FPT Approximations for Fair k-Min-Sum-Radii

Mattis Gerteis
Heinrich-Heine-Universität Düsseldorf

8. Mai 2025

Guessing Radii from a given k-center solution

Algorithm 1 Very Broad Strokes

Input: k-center solution F_0, largest Radius of k-center solution r_1^*, approximation factor β, k, ϵ
Output: set R of $O(log^k_{1+epsilon}(k/\epsilon))$ radius profiles

1: $//\tilde{r}_1 =$ as the optimal largest radius is within one interval of:
2: $lowerBound = (1+\epsilon)^{j-1} * r_1^*/\beta$
3: $upperBound = F_0k$
4: step through interval with step size $(1+\epsilon)^{j-1}\frac{F_0}{\beta} \mid 1 \leq j < k$
5: **for all** steps in this interval **do**
6: use that step as upper bound and $\frac{\epsilon}{k} * r_n$ as lower bound
7: find the rest of the intervals as before
8: repeat for all subintervals again until a depth of k is reached

---

Algorithm 2 Medium Granularity

Input: k-center solution F_0, largest Radius of k-center solution r_1^*, approximation factor β, k, ϵ
Output: set R of $O(log^k_{1+epsilon}(k/\epsilon))$ radius profiles

1: $//\tilde{r}_1 =$ as the optimal largest radius is within one interval of:
2: $R_1 = \emptyset$
3: $lowerBound = (1+\epsilon)^{j-1} * r_1^*/\beta$
4: append $lowerBound$ to R_1
5: $j = 1$
6: **while** $j < \log_{1+\epsilon}(\beta k)$ and last element of $R_1 < F_0k$ **do**
7: append $(1+\epsilon)^{j-1}\frac{F_0}{\beta}$ to R_1
8: j++
9: append F_0k to R_1
10: $//$ R_1 now contains all guesses for the largest radius
11: $j = 2$
12: **for all** r_n in R_1 **do**
13: use r_n as upper bound and $\frac{\epsilon}{k} * r_n$ as lower bound
14: find the rest of the intervals as in line 6-8 and append to R_{1n}
15: repeat for each new radius to calculate profiles $R_{11}...R_{nml...}$ and increment j until a depth of k is reached

---

Algorithm 3 Exact Algorithm

Input: k-center solution F_0, largest Radius of k-center solution r_1^*, approximation factor β, k, ϵ

Output: set R of $O(log_{1+epsilon}^k(k/\epsilon))$ radius profiles

```
R = ∅
//r̃_1 = as the optimal largest radius is within one interval of:
R_1 = ∅
lowerBound = (1 + ε)^{j−1} ∗ r_1^*/β
append lowerBound to R_1
j = 1
while j < log_{1+ε}(βk) and last element of R_1 < F_0 k do
    append (1 + ε)^{j−1} F_0/β to R_1
    j++
append F_0 k to R_1
// R_1 now contains all guesses for the largest radius
T = empty tree with root node
for all r in R_1 do
    add r as child to root
while tree depth ≤ k do
    dfs traversal
    if node does not have children and node depth < k then
        add ε/k ∗ r as child with r as value of the node
        j =node depth
        while j < log_{1+ε}(βk) and last added child < r do
            add (1 + ε)^{j−1} r/β as child
            j++
    else
        continue
R = list of dfs searches, with each leaf node reached as entry
return R
```