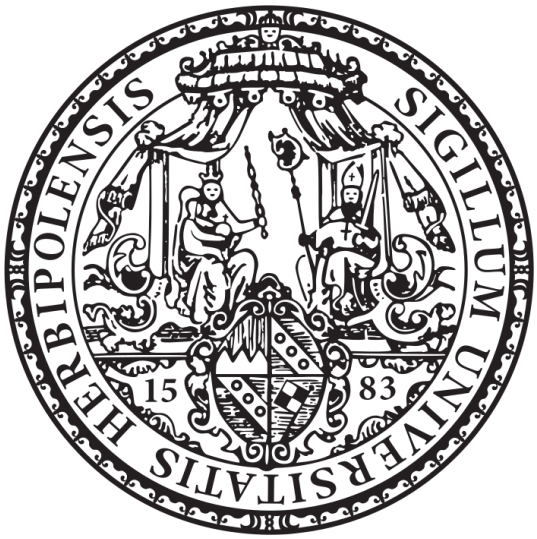
Bachelor Thesis

Approximation Algorithms for Variants of the *k*-Median Problem

Benedikt Riegel

Date of Submission:	9. December 2020
Advisors:	Priv.-Doz. Dr. habil. Joachim Spoerhase
	Dr. Kamyar Khodamoradi.



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

In this work we examine two general variants of the k-Median. The exact uncapacitated k-facility location problem finds precisely k centres in a metric space, while minimizing not only the service cost of a point to its closest centre, but also the opening cost of the chosen centres. For this problem, we provide a proof, based on a proof by Zhang [Zha07] that a local search algorithm with only a single-swap operation gives an approximation of 7. Furthermore, we provide another algorithm with a 3.25-approximation in expectation, by modifying an algorithm by Charikar and Li [CL12].

The reconciliation k-Median is another general variant of the k-Median. It also finds precisely k centres in a metric space, which not only minimize the service cost of a point to its closest centre, but also minimize the distances between themselves. To approximate this problem, one could combine one of our approximation algorithms for exact uncapacitated k-facility location with a reduction algorithm by Spoerhase and Khodamoradi [SK], to receive a 6.5-approximation in expectation or a 14-approximation. Finally, we prove that the reconciliation k-Median problem cannot be approximated with a multi-swap local search algorithm.

Zusammenfassung

In dieser Arbeit betrachten wir zwei allgemeiner gefasste Varianten des k-Median. Das exact uncapacitated k-facility location Problem sucht nach genau k vielen Zentren in einem metrischen Raum, die nicht nur die Abstände zwischen den Punkten in dem Raum und ihrem nächstgelegenen Zentrum minimieren, sondern auch die Kosten minimieren, die bei der Wahl eines Zentrums entstehen. Basierend auf dem Beweis von Zhang [Zha07] beweisen wir, dass ein local search Algorithmus mit nur einer single-swap Operation eine Approximation von 7 liefert. Des Weiteren, modifizieren wir den Algorithmus von Charikar und Li [CL12], sodass er seine mittlere Approximationsgüte von 3.25 weiterhin behält, aber exakt k viele Zentren zurückgibt.

Der reconciliation k-Median ist ebenfalls eine Variante des k-Median. Auch dieses Problem sucht nach genau k vielen Zentren in einem metrischen Raum, die die Abstände zwischen den Punkten in dem Raum und ihrem nächstgelegenen Zentrum minimieren, mit dem Zusatz, dass die Abstände zwischen den Zentren selbst auch minimiert werden. Um dieses Problem zu approximieren, kann man nun einen unserer exact uncapacitated k-facility location Algorithmen mit dem von Spoerhase und Khodamoradi entwickelten

Algorithmus [SK] kombinieren. Diese Kombination ergibt entweder eine erwartete Approximation von 6.5 oder eine 14-Approximation. Abschließend beweisen wir, dass der reconciliation k-Median nicht mit einem multi-swap local search Algorithmus approximiert werden kann.

Contents

1 Introduction

Clustering is an important method when analysing datasets. It is used in the fields of data mining [Ber06], pattern recognition [AIK18], unsupervised learning [CBJD18] and others. Clustering algorithms create clusters, with more similar data points grouped together into the same cluster.

One well studied clustering problem is the k-Median, where given a set F of facilities and a set C of clients in a metric space, and the distances between them (with a distance function d), it opens at most k facilities, minimizing the sum of the distances for each client to its nearest open facility. The distance between a client and its nearest facility (serving facility) is also called the service cost. Note that it is a classical clustering problem, if the set of facilities equals the set of clients.

A variant of the k-Median is the reconciliation k-Median (Rec.-k-Median), minimizing not only the sum of the service costs of each client, but also minimizing the distances between the open facilities themselves (reconciliation cost). This allows for more complex problems to be solved or analysed, e.g. the election of a k-member committee. The goal in electing a k-member committee, e.g. in politics is to represent every individual, i.e. citizen, as close as possible. But a committee, where the members represent different extremes will most likely need a lot of time to make compromises, therefore, slowing down the decision-making processes, e.g. the enacting of laws. Therefore, it would be best for the committee members to not only represent the individuals as close as possible, but also for themselves to be as close as possible. This can of course be modelled by a Rec.-k-Median instance, with the facilities and the clients both being the set of the individual people, and the distances between them being their agreeableness on different topics.

Since the Rec.-k-Median is NP-hard [SK], it can only be approximated in polynomial time. For this, the reduction to the exact uncapacitated k-facility location (Exact-k-UFL) problem, proves to be viable. The Exact-k-UFL problem stems from the uncapacitated k-facility location (k-UFL) problem, which is a generalization of the uncapacitated facility location (UFL) problem and k-Median. An UFL instance has in addition to the two sets F and C and their distances, an opening cost for each facility. UFL then opens facilities that minimize again the sum of the service costs, as well as the sum of opening costs of each open facility. The k-UFL problem minimizes the same sums, with the extra restriction that at most k facilities can be opened. The Exact-k-UFL problem also minimizes the same sums, but opens precisely k facilities.

1.1 Related Work

Charikar and Guha managed to approximate the k-Median problem for the first time, giving a $6\frac{2}{3}$-approximation [CGvTS02]. Currently the best algorithm is a $(2.611+\epsilon)$ approximation [BPR$^+$] and Jain et al. [JMS02] give a lower bound on the approximability of $1+2/e$ for the k-Median, unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$). A new approach is to combine parameterized complexity with approximation problems, allowing them to run in $f(k)n^{O(1)}$ instead of the classical polynomial time, where n is the size of the input and $f(k)$ is an arbitrary function, depending on one or more parameters of the problem. Parameterized complexity has the goal to let an algorithm run in exponential time in only a few specified parameters. With this approach Cohen-Addad et al. [CGK$^+$19] approximate it with a factor of $(1+2/e+\epsilon)$ (in $f(k,\epsilon)n^{O(1)}$ time), but Byrka et al. [BDM$^+$20] improved the factor even further to $(1+\epsilon)$ (in $f(|F|-k,\epsilon)n^{O(1)}$ time), by closing $|F|-k$ many facilities, instead of opening k many.

The UFL problem has very good approximations, with the best being a 1.488-approximation by Li [Li13], which is really close to the lower approximation bound of 1.463 given by Guha and Khuller, unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).

The k-UFL problem was first approximated by Jain and Vazirani [JV01] with a factor of 6 and later improved to a factor of $2+\sqrt{3}+\epsilon$ by Zhang [Zha07].

Since we could not find anything on the Exact-k-UFL problem, it could be possible that this work is the first to examine the Exact-k-UFL problem.

The first and up until now only work on the Rec.-k-Median was by Ordozgoiti and Gionis [OG19]. They introduced a local search algorithm for the problem and prove that if each facility in the solution provided by the algorithm serves at least $\lceil 2\lambda \rceil k$ many clients, then the solution is only a factor of 114λ away from the optimum.

1.2 Our Results

Firstly, we examine a Rec.-k-Median 2α-approximation algorithm developed by Spoerhase and Khodamoradi [SK], which reduces Rec.-k-Median to an Exact-k-UFL, where α is the approximation factor of the algorithm used, to solve the Exact-k-UFL instances. Since Spoerhase and Khodamoradi do not provide an algorithm for Exact-k-UFL, we contribute an Exact-k-UFL algorithm that has a 3.25-approximation in expectation, by modifying the k-UFL algorithm of Charikar and Li [CL12] that also has a 3.25-approximation in expectation, to return exactly k facilities.

