Colorful Priority k-Supplier

Chandra Chekuri* Junkai Song†

June 24, 2024

Abstract

In the *Priority k-Supplier* problem the input consists of a metric space $(F \cup C, d)$ over set of facilities F and a set of clients C, an integer $k > 0$, and a non-negative radius r_v for each client $v \in C$. The goal is to select k facilities $S \subseteq F$ to minimize $\max_{v \in C} \frac{d(v,S)}{r_v}$ where $d(v, S)$ is the distance of v to the closes facility in S. This problem generalizes the well-studied k-Center and k-Supplier problems, and admits a 3-approximation [Ple87, BCCN22]. In this paper we consider two outlier versions. The *Priority k-Supplier with Outliers* problem [BCCN22] allows a specified number of outliers to be uncovered, and the *Priority Colorful k-Supplier* problem is a further generalization where clients are partitioned into c colors and each color class allows a specified number of outliers. These problems are partly motivated by recent interest in fairness in clustering and other optimization problems involving algorithmic decision making.

We build upon the work of [BCCN22] and improve their 9-approximation *Priority k-Supplier with Outliers* problem to a $1+3\sqrt{3} \approx 6.196$-approximation. For the *Priority Colorful k-Supplier* problem, we present the first set of approximation algorithms. For the general case with c colors, we achieve a 17-pseudo-approximation using $k+2c-1$ centers. For the setting of $c = 2$, we obtain a 7-approximation in random polynomial time, and a $2 + \sqrt{5} \approx 4.236$-pseudo-approximation using $k + 1$ centers.

1 Introduction

The discrete k-Center problem is a well-studied clustering problem with several applications. The input consists a finite metric space (X, d) and an integer k. The goal is to choose k centers $S \subseteq X$ to minimize $\max_{v \in X} d(v, S)$ where $d(v, S)$ is the distance of v to the set S (defined as $\min_{u \in S} d(u, v)$). This is an NP-Hard problem that admits a 2-approximation algorithm [HS86, Gon85] and moreover this approximation ratio is tight assuming $P \neq NP$ [HS86]. The k-Supplier problem is a generalization in which $X = F \uplus C$ with F denoting a set of facilities and C denoting a set of clients. Now the goal is to find a set S of k facilities to minimize $\max_{v \in C} d(v, S)$. The k-Supplier problem admits a 3-approximation and this factor is also tight under $P \neq NP$ [HS86]. Plesník [Ple87] considered *priority* versions of these problems. The input now consists of radius requirement r_v for each client $v \in C$. The goal is to select centers to minimize $\max_{v \in C} \frac{d(v,S)}{r_v}$. [1] Pleaník generalized the 2-approximation algorithm for k-Center of [HS86] to Priority k-Center.

*Dept. of Computer Science, University of Illinois, Urbana-Champaign, IL 61820. Work on this paper partially supported by NSF grant CCF-1910149. chekuri@illinois.edu

†The University of Hong Kong. Part of this work was done while visiting University of Illinois, Urbana-Champaign. dsgsjk@connect.hku.hk

[1] Plesník used the terminology "weighted" k-Center for his problem. [BCCN22] et al. used the terminology of "priority" to avoid confusion with another problem, and due to the reformulation via the notion of client radii which correspond to the inverse of weights.

Bajpai et al. [BCCN22] recently observed that Plesník's algorithm and analysis can be extended to obtain a 3-approximation for Priority k-Supplier and some further generalizations that constrain the centers in more complex ways (matroid and knapsack constraints).

Bajpai et al. [BCCN22] also considered the *outlier* versions of priority clustering problems. They were motivated by recent interest in incorporating fairness in clustering objectives. In the *Priority k-Supplier with Outliers* ($\mathsf{P}k\mathsf{SO}$) problem [BCCN22], an additional parameter m is given and the goal is to cover at least m clients. That is, minimize α such that for at least m clients $v \in C$, $d(v, S) \leq \alpha \cdot r_v$. In this paper we consider a further generalization, the *Priority Colorful k-Supplier* ($\mathsf{PC}k\mathsf{S}$) problem. In this version, the client set C is partitioned into c colors $\{C_1, C_2, \ldots, C_c\}$ and c integer parameters $m_1, m_2, \ldots, m_c$ are given. The goal is to cover at least m_i clients from each color i. That is, minimize α such that for at least m_i clients $v \in C_i$, $d(v, S) \leq \alpha \cdot r_v$ for any $1 \leq i \leq c$. The colorful version of k-Center was introduced in [BIPV19] and has seen several results [AKZ22, JSS22]. As far as we know, $\mathsf{PC}k\mathsf{S}$ has not been formally studied in previous work.

For the k-Center with Outlier problem, Charikar et al [CKMN01] obtained a clever greedy 3-approximation algorithm. The approximation ratio was later improved to 2 due to Chakrabarty et al [CGK20] using an LP-based approach. The Colorful k-center probelm is proposed by Bandyapadhyay et al [BIPV19] and they introduced a pseudo-approximation algorithm that yields a 3-approximation using at most $k + c - 1$ centers. Subsequently, Jia et al [JSS22] obtained a true 3-approximation algorithm when c is a fixed constant. Anegg et al [AKZ22] obtain an $O(1)$-approximation algorithm when c is a fixed constant for generalizations including matroid and knapsack versions.

The Priority k-Supplier with Outliers ($\mathsf{P}k\mathsf{SO}$) problem is studied in [BCCN22]. In the paper, the authors made use of the LP-based framework from [CGK20, CN19] and obtained a 9-approximation algorithm. Futhermore, they showed that this result can be extened to the matroid and the knapsack settings, obtaining a 9-approximation and a 17-approximation respectively[2]. In terms of hardness of approximation, we only know a hardness factor of 3 which is inherited from the k-Supplier problem. There is a natural LP relaxation for the problem and no factor worse than 3 is known on its integrality gap.

Results and ideas: For $\mathsf{P}k\mathsf{SO}$ we improve upon the algorithm in [BCCN22], deriving $1+3\sqrt{3} \approx 6.196$-approximation. [BCCN22] considered special cases when the number of distinct client radii is small. They obtained a 3-approximation for two distinct radii (which is tight), and a 5-approximation for three radii. We improve the approximation ratio for three radii from 5 to 3.94. When the radii are powers of b, the approximation ratio is improved from $\frac{3b-1}{b-1}$ to $\frac{3b^2-1}{b^2-1}$.

For $\mathsf{PC}k\mathsf{S}$ there were no previous results. We derive a 17-pseudo-approximation that uses $k + 2c - 1$ centers. For a special case of $\mathsf{PC}k\mathsf{S}$ where there are only 2 colors and the vertices with the same color has a same radius, we obtain a $2 + \sqrt{5} \approx 4.236$-pseudo-approximation using $k + 1$ centers. In addition, based on the framework due to [AKZ22], we obtain a true 7-approximation in randomized polynomial time.

