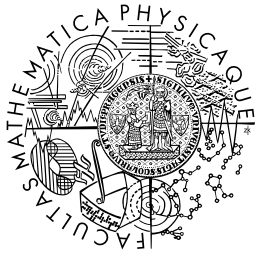


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Tung Anh Vu

Algorithms for Low Highway Dimension Graphs

Department of Applied Mathematics

Supervisor of the master thesis: Andreas Emil Feldmann, Dr.
Study programme: Computer Science
Study branch: Theoretical Computer Science

Prague 2021

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor Andreas Emil Feldmann for guiding me throughout the process of writing this thesis.

Title: Algorithms for Low Highway Dimension Graphs

Author: Bc. Tung Anh Vu

Department: Department of Applied Mathematics

Supervisor: Andreas Emil Feldmann, Dr., Department of Applied Mathematics

Abstract: In this work we develop algorithms for the k-Supplier with Outliers problem. In a network, we are given a set of *suppliers* and a set of *clients*. The goal is to choose k suppliers so that the distance between every served client and its nearest supplier is minimized. Clients that are not served are called *outliers* and the number of allowed outliers is given on input.

As k-Supplier with Outliers has numerous applications in logistics, we focus on parameters which are suitable for *transportation networks*. We study graphs with low *highway dimension*, which was proposed by Abraham et al. [SODA 2010], and low *doubling dimension*.

It is known that unless P = NP, k-Supplier with Outliers does not admit a $(3-\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$. The k-Supplier with Outliers problem is W[1]-hard on graphs of constant doubling dimension for parameters k and highway dimension. We overcome both of these barriers through the paradigm of *parameterized approximation* algorithms.

In the case of highway dimension, we develop a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$ with running time $f(k,p,h,\varepsilon)\cdot n^{O(1)}$ where p is the number of allowed outliers, h is the highway dimension of the input graph, and f is some computable function. In the case of doubling dimension, we develop a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$ with running time $(k+p)^k \cdot \varepsilon^{-O(kd)} \cdot n^{O(1)}$ where p is the number of allowed outliers, and d is the doubling dimension of the input graph. In fact, the latter algorithm can be extended to a more general problem called Capacitated k-Supplier with Outliers.

Additionally, we consider a generalization of k-Supplier with Outliers called Non-Uniform k-Supplier. It was shown that Non-Uniform k-Supplier does not admit constant-approximation algorithms with a polynomial running time, unless P = NP. We extend this hardness result to the setting where the highway dimension and the doubling dimension are constant.

Keywords: highway dimension, doubling dimension, parameterized approximation, k-Supplier with Outliers problem

Contents

Introduction

In this work we consider the k-Supplier problem, which was first introduced by Hochbaum and Shmoys [HS86]. The input is specified by a graph $G = (V, E)$ with positive edge lengths $d : E \to \mathbb{R}^+$, a set of *suppliers* $V_s \subseteq V$, a set of *clients* $V_c \subseteq V$, and an integer k. The goal is to select k suppliers, so that the distance between any client and its nearest selected supplier is minimal. Formally, a *feasible solution* to the problem is a set $S \subseteq V_s$ such that $|S| \leq k$. For a pair of vertices $u, v \in V$, we denote by $\text{dist}_G(u, v)$ the shortest-path distance from u to v with respect to edge lengths d. We omit the subscript if the graph is clear from context. For a vertex u and a set of vertices $A \subseteq V$, we denote $\text{dist}(u, A) = \min_{v \in A} \text{dist}(u, v)$. The cost of a feasible solution S is $\max_{c \in V_c} \text{dist}(c, S)$, and the goal is to find a solution of minimum cost. Note that the conventional definition of k-Supplier [HS86, WS11] has $V_c = V \setminus V_s$. For our purposes however, it will be useful to have vertices which are neither clients nor suppliers.

The k-Supplier problem is a generalization of the k-Center problem. We can formulate k-Center as a special case of k-Supplier where the set of clients and the set of suppliers coincide. That is, given a graph $G = (V, E)$ on input, we have $V_s = V_c$.

A natural application of k-Supplier would be the following situation: we have predetermined some locations in a network which are suitable for building warehouses. We also survey the network and determine the set of clients we would like to serve. Warehouses are typically built on the outskirts of cities, as opposed to clients whose location does not necessarily follow any pattern. As such, to find a good placement of warehouses it is preferable to model the problem using k-Supplier over k-Center: a good k-Center solution might determine that a warehouse should be placed inside a client's residence. Additionally, the majority of the points in a map of the network, e.g., road intersections, will be neither clients nor suppliers. Hence for applications it is indeed desirable to have points which are neither clients nor suppliers in the definition of k-Supplier.

It is known that k-Center is already an NP-hard problem [HN79]. Hence under the standard complexity assumption P $\neq$ NP, we cannot hope to solve it in time polynomial in the length of the input. Two popular ways of dealing with NP-hard problems is to design *approximation algorithms* [Vaz13, WS11] and *parameterized algorithms* [CFK$^+$15, DF13]. An approximation algorithm computes in polynomial time a *b-approximation*, that is, a solution that is at most b times worse than the optimum. For k-Center, there are two known 2-approximation algorithms, one by Hochbaum and Shmoys [HS86] and another by Gonzalez [Gon85]. Moreover, it is known that there cannot be a $(2 - \varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$ unless P = NP [HN79]. Even worse, for k-Supplier it is known that there cannot be a $(3 - \varepsilon)$-approximation [HS86]. This hardness result is matched by a 3-approximation algorithm by Hochbaum and Shmoys [HS86].

Under the assumption that P $\neq$ NP, any exact algorithm for an NP-hard problem will have a running time superpolynomial in the length of its input. The rationale behind fixed-parameter algorithms is to isolate this superpolynomial growth of the running time to some *parameter* q of the input while retaining

a polynomial dependence on the input length. More precisely, a fixed-parameter algorithm computes an optimum solution in time $f(q) \cdot n^{O(1)}$ where f is some computable function, q is a parameter of the input, and n is the length of the input. A problem that has a fixed-parameter algorithm for a parameter q is called *fixed-parameter tractable* (FPT) for q. For k-Center one immediate choice of a parameter would be the number of centers k. Unfortunately, Demaine et al. [DFHT05] have shown that k-Center is a W[2]-hard problem for parameter k. Furthermore, Marx [Mar05] has shown that k-Center is W[1]-hard for parameter k in two-dimensional Manhattan metrics. Under the standard assumption FPT $\subsetneq$ W[1] $\subsetneq$ W[2], this means that k-Center is not FPT for the parameter k in either of these settings.

As fixed-parameter algorithms have a superpolynomial running time in their parameter, for practical purposes one should think of parameters that are small for typical instances. Guided by the example we gave earlier of an application of k-Supplier, we shall consider parameters that are appropriate for *transportation networks*.

Abraham et al. [AFGW10] introduced the *highway dimension* in order to explain the fast running times of various shortest-path heuristics. The definition of highway dimension is motivated by the following empirical observation of Bast et al. [BFM06, BFM$^+$07]. Imagine that we want to travel in a road network from some point A to a sufficiently far point B along the quickest route. The observation is that if we travel along the quickest route, we will inevitably pass through a sparse set of "access points". Highway dimension then measures the sparsity of this set of access points around any vertex of a graph. There are several formal definitions for the highway dimension that differ slightly, see [AFGW10, ADF$^+$11, ADF$^+$16]. Here, we give the definition that we will use in our algorithms. Let $G = (V, E)$ be a graph with edge lengths $d : E \to \mathbb{R}_0^+$. For a shortest path $\pi = (u_1, u_2, \ldots, u_\ell)$ between its endpoints, we define its *length* as $d(\pi) = \sum_{i=1}^{\ell-1} d((u_i, u_{i+1}))$. For a positive real number $r \in \mathbb{R}^+$ the *ball* centered at u of radius r is the set $B_u(r) = \{v \in V : \mathrm{dist}_G(u, v) \leq r\}$, i.e., vertices whose distance from u is at most r.

Definition 1 ([FFKP18])**.** *Let $G = (V, E)$ be an undirected graph with edge lengths $d : E \to \mathbb{R}_0^+$. The* highway dimension *of G is the smallest integer h such that, for some universal constant $\gamma \geq 4$, for any radius $r \in \mathbb{R}^+$, and any vertex u there is a hitting set $Z \subseteq B_u(\gamma r)$ of size h for the set of all shortest paths π satisfying $d(\pi) > r$ and $\pi \subseteq B_u(\gamma r)$.*

Note that the definition we use is slightly more general than the original one stated in [AFGW10]. Abraham et al. [AFGW10] specifically chose $\gamma = 4$ but they also note that this choice is, to some extent, arbitrary. Nevertheless, Feldmann et al. [FFKP18, Lemma 9.6] have shown that the highway dimension of a graph is highly sensitive to the choice of γ in Definition 1: their result says that for any constant $\gamma \geq 4$ there exists a graph that, according to Definition 1, has highway dimension 1 with respect to γ and highway dimension $\Omega(n)$ with respect to any $\gamma' > \gamma$. Their proof only mentions the case when $\gamma > 4$ but it proves the case when $\gamma = 4$ as well.

Another parameter that we will consider is the *doubling dimension*. Let us first give the formal definition.

Definition 2 ([GKL03])**.** *The* doubling constant *of a metric space* (X, dist) *is the smallest value* λ *such that every ball in* X *can be covered by* λ *balls of half the radius. The* doubling dimension *of* X *is then defined as* $\text{dd}(X) = \log_2 \lambda$.

Folklore results show that every metric for which the distance function is given by the ℓ_q-norm in D-dimensional space $\mathbb{R}^D$ has doubling dimension $O(D)$ [GKL03, FM20]. As a transportation network is embedded on a large sphere (namely the Earth), a reasonable model is to assume that the shortest-path metric abides to the Euclidean ℓ_2-norm. Buildings in cities form so called "city blocks", which form a grid of streets. Therefore it is reasonable to assume that the distances in cities are given by the Manhattan ℓ_1-norm. Road maps can be thought of as a mapping of a transportation network into $\mathbb{R}^2$. It is then reasonable to assume, that transportation networks have constant doubling dimension.

An immediate question is whether k-Center is FPT when parameterized by highway dimension or doubling dimension. Under the standard assumption FPT $\neq$ W[1], Feldmann and Marx [FM20] have shown that the answer is negative. In fact, the result they show is even stronger: they show that the k-Center problem is W[1]-hard on weighted planar graphs of constant doubling dimension for the combined parameter k, highway dimension, and *pathwidth*[1].