Secondly, we prove that a single-swap local search algorithm for Exact-k-UFL has an approximation guarantee of 7, by following the proof provided by P. Zhang [Zha07] for a k-UFL local search algorithm.

Finally, we investigate a p-swap local search algorithm for Rec.-k-Median, allowing at most p facilities to be swapped at once. We prove, that the cost of a local optimum of this local search algorithm has no upper bound, therefore disproving Theorem 2 in [OG19], where the authors claim that a local optimum S is bounded by $(\lambda k+5)$ multiplied by the costs of the global optimum.

2 Definitions

In the following we define the problems examined in this thesis, as well as some notations. For every problem, we consider a metric space $(F \cup C, d)$, where the set of facilities F and the set of clients C are not necessarily disjunct and $d : (F \cup C)^2 \to \mathbb{R}$ is a distance function. Furthermore, $\phi_S(j)$ denotes the facility $i \in S$, closest to the client $j \in C$. Using the notations from above, we can define Rec.-k-Median, k-UFL and Exact-k-UFL as follows.

Definition 1 (The Reconciliation k-Median)**.** *Let (F, C, d, k, λ) be a Rec.-k-Median instance, with $(F \cup C, d)$ being a metric space as described above, $k \in \mathbb{N}$ and $\lambda \in \mathbb{R}^+$. Find an optimal set O, minimizing the following cost function:*

$$\text{cost}(O) = \min_{\substack{S \subseteq F \\ |S|=k}} \left\{ \sum_{j \in C} d(j, \phi_S(j)) + \frac{1}{2}\lambda \sum_{i,i' \in S} d(i, i') \right\}$$

Definition 2 (Uncapacitated k-Facility Location)**.** *Let (F, C, d, k, f) be a k-UFL instance, with $(F \cup C, d)$ being a metric space as described above, $k \in \mathbb{N}$ and f assigning the opening costs f_i to every facility $i \in F$. Find an optimal set O, minimizing the following cost function:*

$$\text{cost}(O) = \min_{\substack{S \subseteq F \\ |S| \leq k}} \left\{ \sum_{j \in C} d(j, \phi_S(j)) + \sum_{i \in S} f_i \right\}$$

Definition 3 (Exact Uncapacitated k-Facility Location)**.** *Let (F, C, d, k, f) be an Exact-k-UFL instance, with $(F \cup C, d)$ being a metric space as described above, $k \in \mathbb{N}$ and f assigning the opening costs f_i to every facility $i \in F$. Find an optimal set O, minimizing the following cost function:*

$$\text{cost}(O) = \min_{\substack{S \subseteq F \\ |S|=k}} \left\{ \sum_{j \in C} d(j, \phi_S(j)) + \sum_{i \in S} f_i \right\}$$

Additionally, we define the reconciliation cost $\text{cost}_{\text{RC}}(S)$, the facility cost $\text{cost}_f(S)$ and the service cost $\text{cost}_s(S)$ to be:

$$\begin{aligned} \text{cost}_s(S) &= \sum_{j \in C} d(j, \phi_S(j)) \\ \text{cost}_{\text{RC}}(S) &= \frac{1}{2}\lambda \sum_{i,i' \in S} d(i, i') \\ \text{cost}_f(S) &= \sum_{i \in S} f_i \end{aligned}$$

Also we denote:

$$[t] = \{1, 2, 3, \ldots, t\}$$

To increase readability, we omit the notations necessary for the local search analysis from this Chapter 2, but instead introduce them at the beginning of Chapter 4.

3 Reducing Rec.-k-Median to Exact-k-UFL

Since Rec.-k-Median is a NP-hard [OG19] minimization problem, an approximation algorithm is needed, to compute a solution in polynomial time. In order to approximate this problem Spoerhase and Khodamoradi introduced a novel algorithm. The algorithm transforms for every facility $m \in F$ the Rec.-k-Median instance (F, C, d, k, λ) into an Exact-k-UFL instance (F, C, d, k, f_m), with $f_m(i) = \lambda(k-1)d(i, m)$. Afterwards, it solves the $|F|$ different Exact-k-UFL instances and returns the best solution, as the solution for the Rec.-k-Median instance. This gives a 2α-approximation, with α being the approximation factor of the Exact-k-UFL algorithm. Even though an Exact-k-UFL instance is the same as a k-UFL instance, the need for precisely k facilities to be in the solution, does not allow us to simply use existing k-UFL algorithms.

3.1 The k-UFL Algorithm by Charikar and Li

First, we have to understand the algorithm by Charikar and Li [CL12], before modifying it to return exactly k facilities. It gives a 3.25-approximation for k-UFL in expectation and is subdivided into the following steps:

1. **Linear programme(LP):** First, the following LP for k-UFL is solved. A facility i is open, if it is in the solution, therefore y_i represents how much i is opened. $x_{i,j}$ denotes, how much client j is served by i.

$$\begin{aligned}
\min \quad & \sum_{i \in F, j \in C} d(i,j)x_{j,i} + \sum_{i \in F} f_i y_i \\
\text{s.t.} \quad & \sum_{i \in F} x_{i,j} = 1, \quad \forall j \in C, \\
& \sum_{i \in F} y_i = k, \\
& x_{i,j} \leq y_i, \quad \forall i \in F, j \in C, \\
& x_{i,j}, y_i \in [0,1], \forall i \in F, j \in C
\end{aligned}$$

2. **Splitting phase:** In this phase we split every facility $i \in F$, if needed, such that $x_{i,j} = y_i$ for every client $j \in C$. If $i \in F$ got split, it is called a split facility and the $t \in \mathbb{N}$ many facilities $\{i.1, i.2, \ldots, i.t\}$ that were created through this split, are called fractional facilities. After the split, $y_i = y_{i.1} + y_{i.2} + \cdots + y_{i.t}$, which means that the sum of all y's still equals k. From now on we call the original set of facilities F_{old} and the new set of facilities F_{new}.

3. **Filtering phase:** In this phase a subset $C' \subseteq C$ is determined, where $j, j' \in C'$ are far enough apart from each other.

4. **Bundling phase:** Here ever $j \in C'$ is assigned a set of facilities $U_j \subseteq F_{\text{new}}$, where $i \in U_j$ is close enough to j and i serves j. This leads to $\frac{1}{2} \leq \text{vol}(U_j) \leq 1$, with $\text{vol}(U_j) = \sum_{i \in U_j} y_i$. In this procedure facilities can stay unbundled.

5. **Matching phase:** Now the closest $j, j' \in C'$ are matched together and added to the set M. Here one $b \in C'$ can stay unmatched.

6. **Sampling phase:** Finally, some facilities are opened, depending on a procedure described next.

The description of the steps above are limited to the necessary information, to understand the sampling phase and the modifications it undergoes, to first open less or equal to k facilities and then in Section 3.2 to open exactly k facilities. However, we first have to thoroughly define what "opening" and "closing" a facility, bundle or matched pair means:

1. **Opening/Closing a facility i:** Opening a facility i means to add the facility i to the solution. Closing a facility i means to not add it to the solution.

2. **Opening/Closing a bundle j:** To open a bundle j means to randomly open exactly one facility $i \in U_j$, with the probability $y_i/\text{vol}(U_j)$. Closing a bundle j implies that every facility in U_j is closed.

3. **Opening/Closing a matched pair (j, j'):** Opening a pair (j, j') signifies to open both bundles j and j'. Closing a pair (j, j') is equivalent to open either bundle j or j'. When the matched pair is closed, j is opened with probability $1 - \text{vol}(U_{j'})$ and j' is opened with probability $1 - \text{vol}(U_j)$.

If a facility, bundle or pair is not opened, then it is closed.