Our results build on the framework in [BCCN22] that has several ingredients: (i) solving an LP relaxation to obtain a fractional solution (ii) rounding the client radii to powers of b for some parameter $b > 1$ to create well-separated radius classes (iii) pre-processing the clients in each radius class based on the LP solution to create well-separated cluster centers in each radius class (iv) creating a contact DAG on the cluster centers, distances between them, and the coverage information from the LP solution, and solving an auxiliary flow problem that allows the algorithm

[2]The approximation ratio for the knapsack version claimed in [BCCN22] is 14 but there is a minor error in the analysis that when corrected yields a slightly weaker bound of 17. See Section 4 for detailed discussion.

to identify the centers that form the final output. For the knapsack constraint, instead of using a contact DAG, the framework creates a contact forest and uses dynamic programming on the forest and a cut-and-round approach to solve the underlying LP relaxation — this more complex approach is needed since the underlying flow problem on the contact DAG would not yield an integer polytope/solution. For PkSO we obtain an improved approximation by altering the way in which the contact DAG is created and balancing the parameters based on this different approach. We note that this optimization is specific to PkSO and does not extend to the the more general matroid constraint. For PCkS we build on the the insight from [BIPV19] who showed that one can use the LP solution to process the given instance to obtain a feasible solution to a related LP that has a small number of constraints; one can then take a basic feasible solution of this new LP which yields a pseudo-approximation for the original problem. In our setting the priorities create a non-trivial challenge. We use the LP solution to a create a contact forest (we borrow the approach from the knapsack version of PkSO that we already mentioned) — although the approximation ratio is worse if we use a contact forest instead of a contact DAG, we take advantage of the forest structure to show the existence of a basic feasible solution to an auxiliary LP that has only a few fractional values. This gives us the desired pseudo-approximation.

Tabel 1 summarizes the approximation ratios from the past work and this paper.

Problems	**Prior**	**This Paper**
Priority k-Center	2 [Ple87]	
Priority k-Supplier	3 [BCCN22]	
k-Center with Outliers	2 [CGK20]	
k-Supplier with Outliers	3 [CGK20]	3
Priority k-Supplier with Outliers		
General	9 [BCCN22]	$1+3\sqrt{3} \approx 6.196$
2 types of radii	3 [BCCN22]	3
3 types of radii	5 [BCCN22]	3.94
Radii are powers of b	$\frac{3b-1}{b-1}$ [BCCN22]	$\frac{3b^2-1}{b^2-1}$
Priority Knapsack Supplier with Outliers	14 [BCCN22]	17
Colorful k-Center		
2 colors	$(2, k+1)$ [BIPV19]	
	$(3, k)$ [JSS22]	
c colors	$(2, k+c-1)$ [BIPV19]	
	$(O(1), k)$ [AKZ22]	
Priority Colorful k-Supplier		
2 colors & same color, same radius		$(2+\sqrt{5}, k+1)$
		$(7, k)$
c colors		$(17, k+2c-1)$

Table 1: Summary of known and new results.

Organization: Section 2 contains formal definitions of the problems, the LP relaxation we need and a basic filitering technique. Section 3 describes our improvement for PkSO. Section 4 describes the Knapsack version of PkSO from [BCCN22], for the sake of completeness, since we need the details of that approach. Section 5 describes our algorithms for PkSO. We conclude in Section 6 with a discussion of some open problems.

2 Preliminaries

Definition 1 (Priority k-Supplier with Outliers (PkSO))**.** The input is a metric space $(X = C \cup F, d)$, a radius function $r : C \to \mathbb{R}_{>0}$, and parameters $k, m \in \mathbb{N}$. The goal is to find $S \subseteq F$ of size at most k to minimize α such that for at least m vertices $v \in C$, $d(v, S) \leq \alpha \cdot r_v$.

Since there are only polynomially many choices for an optimal α, we restrict our attention to the decision version of PkSO and assume that the optimal value α^* is equal to 1 by appropriately scaling the radii. We further assume that for each $v \in C$, there exists some $f \in F$ such that $d(f, v) \leq r_v$, otherwise we have a certificate that $\alpha^* > 1$.

The following is the natural LP relaxation for the decision version for of PkSO with $\alpha = 1$. x_f indicates how much a facility f is opened as a center and $\mathsf{cov}(v)$ indicates how much a vertex v is covered.

$$\begin{aligned}
&\sum_{v \in C} \mathsf{cov}(v) \geq m && && \text{(P}k\text{SO LP)}\\
&\sum_{f \in F} x_f \leq k\\
&\mathsf{cov}(v) = \min\Big(\sum_{\substack{f \in F:\\ d(f,v) \leq r_v}} x_f, 1\Big) && && \forall v \in C\\
&0 \leq x_f \leq 1 && && \forall f \in F
\end{aligned} \tag{1}$$

In the preceding formulation the constraint $\mathsf{cov}(v) = \min(\sum_{\substack{f \in F:\\ d(f,v) \leq r_v}} x_f, 1)$ is not linear but can be replaced with two linear constraints; we keep this notation for ease of understanding.

Filter is a standard procedure in clustering problems to partition the vertex set into several well-separated clusters. In Algorithm 1, the vertices are ordered according to their cov values, and a set of representatives R and corresponding clusters $\{D(v) : v \in R\}$ are obtained. Similar algorithms are also used in [HS86] and [Ple87]: in [HS86], the vertex ordering is arbitrary, and in [Ple87], the vertices are sorted based on their radii.

Algorithm 1 Filter

Input: Metric $(X = C \cup F, d)$, radius function r, and LP solution cov

$U \leftarrow C$
$R \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  $v \leftarrow \arg\max_{u \in U} \mathsf{cov}(u)$
  $R \leftarrow R \cup \{v\}$
  $D(v) \leftarrow \{u \in U : d(u, v) \leq r_u + r_v\}$
  $U \leftarrow U \setminus D(v)$
end while

Output: R, $\{D(v) : v \in R\}$

Lemma 1. The output of Algorithm 1 has the following properties:
(a) $\{D(v) : v \in R\}$ is a partition of C.
(b) $\forall u, v \in R, d(u, v) > r_u + r_v$.
(c) $\forall v \in R, u \in D(v), d(u, v) \leq r_u + r_v$ and $\mathsf{cov}(v) \geq \mathsf{cov}(u)$.

To warm up, we provide a 3-approximation algorithm for k-Supplier with Outliers from [CGK20, BCCN22]. Note that in the non-priority setting, the radius of every vertex is the same.

Theorem 2. There is a 3-approximation algorithm for k-Supplier with Outliers.

Proof. We first obtain a solution of PkSO LP and run Algorithm 1. Consider the following auxiliary LP where y_v indicates how much v is opened as a center (note that v is a client here, by opening v we mean open some $f \in F$ where $d(f, v) \le r_v$):

$$\begin{aligned} \max \quad & \sum_{v \in R} |D(v)| y_v \\ & \sum_{v \in R} y_v \le k \\ & 0 \le y_v \le 1 \qquad\qquad \forall v \in R \end{aligned}$$

We claim that $y_v = \mathsf{cov}(v)$ is a feasible fractional solution with objective value at least m. For every $f \in F$, let $A_f = \{v \in C : d(f, v) \le r_v\}$ be the set of vertices f can cover. Note that $|A_f \cap R| \le 1$ since otherwise for $u, v \in A_f \cap R$, we have $d(u, v) \le d(u, f) + d(f, v) \le r_u + r_v$, which contradicts to Lemma 1(b). Hence

$$\begin{aligned} \sum_{v \in R} \mathsf{cov}(v) &= \sum_{v \in R} \sum_{\substack{f \in F: \\ d(f,v) \le r_v}} x_f \\ &= \sum_{v \in R} \sum_{f : v \in A_f} x_f \\ &= \sum_{f \in F} |A_f \cap R| x_f \\ &\le \sum_{f \in F} x_f \le k. \end{aligned}$$

Hence $y_v = \mathsf{cov}(v)$ is feasible. On the other hand, we have

$$\begin{aligned} \sum_{v \in R} |D(v)| \mathsf{cov}(v) &\ge \sum_{v \in R} \sum_{u \in D(v)} \mathsf{cov}(u) && \text{(By Lemma 1(c))} \\ &= \sum_{v \in C} \mathsf{cov}(v) && \text{(By Lemma 1(a))} \\ &\ge m. \end{aligned}$$

The auxiliary LP defines an integer polyhedron since the constraint system corresponds to that of a uniform matroid of rank k (see [S$^+$03]). Since there exists a fractional solution with objective value at least m, we can obtain an integral solution with objective value at least m by choosing k sets from $\{D(v) : v \in R\}$ with largest cardinality. For each chosen v, open any facility f such that $d(f, v) \le r_v$. For all chosen v, every $u \in D(v)$ is covered within 3 times its radius since

$$d(u, f) \le d(u, v) + d(v, f) \le (r_u + r_v) + r_v = 3r_u.$$

Therefore, at least m vertices are covered using at most k facilities within 3 times the radius. □

3 Priority k-Supplier with Outliers

In this section, we introduce our improved algorithm utilizing the framework in [BCCN22], which gives $(1+3\sqrt{3})$-approximation for PkSO. Part of the section is replicating [BCCN22] for the sake of completeness.

