)

$j \leftarrow 1$, $i \leftarrow 1$

$Y[j] \leftarrow Z[i]$

for $i \leftarrow 2$ to k:

if $Z[i] \geq (1+\delta) Y[j]$

$j \leftarrow j+1$

$Y[j] \leftarrow Z[i]$

return $Y[1..j]$

For the analysis of approximation, the following claim is proved:

$\boxed{\text{For any element } s \in S_i, \text{ there is an element } r \in R_i \text{ such that } r \leq s \leq r \cdot (1+\frac{\varepsilon}{2n})^i}$

Intuition: Note that for $Y = \text{Filter}(Z, \delta)$ it holds that for every element $s \in Z$ there exists $r \in Y$ s.t. $\frac{r}{1+\delta} \leq s \leq r$.

To go from this to the statement above, note that R_i is (essentially) S_i after applying Filter for i steps on it.

At the end since $i \leq n$, it holds that for any $s \in S_n$ there is $r \in R_n$ where

$$r \leq s \leq \underbrace{(1+\frac{\varepsilon}{2n})^n r}_{} \leq (1+\varepsilon) r$$

comes from $e^x \geq 1+x$ for all x
and $e^x \leq 1+2x$ for $0 \leq x \leq 1$

For the runtime, $|R_i| = O(\frac{\lg |S_i|}{\delta})$

$|S_i| \leq \min\{2^n, t\}$

$\Rightarrow |R_i| = O\left(\frac{n + \lg t}{\varepsilon/2n}\right) = O\left(\frac{n^2 + n \lg t}{\varepsilon}\right)$

we have n sets $R_1, \ldots, R_n$

The algorithm runs $O\left(\frac{n^3 + n^2 \lg t}{\varepsilon}\right)$ time.