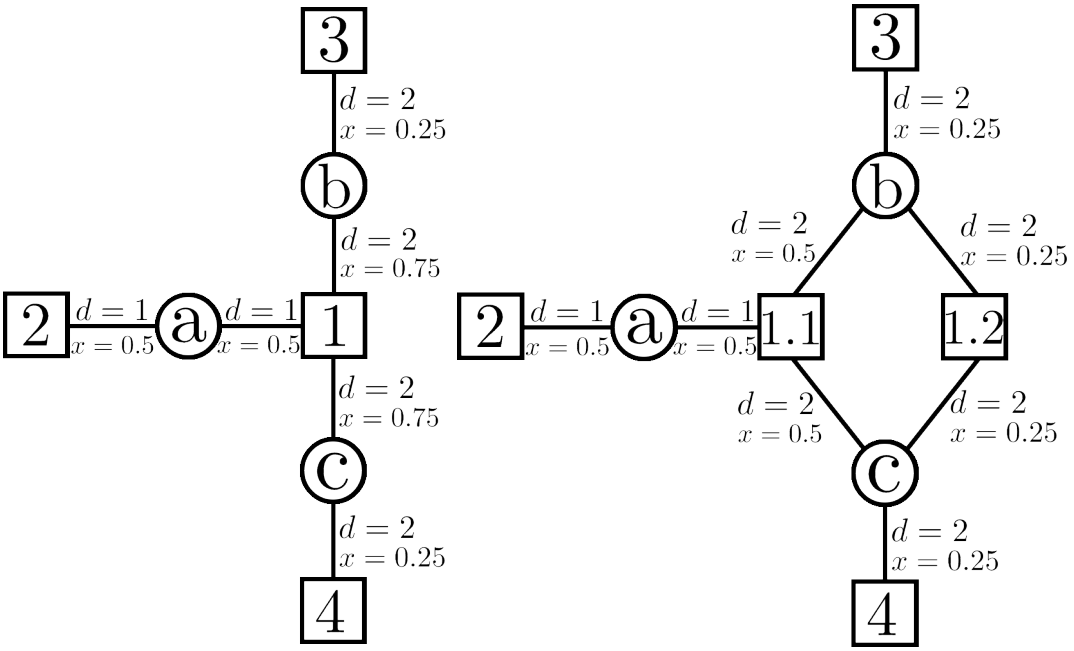
In the original sampling phase every $(j, j') \in M$ would be opened randomly with probability $\text{vol}(U_j) + \text{vol}(U_{j'}) - 1$, the unmatched bundle b would be opened randomly with probability $\text{vol}(U_b)$ and every unbundled facility i with probability y_i. Notice that this results in:

$$\forall j \in C' : \Pr(\text{"j is open"}) = \text{vol}(U_j)$$
$$\forall i \in F_{\text{new}} : \Pr(\text{"i is open"}) = y_i$$

This results in k facilities from F_{new} to be opened in expectation.

The authors now modify this procedure, to return exactly k facilities from F_{new}. In their procedure, two fractional facilities from the same split facility can be picked, resulting in $\leq k$ facilities from F_{old} to be picked. Let $F_{\text{unbundled}}$, $F_{\text{unmatched}}$ and A be defined as follows:

$$F_{\text{unbundled}} = \{i : i \text{ is an unbundled facility}\}$$
$$F_{\text{unmatched}} = \{b : U_b \text{ is an unmatched bundle}\}$$
$$A = F_{\text{unbundled}} \cup F_{\text{unmatched}} \cup M$$



(a) LP solution before the split process

(b) LP solution after the split process

Fig. 3.1: An example with $F = \{1, 2, 3, 4\}$, $C = \{a, b, c\}$, $k = 3$ and the opening cost f for each facility is 1. The distances that are not drawn, are the minimal paths through this graph, e.g. $d(b, a) = 3$, hence the triangle inequality still holds. The x's represent how connected a client is to a facility in the LP solution. The y of a facility in the LP solution is the maximum x adjacent to the facility, e.g. $y_1 = 0.75$.

First the authors define a random variable X_a and a variable x_a for every $a \in A$.

$$X_a = \begin{cases} 1, & \text{if } a \text{ is open} \\ 0, & \text{if } a \text{ is not open} \end{cases}$$

$$x_a = \begin{cases} y_a, & \text{if } a \text{ is an unbundled facility} \\ \text{vol}(U_a), & \text{if } a \text{ is an unmatched bundle} \\ \text{vol}(U_j) + \text{vol}(U_{j'}) - 1, & \text{if } a = (j, j') \in M \end{cases}$$

This results in $\sum_{a \in A} x_a = k - |M|$ to be true and if $\sum_{a \in A} X_a = k - |M|$, then there are exactly k open facilities of F_{new}. It would only open precisely k facilities of F_{old}, if for each split facility, their fractional facilities are in the same bundle, since this would result in only one fractional facility of a split facility to be opened at most. Figure 3.1 shows an example, where this is not the case, since $C' = \{a\}$, $U_a = \{1.1, 2\}$ and the set of unbundled facilities is $\{1.2, 3, 4\}$ and therefore proving that it can open less than k facilities of F_{old}. To achieve the desired sum $\sum_{a \in A} X_a = k - |M|$, the tree-based dependent rounding procedure of [Sri01] is used on the variables, since it has the following properties:

(a) $\Pr[X_a = 1] = x_a$.

(b) $\Pr[|\{a : X_a = 1\}| = k - |M|] = 1$

(c) the following negative correlations holds for all $S \subseteq A$:

$$\Pr\left[\bigwedge_{a \in S} X_a = 0\right] \leq \prod_{a \in S} \Pr[X_a = 0]$$

$$\Pr\left[\bigwedge_{a \in S} X_a = 1\right] \leq \prod_{a \in S} \Pr[X_a = 1]$$

After use, all X_a's are rounded to either 0 or 1, with exactly $k - |M|$ X_a's equal to 1 and now, all $a \in A$ with $X_a = 1$ are opened. Finally, every opened facility is returned as the solution S for the k-UFL instance, resulting in

$$\text{cost}_s(S) \leq 3.25\text{cost}_s(S)$$

and the expected facility cost $\text{cost}_f(S)$ is the facility cost of the LP solution, hence the algorithm approximates k-UFL with a factor of 3.25 in expectation.

3.2 Exact-*k*-UFL Modification

A solution of Exact-k-UFL has to have a cardinality of precisely k, so we have to modify the algorithm described in Chapter 3. Instead of using the tree-based dependent rounding procedure of Srinivasan [Sri01], we use the dependent rounding in bipartite graph presented by Gandhi et al. [GKPS02].

The algorithm in [GKPS02] receives a bipartite graph (B, C, E) and a list of values $x_{v,u}$ for each edge $(v, u) \in E$ as input. It rounds every $x_{v,u}$ to a random variable $X_{v,u} \in \{0, 1\}$ and returns the resulting set of all $X_{v,u}$.

$$d_v = \sum_{(v,u) \in E} x_{v,u} \text{ and } D_v = \sum_{(v,u) \in E} X_{v,u},$$

where d_v denotes the fractional cost of the vertex v and D_v denotes the integral degree. The algorithm provides the following properties for the rounding procedure:

(a) $\Pr[X_{v,u} = 1] = x_{v,u}$.

(b) Degree-preservation. $\Pr[D_v \in \{\lfloor d_v \rfloor, \lceil d_v \rceil\}$. Notice, if $d_v \in \mathbb{N}$, then $D_v = d_v$

(c) the negative correlations hold for any $v \in B \cup C$ and for any $S \subseteq \{X_{(v,i)} : (v, i) \in E\}$:

$$\Pr\left[\bigwedge_{X \in S} X = 0\right] \leq \prod_{X \in S} \Pr[X = 0]$$

$$\Pr\left[\bigwedge_{X \in S} X = 1\right] \leq \prod_{X \in S} \Pr[X = 1]$$

Notice that the bipartite graph procedure yields the same result as the tree-based rounding procedure, if the input is a star graph. This becomes apparent for $B = \{r\}$ (r being the root of the star), $C = A$ (A from Section 3.1) and $x_{a,r} = x_a$ (x_a also from Section 3.1). And indeed, the authors of [GKPS02] even state the fact, that the tree-based rounding procedure [Sri01] is simply a star in the bipartite graph rounding procedure. To get exactly k facilities from the original set F_{old}, we define structures to construct a bipartite graph that gives us the desired properties, to open the facilities correctly and to still preserve the 3.25-approximation in expectation.

First, a root r has to be added to the graph. Every unbundle facility, unmatched bundle and matching is then connected to r with the following structures.

Unbundled facility: An unbundled facility i is simply attached to r, with $x_{r,i} = y_i$. Therefore it is opened with the probability y_i (see Figure 3.2c).