Definition 2 (Weighted k-Path Packing (WkPP))**.** The input is a DAG $G=(V,E)$, a value function $\lambda : V \to \{0,1,\dots,n\}$ for some integer n, and a parameter k. The goal is to find a set of k paths $P \subseteq \mathcal{P}(G)$ that maximizes:

$$\mathsf{val}(P) = \sum_{v \in \bigcup_{p\in P} p} \lambda(v).$$

The problem is polynomial time solvable by a reduction to Min-Cost Max-Flow. We build a new graph $G'=(V',E')$ based on $G=(V,E)$. For each vertex $v \in V$, split it into two copies v_1, v_2. This is to ensure each vertex is count only once even if it is covered by more than one paths.

We set $V' = \bigcup_{v\in V}\{v_1,v_2\} \cup \{s,t\}$ and

$$E' = \bigcup_{v\in V} \begin{Bmatrix} (v_1,v_2) \text{ with capacity 1 and cost } -\lambda(v) \\ (v_1,v_2) \text{ with capacity } \infty \text{ and cost 0} \\ (s,v_1) \text{ with capacity } \infty \text{ and cost 0} \\ (v_2,t) \text{ with capacity } \infty \text{ and cost 0} \end{Bmatrix} \cup \bigcup_{(u,v)\in E} (u_2,v_1) \text{ with capacity } \infty \text{ and cost 0.}$$

Additionlly, the flow passing through sink t is capped by k. It is easy to see that WkPP is equivalent to the Min-Cost Max-Flow from s to t on G'.

The following LP is the Min-Cost Max-Flow LP for WkPP instance. y_e for $e \in E'$ indicates the amount of flow passing through e and $\mathsf{flow}(v)$ for $v \in V$ indicates the amount of flow passing through the arc (v_1,v_2) with capacity 1 and cost $-\lambda(v)$.

$$\begin{aligned}
\max \quad & \sum_{v\in V} \lambda(v)\mathsf{flow}(v) && && \text{(W}k\text{PP LP)} \\
& \sum_{e\in\delta^-(v_1)} y_e = \sum_{e\in\delta^+(v_2)} y_e && \forall v \in V \\
& \mathsf{flow}(v) = \min(\sum_{e\in\delta^-(v_1)} y_e, 1) && \forall v \in V \\
& \mathsf{flow}(t) = \sum_{e\in\delta^-(t)} y_e \le k \\
& 0 \le y_e \le 1 && \forall e \in E'
\end{aligned}$$

Now we are ready to present our algorithm to solve PkSO. We partition the vertex sets into several layers $L_1,\dots,L_t$ (in some ways that will be specified later). In each layer i, run Algorithm 1 on L_i and get the set of representatives R_i and corresponding clusters $\{D(v) : v \in R_i\}$. Let $V = \bigcup_{i=1}^{t} R_i$, we build the contact DAG in the following way.

Definition 3 (contact DAG)**.** Contact DAG $G=(V,E)$ is a DAG on vertex set V where each vertex $v \in V$ has a weight $\lambda(v) = |D(v)|$. For $u \in R_i$ and $v \in R_j$ where $i > j$:

$$(u,v) \in E \iff \exists f \in F : d(f,u) \le r_u \text{ and } d(f,v) \le r_v.$$

Algorithm 2 Find Paths 1

Input: Metric $(X = C \cup F, d)$, radius function r, and LP solution cov

1: **for** $i = 1$ to t **do**
2: $\quad R_i, \{D(v) : v \in R_i\} \leftarrow \text{Filter}((L_i \cup F, d), r, \mathsf{cov})$
3: **end for**
4: Construct contact DAG $G = (V, E)$ according to Definition 3
5: Get a solution P for WkPP on G, i.e. an integral solution to the WkPP LP

Output: P

By solving WkPP on the contact DAG, we obtain a collection of k paths P. The above steps are summarized in Algorithm 2. Then, we will choose a center to open for each path, and the resulting union of the paths represents the vertices that we cover.

Lemma 3. The output of Algorithm 2 P satisfies

$$\mathsf{val}(P) = \sum_{v \in \bigcup_{p \in P} p} \lambda(v) \geq m.$$

Proof. Note P implies an integral solution to the WkPP LP, with $\mathsf{val}(P)$ equals to the objective value. Similar to the proof of Theorem 2, we show that there is a feasible fractional solution to the WkPP LP with objective value at least m. Since the WkPP LP is integral, thereby we have $\mathsf{val}(P) =$ optimal integral solution $\geq m$.

We first obtain a solution of PkSO LP. For any facility $f \in F$, let $A_f = \{v \in V : d(f, v) \leq r_v\}$ be the set of vertices that f can cover. For all R_i, $|A_f \cap R_i| \leq 1$ due to Lemma 1(b). Let p_f be the path in contact DAG connecting vertices in A_f and s, t in topological order. We add x_f to all y_e for $e \in p_f$ and increase $\mathsf{flow}(v)$ by x_f for all $v \in A_f$ (taking minimum with 1).

First we claim that this is a feasible solution to WkPP LP. Clearly it satisfies flow conservation $\sum_{e \in \delta^-(v_1)} y_e = \sum_{e \in \delta^+(v_2)} y_e$ since each time we add values to a path from s to t. Also $\mathsf{flow}(t) \leq k$ since $\mathsf{flow}(t) = \sum_{f \in F} x_f \leq k$.

Next we argue that the objective value is at least m. Note that

$$\mathsf{flow}(v) = \min(\sum_{\substack{f \in F: \\ v \in A_f}} x_f, 1) = \min(\sum_{\substack{f \in F: \\ d(f,v) \leq r_v}} x_f, 1) = \mathsf{cov}(v).$$

and thus we have

$$\begin{aligned}
\sum_{v \in V} \lambda(v)\mathsf{flow}(v) &= \sum_{v \in V} \lambda(v)\mathsf{cov}(v) \\
&= \sum_{v \in V} |D(v)|\mathsf{cov}(v) \\
&\geq \sum_{v \in V} \sum_{u \in D(v)} \mathsf{cov}(u) \\
&= \sum_{v \in C} \mathsf{cov}(v) \geq m.
\end{aligned}$$

Since the WkPP LP is integral, the optimal integral solution found by Algorithm 2 has an objective value at least m. ∎

Theorem 4. There is a $\frac{3b^2-1}{b^2-1}$-approximation algorithm if the radii are power of b.

Proof. Suppose radii are from b^0 to b^{t-1} where t is even, let $B_i = \{v \in X : r_v = b^{i-1}\}$. We partition the vertex set C such that $L_0 = B_{t-1}, L_1 = B_{t-3}, \ldots, L_{\frac{t}{2}-1} = B_1, L_{\frac{t}{2}} = B_0, L_{\frac{t}{2}+1} = B_2, \ldots, L_{t-1} = B_{t-2}$ and run Algorithm 2. According to Lemma 3, we have $\sum_{v \in \bigcup_{p \in P}} |D(v)| \geq m$. Therefore, we only need to show that $\bigcup_{v \in \bigcup_{p \in P}} D(v)$ can be covered within $\frac{3b^2-1}{b^2-1}$ times the radius.