So far, it seems like the k-Center problem is quite resistant to approximation and fixed-parameter algorithms, aside from the aforementioned 2-approximation algorithms. However, what if we combine the two approaches? What if we want a fixed-parameter algorithm that can return an approximate solution instead of an optimum solution, and whose approximation factor is significantly better than the 2 which we can achieve in polynomial time? In such cases, we develop *fixed-parameter b-approximation algorithms* (b-FPA) that compute a b-approximation in time $f(p) \cdot n^{O(1)}$ for parameter p. A family of algorithms for a problem such that for every constant $\varepsilon > 0$ there exists an $(1+\varepsilon)$-approximation algorithm for the said problem is called an *approximation scheme*. If an approximation scheme has a running time in the form of $f(\varepsilon, p) \cdot n^{O(1)}$ where f is some computable function and p is a parameter of the input, then we call such an approximation scheme an *efficient*[2] *parameterized approximation scheme* (EPAS) for parameter p.

For the parameters we have considered so far, the answer is negative. Feldmann [Fel19] has shown that unless FPT = W[2], there is no $(2-\varepsilon)$-FPA for k-Center when the parameter is k. The following results imply that we cannot parameterize solely by the doubling dimension: Feder and Greene [FG88] have shown that unless P = NP, there cannot be a $(1.822-\varepsilon)$-approximation algorithm for two-dimensional Euclidean metrics, and that there cannot be a $(2-\varepsilon)$-approximation algorithm for two-dimensional Manhattan metrics. On the other hand, the situation is not clear-cut in the case of highway dimension. To the best of our knowledge, it is not yet ruled out that there exists a $(2-\varepsilon)$-FPA algorithm when parameterizing by highway dimension. The best result so far is by Feldmann [Fel19] which shows that it is NP-hard to $(2-\varepsilon)$-approximate k-Center on graphs with highway dimension $O(\log^2 n)$. He [Fel19] notes that while this

[1] See Section 1.3 for definitions of pathwidth and treewidth.

[2] Efficient in this context means that the dependence on ε in running time of the algorithm can be bounded by some computable function of the parameter and ε. In particular, an efficient parameterized approximation scheme cannot have a running time in the form of, e.g., $O(n^{O(1/\varepsilon)})$.

does not rule out $(2-\varepsilon)$-FPA algorithms for the highway dimension parameter, if such an algorithm exists, then, it cannot have a running time of $2^{2^{o(\sqrt{h})}} \cdot n^{O(1)}$, unless the Exponential Time Hypothesis (ETH) [IP01] fails.

While the situation seems dire, there is still one more approach that we have not discussed yet. That is, if we combine multiple parameters together. The outlook for the k-Center problem is much more positive in this setting. For parameters k and highway dimension there is an EPAS by Becker et al. [BKS18], for parameters k and doubling dimension there is an EPAS by Feldmann and Marx [FM20], and for parameter *treewidth*[1] there is an EPAS by Katsikarelis et al. [KLP19].

We will extend these results to the k-Supplier with Outliers problem, which was first introduced by Charikar et al. [CKMN01]. The motivation behind this problem is that "a few very distant clients, called *outliers*, can exert a disproportionately strong influence over the final solution" [CKMN01]. Hence it is desirable to leave these few outliers unserviced in favor of obtaining a solution of significantly lower cost. Formally, in addition to the input of the k-Supplier problem, the input contains an integer $p \in \mathbb{N}_0$. A feasible solution is a subset $S \subseteq V_\mathrm{s}$ of size at most k and its cost is

$$\mathrm{cost}(S) = \min_{\substack{V'_\mathrm{c} \subseteq V_\mathrm{c} \\ |V'_\mathrm{c}| \geq n-p}} \max_{u \in V'_\mathrm{c}} \mathrm{dist}(u, S)$$

where V'_c are subsets of clients of size at least $n-p$. Clients $V_\mathrm{c} \setminus V'_\mathrm{c}$ are called *outliers*. The goal is to find a solution of minimum cost. Note that setting $p = 0$ gives the original k-Supplier problem. The k-Center with Outliers problem is a special case when $V_\mathrm{s} = V_\mathrm{c}$.

We will show a $\frac{3}{2}$-approximation algorithm for k-Center with Outliers when the parameters are k, the number of outliers, and the highway dimension of the input graph. The result is an extension of a $\frac{3}{2}$-approximation algorithm for k-Center by [Fel19]. Given an instance $\mathcal{I}$ of some problem, we denote by $\mathrm{OPT}(\mathcal{I})$ the cost of the optimum solution of the instance $\mathcal{I}$. The result is the following theorem.

Theorem 3. *Let $\mathcal{I} = (G, k, p)$ be an instance of the k-Center with Outliers problem where G has highway dimension h. There exists an algorithm that outputs a solution of cost at most $\frac{3}{2}\mathrm{OPT}(\mathcal{I})$ in time $2^{O(k(h \log h)+p)} \cdot n^{O(1)}$.*

For a stricter definition of highway dimension, we extend the EPAS parameterized by k and highway dimension for k-Center by Becker et al. [BKS18] to k-Supplier with Outliers. This "strictness" concerns the choice of the constant γ in Definition 1. In both the original algorithm [BKS18] and in our algorithm, we will require that the constant γ is strictly greater than 4, while the algorithm given by Theorem 3 can work with the definition of highway dimension where $\gamma = 4$ as well. As we have mentioned previously, Feldmann et al. [FFKP18] have shown that a graph can have significantly different highway dimensions depending on the choice of the constant γ. Thus, while the approximation ratio of the algorithm given by Theorem 3 is weaker than the approximation ratio of an EPAS, it uses a more general definition of highway dimension. Another advantage of the algorithm given by Theorem 3 is its running time. Both algorithms are

exponential in k, the number of outliers, and highway dimension, however, the base of the exponent in the running time in Theorem 3 is constant while the base of the exponent of the EPAS depends on k, the number of outliers, the highway dimension, and ε. The result is the following theorem.

Theorem 4. *Let $\mathcal{I} = (G, k, p)$ be an instance of the k-Supplier with Outliers problem where G has highway dimension h. There exists a computable function $f(\cdot,\cdot,\cdot,\cdot)$ and an algorithm such that for any $\varepsilon > 0$ it outputs a solution of cost $(1+\varepsilon)\mathrm{OPT}(\mathcal{I})$ in time $f(h,k,p,\varepsilon) \cdot n^{O(1)}$. Moreover we have $f \in \Omega\left(\left(\frac{1}{\varepsilon}(h+k+p)^2\right)^{O\left((h+k+p)^2 \log(1/\varepsilon)\right)}\right)$.*

When considering doubling dimension instead of highway dimension, we extend the EPAS parameterized by k and doubling dimension for the k-Center problem by Feldmann and Marx [FM20] to an EPAS for the Capacitated k-Supplier with Outliers problem. In the example of an application of k-Supplier we gave earlier, a warehouse can supply an unbounded number of clients. This might not be the case in real-world applications as warehouses can only store a limited amount of goods. In Capacitated k-Supplier with Outliers this issue is mitigated by placing a limit on the number of clients each supplier can serve. Let us define the problem formally. In the Capacitated k-Supplier with Outliers problem [CHK12], in addition to the input of the k-Supplier with Outliers problem, we have a *capacity function* $L : V_\mathrm{s} \to \mathbb{N}_0$. A feasible solution is a set of clients $V'_\mathrm{c} \subseteq V_\mathrm{c}$, a set of suppliers $V'_\mathrm{s} \subseteq V_\mathrm{s}$, and a function $\phi : V'_\mathrm{c} \to V'_\mathrm{s}$ which maps every client of V'_c to some supplier of V'_s such that:

- $|V'_\mathrm{c}| \geq n - p$,
- $|V'_\mathrm{s}| \leq k$,
- $|\phi^{-1}(s)| \leq L(s)$ for every $s \in V'_\mathrm{s}$.

The cost of such a feasible solution is $\max_{u \in V'_\mathrm{c}} \mathrm{dist}(u, \phi(u))$ and the goal is to find a solution of minimum cost. If we set $L(s) = |V_\mathrm{c}|$ for every supplier $s \in V_\mathrm{s}$, then we get the original (uncapacitated) k-Supplier with Outliers problem. Our result is the following theorem. Note that the running time depends only on the doubling dimension of the suppliers. This can be especially advantageous, if the set of clients does not exhibit any particular structure.

Theorem 5. *Let $\mathcal{I} = (G, k, p, L)$ be an instance of the Capacitated k-Supplier with Outliers problem, $(V_\mathrm{s}, \mathrm{dist})$ be the shortest-path metric induced by the supplier set V_s, and d be the doubling dimension of $(V_\mathrm{s}, \mathrm{dist})$. There exists an algorithm such that for any $\varepsilon > 0$ it outputs a solution of cost $(1+\varepsilon)\mathrm{OPT}(\mathcal{I})$ in time $(k+p)^k \cdot \varepsilon^{-O(kd)} \cdot n^{O(1)}$.*

To obtain the algorithm given by Theorem 4, that is, an EPAS for k-Supplier with Outliers when the parameters are k, the number of outliers, and highway dimension, we use a framework by Becker et al. [BKS18]. To be able to use the framework, we need as a subroutine an EPAS for k-Supplier with Outliers when the parameter is the treewidth of the input graph. While we

are able to show such an algorithm for the k-Supplier with Outliers problem (Theorem 17), we also show that such an algorithm for the Capacitated k-Supplier with Outliers problem would imply that FPT = W[1]. Therefore we are not able to obtain a result analogous to Theorem 5 when parameterizing by highway dimension. Formally, the result is the following theorem.

Theorem 6. *For any $\varepsilon > 0$ it is* W[1]*-hard to $(2-\varepsilon)$-approximate the Capacitated k-Supplier with Outliers problem for parameters k and treewidth.*

Note the difference in the dependence on the number of outliers in the running time between Theorem 4 and Theorem 5. While in Theorem 4, the parameter p appears in the exponent, in Theorem 5 the parameter p appears only in the base of the exponent. We leave open whether the dependence on p can be improved in the case of highway dimension. We also leave open whether the dependence on p can be improved in either case so that the running time is polynomial in p.