Unmatched bundle: The unmatched bundle b is attached to the root via a vertex μ, with $x_{r,\mu} = \text{vol}(U_b)$. Furthermore a vertex φ is connected to μ, with $x_{\mu,\varphi} = 1 - \text{vol}(U_b)$, resulting in $d_\mu \in \mathbb{N}$. Therefore

$$d_\mu = D_\mu = X_{r,\mu} + X_{\mu,\varphi} = 1$$

and the edge (μ, φ) can be seen as a negation to (r, μ), since either $X_{r,\mu}$ or $X_{\mu,\varphi}$ is 1, and the other is 0. Attached to φ are all the facilities $b_i \in U_b$, with $x_{\varphi,b_i} = y_{b_i}$. Thus

$$d_\varphi = D_\varphi = 1$$

and at most only one X_{φ,b_i} with $b_i \in U_b$ can be equal to 1. The bundle b is open, if $X_{r,\mu} = 1$ and a facility $b_i \in U_b$ is open, if $X_{\varphi,b_i} = 1$. Hence

$$\begin{aligned}
&\text{"bundle } b \text{ is opened"} \implies X_{r,\mu} = 1 \implies X_{\mu,\varphi} = 0 \\
&\implies X_{\varphi,b_i} = 1 \text{ for exactly one } b_i \in U_b \\
&\text{"bundle } b \text{ is closed"} \implies X_{r,\mu} = 0 \implies X_{\mu,\varphi} = 1 \implies X_{\varphi,b_i} = 0 \ \forall b_i \in U_b
\end{aligned}$$

and therefore, the structure complies with the rules given in Section 3.1 (see Figure 3.2a).

Matched pair: The matched pair (j, j') is attached to the root via a vertex γ and attached to γ is the bundle j via a vertex α and the bundle j' via a vertex β. The edges have the following weights

$$\begin{aligned}
x_{r,\gamma} &= \text{vol}(U_j) + \text{vol}(U_{j'}) - 1 \\
x_{\gamma,\alpha} &= 1 - \text{vol}(U_j) \\
x_{\gamma,\beta} &= 1 - \text{vol}(U_{j'})
\end{aligned}$$

where $x_{\gamma,\alpha}$ and $x_{\gamma,\beta}$ together represent the negation of $x_{r,\gamma}$. This is very similar to the unmatched bundle and just like with the unmatched bundle, every facility $j_i \in U_j$ is attached to α with $x_{\alpha,j_i} = y_{j_i}$ and every facility $j'_i \in U_{j'}$ is attached to β with $x_{\beta,j'_i} = y_{j'_i}$. Since $D_\gamma = 1$, only one of the edges (r, γ), (γ, α) and (γ, β) will have a $X = 1$.

$$\begin{aligned}
X_{\gamma,\alpha} = 0 &\iff \text{bundle } j \text{ is open} \\
X_{\gamma,\beta} = 0 &\iff \text{bundle } j' \text{ is open}
\end{aligned}$$

This results in:

The matched pair (j, j') is open $\implies X_{r,\gamma} = 1 \implies X_{\gamma,\alpha} = X_{\gamma,\beta} = 0$
$\implies$ both bundles j and j' are open
The matched pair (j, j') is closed $\implies X_{r,\gamma} = 0 \implies$ either $X_{\gamma,\alpha} = 0$ or $X_{\gamma,\beta} = 0$
$\implies$ either bundle j or j' is open

and therefore, the structure complies with the rules given in Section 3.1 (see Figure 3.2b)

Buffer vertices: To avoid having more than one edge between two vertices after introducing the split facilities, two buffer vertices ω and ψ are added for each fractional facility i. ω is attached to i and ψ is attached to ω, with

$$x_{i,\omega} = 1 - y_i$$
$$x_{\omega,\psi} = y_i$$

Notice that now the edge (ω, ψ) also represents, if facility i is open or not.

Split facilities: If $i \in F_{old}$ got split into $\{i.1, i.2, \ldots, i.t\}$, then we fuse the vertices of the fractional facilities $\psi_{i.1}, \psi_{i.2}, \ldots, \psi_{i.t}$ together to one super vertex i. This results in

$$d_i = \sum_{h\in[t]} d_{\psi_{i.h}} = \sum_{h\in[t]} y_{i.h} = y_i \leq 1$$

This ensures that at most only one fractional facility is opened, for each split facility. (see Figure 3.2d)
All the edges at the root sum up to $k - |M|$, resulting in exactly k facilities being opened and the different structures ensure the opening rules given by the algorithm in Section 3.1 to be followed. Since the vertices of the split facilities ensure that at most one of their fractional facilities is open, the output is exactly k open facilities of the original facility set F_{old}. Figure 3.3 shows the bipartite graph of the example in Figure 3.1. It is now easy to see that the edges leading to a facility vertex represent the facilities in F_{new} and the facility vertices themselves symbolize the facilities in F_{old}. In order to show that this still computes a 3.25-approximation in expectation, we have to prove the same properties stated in Lemma 9 and 10 of [CL12].

Lemma 4. *Let $T \subseteq F_{new}$ or $T \subseteq F_{old}$. We have*

$$Pr\left[\bigwedge_{z\in T} \overline{z}\right] \leq \prod_{z\in T} Pr[\overline{z}],\ Pr\left[\bigwedge_{z\in T} z\right] \leq \prod_{z\in T} Pr[z]$$

where z denotes the event that facility z is open, and $\overline{z}$ is the negation of z.

Proof. We differentiate between the cases $T \subseteq F_{new}$ and $T \subseteq F_{old}$.

1. $T \subseteq F_{new}$: First, we add two buffer vertices ω and ψ, for every none-fractional facility i'. We also add an edge (i', ω), with $x_{i',\omega} = 1 - y_{i'}$ and an edge (ω, ψ), with $x_{\omega,\psi} = y_{i'}$. Then, we fuse all ψ vertices of each none fractional facility and all

the vertices i of each split facility into one super vertex. This loosens the negative correlation between fractional facilities, since more than one fractional facility from one split facility can be opened now. Now every edge, adjacent to the super vertex, represents a facility in F_{new}. Let z correspond to the edge e, then we have $X_e = z$, and with property (c) of the rounding procedure and the fact that every e is an edge of the super vertex, we obtain the negative correlation above. The following equation holds for the super vertex and all its adjacent edges e

$$\sum_e x_e = \sum_{i\in} y_e = k$$

Therefore, we still open k facilities, although for F_{new}, but as mentioned, the fractional facilities of one split facility normally have a stronger negative correlation between each other. Hence, the facilities also have this negative correlation in the normal graph.

2. $T \subseteq F_{old}$: In this case, we first have to add two buffer vertices ω and ψ, for every facility $i \in F_{old}$. We add an edge (i, ω), with $x_{i,\omega} = 1 - y_i$ and an edge (ω, ψ), with $x_{\omega,\psi} = y_i$. This does not affect the results of the rounding procedure. Next, we fuse every vertex ψ together into one super vertex. The edges of this super vertex represent the facilities in F_{old}. Let z correspond to the edge e of the super vertex, then we have $X_e = z$, and with property (c) of the rounding procedure, we obtain the negative correlation above. The following equation holds for the super vertex and all its adjacent edges e

$$\sum_e x_e = \sum_{i\in F_{old}} y_e = k$$

Therefore, the rounding procedure gives the same results for this modified graph, as for the original graph.