For each path $p \in P$ from the output of Algorithm 2, consider the worst case that it has a vertex u_i from every L_i. It will be clear right off why this is the worst case. Let f_i be an arbitrary facility covers both u_{i-1} and u_i, i.e. $d(f_i, u_{i-1}) \leq r_{u_{i-1}}$ and $d(f_i, u_i) \leq r_{u_i}$. It always exists by the construction of contact DAG. We open the facility $f_{\frac{t}{2}}$. Fix some $i \leq \frac{t}{2} - 1$. For all $v \in D_{u_i}$, we have

$$\begin{aligned}
\frac{d(v, f_{\frac{t}{2}})}{r_v} &\leq \frac{1}{r_v}\big(d(v, u_i) + d(u_i, f_{i+1}) + d(f_{i+1}, f_{i+2}) + \cdots + d(f_{\frac{t}{2}-1}, f_{\frac{t}{2}})\big) \\
&\leq \frac{1}{r_v}\big((r_v + r_{u_i}) + r_{u_i} + 2 \cdot r_{u_{i+1}} + \cdots + 2 \cdot r_{u_{\frac{t}{2}-1}}\big) \\
&= 3 + 2 \cdot (b^{-2} + b^{-4} + \cdots + b^{2i+2-t}) \\
&= 3 + 2 \cdot \frac{b^{-2} - b^{2i-t}}{1 - b^{-2}} \\
&\leq 3 + 2 \cdot \frac{b^{-2}}{1 - b^{-2}} = \frac{3b^2 - 1}{b^2 - 1}.
\end{aligned}$$

For $i \geq \frac{t}{2}$ and odd t, the proof is analogous.

Note that if u_i is missing from some layers L_i, we can always open the facility in the middle. Formally, for largest $i < \frac{t}{2}$ such that u_i exists and smallest $j \geq \frac{t}{2}$ such that u_j exists, open a facility f covers both u_i and u_j, i.e. $d(f, u_i) \leq r_{u_i}$ and $d(f, u_j) \leq r_{u_j}$. For any i and $v \in D_{u_i}$, the worst case distance between v and f can only be smaller when some u_i's are missing since we can take shortcuts by skipping the missing layers. This completes the proof. □

Theorem 5. There is a $(1 + 3\sqrt{3})$-approximation algorithm for PkSO.

Proof. For some b that will be chosen later, suppose $t = \lceil \log_b r_{max} \rceil$, let $B_i = \{v \in X : b^{i-1} \leq r_v < b^i\}$. We partition the vertex set C such that $L_0 = B_{t-1}, L_1 = B_{t-3}, \ldots, L_{\frac{t}{2}-1} = B_1, L_{\frac{t}{2}} = B_0, L_{\frac{t}{2}+1} = B_2, \ldots, L_{t-1} = B_{t-2}$ and run Algorithm 2. Similar to the proof of Theorem 4, we only need to show that $\bigcup_{v \in \bigcup_{p \in P}} D(v)$ can be covered within $1 + 3\sqrt{3}$ times the radius.

Following the notation in the proof of Theorem 4, for each path $p \in P$, we open $f_{\frac{t}{2}}$ and for all $v \in D_{u_i}, i \leq \frac{t}{2} - 1$, we have

$$\begin{aligned}
\frac{d(v, f_{\frac{t}{2}})}{r_v} &\leq \frac{1}{r_v}\big(d(v, u_i) + d(u_i, f_{i+1}) + d(f_{i+1}, f_{i+2}) + \cdots + d(f_{\frac{t}{2}-1}, f_{\frac{t}{2}})\big) \\
&\leq \frac{1}{r_v}\big((r_v + r_{u_i}) + r_{u_i} + 2 \cdot r_{u_{i+1}} + \cdots + 2 \cdot r_{u_{\frac{t}{2}-1}}\big) \\
&\leq 1 + 2 \cdot b + 2 \cdot (b^{-1} + b^{-3} + \cdots + b^{2i+3-t}) \\
&= 1 + 2 \cdot b + 2 \cdot \frac{b^{-1} - b^{2i+1-t}}{1 - b^{-2}} \\
&\leq 1 + 2 \cdot b + 2 \cdot \frac{b^{-1}}{1 - b^{-2}} = \frac{2b^3 + b^2 - 1}{b^2 - 1}.
\end{aligned}$$

which attains its minimum $1+3\sqrt{3}$ when $b=\sqrt{3}$. □

We remark that the main procedures of our algorithm are the same as those in the algorithm of [BCCN22] (see their Section 3). The main difference is in the way we order the layers. In [BCCN22], they set $L_0 = B_0, L_1 = B_1, \ldots, L_{t-1} = B_{t-1}$ and open a facility on one side. In our case, we open a facility in the middle and place the layers alternately on two sides. This has enabled us to achieve an improved approximation.

When the number of distinct radii is small, we provide a technique that further improves the approximation ratio: we can consider the relationship of the radii of different layers and decide whether contract them or not. We start with the known result for 2 radii.

Theorem 6. There is a 3-approximation algorithm when there are two different radii.

Proof. Denote by r_0, r_1 the 2 radii. We partition the vertex set to $L_0 = \{v \in C : r_v = r_0\}, L_1 = \{v \in C : r_v = r_1\}$ and run Algorithm 2. We only need to show that $\bigcup_{v \in \bigcup_{p \in P}} D(v)$ can be covered within 3 times the radius.

Consider a path $\{u_0, u_1\} \in P$ with $v_0 \in D(u_0), v_1 \in D(u_1)$, which is illustrated in Figure 1. If we open f_1, it is easy to see that $d(v_0, f_1) \leq 3r_0$ and $d(v_1, f_1) \leq 3r_1$. □

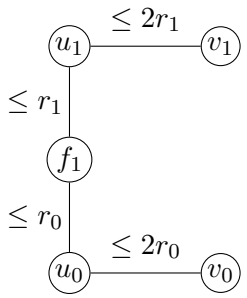

Figure 1: Illustration of 2 radii

Theorem 7. There is a 3.94-approximation algorithm when there are three different radii.

Proof. Denote by r_0, r_1, r_2 the 3 radii and $B_i = \{v \in C : r_v = r_i\}$. As shown in Figure 2, consider three ways to partition the vertex set: (a) $L_0 = B_0, L_1 = B_1, L_2 = B_2$; (b) $L_0 = B_0, L_1 = B_1 \cup B_2$; (c) $L_0 = B_0 \cup B_1, L_1 = B_2$. Let $\alpha = \frac{r_1}{r_0}, \beta = \frac{r_2}{r_1}$. If we open f_1, it is easy to see that the worst case approximation ratio are $\frac{3r_2+2r_0}{r_2} = 3 + \frac{2}{\alpha\beta}, \frac{r_1+2r_2}{r_1} = 1 + 2\alpha, \frac{r_0+2r_1}{r_0} = 1 + 2\beta$ respectively. Given $\alpha, \beta > 1$, $\min(3 + \frac{2}{\alpha\beta}, 1 + 2\alpha, 1 + 2\beta)$ attains its maximum ≈ 3.9311 at $\alpha = \beta \approx 1.47$. □

We remark that the technique is useful for more than 3 radii, but we omit this part due to the tedious case analysis.