Another downside of the definition of k-Supplier we gave earlier is that we assumed all suppliers can provide the same quality of service. This might not be the case in real-world applications, since warehouse staff can have different level of competence, equipment and vehicles have different performance levels, etc. Therefore aside from placing the suppliers in the network, we should also consider the magnitude of the area each supplier is able to serve while maintaining service quality standards. In the Non-Uniform k-Supplier problem, in addition to the input of k-Supplier the input contains k reals $r_1 \geq r_2 \geq \cdots \geq r_k$. A feasible solution S is a tuple of k suppliers $(s_1, \ldots, s_k)$ and its cost is $\max_{u \in V_c} \min_{i=1}^{k} \frac{\text{dist}(u,s_i)}{r_i}$. The goal is to find a solution of minimum cost. We can formulate an instance of k-Supplier by setting $r_1 = \cdots = r_k = 1$. If it is the case that $V_c = V_s$, then we have an instance of the Non-Uniform k-Center problem, which was recently introduced by Chakrabarty et al. [CGK16]. We can formulate an instance of k-Center with Outliers as an instance of Non-Uniform k-Center by setting $r_1 = \cdots = r_k$ and $r_{k+1} = \cdots = r_{k+p} = 0$. Note that this approach cannot be used to generalize k-Supplier with Outliers to Non-Uniform k-Supplier as each outlier would have to be a supplier to be able to open a supplier of radius 0 in it. Chakrabarty et al. [CGK16] show that unless P = NP, there cannot be a polynomial time b-approximation algorithm for any constant $b \geq 1$ for Non-Uniform k-Center. We will show a similar hardness result for the Non-Uniform k-Supplier problem when the parameters are doubling dimension and highway dimension. The following theorem implies that the algorithms given by Theorems 4 and 5 cannot be extended to Non-Uniform k-Supplier.

Theorem 7. *It is NP-hard to b-approximate the Non-Uniform k-Supplier problem for any constant $b \geq 1$, even on instances where both the highway dimension and the doubling dimension of the graph is* 2*.*

1. Preliminaries

1.1 Notation

Let $G = (V, E)$ be a graph with edge lengths $d : E \to \mathbb{R}_0^+$. By $\mathrm{dist}_G(u, v)$ we denote the *shortest-path distance* from u to v. We omit the subscript if the graph is clear from context. For a vertex u and a subset of vertices $W \subseteq V$ we denote $\mathrm{dist}_G(u, W) = \min_{w \in W} \mathrm{dist}_G(u, w)$. For a shortest path $\pi = (u_1, u_2, \ldots, u_\ell)$ between its endpoints, we define its *length* as $d(\pi) = \sum_{i=1}^{\ell-1} d((u_i, u_{i+1}))$. For a positive real number $r \in \mathbb{R}$ the *ball* centered at u of radius r is the set $B_u(r) = \{v \in V : \mathrm{dist}_G(u, v) \leq r\}$, i.e., vertices whose distance from u is at most r. The *open neighbourhood* of a vertex u is the set $N(u) = \{v \in V : (u, v) \in E\}$. The *closed neighbourhood* of a vertex u is the set $N[u] = N(u) \cup \{u\}$.

1.2 Approximation algorithms & parameterized complexity

In this thesis we will design parameterized approximation algorithms. For the reader's convenience, we state some essential notions from those fields. For the rest of this work we shall make the standard assumption that P $\neq$ NP and FPT $\neq$ W[1].

Parameterized complexity. We follow [CFK+15] for the standard definitions from the field of parameterized complexity.

We start by defining a parameterized problem and its corresponding complexity class. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the *parameter*. The size of an instance (x, k) is defined as $|x|+k$, i.e. we assume that k is encoded in unary. A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable (FPT)* if there exists an algorithm $\mathcal{A}$ (called a *fixed-parameter algorithm*), a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called *FPT*.

For an FPT algorithm $\mathcal{A}$, we are often not interested in the polynomial factor of its runtime. In such cases we may express the running time of $\mathcal{A}$ using the *O^*-notation* which suppresses factors polynomial in the input size. A running time $O^*(f(k))$ means that the running time is bounded by $f(k) \cdot n^{O(1)}$ where n is the input size.

To exclude the existence of an FPT algorithm for some problem, we will need the notion of a parameterized reduction. Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that

- (x, k) is a YES instance of A if and only if (x', k') is a YES instance of B,
- $k' \leq g(k)$ for some computable function g, and

- the running time is $f(k) \cdot |x|^{O(1)}$ for some computable function f.

Analogous to the NP-hardness in classical complexity, the corresponding counterpart in the field of parameterized algorithms is W[1]-*hardness*. Under the standard assumption FPT $\neq$ W[1], if there is a parameterized reduction from a W[1]-hard problem A to problem B, then problem B is also W[1]-hard and it does not admit an FPT algorithm. For further information about the W[t] hierarchy, we refer the reader to [CFK+15] and [FG01].

Let us remark that the aforementioned assumption P $\neq$ NP is not sufficient to imply that FPT $\neq$ W[1]. We need a stronger assumption known as Exponential Time Hypothesis (ETH), first formulated by Impagliazzo and Paturi [IP01], whose statement follows: Let $\delta \in \mathbb{R}$ be the infimum of the set of of constants c for which there exists an algorithm solving 3-SAT in time $O^*(2^{cn})$, then $\delta > 0$.

Approximation algorithms. We follow [Vaz13] and [WS11] for the standard definitions from the field of approximation algorithms.

Before defining an approximation algorithm, we need the notion of an optimization problem. An NP-*optimization* problem Π consists of:

- A set of *valid instances* D_Π recognizable in polynomial time.
- Each instance $I \in D_\Pi$ has a set of *feasible solutions* $S_\Pi(I)$. We require that $S_\Pi(I) \neq \emptyset$ and that every solution $s \in S_\Pi(I)$ is of length polynomially bounded in I. Furthermore, there is a polynomial algorithm which, given a pair (I, s), decides whether $s \in S_\Pi(I)$.
- There is a polynomial time computable *objective function* obj_Π that assigns a nonnegative rational number to each pair (I, s) where I is an instance and s a feasible solution to I. We may refer to the objective function as the *cost function* and denote the cost of a solution by $\text{cost}_\Pi(I, s)$. We may omit the subscript Π or the argument I if they are clear from the context.
- Finally, Π is specified to be either a *minimization* or *maximization* problem.

An *optimal solution* for an instance of a minimization (maximization) problem is a feasible solution that achieves the smallest (largest) objective function value. We denote by $\text{OPT}_\Pi(I)$ the cost of the optimal solution of an instance I of the problem Π. We may omit the subscript Π or the argument I if the problem at hand or the instance are clear from the context.

Let $\mathcal{A}$ be an algorithm for an optimization problem Π with a running time polynomial in the size of its input. Given an instance I of the problem Π, we denote by $\mathcal{A}(I)$ the cost of the solution output by $\mathcal{A}$ on input I. Let $\alpha : \mathbb{N} \to \mathbb{R}^+ \cap [1, \infty)$ be a nondecreasing function of the input size. We say that $\mathcal{A}$ is an α-*approximation algorithm* for Π if for all instances I of Π we have $\mathcal{A}(I) \leq \alpha(|I|) \cdot \text{OPT}_\Pi(I)$ when Π is a minimization problem and $\alpha(|I|) \cdot \mathcal{A}(I) \geq \text{OPT}_\Pi(I)$ if Π is a maximization problem.

A natural question for any optimization problem is how well we can approximate it, that is what is the best $\alpha \geq 1$ such that there exists an α-approximation algorithm for it. In some cases, we are able to obtain arbitrarily good approximation algorithms; a *polynomial-time approximation scheme* (PTAS) is a family

of algorithms $\{A_\varepsilon\}$, where there is an algorithm for each constant $\varepsilon > 0$, such that A_ε is a $(1+\varepsilon)$-approximation algorithm. Note that this definition allows A_ε to have running times such as $n^{O(1/\varepsilon)}$; if we require the running time to be bounded by $O(n^c)$ for a constant c independent of ε, then the resulting family of algorithms is called an *efficient polynomial-time approximation scheme* (EPTAS). Still, the constant hidden by the big-O notation in the running time of an EPTAS can depend on ε arbitrarily. If we further restrict the running time of an EPTAS to be polynomial in $\frac{1}{\varepsilon}$, then the resulting family of algorithms is called a *fully polynomial-time approximation scheme* (FPTAS).

Parameterized optimization. Given a problem which is W[1]-hard and there exists a (possibly constant) computable function g of the input size n such that it does not admit a $g(n)$-approximation algorithm, we may hope to obtain an algorithm which combines both approaches to solving it by producing an approximation of the optimum solution in FPT time. We follow [CGG06], [FSLM20] and [Mar08] for standard definitions from the field of parameterized complexity and parameterized optimization.

Let Π be an NP-optimization problem with a possibly empty set of feasible solutions and $\kappa : D_\Pi \to \mathbb{N}$ be a *parameterization* that assigns a parameter to each instance of the problem. Let $\mathcal{A}$ be an algorithm for an optimization problem Π. If the set of feasible solutions of an instance is empty, then we allow the behaviour of $\mathcal{A}$ to be undefined. For the rest of this part, assume that the set of feasible solutions is nonempty. Given an instance I of the problem Π, we denote by $\mathcal{A}(I)$ the cost of the solution output by $\mathcal{A}$ on input I. Let $\alpha : \mathbb{N} \to \mathbb{R}^+ \cap [1, \infty)$ be a nondecreasing function of the input size. We say that $\mathcal{A}$ is an FPT *α-approximation algorithm* if for all instances I of Π we have $\mathcal{A}(I) \leq \alpha(|I|) \cdot \mathrm{OPT}_\Pi(I)$ when Π is a minimization problem and $\alpha(|I|) \cdot \mathcal{A}(I) \geq \mathrm{OPT}_\Pi(I)$ if Π is a maximization problem, and the running time of $\mathcal{A}$ is bounded by $f(\kappa(I)) \cdot |I|^{O(1)}$ for some computable function f.

Analogously to polynomial-time approximation schemes, we define a *parameterized approximation scheme* (PAS) to be a family of algorithms $\{A_\varepsilon\}$, where there is an algorithm for each $\varepsilon > 0$, such that A_ε is a $(1+\varepsilon)$-approximation algorithm with running time $f(k, \varepsilon) \cdot n^{g(1/\varepsilon)}$ for some computable functions f and g where n is the input size and k is the parameter. Note that such an algorithm is an FPT algorithm only if we treat ε as a constant. To obtain an FPT algorithm for the case when ε is a parameter, we restrict the running time to $f(k, \varepsilon) \cdot n^{O(1)}$. The resulting family of algorithms with such a running time is called a *efficient parameterized approximation scheme* (EPAS).