□

Lemma 5. *The approximation ratio of the algorithm, using the bipartite graph dependence rounding procedure, is at most that of the original algorithm.*

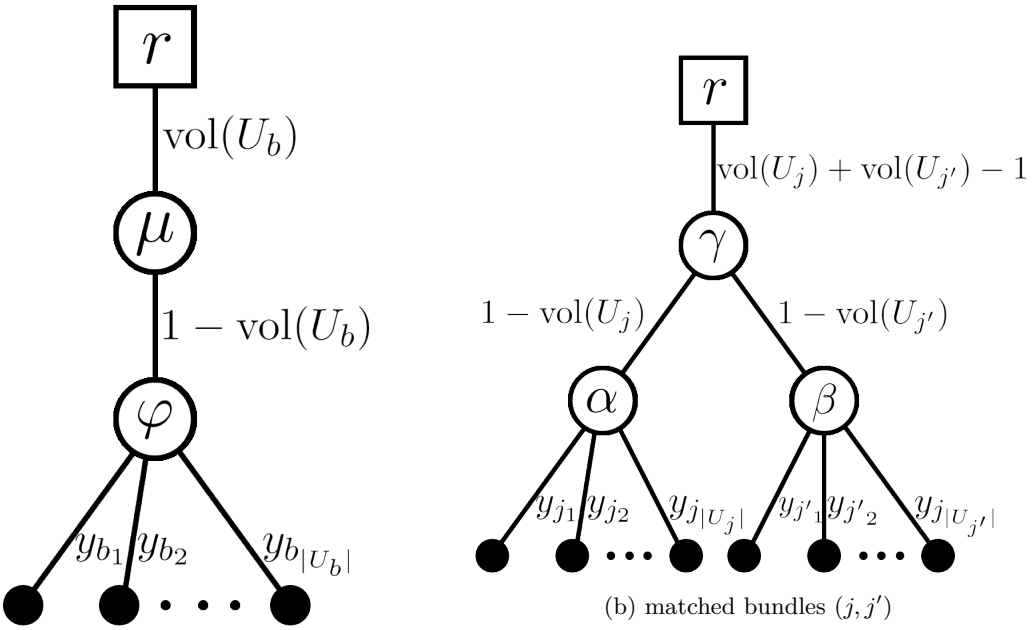
Proof. For a client j we order every facility in the ascending order of distances to j. Let $z_1, z_2, \ldots, z_m$ be this order. Because j gets served by the first open facility in this order, it suffice to show that for every $s \in [m]$, the probability of $\mathrm{P}_{\text{exact}}$ that the first s facilities are closed in the Exact-k-UFL algorithm, is at most the correspondent probability P_{old} of the original algorithm. For every $s \in [m]$ we can use Lemma 4 to show:

$$\mathrm{P}_{\text{exact}} = \Pr\left[\bigwedge_{h=1}^{s} z_h\right] \leq \prod_{h=1}^{s} \Pr[z_h] = \mathrm{P}_{\text{old}}$$

□

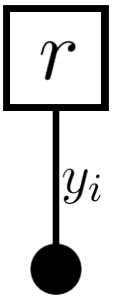
Theorem 6. *The algorithm, using the bipartite graph rounding procedure, has a* 3.25*-approximation in expectation.*

Proof. Lemma 4 and Lemma 5 prove the properties necessary, stated by Charikar and Li [CL12], for the algorithm to have a 3.25-approximation in expectation. □

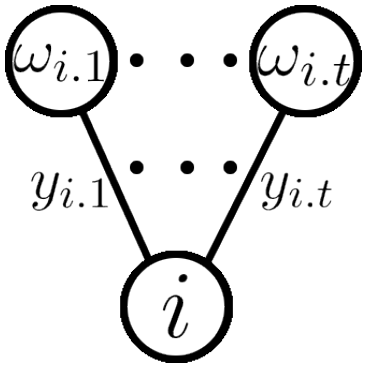


(a) unmatched bundle b

(b) matched bundles (j, j')



(c) unbundled facility i



(d) split facility i and it's fractional facilities $i.1, \ldots, i.t$. There are no ψ's, since they fused into i.

Fig. 3.2: The different structures for the bipartite graph.

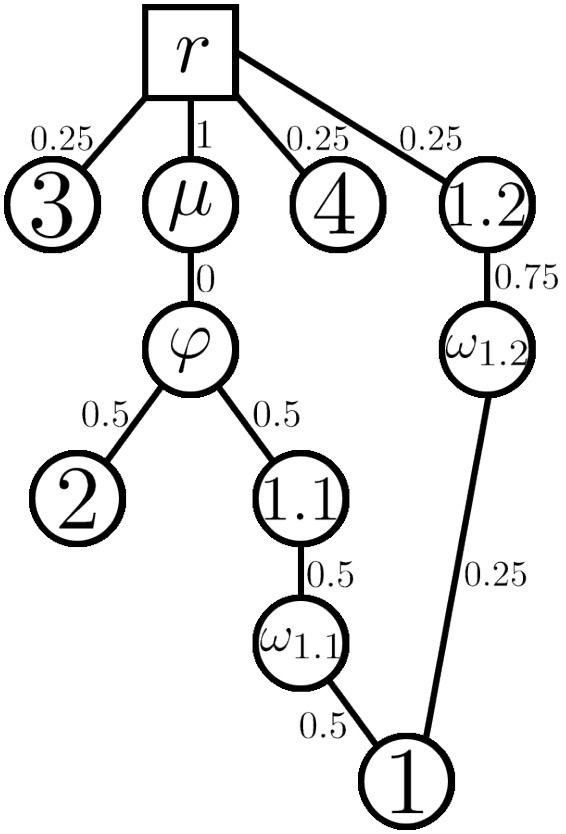


Fig. 3.3: Bipartite graph of the example in Figure 3.1

4 Reconiciliation *k*-Median Local Search

The reduction algorithm by Spoerhase and Khodamoradi allows us to use any Exact-k-UFL approximation algorithm. So, if we can get an Exact-k-UFL local search algorithm with a constant factor approximation, then this would raise the question, if we could also get a constant factor approximation for a Rec.-k-Median local search algorithm.

A local search algorithm, first, takes a random, but valid solution $S \subseteq F$. If there is a $S' \in N(S)$ with less cost than S, then replace S with S'. $N(S)$ is the set of S' that can be reached from S by performing only one predefined local operation. Since the feasible solutions for Rec.-k-Median and Exact-k-UFL always have the cardinality of k, we only need a local p-swap operation.

Definition 7 (swap(A, B))**.** *A swap(A, B) operation is only applicable, if $A \subseteq S \wedge B \subseteq F - S$ and $|A| = |B| \leq p$, where $p \in \mathbb{N}$ has to be predefined and stays fixed for the algorithm. If used on a solution S, each facility in A is removed from S and each facility in B is added to S.*

This results in $N(S) = \{S + A - B : A \subseteq S \wedge B \subseteq F - S \wedge |A| = |B| \leq p\}$. The local search algorithm for Exact-k-UFL and Rec.-k-Median only differ in how the cost of a solution is calculated, but a general version is depicted with Algorithm 1.

4.1 Exact-*k*-UFL Local Search

In this section we analyse, if local search for Exact-k-UFL gives a constant factor approximation, since this and the algorithm [SK] would lead to the question, if a local search algorithm for Rec.-k-Median can also give a constant factor approximation.

To prove a constant factor approximation, we follow the proof of Zhang [Zha07] and slightly modify it, and extend some Lemmas, to match the case of Exact-k-UFL. First, the following notations have to be clarified that we omitted from Chapter 2 due to readability. In this section $O = \{o_1, o_2, \ldots, o_k\}$ is always the global optimum and

Algorithm 1: LocalSearch(F, C, k, cost)

Input: set F, set C, natural number k, function cost
Output: S, if cost$(S) \leq$ cost(S') for all $S' \in N(S)$

```
1 S = random subset of F with cardinality k
2 while There is a S' ∈ N(S) and cost(S') < cost(S) do
3 |   S = S'
4 return S
```

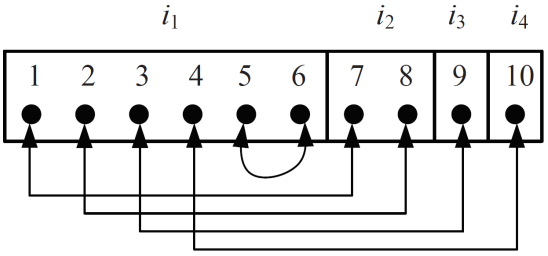


Fig. 4.1: An example for π with $N_O(o) = [10]$, where i_1 captures o and $N_O(o)$ gets partitioned by i_1, i_2, i_3 and i_4(picture from [Zha07]).

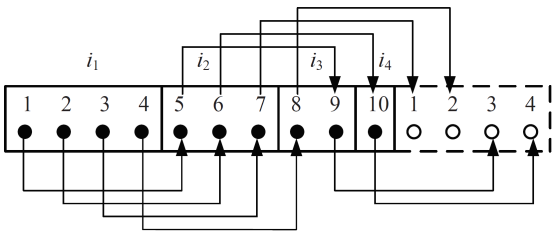


Fig. 4.2: An example for π with $N_O(o) = [10]$, where no i captures o and $N_O(o)$ gets partitioned by i_1, i_2, i_3 and i_4 (picture from [Zha07]).