4 Priority Knapsack Supplier with Outliers

In this section, we describe a modification to the algorithm in Section 3 following [BCCN22], by trading off a slight approximation ratio loss to force the contact DAG to be a forest. In [BCCN22], the authors claimed a 14-approximation for PKnapSO. However, there is a minor error in Claim 19 of that paper. We reproduce their result in this section to fix the error, and since we also need to use the DAG-to-forest technique in our algorithm for PCkS.

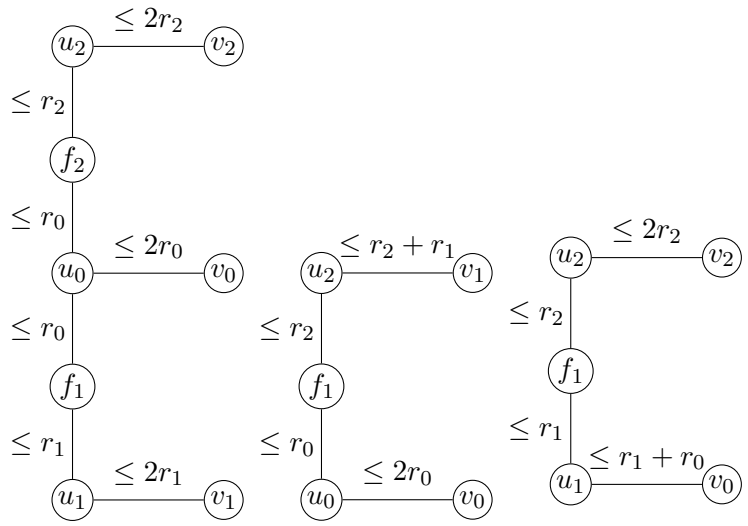

Figure 2: Illustration for the case of 3 distinct radii

Definition 4 (Priotrity Knapsack Supplier with Outliers (PKnapSO))**.** The input is a metric space $(X = C \cup F, d)$, a radius function $r : C \to \mathbb{R}_{>0}$, a weight function $w : F \to \mathbb{R}_{\geq 0}$, and parameters $k, m \in \mathbb{N}$. The goal is to find $S \subseteq F$ with $w(S) := \sum_{f \in S} w(f) \leq k$ to minimize α such that for at least m vertices $v \in C, d(v, S) \leq \alpha \cdot r_v$.

Since the natural LP relaxation of WKnapPP has an unbounded integrality gap [CLLW16], we are going to use an exponential size configuration LP and solve it implicitly using the ellipsoid method. Let $\mathscr{F} := \{S \subseteq F : w(S) \leq k\}$. Consider the following convex hull of the integral solutions for PKnapSO. z_S indicates how much a set of facilities $S \in \mathscr{F}$ is opened as the centers.

$$
\begin{aligned}
\mathscr{P}_{\mathsf{cov}} = \{ \quad & (\mathsf{cov}(v) : v \in C) : \\
& \mathsf{cov}(v) = \min\Big(\sum_{\substack{S \in \mathscr{F}: \\ d(v,S) \leq r_v}} z_S, 1 \Big) && \forall v \in C, \\
& \sum_{S \in \mathscr{F}} z_S \leq 1, \\
& 0 \leq z_S \leq 1 && \forall S \in \mathscr{F} \quad \}.
\end{aligned}
$$

Note that the dimension of $\mathscr{P}_{\mathsf{cov}}$ is polynomial although there are exponentially many auxiliary variables z_S's.

Definition 5 (Weighted Knapsack Path Packing (WKnapPP))**.** The input is a directed *out-forest* $G = (V, E)$, a value function $\lambda : V \to \{1, 2, \ldots, n\}$ for some integer n, a weight function $w' : V \to \mathbb{R}_{\geq 0}$ and a parameter k. The goal is to find a set of paths $P \subseteq \mathcal{P}(G)$ such that $\sum_{p \in P} w'(\mathrm{sink}(p)) \leq k$ that maximizes:

$$\mathsf{val}(P) = \sum_{v \in \bigcup_{p \in P} p} \lambda(v).$$

For $v \in V$, its weight $w'(v)$ is defined as

$$w'(v) = \min_{\substack{f \in F: \\ d(f, \mathrm{sink}(p)) \leq r_{\mathrm{sink}(p)}}} w(f).$$

Note that we need a forest instead of a DAG here since solving the WKnapPP problem is hard on DAGs. When it is a forest one can use a dynamic programming algorithm to solve the problem in polynomial time (c.f. Appendix A of [BCCN22]).

4.1 DAG to Forest

To ensure that the contact DAG is a directed out-forest, we use Algorithm 3 as a subroutine instead of Algorithm 1.

Algorithm 3 Modified Filter

Input: Metric $(X = C \cup F, d)$, radius function r, LP solution cov, and a parameter ℓ

$U \leftarrow C$
$R \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  $v \leftarrow \arg\max_{u \in U} \mathsf{cov}(u)$
  $R \leftarrow R \cup \{v\}$
  $D(v) \leftarrow \{u \in U : d(u, v) \leq r_u + r_v + \ell\}$
  $U \leftarrow U \setminus D(v)$
end while

Output: R, $\{D(v) : v \in R\}$

Suppose $t = \lceil \log_4 r_{max} \rceil$, we partition the vertex set C into t layers such that $L_i = \{v \in C : 4^{i-1} \leq r_v < 4^i\}$. In each layer i, run Algorithm 3 with additional parameter 4^i on L_i and get the set of representatives R_i and corresponding clusters $\{D(v) : v \in R_i\}$. Let $V = \bigcup_{i=1}^{t} R_i$, build the contact forest in the following way:

Definition 6 (contact forest)**.** Contact DAG $G = (V, E)$ is a DAG on vertex set V where each vertex $v \in V$ has a value $\lambda(v) = |D(v)|$ and a weight $w(v) = \min_{f \in F : d(f,v) \leq r_v} w(f)$. For $u \in R_i$ and $v \in R_j$ where $i > j$:

$$(u, v) \in E \iff d(u, v) \leq r_u + r_v + 4^j.$$

The contact forest is derived by removing all forward edges from E, i.e. remove $(u, v) \in E$ if there exists a path from u to v of length greater than 2 in G.

Lemma 8. Contact forest is a directed out forest.

Proof. Suppose to the contrary, there are $u \in R_i, v \in R_j, w \in R_k$ where $i \geq j > k$ and $(u, w) \in E, (v, w) \in E$, we have

$$\begin{aligned} d(u, v) &\leq d(u, w) + d(w, v) \\ &= r_u + r_w + 4^k + r_v + r_w + 4^k \\ &\leq r_u + r_v + 2r_w + 2 \cdot 4^k \\ &\leq r_u + r_v + 4^{k+1} \\ &\leq r_u + r_v + 4^j. \end{aligned}$$

If $i = j$, this implies i and j should be in the same cluster in Algorithm 3 and hence should not be in R_i simultaneously. If $i > j$, this implies $(u, v) \in E$ and hence (u, w) is a forward edge that should be removed. □

4.2 Round-or-cut

To solve the LP, we utilize the round-or-cut framework in [CN19]. We use the ellipsoid algorithm on $\mathscr{P}_{\mathsf{cov}}$. In each iteration, after getting a solution cov, we either find a feasible collection of paths which implies a approximate PKnapSO solution, or we can give the ellipsoid algorithm a separating hyperplane. See Algorithm 4.