1.3 Graph parameters

Treewidth. The following definition of treewidth is given in [CFK+15]. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree whose every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a *bag*, such that the following three conditions hold:

(T1) Every vertex of G is in at least one bag, that is $\bigcup_{t \in V(T)} X_t = V(G)$.

(T2) For every $(u, v) \in E(G)$, there exists a node $t \in T$ such that bag X_t contains both u and v.

(T3) For every $u \in V(G)$, the set $T_u = \{t \in V(T) : u \in X_t\}$, that is the set of nodes whose corresponding bags contain u, induces a connected subtree of T.

To improve comprehensibility, we shall refer to vertices of the underlying tree as *nodes*. It follows from the third condition that for all $i, j, k \in V(T)$, if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$. The *width* of a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ equals $\max_{t \in V(T)}(|X_t| - 1)$, that is, the maximum size of its bag minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum possible width of a tree decomposition of G.

If the underlying tree T of a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ is a path, then $\mathcal{T}$ is called a *path decomposition*. The *width* of a path decomposition is the maximum size of its bag minus 1. The *pathwidth* of a graph G, denoted by $\mathrm{pw}(G)$, is the minimum possible width of a path decomposition of G. Clearly $\mathrm{tw}(G) \leq \mathrm{pw}(G)$ for any graph G.

For algorithmic purposes, it is often more convenient to work with nice tree decompositions. A rooted tree decomposition $(T, \{X_t\}_{t \in V(T)})$ with root $r \in V(T)$ is a *nice tree decomposition* if each of its leaves $\ell \in V(T)$ contains an empty bag (that is $X_\ell = \emptyset$) and inner nodes are one of the following three types:

- **Introduce node:** a node t with exactly one child t' such that $X_t = X_{t'} \cup \{u\}$ for some vertex $u \notin X_{t'}$. We say that u is *introduced* at t.
- **Forget node:** a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{v\}$ for some vertex $v \in X_{t'}$. We say that v is *forgotten* at t.
- **Join node:** a node t with exactly two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

It is known that if graph G admits a tree decomposition of width at most k, then it also admits a nice tree decomposition of width at most k. Furthermore, given a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G of width at most k, one can in $O(k^2 \cdot \max(|V(T)|, |V(G)|))$ time compute a nice tree decomposition of G of width at most k that has at most $O(k\,|V(G)|)$ nodes, for more details, see [CFK+15, Lemma 7.4]. For this reason, we shall always assume without loss of generality, that input tree decompositions of our algorithms are nice. By V_t we denote vertices which appear in bags in the subtree rooted at the vertex corresponding to X_t and by $G[X_t]$ and $G[V_t]$ we denote the subgraph induced by vertices in bag X_t and vertices V_t respectively.

Doubling dimension. We recall the definition of doubling dimension.

Definition 2 ([GKL03])**.** *The* doubling constant *of a metric space* (X, dist) *is the smallest value* λ *such that every ball in* X *can be covered by* λ *balls of half the radius. The* doubling dimension *of* X *is then defined as* $\mathrm{dd}(X) = \log_2 \lambda$.

Highway dimension We dedicate this section to various definitions of highway dimension, relations between the definitions and other graph parameters. The notion of highway dimension is motivated by the observation of Bast et al. [BFM06, BFM+07] that in road networks, all shortest paths leaving a certain region pass through one of a small number of vertices. There are multiple definitions capturing the notion of highway dimension, we list them all for completeness.

The first definition of highway dimension can be found in [AFGW10]. We recall the definition from the introduction.

Definition 1 ([FFKP18])**.** *Let $G = (V, E)$ be an undirected graph with edge lengths $d : E \to \mathbb{R}_0^+$. The* highway dimension *of G is the smallest integer h such that, for some universal constant $\gamma \geq 4$, for any radius $r \in \mathbb{R}^+$, and any vertex u there is a hitting set $Z \subseteq B_u(\gamma r)$ of size h for the set of all shortest paths π satisfying $d(\pi) > r$ and $\pi \subseteq B_u(\gamma r)$.*

The second definition of highway dimension, which uses a closely related notion of *shortest path covers*, appears in [ADF+11], we give a slightly more compact formulation from [Fel19].

Definition 8 ([ADF+11, Fel19])**.** *Let $G = (V, E)$ be an undirected graph with edge lengths $d : E \to \mathbb{R}_0^+$. For a constant* scale *$r \in \mathbb{R}^+$, let $\mathcal{P}_{(r,2r]} \subseteq 2^V$ contain all vertex sets given by shortest paths in G whose lengths lie in the interval $(r, 2r]$. A* shortest path cover *$\textsc{spc}(r) \subseteq V$ is a hitting set for the set system $\mathcal{P}_{(r,2r]}$, i.e. for each $\pi \in \mathcal{P}_{(r,2r]}$ we have $\pi \cap \textsc{spc}(r) \neq \emptyset$. We call the vertices in $\textsc{spc}(r)$* hubs*. A hub set $\textsc{spc}(r)$ is called* locally h-sparse*, if for every vertex $v \in V$ the ball $B_v(2r)$ of radius $2r$ around v contains at most h hubs. The* highway dimension *of G is the smallest integer h such that there is a locally h-sparse shortest path cover $\textsc{spc}(r)$ for every scale $r \in \mathbb{R}^+$ in G.*

The third definition of highway dimension, which also appears in [ADF+16], uses a notion of *r-significant shortest paths*.

Definition 9 ([ADF+16])**.** *Given a shortest path $\pi = (v_1, \ldots, v_k)$ and $r > 0$, an* r-witness *path π' is a shortest path with length more than r such that π' can be obtained from π by adding at most one vertex to each end. That is, either $\pi' = \pi$, or $\pi' = (v_0, v_1, \ldots, v_k)$, or $\pi' = (v_1, \ldots, v_k, v_{k+1})$, or $\pi' = (v_0, v_1, \ldots, v_k, v_{k+1})$. If π has an r-witness path π' it is said to be* r-significant *and π is* (r, d)-close *to a vertex v if $\operatorname{dist}(\pi', v) \leq d$. The* highway dimension *of a graph G is the smallest integer h such that for all $r > 0$ and $v \in V$, there is a hitting set of size at most h for the r-significant paths that are $(r, 2r)$-close to v.*

We denote highway dimensions according to Definitions 1, 8 and, 9 as hd_1, hd_2 and hd_3 respectively. Feldmann et al. [FFKP18] have shown that $\mathrm{hd}_1 \leq \mathrm{hd}_3(\mathrm{hd}_3 + 1)$. Blum [Blu19] has shown that

- $\mathrm{hd}_2 \leq \mathrm{hd}_1$ and $\mathrm{hd}_3 \leq \mathrm{hd}_2$,
- all stated definitions of highway dimension are incomparable to treewidth,
- hd_1 and hd_2 are incomparable to doubling dimension.

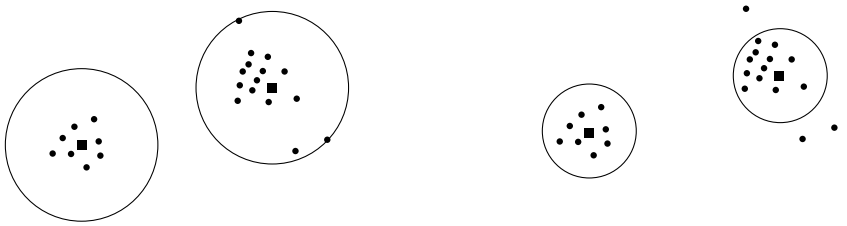
Feldmann et al. [FFKP18] have also shown that computing hd_1 and hd_2 is NP-hard and Blum [Blu19] has shown that computing hd_3 is NP-hard.

2. k-Center with Outliers

Let us recall the definition of the k-Center with Outliers (kCwO) problem. As input we receive a graph $G = (V, E)$ with positive edge lengths $d : E \to \mathbb{R}^+$, and integers $k \in \mathbb{N}$ and $p \in \mathbb{N}_0$. A feasible solution is a set of vertices $C \subseteq V$ of size $|C| \leq k$. The cost of a solution C is

$$\text{cost}(C) = \min_{\substack{V' \subseteq V \\ |V'| \geq n-p}} \max_{u \in V'} \text{dist}(u, C)$$

over all subsets $V' \subseteq V$ of size at least $n - p$. The objective is to find a feasible solution of minimum cost. Note that setting $p = 0$ yields the original k-center problem.



2-center solution 2-center solution with three outliers

Figure 2.1: An example to illustrate how allowing outliers may yield a solution of a lower cost. Dots and boxes are the points of the metric. Boxes are the selected centers and the circles centered at them depict the points they cover.

In some cases it is more convenient to work with the decision variant of kCwO. The input of the decision variant of kCwO has an additional integer ϱ and our goal is to decide whether there exists a feasible solution of cost at most ϱ. If we can solve the decision variant of kCwO, we can solve the optimization version as follows. By definition of the problem, the cost of the optimum solution is equal to the distance between some pair of vertices. Let $d_1 < d_2 < \cdots < d_\ell$ be all inter-vertex distances of G. We gradually apply the algorithm for the decision variant with costs $d_1, d_2, \ldots, d_\ell$ and the cost of the first YES instance is the cost of the optimum solution.

Hochbaum and Shmoys [HS86] have shown that for any $\varepsilon > 0$ it is NP-hard to $(2 - \varepsilon)$-approximate the k-Center problem. As k-Center with Outliers is a generalization of k-Center, the same hardness result applies. The hardness result for the kCwO problem is matched by 2-approximation algorithms for this problem: there are two different approaches by Chakrabarty et al. [CGK16] and Harris et al. [HPST19].

2.1 3/2-approximation algorithm for low highway dimension graphs

In this section we extend the FPT $\frac{3}{2}$-approximation algorithm for the k-Center problem given by Feldmann [Fel19] to obtain an FPT $\frac{3}{2}$-approximation algorithm

for kCwO. For this algorithm, we use Definition 1 of highway dimension. Let us recall the main result of this section

Theorem 3. *Let $\mathcal{I} = (G, k, p)$ be an instance of the k-Center with Outliers problem where G has highway dimension h. There exists an algorithm that outputs a solution of cost at most $\frac{3}{2}\mathrm{OPT}(\mathcal{I})$ in time $2^{O(k(h\log h)+p)} \cdot n^{O(1)}$.*

Feldmann [Fel19] observes that the vertices of low highway dimension graphs are highly structured for any scale r. To describe the structure, we need the following definition.