$S = \{i_1, i_2, \ldots, i_k\}$ is always a local optimum. Let U be any solution, then the following notations can be introduced:

1. $U_j = d(j, \phi_U(j))$ denotes the service cost for j in U
2. $N_U(i) = \{j \in C : \phi_U(j) = i\}$, i.e. $N_U(i)$ is the set of clients, which are connected to the facility i in solution U.
3. $N_i^o = N_S(i) \cap N_O(o)$ contains all clients that are served by i and o.
4. We say i captures o, if $|N_i^o| > \frac{1}{2}|N_O(o)|$. In addition, i is called good, if it does not capture any o and bad, if it does.
5. We also use a bijective mapping $\pi : N_O(o) \to N_O(o)$. For every $i \in \{i : N_i^o \neq \emptyset\}$ that does not capture o, we have that every $j \in N_i^o$ is mapped outside of N_i^o, i.e. $\pi(j) \notin N_i^o$. If i captures o, then for each $j, \pi(j) \in N_i^o$, we have $\pi(j)$ is mapped back onto j, i.e. $\pi(\pi(j)) = j$. When constructing π as described in [Zha07], it also yields the property that if i captures o, then $j = \pi(\pi(j))$ for every $j \in N_O(o)$. Examples of the behaviour of π is shown in Figure 4.1 and Figure 4.2.

Lemma 8. *Let $j \in N_S(i)$ and $\pi(j) \notin N_S(i)$. After a swap(i, o), the new service cost for the client j can be bounded by $S_{\pi(j)} + O_{\pi(j)} + O_j$.*

Proof. We consider the cases $j \in N_i^o$ and $j \notin N_i^o$ separately:

1. $j \in N_i^o$: Let i' be the nearest facility serving the client $\pi(j)$. After o is swapped in for i, each $j \in N_i^o$ will be served by its new nearest facility i^*. Since d is a metric, it obeys the triangle inequality, leading to the following:

$$\begin{aligned} d(j, i^*) \leq d(j, i') \leq d(\pi(j), i') + d(j, \pi(j)) \leq d(\pi(j), i') + d(\pi(j), o) + d(j, o) \\ = S_{\pi(j)} + O_{\pi(j)} + O_j \end{aligned}$$

2. $j \notin N_i^o$: Let o' be the facility, with $j \in N_i^{o'}$ and therefore $\pi(j) \in N_i^{o'}$. Furthermore, let i' be the nearest facility serving the client $\pi(j)$. After o is swapped in for i, each $j \notin N_i^o$ will be served by its new nearest facility i^*. Again, we can use triangle inequality to obtain the following:

$$\begin{aligned} d(j, i^*) \leq d(j, i') \leq d(\pi(j), i') + d(j, \pi(j)) \leq d(\pi(j), i') + d(\pi(j), o') + d(j, o') \\ = S_{\pi(j)} + O_{\pi(j)} + O_j \end{aligned}$$

□

Lemma 9. *$S_j \leq S_{\pi(j)} + O_{\pi(j)} + O_j$ for each $j \in C$.*

Proof. Let i' be the nearest facility serving the client $\pi(j)$ and o be the facility for which $j \in N_i^o$. Because $\pi : N_O(o) \to N_O(o)$, we know $\pi(j) \in N_i^o$

$$\begin{aligned} S_j = d(i, j) \leq d(i', j) \leq d(\pi(j), i') + d(j, \pi(j)) \leq d(\pi(j), i') + d(\pi(j), o) + d(j, o) \\ = S_{\pi(j)} + O_{\pi(j)} + O_j \end{aligned}$$

□

Lemma 10. *If o is the nearest facility that i captures and i also captures $o' \neq o$, then after a swap(i, o), the new service cost for each $j \in N_i^{o'}$ with $\pi(j) \in N_S(i)$, can be bounded by $2S_j + O_j$.*

Proof. Let i^* be the new nearest facility to j. Because of triangle inequality and the fact that o is the nearest facility that i captures, the following holds

$$d(j, i^*) \leq d(j, o) \leq d(j, i) + d(i, o) \leq d(j, i) + d(i, o') \leq d(j, i) + d(j, i) + d(j, o') = 2S_j + O_j$$

□

For sake of completeness, we also show the following:

Lemma 11. *After a swap(i, o), the new service cost for a client $j \in N_O(o)$, with $\pi(j) \in N_S(i)$ can be bounded by O_j.*

Proof. Let i^* be the new nearest facility to j. Since $j \in N_O(o)$ is served by o in the solution O, we obtain:

$$d(j, i^*) \leq d(j, o) = O_j$$

□

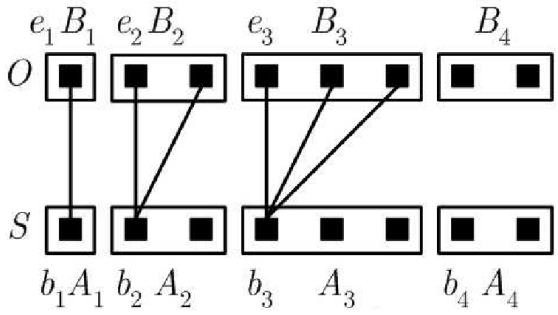


Fig. 4.3: Partition of S and O (picture from [Zha07])

Lemma 12. *The facility cost $cost_f(S)$ is bounded by $cost_f(O) + 2cost_s(O)$.*

Proof. First, partition S into subsets $(A_1, A_2, \ldots, A_m)$ and O into subsets $(B_1, B_2, \ldots, B_m)$, to get pairs (A_i, B_i) with $|A| = |B|$ $\forall i \in [m]$. To do this, we pick any bad facility $b \in S$ and add it to A. Afterwards, add every facility $o \in O$ that i captures to B and fill A with arbitrary good facilities in S, until $|A| = |B|$. Repeat this for every bad facility in A. Let A_m be the set of good facilities left in S and B_m the set of facilities left in O. Denote with $e \in B$ the facility closest to b. This method of partitioning works, because no two $i \in S$ can capture the same o and therefore the number of bad facilities in S are $\leq |A|$. An example partition for $m = 4$ can be found in Figure 4.3. We can now apply Lemma 11, Lemma 10 and Lemma 8 to bound the cost after swap(b, e) by:

$$
\begin{aligned}
&-f_b + f_e + \sum_{\substack{j \in N_b^e \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{\substack{j \in N_S(b) - N_O(e) \\ \pi(j) \in N_S(b)}} (2S_j + O_j - S_j) \\
&\qquad + \sum_{\substack{j \in N_S(b) \\ \pi(j) \notin N_S(b)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\
\Longrightarrow \qquad &-f_b + f_e + \sum_{\substack{j \in N_b^e \\ \pi(j) \in N_S(b)}} 2O_j + \sum_{\substack{j \in N_S(b) - N_O(e) \\ \pi(j) \in N_S(b)}} (S_j + O_j) \\
&\qquad + \sum_{\substack{j \in N_S(b) \\ \pi(j) \notin N_S(b)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0
\end{aligned}
$$

Moreover, the cost of a solution after one swap(i, o) with $i \in A - b$ and $o \in B - e$, can be bounded with Lemma 11 and Lemma 8 by:

$$
-f_i + f_o + \sum_{\substack{j \in N_b^o \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{\substack{j \in N_i^o \\ \pi(j) \notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0
$$

Since $O_j - S_j \leq 2O_j$ and

$$\{j \in N_b^o : o \in B - e \wedge \pi(j) \in N_S(b)\} = \{j \in N_S(b) - N_O(e) : \pi(j) \in N_S(b)\},$$

summing all of the different swaps between A and B up, gives the following bound:

$$\begin{aligned}
&-\sum_{i\in A} f_i + \sum_{o\in B} f_o + \sum_{\substack{j\in N_b^e \\ \pi(j)\in N_S(b)}} 2O_j + \sum_{\substack{j\in N_S(b)-N_O(e) \\ \pi(j)\in N_S(b)}} (S_j + O_j) \\
&+ \sum_{o\in B-e} \sum_{\substack{j\in N_b^o \\ \pi(j)\in N_S(b)}} (O_j - S_j) + \sum_{i\in A} \sum_{\substack{j\in N_S(b) \\ \pi(j)\notin N_S(b)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\
\implies &-\sum_{i\in A} f_i + \sum_{o\in B} f_o + \sum_{\substack{j\in N_S(b) \\ \pi(j)\in N_S(b)}} 2O_j + \sum_{i\in A} \sum_{\substack{j\in N_S(i) \\ \pi(j)\notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0
\end{aligned}$$