Algorithm 4 Find Paths 2

Input: Metric $(X = C \cup F, d)$, radius function r

1: Start an ellipsoid algorithm $\mathcal{E}$ on $\mathscr{P}_{\mathsf{cov}}$
2: **while** true **do**
3: $\quad \{\mathsf{cov}(v) : v \in C\} \leftarrow \mathcal{E}$
4: $\quad$ **for** $i = 1$ to t **do**
5: $\quad\quad R_i, \{D(v) : v \in R_i\} \leftarrow$ Modified Filter$((L_i \cup F, d), r, \mathsf{cov}, 4^i)$
6: $\quad$ **end for**
7: $\quad$ Construct contact forest $G = (V, E)$ according to Definition 6
8: $\quad$ Get a solution P for WKnapPP on G
9: $\quad$ **if** $\mathsf{val}(P) \geq m$ **then**
10: $\quad\quad$ **return** P
11: $\quad$ **else**
12: $\quad\quad \mathcal{E} \leftarrow$ a separating hyperplane $\sum_{v \in C} \lambda(v)\mathsf{cov}(v) < m$.
13: $\quad$ **end if**
14: **end while**

Output: P

Lemma 9 (c.f. Lemma 10 of [CN19])**.** If $\{\lambda(v) \in \mathbb{R} : v \in C\}$ satisfies

$$\sum_{\substack{v \in C: \\ d(v,S) \leq r_v}} \lambda(v) < m \quad \forall S \in \mathscr{F},$$

then any $\mathsf{cov} \in \mathscr{P}_{\mathsf{cov}}$ satisfies

$$\sum_{v \in C} \lambda(v)\mathsf{cov}(v) < m.$$

Proof.

$$\begin{aligned}\sum_{v \in C} \lambda(v)\mathsf{cov}(v) &\leq \sum_{v \in C} \lambda(v) \sum_{\substack{S \in \mathscr{F}: \\ d(v,S) \leq r_v}} z_S \\ &= \sum_{S \in \mathscr{F}} z_S \sum_{\substack{v \in C: \\ d(v,S) \leq r_v}} \lambda(v) \\ &< \sum_{s \in \mathscr{F}} z_S \cdot m \leq m.\end{aligned}$$

□

Lemma 10 (c.f. Lemma 23 of [BCCN22])**.** Each time at Line 12 of Algorithm 4, we have

$$\sum_{\substack{v \in C: \\ d(v,S) \leq r_v}} \lambda(v) \leq m \quad \forall S \in \mathscr{F}$$

and

$$\sum_{v \in C} \lambda(v)\mathsf{cov}(v) \geq m$$

where

$$\lambda(v) = \begin{cases} |D(v)| & v \in V \\ 0 & \text{otherwise} \end{cases}.$$

Proof. Fix any $S \in \mathscr{F}$. For some $f \in S$, let $A_f = \{v \in V : d(f, v) \leq r_v\}$ be the set of vertices f can cover and p_f be the path in contact forest connecting vertices in A_f in topological order. Clearly $P' = \bigcup_{f \in S} p_f$ is a feasible solution to $\mathsf{WKnapPP}$. Since P is the optimal solution and $\mathsf{val}(P) < m$, we have

$$m > \mathsf{val}(P) \geq \mathsf{val}(P') \geq \sum_{v \in \bigcup_{f \in S} A_f} \lambda(v) = \sum_{\substack{v \in V: \\ d(v,S) \leq r_v}} \lambda(v) = \sum_{\substack{v \in C: \\ d(v,S) \leq r_v}} \lambda(v).$$

On the other hand, we have

$$\begin{aligned} \sum_{v \in C} \lambda(v)\mathsf{cov}(v) &= \sum_{v \in V} \lambda(v)\mathsf{cov}(v) \\ &= \sum_{v \in V} |D(v)|\mathsf{cov}(v) \\ &\geq \sum_{v \in V} \sum_{u \in D(v)} \mathsf{cov}(u) \\ &= \sum_{v \in C} \mathsf{cov}(v) \geq m. \end{aligned}$$

□

Theorem 11. There is a 17-approximation algorithm for $\mathsf{PKnapSO}$.

Proof. Combining Lemma 9 and Lemma 10, we see that $\sum_{v \in C} \lambda(v)\mathsf{cov}(v) < m$ is indeed a separating hyperplane. Hence the correctness of Algorithm 4.

For the approximation guarantee, we provide a similar proof to Theorem 4. For a path $p \in P$, consider the worst case that it has a vertex u_i from each layer L_i. Open the facility

$$f^* = \arg \min_{\substack{f \in F: \\ d(f, \mathrm{sink}(p)) \leq r_{\mathrm{sink}(p)}}} w(f).$$

For a vertex $v \in D(u_i)$, we have

$$\begin{aligned} \frac{d(v, f^*)}{r_v} &\leq \frac{1}{r_v}\big(d(v, u_i) + d(u_i, u_{i-1}) + \cdots + d(u_2, u_1) + d(u_1, f^*)\big) \\ &\leq \frac{1}{r_v}\big((r_v + r_{u_i} + 4^i) + (r_{u_i} + r_{u_{i-1}} + 4^{i-1}) + \cdots + (r_2 + r_1 + 4) + r_1\big) \\ &\leq \frac{1}{r_v}\big((r_v + 2 \cdot 4^i) + (4^i + 2 \cdot 4^{i-1}) + \cdots + (4^2 + 2 \cdot 4) + 4\big) \\ &\leq 1 + 3 \cdot (4 + 1 + 4^{-1} + \cdots + 4^{2-i}) \\ &= 1 + 4 \cdot (4 - 4^{1-i}) \leq 17. \end{aligned}$$

□

5 Priority Colorful k-Supplier

In this section, we discuss the Priority Colorful k-Supplier problem.

Definition 7 (Priority Colorful k-Supplier (PCkS))**.** The input is a metric space $(X = C \cup F, d)$, a radius function $r : C \to \mathbb{R}_{>0}$, a partition $\{C_1, C_2, \ldots, C_c\}$ of C into c colors, a coverage requirement $0 \le m_i \le |C_i|$ for each color $1 \le i \le c$, and a parameter k. The goal is to find $S \subseteq F$ of size at most k to minimize α such that there are at least m_i vertices v from each color i satisfying $d(v, S) \le \alpha \cdot r_v$.

Definition 8 (PCkS LP)**.**

$$\sum_{v \in C_i} \mathsf{cov}(v) \ge m_i \qquad 1 \le i \le c$$

$$\sum_{f \in F} x_f \le k$$

$$\mathsf{cov}(v) = \min\Big(\sum_{\substack{f \in F: \\ d(f,v) \le r_v}} x_f, 1\Big) \qquad \forall v \in C$$

$$0 \le x_v \le 1 \qquad \forall v \in C$$

Follow the construction in Section 4.1, we can obtain a directed out-forest.

Definition 9 (Weighted Colorful k-Path Packing (WCkPP))**.** The input is a directed out-forest $G = (V, E)$, for each $1 \le i \le c$ a value function $\lambda_i : V \to \{0, 1, \ldots, n\}$ for some integer n, and parameters k, m_i for $1 \le i \le c$. The goal is to find a set of k paths $P \subseteq \mathcal{P}(G)$ such that:

$$\sum_{v \in \bigcup_{p \in P} p} \lambda_i(v) \ge m_i \quad \forall 1 \le i \le c.$$

Denote by L the leaves in the forest and T_v the subtree rooted at v. Since $\lambda_i(v) \ge 0$, we only consider picking the paths from some leaf $v \in L$ to the root. It is clear that we will always choose paths from a leaf to the root.