Definition 10 ([Fel19, Definition 3])**.** *Let $G = (V, E)$ be a graph, $r \in \mathbb{R}^+$ be a fixed constant, and $\textsc{spc}(r) \subseteq V$ be a fixed shortest path cover for scale r. We call an inclusion-wise maximal set $T \subseteq \{v \in V : \mathrm{dist}(v, \textsc{spc}(r)) > r\}$ with $\mathrm{dist}(u, w) \leq r$ for all $u, w \in T$ a* cluster*, and we denote the set of all clusters by $\mathcal{T}$. The* non-cluster *vertices are those which are not contained in any cluster of $\mathcal{T}$.*

Informally speaking, for a fixed scale $r \in \mathbb{R}^+$ we group vertices which are at distance more than r from their nearest hub into (inclusion-wise maximal) clusters of diameter at most r. Vertices which do not lie in clusters are at distance at most r from their nearest hub. Additionally he [Fel19] observes that clusters must be at distance more than $2r$ from each other. The following lemma formalizes the structure of cluster and non-cluster vertices, the proof of which can be found in [Fel19]. Note that the first two properties of the lemma follow directly from Definition 10. For an illustration, see Figure 2.2.

Lemma 11 ([Fel19, Lemma 4])**.** *Let $\mathcal{T}$ be the cluster set for a scale $r \in \mathbb{R}^+$ and a shortest path cover $\textsc{spc}(r)$. For each non-cluster vertex u, $\mathrm{dist}(u, \textsc{spc}(r)) \leq r$. The diameter of any cluster $T \in \mathcal{T}$ is at most r, and $\mathrm{dist}(T, T') > 2r$ for any distinct pair of clusters $T, T' \in \mathcal{T}$.*

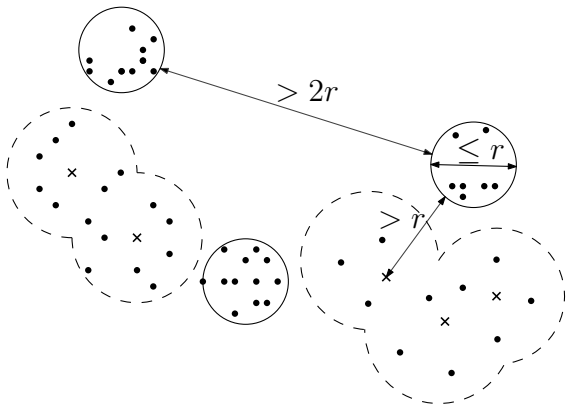


Figure 2.2: An example of a separation of a graph into clusters and non-clusters given by a shortest path cover for some scale r. Clusters (full circles) are far from each other, have a small diameter, and are far from hubs (crosses).

Overview of the original algorithm. Let us give an overview of the algorithm of Feldmann [Fel19]. Let $T \subseteq V$ be a cluster as defined in Definition 10. A set of vertices C *covers* T with balls of radius r if $T \subseteq \bigcup_{c \in C} B_c(r)$. Let C^* be an optimum solution of cost ϱ of a k-Center instance (G, k). First he defines $C_1^* \subseteq C^*$ to be the set of non-cluster centers, i.e. $C_1^* = C^* \cap (V \setminus \{v \in T : T \in \mathcal{T}\})$. The algorithm computes a $\frac{3}{2}$-approximate center set C_1 such that $\bigcup_{c \in C_1^*} B_c(\varrho) \subseteq \bigcup_{c \in C_1} B_c(\frac{3}{2}\varrho)$, that is, every vertex covered by C_1^* is covered by C_1. Using the computed set C_1, he defines $C_2^* \subseteq C^*$ to be the set of centers which lie in clusters and cover all clusters uncovered by the approximate center set C_1. Then the algorithm computes an approximate center set C_2 such that $\bigcup_{c \in C_2^*} B_c(\varrho) \subseteq \bigcup_{c \in C_2} B_c(\frac{3}{2}\varrho)$. Finally, the subset $C_3^* \subseteq C^*$ is the set of centers which lie in clusters and cover the remaining vertices uncovered by C_1 and C_2. For $i \in \{1, 2, 3\}$ we denote R_i^* the region covered by centers C_i^* by balls of radius ϱ, that is $R_i^* = \bigcup_{c \in C_i^*} B_c(\varrho)$. Analogously, we denote R_i the region covered by centers C_i by balls of radius $\frac{3}{2}\varrho$ where the $\frac{3}{2}$ factor comes from the approximation ratio. Using this notation, we have $R_1^* \subseteq R_1$ and $R_2^* \subseteq R_2$. Note that the sets C_1^*, C_2^*, and C_3^* are disjoint.

We briefly sketch how the approximate sets of centers C_i are computed. The algorithm starts by guessing the cost ϱ of the optimum solution by trying all inter-vertex distances in increasing order. Assume that the algorithm guesses the cost of the optimum solution and let $r = \frac{1}{2}\varrho$, $\textsc{spc}(r)$ be the shortest path cover for scale $\frac{1}{2}\varrho$, and s be the local sparsity of $\textsc{spc}(r)$. Since C_1^* are non-cluster vertices, from Lemma 11 for each $c \in C_1^*$ we have $\mathrm{dist}(c, \textsc{spc}(r)) \leq \frac{1}{2}\varrho$. The algorithm guesses a minimum sized subset of hubs $H \subseteq \textsc{spc}(r)$ such that $C_1^* \subseteq \bigcup_{h \in H} B_h(\frac{1}{2}\varrho)$ and sets $C_1 = H$. Observe that if the guess is correct, then we have $R_1^* \subseteq R_1$. To bound the running time incurred by guessing H, Feldmann [Fel19] shows the following lemma.

Lemma 12 ([Fel19, Lemma 5])**.** *Let ϱ be the optimum cost of the k-Center problem in a given instance G. If a shortest path cover $\textsc{spc}(\frac{1}{2}\varrho)$ of G for scale ϱ is locally s-sparse, then $\left|\textsc{spc}(\frac{1}{2})\right| \leq ks$.*

The algorithm skips the current guess of optimum cost if $|\textsc{spc}(r)| > ks$, where s is the local sparsity of $\textsc{spc}(r)$, in accordance with Lemma 12. Let $\mathcal{U}$ be the set of clusters which are not fully covered by R_1, that is $\mathcal{U} = \{T \in \mathcal{T} : T \setminus R_1 \neq \emptyset\}$. The next goal of the algorithm is to cover $\mathcal{U}$. Lemma 11 shows that the diameter of each cluster is at most $r = \frac{1}{2}\varrho$ and that the distance between any pair of clusters is more than $2r = \varrho$. The second property implies that to cover a cluster $T \in \mathcal{U}$ it must contain a center and the first property shows that picking any vertex of T as a center covers T entirely. As such the algorithm picks C_2 by selecting an arbitrary vertex in each cluster of $\mathcal{U}$. Let us remark that the existence of the optimum solution guarantees $|\mathcal{U}| \leq k$. Finally to cover the remaining vertices, the algorithm computes C_3 by solving a certain instance of Set Cover.

The following definition of the Set Cover problem appears in [WS11]. The input to the Set Cover problem is a *ground set* of elements $U = \{e_1, \ldots, e_n\}$ and some subsets of those elements $\mathcal{S} = \{S_1, \ldots, S_m\}$ where each $S_i \subseteq U$. The goal is to find a minimum sized collection of subsets $\mathcal{S}$ that covers U; that is, we wish to find a minimum sized set $I \subseteq \{1, \ldots, m\}$ such that $\bigcup_{i \in I} S_i \supseteq U$. It is

known that Set Cover can be solved in time $2^{|U|}(|U| + |\mathcal{S}|)^{O(1)}$. We will need a more concrete result for this problem, as such we state the theorem formally.

Theorem 13 ([CFK+15], [FKW04])**.** *Given a set system $(U, \mathcal{S})$ we can compute a table $\mathbb{T}$ which for any subset $U' \subseteq U$ contains the smallest set cover for $(U', \mathcal{S})$ in the entry $\mathbb{T}[U']$. For any subset $U' \subseteq U$, the optimum set cover for U' can be retrieved in constant time from $\mathbb{T}$ and $\mathbb{T}$ can be computed in time $2^{|U|}(|U|+|\mathcal{S}|)^{O(1)}$.*

The algorithm

Now we turn to describing our algorithm. Refer to Algorithm 1 for a description using pseudocode. Let (G, k, p) be an instance of the kCwO problem, and C^* be an optimum solution of cost ϱ which covers the subset of vertices $V^* \subseteq V$ of size at least $n - p$. For a fixed shortest path cover $\textsc{spc}(\frac{1}{2}\varrho)$, let $\mathcal{T}$ be the set of clusters induced by $\textsc{spc}(\frac{1}{2}\varrho)$. Let $\mathcal{T}^*$ be the set of clusters fully covered by C^*, i.e. $\mathcal{T}^* = \{T \in \mathcal{T} : T \setminus \bigcup_{c \in C^*} B_c(\varrho) = \emptyset\}$. Similarly to the analysis of the original algorithm, we will separate C^* into disjoint sets C_1^*, C_2^*, and C_3^* and we will also compute three sets of approximate centers C_1, C_2, and C_3. For $i \in \{1, 2, 3\}$ we denote R_i^* the region covered by balls of radius ϱ around centers in C_i^* and R_i the region covered by balls of radius $\frac{3}{2}\varrho$ around centers in C_i. The set C_1^* consists of non-cluster centers of C^*, that is $C_1^* = C^* \cap (V \setminus \{v \in T : T \in \mathcal{T}\})$. We will compute an approximate center set C_1 such that $R_1^* \subseteq R_1$. Then the set C_2^* is the set of centers which lie in clusters and cover those clusters of $\mathcal{T}^*$ that are not fully covered by C_1 and C_3^* is the set of remaining centers $C^* \setminus (C_1^* \cup C_2^*)$.

Similar to the k-Center algorithm, we will need to bound the size of the shortest path cover for scale $\frac{1}{2}\varrho$. For that purpose, we present the following lemma.