Because every facility $i \in A_m$ is good, we have $\forall i \in A_m, \forall j \in N_S(i) : \pi(j) \notin N_S(i)$. Therefore, A_m and B_m can be bounded with Lemma 8 by:

$$-\sum_{i\in A} f_i + \sum_{o\in B} f_o + \sum_{i\in A} \sum_{\substack{j\in N_S(i) \\ \pi(j)\notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0$$

Finally adding all swaps between A and B, results in the following bound:

$$\begin{aligned}
&-\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{t=1}^{m} \sum_{\substack{j\in N_S(b_t) \\ \pi(j)\in N_S(b_t)}} O_j \\
&+\sum_{i\in A} \sum_{\substack{j\in N_S(i) \\ \pi(j)\notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\
\implies &-\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{t=1}^{m} \sum_{\substack{j\in N_S(b_t) \\ \pi(j)\in N_S(b_t)}} O_j \\
&+ \sum_{\substack{j\in C \\ \pi(j)\notin N_S(\phi_S(i))}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j)) \geq 0 \\
\implies &-\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{t=1}^{m} \sum_{\substack{j\in N_S(b_t) \\ \pi(j)\in N_S(b_t)}} O_j + 2 \sum_{\substack{j\in C \\ \pi(j)\notin N_S(\phi_S(i))}} O_j \geq 0 \\
\implies &-\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{j\in C} O_j \geq 0 \\
\implies &-cost_f(S) + cost_f(O) + 2cost_s(O) \geq 0 \\
\implies &cost_f(S) \leq cost_f(O) + 2cost_s(O)
\end{aligned}$$

□

Lemma 13. *If only one swap at a time is allowed, the service cost for the local optimum S can be bounded by $cost_s(S) \leq cost_f(O) + 5cost_s(O)$.*

Proof. First, partition S into subsets $(A_1, A_2, \ldots, A_m)$ and O into subsets $(B_1, B_2, \ldots, B_m)$, with the same method presented in Lemma 12. Now, we again bound the cost after a swap(i, o), with the key difference that $i \in A - b$, $o \in B$. To obtain a term that sums up all S_j's for $j \in C$, which is the service cost $cost_s(S)$, we bound the cost after a swap(i, o) even looser, then before in Lemma 12. Since no $i \in A - b$ captures an $o \in O$, we know that $\pi(j) \notin N_S(i)$ for each $j \in N_S(i)$. Ergo we can bound a swap(i, o) with Lemma 8 and Lemma 9 by:

$$\begin{aligned} & -f_i + f_o + \sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(i) - N_O(o)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\ \implies \quad & -f_i + f_o + \sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(i)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \end{aligned}$$

Summing up swap(i, o) for $i \in A - b, o \in B - e$ and one swap(i', e) for some $i' \in A - b$ and again using Lemma 9 results in

$$\begin{aligned} & - \sum_{i \in A-b} f_i + \sum_{o \in B} f_o + \sum_{o \in B} \sum_{j \in N_O(o)} (O_j - S_j) \\ & + \sum_{i \in A-b} \sum_{j \in N_S(i)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) + \sum_{j \in N_S(i')} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\ \implies & - \sum_{i \in A-b} f_i + \sum_{o \in B} f_o + \sum_{o \in B} \sum_{j \in N_O(o)} (O_j - S_j) \\ & + 2 \sum_{i \in A} \sum_{j \in N_S(b)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\ \implies & \sum_{o \in B} f_o + 5 \sum_{j \in C} O_j - \sum_{j \in C} S_j \geq 0 \\ \implies & \sum_{j \in C} S_j \leq \sum_{o \in B} f_o + 5 \sum_{j \in C} O_j \\ \implies & cost_s(S) \leq cost_f(O) + 5cost_s(O) \end{aligned}$$

□

Theorem 14. *A local search algorithm for Exact-k-UFL with a single swap operation, has a locality gap of at most 7.*

Proof.

$$\begin{aligned} cost(S) = cost_f(S) + cost_s(S) &\leq cost_f(O) + 2cost_s(O) + cost_f(O) + 5cost_s(O) \\ &\leq 7(cost_f(O) + cost_s(O)) = 7cost(O) \end{aligned}$$

□

Notice that by combining the local search algorithm with the Rec.-k-Median algorithm proposed by [SK], we obtain a 14-approximation for Rec.-k-Median.

4.2 Single-Swap Rec.-*k*-Median Local Search Analysis

After proving, that the local search algorithm for the Exact-k-UFL is at most a 7-approximation, it can be used in combination with the reduction algorithm [SK], to obtain a 14-approximation for Rec.-k-Median. This raises the question, if a local search algorithm used directly on a Rec.-k-Median instance can still maintain a constant factor approximation. As we will show in the following, it is quite the opposite, with the local search algorithm for Rec.-k-Median having no bound at all. The local search algorithm considered, is actually Algorithm 1, with $p = 1$ and

$$\text{cost}(S) = cost_s(S) + cost_{RC}(S) = \sum_{j \in C} d(j, \phi_S(j)) + \frac{1}{2}\lambda \sum_{i,i' \in S} d(i, i')$$

Theorem 15. *Rec.-k-Median's local search algorithm, with only a 1-swap operation, is unbounded.*

Proof. Let $F = A \cup B$, $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, $C = \{1, 2\}$, $k = 2$ and $z \in \mathbb{R}$. Let the distance function d be as follows:

1. $\forall i, j, h \in [2], i \neq h : d(a_i, a_i) = d(b_i, b_i) = 0$, $d(a_i, a_h) = d(b_i, b_h) = 1$ and $d(a_i, b_j) = z$
2. $\forall i \in [2] \wedge j \in C : d(a_i, j) = 1$ and $d(b_i, j) = z$

The first item defines the distances between facilities, where facilities in the same set (A or B) have a distance of 1 and facilities in different sets, have a distance of z. The second item denotes the distances between facilities and clients, where every facility in A has a distance of 1 to every client and every facility in B has a distance of z to every client. This renders all the facilities in A symmetrical to each other. This holds up for the facilities in B as well. Then the global optimum $O = A$ and the goal is for B to be a local optimum S.

$$cost(O) = \lambda + 2 < \lambda + 2z = cost(S)$$

So $z > 1$. Now the neighbourhood of S has to have a higher cost, than S. Because of the symmetry in A and in B, every neighbour of S has the same cost.

$$\forall i, j, x, y \in [2] : cost(S - b_i + a_j) = cost(S - b_x + a_y) = \lambda z + 2$$

Hence, the following has to be true for all $i, j \in [2]$:

$$\begin{aligned} cost(S + a_i - b_j) &> cost(S) \\ \lambda z + 2 &> \lambda + 2z \\ z(\lambda - 2) &> \lambda - 2 \\ z &> 1, \quad \text{for } \lambda > 2 \end{aligned}$$

Therefore as z approaches infinity, so does the gap between O and S. □

Theorem 14 also disproves Ordozgoiti's and Gionis's theorem 3 in [OG19].

4.3 Multi-Swap Rec.-*k*-Median Local Search Analysis

We have seen that the single-swap local search algorithm does not provide an upper bound for a solution of a Rec.-k-Median instance. In the following we will show that even a multi-swap local search algorithm cannot provide a bound either. Multi-swap local search works exactly like the single-swap algorithm in Section 4.2, with the key difference, that we have a p-swap operation, where p is the number of facilities that can be swapped at once. p is fixed for the algorithm and $p \in \mathbb{N}$.