The following is the natural LP relaxation of the WCkPP problem. y_v for $v \in L$ indicates how much the path from v to the root is chosen. z_v for $v \in V \setminus L$ indicates how much v is covered by the chosen paths. We separately take out the constraint for the first color as the objective value, so that we can save one constraint and thereby save one additional center.

$$\max \quad \sum_{v \in L} \lambda_1(v) y_v + \sum_{v \in V \setminus L} \lambda_1(v) z_v \qquad \text{(WC}k\text{PP LP)}$$

$$\sum_{v \in L} \lambda_i(v) y_v + \sum_{v \in V \setminus L} \lambda_i(v) z_v \ge m_i \qquad 2 \le i \le c \qquad (2)$$

$$z_v \le \sum_{u \in T_v} y_u \qquad \forall v \in V \setminus L \qquad (3)$$

$$\sum_{v \in L} y_v \le k \qquad (4)$$

$$0 \le y_v \le 1 \qquad \forall v \in L \qquad (5)$$

$$0 \le z_v \le 1 \qquad \forall v \in V \setminus L \qquad (6)$$

Similar to the algorithm in Section 4.1, suppose $t = \lceil \log_4 r_{max} \rceil$, we partition the vertex set C into t layers such that $L_i = \{v \in C : 4^{i-1} \leq r_v < 4^i\}$. In each layer i, run Algorithm 3 with additional parameter 4^i on L_i and get the set of representatives R_i and corresponding clusters $\{D(v) : v \in R_i\}$. Then build the contact forest according to Definition 6.

Algorithm 5 Find Paths 3

Input: Metric $(X = C \cup F, d)$, radius function r, and LP solution cov

1: **for** $i = 1$ to t **do**
2: $\quad R_i, \{D(v) : v \in R_i\} \leftarrow$ Modified Filter$((L_i \cup F, d), r, \mathsf{cov}, 4^i)$
3: **end for**
4: Construct contact forest $G = (V, E)$ according to Definition 6
5: Get an extreme point solution (y^*, z^*) to WCkPP LP
6: $S \leftarrow \{v \in L : y^*_v > 0\}$

Output: S

Lemma 12. There exists a solution to WCkPP LP with objective value at least m_1.

Proof. For $f \in F$, let $A_f = \{v \in V : d(f, v) \leq r_v\}$ be the set of vertices f can cover. By the nature of filtering algorithm, A_f contains at most one vertex from each C_i. Suppose v is the vertex in A_f with smallest r_v. We increase some $y_u < 1$ for $u \in T_v \cap L$ by a total amount of x_f and increase z values for the ancestors of u accordingly. All these increase are capped by 1. In this process, it is guaranteed that for all $v \in L$:

$$y_v \geq \sum_{v \in A_f} x_f = \sum_{\substack{f \in F: \\ d(f,r) \leq r_v}} x_f = \mathsf{cov}(v).$$

and for all $v \in V \setminus L$:

$$z_v \geq \sum_{v \in A_f} x_f = \sum_{\substack{f \in F: \\ d(f,r) \leq r_v}} x_f = \mathsf{cov}(v).$$

Hence for any $1 \leq i \leq c$, we have

$$\begin{aligned}
\sum_{v \in L} \lambda_i(v) y_v + \sum_{v \in V \setminus L} \lambda_i(v) z_v &\geq \sum_{v \in V} \lambda_i(v) \mathsf{cov}(v) \\
&= \sum_{v \in V} |D(v) \cap C_i| \mathsf{cov}(v) \\
&\geq \sum_{v \in V} \sum_{u \in D(v) \cap C_i} \mathsf{cov}(u) \\
&= \sum_{v \in C_i} \mathsf{cov}(v) \geq m_i.
\end{aligned}$$

The proof is completed by noting that

$$\sum_{v \in L} y_v \leq \sum_{f \in F} x_f \leq k.$$

□

Lemma 13. In any extreme point feasible solution to a linear program, the number of linearly independent tight constraints is equal to the number of variables.

Lemma 14. For an extreme point solution (y^*, z^*) to WCkPP LP, there are at most $2c$ strictly fractional y_v^*.

Proof. In the WCkPP LP, except for the constraints 2 and 4, there are at least $|V| - c$ independent tight constraints.

Denote by $\mathcal{T}$ the set of trees in the forest. For each tree $t \in \mathcal{T}$, let

$$\gamma_t = \text{size}(t) - \text{linearly independent tight constraints in } t$$

where size(t) denotes the number of vertices in t. Note that $\gamma_t \geq 0$ and $\sum_{t\in\mathcal{T}} \gamma_t \leq |V| - (|V| - c) = c$. Denote by τ_t the number of strictly fractional leaves in t. We call a vertex $v \in V \setminus L$ *tight* if constraints 3 and 6 for v is tight simultaneously. We pay special attention to tight vertices since we can begin with $\gamma_t = \tau_t$ and γ_t can only be decreased by 1 due to the existance of a tight vertex in t.

Recall that L denotes the leaves in the forest and T_v denotes the subtree rooted at v.

Claim 15. If u, v are both tight and $u \in T_v$, then v does not provide an additional linearly independent tight constraint.

This is because both u, v are tight implies $x_w = 0$ for all $w \in (T_v \setminus T_u) \cap L$. The constraint 3 for v is a linear combination of constraints 3 and 6 for u and constraint 5 for all $w \in (T_v \setminus T_u) \cap L$. Since all of them are tight, v does not provide an additional linearly independent tight constraint.

Claim 16. If v is tight, then there are 0 or at least 2 fractional leaves in T_v.

Combining the above two claims, we conclude that if we start with $\gamma_t = \tau_t$, it can only be decreased by 1 at a tight vertex v if $\nexists u \in T_v$ such that u is tight and there are at least 2 fractional leaves in T_v. Hence we have $\gamma_t \geq \frac{\tau_t}{2}$ for every $t \in \mathcal{T}$ and therefore

$$\sum_{t\in\mathcal{T}} \tau_t \leq 2 \sum_{t\in\mathcal{T}} \gamma_t \leq 2c.$$

□

Theorem 17. There is a 17-approximation algorithm for PCkS using at most $k + 2c - 1$ centers.

Proof. For each $v \in S$, the output of Algorithm 5, we open any facility $f \in F$ such that $d(f, v) \leq r_v$ and covers all clusters on the path from v to the root. By Lemma 12, it satisfies the coverage requirements m_i. Also we are using at most $k+2c-1$ centers by Lemma 14. For the approximation guarantee, see the proof of Theorem 11. □

In the following of this section, we focus on a special case of PCkS problem. We assume that the vertices with the same color have a same radius. That is, the partition $\{C_1, C_2, \ldots, C_c\}$ is

$$C_1 = \{v \in C : r_v = r_1\}, C_2 = \{v \in C : r_v = r_2\}, \ldots, C_c = \{v \in C : r_v = r_c\}.$$

for some $r_1, r_2, \ldots, r_c$. We refer to this special case as Uniform Priority Colorful k-Supplier (UPCkS) problem.

We provide 2 improved algorithms for UPCkS when there are only 2 kinds of colors. The first one is a modification of the algorithm for PCkS and the second one follows the framework in [AKZ22].

Theorem 18. There is a $(2+\sqrt{5})$-approximation algorithm using at most $k+1$ centers for UPCkS where there are only 2 colors.

Proof. Let r_1 be the radius of vertices in C_1 and r_2 be that for C_2. Assume $r_1 < r_2$. If $r_2 \leq \frac{1+\sqrt{5}}{2} r_1$, we can use the algorithm introduced in [BIPV19] for non-priority case, which yeilds $1 + 2 \cdot \frac{1+\sqrt{5}}{2} = 2 + \sqrt{5}$ approximation.