Lemma 14. *Let (G, k, p) be an instance of the kCwO problem and ϱ its optimum cost. If a shortest path cover $\textsc{spc}(\frac{1}{2}\varrho)$ of G is locally s-sparse, then $\left|\textsc{spc}(\frac{1}{2}\varrho)\right| \leq ks + p$.*

Proof. The optimum solution C^* covers the set of vertices $V^* \subseteq V$ of size at least $n - p$ by k balls of radius ϱ. The set V^* is given by $\bigcup_{c \in C^*} B_c(\varrho)$. By Definition 8 and the local sparsity of the shortest path cover, each ball of the optimum solution centered at a vertex of C^* can contain at most s hubs. This gives $\left|\textsc{spc}(\frac{1}{2}\varrho) \cap V^*\right| \leq ks$. All hubs which lie in V^* have been accounted for, it remains to bound the number of hubs in $V \setminus V^*$. Since V^* are vertices covered by the optimum solution, we must have $\left|\textsc{spc}(\frac{1}{2}\varrho) \cap (V \setminus V^*)\right| \leq |V \setminus V^*| \leq p$. In total we have $\left|\textsc{spc}(\frac{1}{2}\varrho)\right| = \left|\textsc{spc}(\frac{1}{2}\varrho) \cap V^*\right| + \left|\textsc{spc}(\frac{1}{2}\varrho) \cap (V \setminus V^*)\right| \leq ks + p$. □

Computing a shortest path cover. We guess the cost of the optimum solution by trying all $\binom{n}{2}$ inter-vertex distances in increasing order. For each guess ϱ' we set scale $r = \frac{1}{2}\varrho'$ and compute a shortest path cover $\textsc{spc}(r)$. Abraham et al. [ADF+11] have shown that given a graph of highway dimension h according to Definition 1, one can compute a locally $O(h \log h)$-sparse shortest path cover in polynomial time assuming there is a unique shortest path between each pair of vertices. Folklore results [ADF+11] show that this assumption can be

made without loss of generality, as one can perturb the input to ensure uniqueness. Therefore for $s = O(h \log h)$ we compute a locally s-sparse shortest path cover $\textsc{spc}(r)$ in polynomial time. To keep the running time of the algorithm low, we check that the size of the shortest path cover is not too large. In particular, by Lemma 14 if it is the case that $|\textsc{spc}(r)| > ks + p$, then we know that the current guess ϱ' cannot be the cost of the optimum solution and we proceed to the next guess. For the rest of this section, assume that we have found the optimum cost ϱ.

Computing C_1. Computing C_1 will be done identically to the original algorithm. Recall that C_1^* are non-cluster centers. By Lemma 11, for each center $c \in C_1^*$ there exists a hub $h \in \textsc{spc}(\frac{1}{2}\varrho)$ such that $\text{dist}(h, c) \leq \frac{1}{2}\varrho$. We guess a minimum sized subset of hubs $H \subseteq \textsc{spc}(\frac{1}{2}\varrho)$ such that $C_1^* \subseteq \bigcup_{h \in H} B_h(\frac{1}{2}\varrho)$ and choose H to be the first set of approximate centers C_1.

We have $|C_1| \leq |C_1^*|$ since we choose a minimum sized subset of hubs such that there is a hub $h \in H$ at most $\frac{1}{2}\varrho$ away from each center of C_1^*. To obtain $R_1^* \subseteq R_1$ we show that for each vertex $u \in R_1^*$ we also have $u \in R_1$. Consider an optimum center $c \in C_1^*$ such that $u \in B_c(\varrho)$. By our choice of H there exists a hub $h \in H$ such that $\text{dist}(h, c) \leq \frac{1}{2}\varrho$. Using triangle inequality we have $\text{dist}(h, u) \leq \text{dist}(h, c) + \text{dist}(c, u) \leq \frac{3}{2}\varrho$ and it follows that $R_1^* \subseteq R_1$.

Computing C_2. By the definition of C_1^* we know that centers $C^* \setminus C_1^*$ lie in clusters. At this point our algorithm will diverge from the original algorithm since in the k-Center problem all clusters must be covered, but this is not necessarily the case for kCwO as some cluster vertices may be outliers.

Let $\mathcal{U} = \{T \in \mathcal{T} : T \setminus R_1 \neq \emptyset\}$, that is the set of clusters uncovered by C_1, and similarly for the optimum solution let $\mathcal{U}^* = \{T \in \mathcal{T} : T \setminus R_1^* \neq \emptyset\}$. It follows from $R_1^* \subseteq R_1$ that $\mathcal{U} \subseteq \mathcal{U}^*$. Let us examine how the optimum solution C^* interacts with the uncovered clusters $\mathcal{U}^*$, and in particular how it covers clusters $\mathcal{T}^*$. From Lemma 11 we have that the distance between any pair of clusters is more than ϱ. Consider an optimum center $c \in C_2^*$ placed in a cluster $T \in \mathcal{T}$ and let $T' \in \mathcal{T}$ be another cluster distinct from T, then it follows from the lemma that $B_c(\varrho) \cap T' = \emptyset$. We can conclude that to cover clusters $\mathcal{T}^*$, the optimum solution opens at least one center in each cluster of $\mathcal{T}^*$. Lemma 11 also shows that the diameter of any cluster is at most $\frac{1}{2}\varrho$, therefore any cluster can be entirely covered by opening a single center inside it. On the contrary, if v is an outlier in the optimum solution and v lies in cluster $T' \in \mathcal{T}$, then T' cannot contain a center of the optimum solution. As a consequence, all vertices of T', aside from those already covered by C_1^*, are outliers as well. Clusters are non-empty by definition, hence we obtain $|\mathcal{T} \setminus \mathcal{T}^*| \leq p$, i.e., in the optimum solution at most p clusters can contain outliers. The optimum solution uses at most k centers, thus giving us $|\mathcal{U}| \leq |\mathcal{U}^*| \leq k + p$.

The algorithm first checks whether $|\mathcal{U}| \leq k + p$. If this is not the case, then we know that the preceding guess of set H is incorrect and we continue to the next one. The algorithm computes the set C_2 by guessing which clusters of $\mathcal{U}$ are covered in the optimum solution and picks an arbitrary vertex in each of the guessed clusters as the set C_2. If the number of cluster vertices we decide not to cover exceeds p, then we must reject the current guess. Note that the algorithm must cover all clusters $T \in \mathcal{T}$ which have more than p uncovered vertices by C_1,

i.e., if $|T \setminus R_1| > p$, otherwise the solution would contain more than p outliers. Let $T \in \mathcal{T}^*$ be a cluster covered by the optimum solution and $C' = C^* \cap T$, that is the set of optimum centers which lie in cluster T, then from the upper bound of $\frac{1}{2}\varrho$ on the diameter of a cluster it follows that $\bigcup_{u \in C'} B_u(\varrho) \subseteq B_v(\frac{3}{2}\varrho)$ for any vertex $v \in T$, hence we have $R_2^* \subseteq R_2$ and $|C_2| \leq |C_2^*|$.

Computing C_3. Let us summarize our progress so far and recall some important properties for computing C_3. In the previous steps we computed center sets C_1 and C_2. The region these approximate center sets cover is a superset of the region covered by their optimum counterparts, i.e., $R_1^* \subseteq R_1$ and $R_2^* \subseteq R_2$. These approximate centers also cover the set $\mathcal{T}^*$ of clusters covered by the optimum solution, hence we can conclude that the remaining vertices covered by the optimum solution $V^* \setminus (R_1 \cup R_2)$ are non-cluster vertices. Additionally, by the definition of C_1^* we know that C_3^* are cluster vertices. We will compute a set C_3 which covers the remaining vertices V^* and $|C_3| \leq |C_3^*|$. Note that since sets C_1^*, C_2^* and C_3^* are disjoint, the upper bounds on the sizes of the approximate center sets show that $|C_1 \cup C_2 \cup C_3| \leq k$.

Let $V' = V^* \setminus (R_1 \cup R_2)$. Our goal is to compute a set of centers C_3 such that $V' \subseteq \bigcup_{c \in C_3} B_c(\frac{3}{2}\varrho)$ and $|C_3| \leq |C_3^*|$. We guess the set of hubs $H' \subseteq (\textsc{spc}(\frac{1}{2}\varrho) \setminus H)$ that lie in region R_3^*. The set H is excluded because we are looking for hubs at distance at most $\frac{1}{2}\varrho$ from vertices in R_3^*, which are not yet covered by our solution $R_1 \cup R_2$, however, vertices close to H are already covered by C_1. Then we compute a center set C_3 of minimum size such that $H' \subseteq \bigcup_{c \in C_3} B_c(\varrho)$. We achieve this by reducing covering H' to the Set Cover problem with fixed universe size.

We construct the following Set Cover instance: the universe U consists of hubs $\textsc{spc}(\frac{1}{2}\varrho)$ and the set system is defined as $\mathcal{S} = \{B_v(\varrho) \cap \textsc{spc}(\frac{1}{2}\varrho) : (\exists T \in \mathcal{T})(v \in T)\}$, i.e., balls around cluster vertices of radius ϱ restricted to hubs. To optimize the running time of the algorithm, we can precompute the table $\mathbb{T}$ given by Theorem 13 before computing C_1 and in particular before guessing set H. As a result, we obtain a table $\mathbb{T}$ from which we can obtain an optimum center set for covering any subset of hubs by balls around cluster vertices. As the next lemma shows, we obtain the required properties for C_3. This lemma is identical to the one used by the original algorithm [Fel19]. Moreover, its proof applies for our problem without any modifications.

Lemma 15 ([Fel19, Lemma 7])**.** *Assume the algorithm guessed the cost of the optimum solution ϱ, the correct scale $r = \frac{1}{2}\varrho$, and the correct sets H and H'. Let $C_3 = \{v \in \bigcup_{T \in \mathcal{T}} T : B_v(\varrho) \cap \textsc{spc}(r) \in \mathbb{T}[H']\}$, that is vertices which induce the sets covering H'. Then it holds that $H' \subseteq \bigcup_{v \in C_3} B_v(\varrho)$ and $|C_3| \leq |C_3^*|$.*

It remains to show that our solution is a feasible solution to the given kCwO instance, which we show in the next lemma. In particular, we need to show that we cover at least $n - p$ vertices and to prove this it is sufficient to show that we cover all vertices V^*. Up to some technicalities, the proof of our lemma is identical to the one used in the original algorithm, cf. [Fel19, Lemma 8]. Nevertheless, since the proof of the lemma is one of the main technical difficulties of the algorithm, we present the entire proof.

Lemma 16 ([Fel19, cf. Lemma 8])**.** *Assume the algorithm guessed the correct scale $r = \frac{1}{2}\varrho$ and the correct sets H and H'. The approximate center sets C_1, C_2 and C_3 cover vertices V^*, i.e., $(R_1 \cup R_2 \cup R_3) \supseteq V^*$.*

Proof. We prove the lemma by contradiction. Refer to Figure 2.3 for an illustration of some bounds on the distances we will derive in this proof. Recall that V^* is the set of vertices covered by the optimum solution C^*. Assume there exists a vertex $v \in V^*$ which is not covered by our solution, i.e. $v \notin R_1 \cup R_2 \cup R_3$. We have already argued that all clusters $\mathcal{T}^*$ are covered by centers C_1 and C_2, thus v is a non-cluster vertex. We will identify a hub $y \in \textsc{spc}(\frac{1}{2}\varrho)$ on the shortest path between v and an optimum center $c \in C^*$ which covers v. We will show that this hub lies in H', which in turn means that v lies in R_3, since v will turn out to be close to y.