Theorem 16. *Rec.-k-Median's multi-swap local search algorithm, using a p-swap operation is unbounded.*

Proof. Let $F = A \cup B$, $A = \{a_1, a_2, \ldots, a_k\}$, $B = \{b_1, b_2, \ldots, b_k\}$, $C = \{1, 2\}$, $k > p$ and $z \in \mathbb{R}$. Let the distance function d be as follows:

1. $\forall i, j, h \in [k], i \neq h : d(a_i, a_i) = d(b_i, b_i) = 0$, $d(a_i, a_h) = d(b_i, b_h) = 1$ and $d(a_i, b_j) = z$
2. $\forall i \in [k] \wedge j \in C : d(a_i, j) = 1$ and $d(b_i, j) = z$

Again, having distance 1 between facilities in the same set (A or B) and the distance z between facilities in different sets. Clients and facilities in A have a distance of 1 and Clients and facilities in B have a distance of z (see Figure 4.4). Then, the global optimum $O = A$ and the goal is for B to be a local optimum S.

$$cost(O) = \lambda\frac{k^2 - k}{2} + 2 < \lambda\frac{k^2 - k}{2} + 2z = cost(S)$$

Let $A_i \subseteq A$ and $B_i \subseteq B$, with $|A_i| = |B_i| = i$. Due to the symmetry of the facilities in A and in B, it does not matter which $a \in A$ is included in A_i and which $b \in B$ is included in B_i. For $q \in \mathbb{N}$ and $q \leq p$, we consider the neighbourhood of S, consisting of the neighbours that are one swap(A_q, B_q) away. Due to the symmetry between a's and the symmetry between b's, all neighbours obtained by a swap(A_q, B_q) have the same cost. Since $S + A_q - B_q$ has exactly q many a's and $k - q$ many b's, the edges are as follows:

$$\begin{aligned} \frac{(k^2 - k)}{2} &\text{ many edges in total.} \\ q(k - q) = qk - q^2 &\text{ many edges between } a \in A_q \text{ and } b \in S - B_q. \\ \frac{(k^2 - k - 2qk + 2q^2)}{2} &\text{ many edges between facilities in the same set.} \end{aligned}$$

Thus, we obtain:

$$cost(S + A_q - B_q) = \lambda\left(z(qk - q^2) + \frac{k^2 - k - 2qk + 2q^2}{2}\right) + 2$$

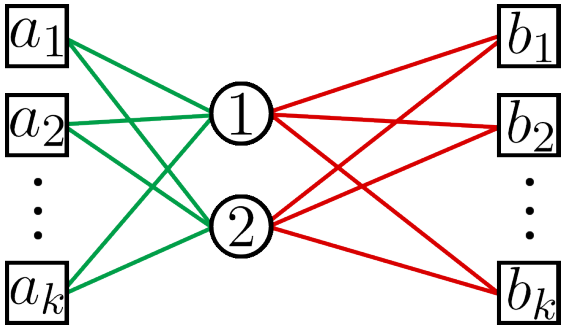


Fig. 4.4: This figure depicts, the distances between the clients C and the facilities F. A red line indicates a distance of z and a green line indicates a distance of 1

To ensure S being a local optimum, the following has to hold for all $q \in [p]$:

$$\begin{aligned} cost(S + A_q - B_q) &> cost(S) \\ \lambda z(qk - q^2) + \lambda \frac{k^2 - k - 2qk + 2q^2}{2} + 2 &> \lambda \frac{k^2 - k}{2} + 2z \\ z(\lambda(qk - q^2) - 2) &> \lambda(\frac{2qk - 2q^2}{2}) - 2 \\ z(\lambda(qk - q^2) - 2) &> \lambda(qk - q^2) - 2 \\ z &> 1, \text{ for } \lambda > \frac{2}{qk - q^2} \end{aligned}$$

The function $f_k(q) = qk - q^2$ is a parabola, with $f_k(q) = 0$ for $q = 0$ and $q = k$, and $f_k(q) > 0$ for $0 < q < k$. The parabola's only turning point, which is a maximum, is at $k/2$, therefore maximizing $2/(qk - q^2)$ for $q = 1$. Hence, λ has to be greater than $\frac{2}{k-1}$. Notice, that for $k = 2$ we get the same $\lambda > 2$ as in Section 4.2. However, as $k \to \infty$, $\lambda \to 0$, enabling the gap between S and O to approach infinity, even for small λ's. □

5 Conclusion

In this work, we presented two algorithms for Exact-k-UFL, one local search algorithm with a 7-approximation and a modified version of the algorithm by Charikar and Li [CL12] with a 3.25-approximation in expectation. We showed that the Rec.-k-Median can be approximated with a factor of at least 14 and in expectation with a factor of 6.5, by combining the reduction algorithm of Spoerhase and Khodamoradi [SK] and the Exact-k-UFL algorithms. Moreover, we have shown that the Rec.-k-Median cannot be approximated by using a multi-swap local search algorithm.

Interesting for future work would be, to prove that a multi-swap local search algorithm for Exact-k-UFL has a bound of $3 + 2/p$, where p is the maximum number of facilities allowed to be swapped with one operation. Furthermore, an evaluation for the combination between the reduction algorithm for Rec.-k-Median [SK] and the algorithm in Section 3.2 would also be interesting[1], to analyse if it proves useful in practice, since it is an expected 6.5-approximation.

[1] We provide a full implementation of both Exact-k-UFL algorithms, in combination with the reduction algorithm, here https://gitlab2.informatik.uni-wuerzburg.de/s362326/bachelor-thesis-benedikt.

Bibliography

[AIK18] M. Ahmed, M. T. Imtiaz, and R. Khan: Movie recommendation system using clustering and pattern recognition network. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 143–147, 2018, 10.1109/CCWC.2018.8301695.

[BDM+20] Jarosław Byrka, Szymon Dudycz, Pasin Manurangsi, Jan Marcinkowski, and Michał Włodarczyk: To close is easier than to open: Dual parameterization to k-median, 2020.

[Ber06] P. Berkhin: *A Survey of Clustering Data Mining Techniques*, pages 25–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, ISBN 978-3-540-28349-2, 10.1007/3-540-28349-8_2. https://doi.org/10.1007/3-540-28349-8_2.

[BPR+] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh: *An Improved Approximation for k-median, and Positive Correlation in Budgeted Optimization*, pages 737–756. 10.1137/1.9781611973730.50. https://epubs.siam.org/doi/abs/10.1137/1.9781611973730.50.

[CBJD18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze: Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[CGK+19] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li: Tight FPT approximations for k-median and k-means. *CoRR*, abs/1904.12334, 2019. http://arxiv.org/abs/1904.12334.

[CGvTS02] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys: A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129 – 149, 2002, https://doi.org/10.1006/jcss.2002.1882, ISSN 0022-0000. http://www.sciencedirect.com/science/article/pii/S0022000002918829.

[CL12] Moses Charikar and Shi Li: A dependent lp-rounding approach for the k-median problem. In *Automata, Languages, and Programming*, pages 194–205, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg, ISBN 978-3-642-31594-7.

[GKPS02] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan: Dependent rounding in bipartite graphs. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 323–332, 2002, 10.1109/SFCS.2002.1181955.

[JMS02] Kamal Jain, Mohammad Mahdian, and Amin Saberi: A new greedy approach for facility location problems. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 731–740, New York, NY, USA, 2002. Association for Computing Machinery, ISBN 1581134959, 10.1145/509907.510012. https://doi.org/10.1145/509907.510012.

[JV01] Kamal Jain and Vijay V. Vazirani: Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, March 2001, 10.1145/375827.375845, ISSN 0004-5411. https://doi.org/10.1145/375827.375845.

[Li13] Shi Li: A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45 – 58, 2013, https://doi.org/10.1016/j.ic.2012.01.007, ISSN 0890-5401. http://www.sciencedirect.com/science/article/pii/S0890540112001459, 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[OG19] Bruno Ordozgoiti and Aristides Gionis: Reconciliation k-median: Clustering with non-polarized representatives. In *The World Wide Web Conference*, WWW '19, page 1387–1397, New York, NY, USA, 2019. Association for Computing Machinery, ISBN 9781450366748, 10.1145/3308558.3313475. https://doi.org/10.1145/3308558.3313475.

[SK] Joachim Spoerhase and Kamyar Khordamoradi: A reduction from reconciliation k-median to uncapacitated k-facility location.

[Sri01] A. Srinivasan: Distributions on level-sets with applications to approximation algorithms. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2001, 10.1109/SFCS.2001.959935.

[Zha07] Peng Zhang: A new approximation algorithm for the k-facility location problem. *Theoretical Computer Science*, 384(1):126 – 135, 2007, https://doi.org/10.1016/j.tcs.2007.05.024, ISSN 0304-3975. http://www.sciencedirect.com/science/article/pii/S0304397507004665, Theory and Applications of Models of Computation.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 9. December 2020

. . . . . .
Benedikt Riegel