If $r_2 > \frac{1+\sqrt{5}}{2} r_1$, we use a similar algorithm to Algorithm 4. We run Algorithm 1 in C_1 and Algorithm 3 in C_2 with additional parameter $2r_1$. Build the contact forest such that for all $u \in R_2$ and $v \in R_1$,

$$(u, v) \in E \iff \exists f \in F : d(f, u) \leq r_u \text{ and } d(f, v) \leq r_v.$$

Note that it is a forest since if there exist $u, v \in R_2$ and $w \in R_1$ such that $(u, w) \in E, (v, w) \in E$, then we have

$$d(u, v) \leq d(u, w) + d(w, v) = 2 \cdot (r_1 + r_2) = r_2 + r_2 + 2r_1.$$

which implies u and v should be in the same cluster.

For each tree of size at least 3 in the forest, let v be the root and L be the set of leaves. Since $\lambda_2(v) = 0$ for all $v \in L$, we can split the tree into a tree of size 2 formed by v and $\arg\max_{u \in L} \lambda_1(u)$, and $|L| - 1$ singletons. Thus we only need to round up at least 2 fractional leaves and we can use at most $k+1$ centers to satisfy the coverage requirement. The proof for the approximation ratio is analogous to that of Theorem 6. □

The following algorithm is based on [AKZ22].

Definition 10 (single color (L, r)-partition)**.** Let $(X = C \cup F, d)$ be a single color metric space. A partition $\mathcal{P} \subseteq 2^C$ is an (L, r)-partition if

- $\text{diam}(A) := \max_{u,v \in A} d(u, v) \leq L \cdot r \quad \forall A \in P$
- For any $Z \subseteq F$, there exists a subfamily $\mathcal{A} \subseteq \mathcal{P}$ and injection $h : \mathcal{A} \to Z$ such that
  - $d(A, h(A)) \leq r \quad \forall A \in \mathcal{A}$
  - $|\bigcup_{A \in \mathcal{A}} A| \geq |\{v \in C : d(v, Z) \leq r\}|$

Theorem 19. There is a 7-approximation algorithm using at most k centers for UPCkS where there are only 2 colors.

Proof. Let r_1 be the radius of vertices in C_1 and r_2 be that for C_2. According to [AKZ22], we can find a $(6, r_1)$-partition $\mathcal{P}_1$ for C_1 and a $(6, r_2)$-partition $\mathcal{P}_2$ for C_2.

Consider a bipartite graph $G = (V = \mathcal{P}_1 \cup \mathcal{P}_2, E)$. For any $A_1 \in \mathcal{P}_1$ and $A_2 \in \mathcal{P}_2$, $(A_1, A_2) \in E$ iff there exists $f \in F$ such that $d(A_1, f) \leq r_1$ and $d(A_2, f) \leq r_2$. We aim to find a matching M (a vertex can be matched to empty) in G such that $|M| \leq k$, $\sum_{A_1 \in \mathcal{P}_1, A_1 \in M} |A_1| \geq m_1$ and $\sum_{A_2 \in \mathcal{P}_2, A_2 \in M} |A_2| \geq m_2$. Recall that we are working on the decision version of UPCkS, we make the following claim:

Claim 20. Such a matching M exists if an optimal solution exists.

Suppose the optimal solution opens facilities $Z \subseteq F$. According to the property of (L, r)-partition, there exists a subfamily $\mathcal{A}_1 \subseteq \mathcal{P}_1$ and injection $h_1 : \mathcal{A}_1 \to Z$, as well as a subfamily $\mathcal{A}_2 \subseteq \mathcal{P}_2$ and injection $h_2 : \mathcal{A}_2 \to Z$. It is easy to see that the following matching is feasible (match to empty if $h_1^{-1}(f)$ or $h_2^{-1}(f)$ doesn't exist)

$$M = \bigcup_{f \in Z} (h_1^{-1}(f), h_2^{-1}(f)).$$

Finding a feasible matching can be reduced the exact weight perfect matching problem. By creating $|\mathcal{P}_2| - k, |\mathcal{P}_1| - k$ dummy vertices on two sides of the bipartite graph respectively and add edges with ∞ weights, we can get rid of the constraint $|M| \leq k$. Since m_1, m_2 and all $|A_1|, |A_2|$ are polynomial in n, we can encode $|A_1|, |A_2|$ into edge weights. By enumerating $\sum_{A_1 \in \mathcal{P}_1, A_1 \in M} |A_1|$ and $\sum_{A_2 \in \mathcal{P}_2, A_2 \in M} |A_2|$, we can then determine an exact weight perfect matching, which can be solved in random polynomial time [Maa22].

For every $e = (A_1, A_2) \in M$, open the facility indicated by e, i.e. some $f \in F$ such that $d(A_1, f) \leq r_1$ and $d(A_2, f) \leq r_2$. Consider any $v \in A_1$, we have

$$d(v, f) \leq \operatorname{diam}(A) + d(A, f) \leq r_1 + 6r_1 = 7r_1.$$

and similarly for any $v \in A_2$. Therefore, opening facilities indicated by every $e \in M$ forms a feasible solution within 7 times the radii. □

6 Conclusions

We improved the approximation for PkSO from 9 to $1 + 3\sqrt{3}$ via the natural LP. However, the known lower bound on the integrality gap is 3. Closing the gap is an interesting open problem. [BCCN22] obtained a 9-approximation even under a matroid constraint on the chosen facilities. Is there a better approximation or lower bound for this more general problem?

We considered PCkS and obtained a bi-criteria approximation that yields a 17 approximation in the cost while violating the number of centers by an additive bound of $2c - 1$ (c is the number of colors). Can the approximation bound of 17 be improved? Is there an $O(1)$ approximation for PCkS that does not violate the number of centers when c is a fixed constant? For the simpler colorful k-center problem there is a 3-approximation that does not violate the number of centers when c is fixed constant [JSS22, AAKZ22]; however the running time of these algorithms is exponential in c and this is indeed necessary under the exponential time hypothesis [AAKZ22].

References

[AAKZ22] Georg Anegg, Haris Angelidakis, Adam Kurpisz, and Rico Zenklusen. A technique for obtaining true approximations for k-center with covering constraints. *Mathematical Programming*, pages 1–25, 2022.

[AKZ22] Georg Anegg, Laura Vargas Koch, and Rico Zenklusen. Techniques for generalized colorful k-center problems. *arXiv preprint arXiv:2207.02609*, 2022.

[BCCN22] Tanvi Bajpai, Deeparnab Chakrabarty, Chandra Chekuri, and Maryam Negahbani. Revisiting priority k-center: Fairness and outliers, 2022.

[BIPV19] Sayan Bandyapadhyay, Tanmay Inamdar, Shreyas Pai, and Kasturi Varadarajan. A constant approximation for colorful k-center. *arXiv preprint arXiv:1907.08906*, 2019.

[CGK20] Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. *ACM Transactions on Algorithms (TALG)*, 16(4):1–19, 2020.

[CKMN01] Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, volume 1, pages 642–651. Citeseer, 2001.

[CLLW16] Danny Z Chen, Jian Li, Hongyu Liang, and Haitao Wang. Matroid and knapsack center problems. *Algorithmica*, 75:27–52, 2016.

[CN19] Deeparnab Chakrabarty and Maryam Negahbani. Generalized center problems with outliers. *ACM Trans. Algorithms*, 15(3), jul 2019.

[Gon85] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[HS86] Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, 33(3):533–550, 1986.

[JSS22] Xinrui Jia, Kshiteej Sheth, and Ola Svensson. Fair colorful k-center clustering. *Mathematical Programming*, 192(1-2):339–360, 2022.

[Maa22] Nicolas El Maalouly. Exact matching: Algorithms and related problems. *arXiv preprint arXiv:2203.13899*, 2022.

[Ple87] Ján Plesník. A heuristic for the p-center problems in graphs. *Discrete Applied Mathematics*, 17(3):263–268, 1987.

[S+03] Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.