To show the existence of y, we begin by arguing that the closest hub $x \in \textsc{spc}(\frac{1}{2}\varrho)$ to v is neither in H or H'. Lemma 11 shows that $\operatorname{dist}(x, v) \leq \frac{1}{2}\varrho$, hence $x \in H$ would imply that v is be covered by our solution, since H is precisely the first set of approximate centers C_1. Suppose that $x \in H'$. By Lemma 15 there exists a center $c \in C_3$ such that $\operatorname{dist}(c, x) \leq \varrho$. Using triangle inequality we have $\operatorname{dist}(c, v) \leq \frac{3}{2}\varrho$. Thus $x \notin H \cup H'$.

From the assumption $v \in V^*$, there exists an optimum center $w \in C^*$ such that $v \in B_w(\varrho)$. Optimum center w must be an element of center set C_3^* since $v \notin R_1 \cup R_2$ and we have $R_1^* \subseteq R_1$ and $R_2^* \subseteq R_2$. We defined H' as the set of hubs that are at distance at most ϱ from the set C_3^*, and so that $H \cap H' = \emptyset$. Since $x \notin H \cup H'$, we have $\operatorname{dist}(x, w) > \varrho$. From the definition of non-cluster vertices and from x being the closest hub to v, we have $\operatorname{dist}(x, v) \leq \frac{1}{2}\varrho$ and $\operatorname{dist}(v, w) > \frac{1}{2}\varrho$. In the optimum solution, center w covers v, hence we get $\operatorname{dist}(v, w) \leq \varrho$. Putting these bounds together, we have $\frac{1}{2}\varrho < \operatorname{dist}(v, w) \leq \varrho$ which implies that there must exist a hub $y \in \textsc{spc}(\frac{1}{2}\varrho)$ which hits the shortest path between v and w. From the upper bound $\operatorname{dist}(v, w) \leq \varrho$ we obtain $\operatorname{dist}(v, y) \leq \varrho$ and $\operatorname{dist}(y, w) \leq \varrho$. We assume that vertex v is not covered by our solution, then we have $y \notin H$, otherwise the bound $\operatorname{dist}(v, y) \leq \varrho$ would imply that $v \in R_1$. However, hub y not being an element of H combined with the bound $\operatorname{dist}(y, w) \leq \varrho$ and $w \in C_3^*$ shows that $y \in H'$.

Since w is a cluster vertex, we have $\operatorname{dist}(y, w) > \frac{1}{2}\varrho$ by Lemma 11. This implies $\operatorname{dist}(v, y) < \frac{1}{2}\varrho$. However, this means that $v \in B_y(\frac{1}{2}\varrho)$ which gives $v \in R_3$ since $y \in H'$ and in particular $\operatorname{dist}(y, C_3) \leq \varrho$. This contradicts the assumption that v was not covered by the approximate center set. □

To check if the solution we compute is feasible, it must hold that $n - p \leq |R_1 \cup R_2 \cup R_3|$. Lemma 16 proves correctness of the algorithm, to prove Theorem 3, it remains to bound its running time.

Proof of Theorem 3. The correctness of Algorithm 1 is proved by Lemma 16, it remains to bound the running time of the algorithm. It is clear that the preprocessing outside the main loop requires polynomial time. The first nontrivial step of the algorithm happens on line 5. Abraham et al. [ADF+11] have shown that one can compute a locally $O(h \log h)$-sparse shortest path cover for graphs of highway dimension h in polynomial time assuming there is a unique shortest path between every pair of vertices. Filling table $\mathbb{T}$ on line 10 requires time $O^*(2^{ks+p})$

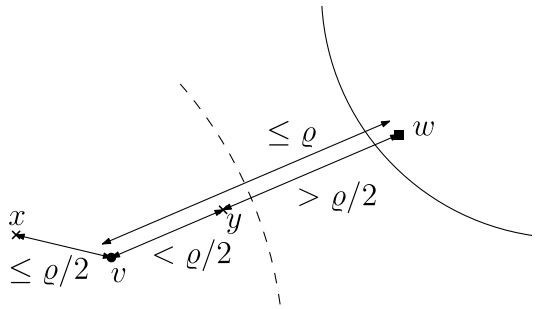


Figure 2.3: The situation in the proof of Lemma 16. Vertex w lies in a cluster, vertex v and hubs x, y are non-cluster vertices. The figure shows some of the bounds on the distances between vertices in the figure.

by Theorem 13 and the same theorem shows that retrieving a solution on line 19 requires constant time. Lemma 14 and the check on line 6 guarantees that shortest path cover $\textsc{spc}(r)$ has size at most $ks + p$. On lines 11 and 18 we guess hub sets H and H' respectively. Since H' is disjoint from H by definition, sets H and H' are disjoint subsets of $\textsc{spc}(r)$. Thus guessing the sets H and H' takes time $O^*(3^{ks+p})$, each hub is either in H or in H' or neither of them. The check on line 15 guarantees that guessing $\mathcal{U}'$ takes time $O^*(2^{k+p})$. The remaining steps of the algorithm can all be performed in polynomial time. In total, the algorithm takes time $O^*(2^{O(ks+p)})$ where $s = O(h \log h)$, hence the claimed running time follows. □

Algorithm 1: fixed parameterized $\frac{3}{2}$-approximation algorithm for κCwO in low highway dimension graphs

Input: Graph G of highway dimension h with optimum κCwO cost ϱ
Output: a κCwO solution C of cost at most $\frac{3}{2}\varrho$

$s \leftarrow O(h \log h)$ `// local sparsity of a polynomially computable shortest path cover`
$\mathbb{A} \leftarrow \texttt{sort}(\{\text{dist}(u,v) : u, v \in V\})$ `// sorted inter-vertex distances in non-decreasing order`
for $i \leftarrow 0$ **to** $\binom{n}{2} - 1$ **do**
  $r \leftarrow \frac{1}{2}\mathbb{A}[i]$
  compute a locally s-sparse $\text{SPC}(r)$ and obtain a cluster set $\mathcal{T}$
  if $|\text{SPC}(r)| > ks + p$ **then** `// too many hubs means` $r \neq \frac{1}{2}\varrho$
    continue
  $V(\mathcal{T}) \leftarrow \bigcup_{T \in \mathcal{T}} T$
  $\mathcal{S} \leftarrow \bigcup_{v \in V(\mathcal{T})} \{B_v(\varrho) \cap \text{SPC}(r)\}$ `// the set system given by hubs in balls of radius` ϱ `around cluster vertices`
  $\mathbb{T} \leftarrow \texttt{SetCoverDP}(\text{SPC}(r), \mathcal{S})$ `// lookup table` $\mathbb{T}$ `contains an optimum set cover for each subset of the universe` $\text{SPC}(r)$
  `// guess the minimum sized set of hubs covering non-cluster centers`
  foreach $H \subseteq \text{SPC}(r)$ **do**
    $C_1 \leftarrow H$
    $R_1 \leftarrow \bigcup_{c \in C_1} B_c(3r)$
    $\mathcal{U} \leftarrow \{T \in \mathcal{T} : T \setminus R_1 \neq \emptyset\}$
    if $|\mathcal{U}| > k + p$ **then continue** `// too many uncovered clusters means` $r \neq \frac{1}{2}\varrho$
    `// guess which clusters of` $\mathcal{U}$ `are covered by` C^*
    foreach $\mathcal{U}' \subseteq \mathcal{U}$ **do**
      $C_2 \leftarrow$ an arbitrary vertex from each cluster in $\mathcal{U}'$
      `// cover the remaining vertices of` V^* `by reducing to` Set Cover
      foreach $H' \subseteq (\text{SPC}(r) \setminus H)$ **do**
        $C_3 \leftarrow \{v \in V(\mathcal{T}) : B_v(2r) \cap \text{SPC}(r) \in \mathbb{T}[H']\}$
        $C \leftarrow C_1 \cup C_2 \cup C_3$
        $R \leftarrow \bigcup_{c \in C} B_c(3r)$
        `// check if the current solution is feasible`
        if $|C| \leq k$ **and** $|R| \geq n - p$ **then return** C

3. k-Supplier with Outliers

Let us start by recalling the definition of the k-Supplier with Outliers (kSwO) problem. The input is specified by a graph $G = (V, E)$ with positive edge lengths $d : E \to \mathbb{R}^+$, a set of *suppliers* $V_s \subseteq V$, a set of *clients* $V_c \subseteq V$, and integers $k \in \mathbb{N}$ and $p \in \mathbb{N}_0$. A feasible solution is a subset of suppliers of size at most k. The cost of a feasible solution S is

$$\operatorname{cost}(S) = \min_{\substack{V'_c \subseteq V_c \\ |V'_c| \geq n-p}} \max_{u \in V'_c} \operatorname{dist}(u, S)$$

where V'_c are subsets of clients of size at least $n - p$. Clients $V_c \setminus V'_c$ that are not covered by the solution are called *outliers*. The goal is to find a feasible solution of minimum cost.

To simplify algorithm descriptions, we will require that $V_s \cap V_c = \emptyset$. In our setting, vertices which are neither clients nor customers do not affect the cost of a solution. Their sole purpose is to enforce additional structure on the input graph, i.e., once we compute the shortest-path metric of the entire graph, we can discard them and solve the problem on the remaining metric given by the clients and the suppliers. Thus in our algorithms, we will tacitly assume that every vertex is either a client or a supplier.

In some cases it might be more convenient to work with the decision version of the k-Supplier problem. In such cases, the input contains an additional value $\varrho \in \mathbb{R}_0^+$ and our goal is to decide whether there exists a feasible solution of cost at most ϱ. From the definition of the problem it is immediate that the cost of the optimum solution must be one of the distances between a client and a supplier. Thus the same approach of using an algorithm for the decision variant of the kCwO problem to solve the optimization variant of the kCwO problem, which we described in the previous chapter, can be used for the k-Supplier problem.

Hochbaum and Shmoys [HS86] have shown that the k-Supplier problem cannot be $(3-\varepsilon)$-approximated for any $\varepsilon > 0$ and they have also shown a matching 3-approximation algorithm in the same work. As the k-Supplier problem is a generalization of the k-Center problem, all parameterized hardness results for k-Center apply for k-Supplier as well.

3.1 EPAS for low treewidth graphs

In this section we present an FPT approximation scheme for the kSwO problem on low treewidth graphs. To simplify the exposition, we restrict ourselves to instances with integral edge lengths. The properties of the algorithm are summarized by the following theorem.

Theorem 17. *Let $\mathcal{I} = (G, k, p)$ be an instance of the kSwO problem, $\mathcal{T} = (T, \{X_i\}_{i \in V(T)})$ a nice tree decomposition of width* tw *of G, ϱ a integral cost, and $\varepsilon > 0$. There exists an algorithm which either*

- *returns a solution of cost $(1+\varepsilon)\varrho$ of the instance $\mathcal{I}$ if there exists a feasible solution of cost ϱ, or*

- *correctly determines that there is no feasible solution of cost ϱ,*

running in time $O^*\left((\frac{\mathrm{tw}}{\varepsilon})^{O(\mathrm{tw})}\right)$.

Our algorithm extends the FPT approximation scheme for the k-Center problem by Katsikarelis et al. [KLP19]. They use a technique introduced by Lampis [Lam14] to approximate problems which are W[1]-hard parameterized by treewidth. He [Lam14] observes that the hardness of such problems "intuitively stems from the fact that large *integers* need to be stored in the dynamic programming table." In the case of the algorithm of Katsikarelis et al. [KLP19], these are distances of vertices to their nearest opened center. In our case, these will be distances of vertices to their nearest opened supplier. The idea of the technique of Lampis [Lam14] is the following: Instead of storing integers in the dynamic programming table, we fix a parameter $\delta > 0$ and represent all integers in $\{1, \ldots, r\}$ by rounding them to the closest integer power of $(1 + \delta)$. It is shown in [Lam14] that for an appropriate selection of δ, the size of the dynamic programming table is reduced from r^{tw} to $(\log r)^{O(\mathrm{tw})}$. He [Lam14] then points out that "during the process of running a dynamic programming algorithm on the approximate values the rounding errors will propagate and potentially pile up to a large error." In the analysis of the original algorithm of Katsikarelis et al. [KLP19], it is shown that these errors in a dynamic programming table can be bounded by a function only in the height of the node of the bag corresponding to the table. We will extend this analysis to our problem as well. In the original algorithm [KLP19], using a result of Bodlaender and Hagerup [BH98], the provided tree decomposition is balanced so that its height is $O(\log |V(T)|)$ while increasing the width only by a constant factor.

We start by describing an exact FPT algorithm for the kSwO problem parameterized by treewidth of the input graph. This algorithm is an extension of an exact FPT algorithm for the k-Center problem [KLP19, Theorem 27]. We will later transform this exact algorithm into an approximation scheme with running time independent from ϱ. The properties of the exact algorithm are summarized by the following theorem.

Theorem 18. *There exists an algorithm which for a given kSwO instance* $\mathcal{I} = (G, k, p)$ *such that graph* G *has treewidth* tw *and cost* $\varrho \in \mathbb{N}$ *decides whether* $\mathcal{I}$ *has a feasible solution of cost* ϱ *running in time* $O^*\left(\varrho^{O(\mathrm{tw})}\right)$.

We give an equivalent formulation of the kSwO problem which is more convenient for our purposes. Let (G, k, p) be an instance of the kSwO problem where $G = (V, E)$ has edge lengths $d : E \to \mathbb{N}$ and vertices are partitioned into clients V_{c} and suppliers V_{s}. A *distance labelling* function of G is a function $\mathrm{dl} : V \to \{0, \ldots, \varrho\} \cup \{\infty\}$. Only a supplier can have label 0. We say that a vertex $u \in V$ is *satisfied* by dl if $\mathrm{dl}(u) = 0$ or u has a finite label and there exists a neighbour $v \in N(u)$ such that $\mathrm{dl}(u) \geq \mathrm{dl}(v) + d(u, v)$. Given a distance labelling function dl, if every client is either satisfied or has label ∞, then we say that dl is *valid*. We define the *cost* of a distance labelling function dl as $\left|\mathrm{dl}^{-1}(0)\right|$ and the *penalty* as $\left|\mathrm{dl}^{-1}(\infty) \cap V_{\mathrm{c}}\right|$. The following lemma shows the equivalence between the two formulations. A special case of this statement for the k-Center problem is proved in the original algorithm, cf. [KLP19, Lemma 26].

Lemma 19. *A* κSwO *instance* $\mathcal{I} = (G, k, p)$ *admits a feasible solution of cost* ϱ *if and only if it admits a valid distance labelling function* $\mathrm{dl} : V \to \{0, \ldots, \varrho\} \cup \{\infty\}$ *of cost* k *and penalty* p*.*

Proof. Let $S \subseteq V_\mathrm{s}$ be a solution of $\mathcal{I}$ of cost ϱ. We construct the required distance labelling function dl as follows:

1. for each vertex $u \in V$ with $\mathrm{dist}(u, S) \leq \varrho$ we set $\mathrm{dl}(u) = \mathrm{dist}(u, S)$,
2. we set the labels of the remaining vertices to ∞.

It is immediate that the cost of dl is at most k. From the definition of the κSwO problem it is also clear that outliers are clients whose distance from S exceeds ϱ, therefore the penalty of dl is at most p. Consider a vertex $u \in V$ and a shortest path π of length at most ϱ between u and its nearest opened supplier $s \in S$. Let v be the neighbour of u on path π. We have $\mathrm{dist}(u, s) = d(u, v) + \mathrm{dist}(v, s)$ and $\mathrm{dl}(u) = \mathrm{dist}(u, s)$ and $\mathrm{dl}(v) = \mathrm{dist}(v, s)$, this shows that every vertex and particularly every client with a finite label is satisfied.

For the opposite direction, let dl be a distance labelling function with properties given by the lemma statement. We select vertices with label 0 as the solution S. By definition of a distance labelling function these can only be suppliers. It suffices to show that the distance of vertices with finite labels from S is at most the value of their label. This is the case because a valid distance labelling function assigns finite labels to all but (up to) p vertices and the maximum finite label is ϱ. We prove this claim by induction on label values. The base case is immediate since vertices with label 0 are precisely the solution S. In the induction step consider a vertex u with a finite label ℓ. Since u is a satisfied vertex, there exists a neighbour $v \in N(u)$ such that $\mathrm{dl}(u) \geq \mathrm{dl}(v) + d(u, v)$. As edge lengths are positive, we have $\mathrm{dl}(u) > \mathrm{dl}(v)$ and using triangle inequality we obtain

$$\mathrm{dist}(u, S) \leq \mathrm{dist}(u, v) + \mathrm{dist}(v, S) \leq \mathrm{dist}(u, v) + \mathrm{dl}(v) \leq \mathrm{dl}(u)$$

where the second inequality follows from the induction hypothesis $\mathrm{dist}(v, S) \leq \mathrm{dl}(v)$. This shows that clients with finite labels are covered by the solution S and since the penalty of dl is at most p, there are at most p outliers. Also the cost of dl is at most k thus we also have $|S| \leq k$. □

According to the proven equivalence, we may refer to a client with label ∞ as an outlier and a supplier with label 0 as an opened supplier.

We are ready to prove Theorem 18. The algorithm will be a standard dynamic programming procedure on a nice tree decomposition of the input graph. As is customary, let us assume that a nice tree decomposition of the input graph G of width $\mathrm{tw}(G)$ is given as a part of the input.

Proof of Theorem 18. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a nice tree decomposition of the input graph G of width $\mathrm{tw}(G)$. For every node of $i \in V(\mathcal{T})$ we maintain a dynamic programming table

$$D_i : \Big((X_i \to \{0, \ldots, \varrho\} \cup \{\infty\}) \times 2^{X_i \cap V_\mathrm{c}} \times \{0, \ldots, k\} \times \{0, \ldots, p\}\Big) \to \{0, 1\}.$$

We may refer to the value 0 in the dynamic programming table as *false* and to the value 1 as *true*. Let i be a node of a nice tree decomposition, recall that V_i is

the set of vertices $u \in V(G)$ such that there exists a bag corresponding to a node in the subtree rooted at i which contains u. For a distance labelling function dl of $X_i, i \in V(T)$, a subset of clients $S \subseteq X_i \cap V_c$ and constants k_i and p_i the entry $D_i[\mathrm{dl}, S, k_i, p_i]$ is going to be 1 if and only if there exists a distance labelling function dl^* of $G[V_i]$ such that:

- dl^* *agrees* with dl on X_i, i.e. $(\forall u \in X_i)(\mathrm{dl}(u) = \mathrm{dl}^*(u))$,
- the cost of dl^* is k_i, that is $\left|\mathrm{dl}^{*^{-1}}(0)\right| = k_i$,
- the penalty of dl^* is p_i, that is $\left|\mathrm{dl}^{*^{-1}}(\infty) \cap V_c\right| = p_i$,
- aside from clients $\mathrm{dl}^{*^{-1}}(\infty) \cap V_c$ with an infinite label, every client $((V_i \setminus X_i) \cup S) \cap V_c$ is satisfied.

In combination with Lemma 19, the input admits a feasible solution of cost ϱ if there exists a distance labelling dl_r of the root node r, and constants $k_r \leq k$ and $p_r \leq p$ such that the entry $D_r[\mathrm{dl}_r, X_r \cap V_c, k_r, p_r]$ has value 1.

To simplify the exposition, assume that every value of the dynamic programming table at each node is initialized to value 0.

Leaf node. For a leaf node ℓ we have $V_\ell = \emptyset$. Thus the only table entry which can have value 1 is $D_\ell[\mathrm{dl}, \emptyset, 0, 0]$ where the domain of dl is an empty set.

Introduce node. Let i be an introduce node with a child node j where $X_i = X_j \cup \{u\}$ and $u \notin X_j$. Let $D_j[\mathrm{dl}_j, S_j, k_j, p_j]$ be an entry of the table of the child node j with value 1. We construct a distance labelling function dl_i which agrees with dl_j on X_j and tries all possible values for u. In particular the constructed distance labelling functions set $\mathrm{dl}_i(u)$ to values $\{1, \ldots, \varrho\} \cup \{\infty\}$ and additionally we try the label 0 for u if u is a supplier. For such a distance labelling function dl_i we compute S_i to be the set of satisfied clients of X_i as follows. We add the entire set S_j to S_i, we add u to S_i if there exists a neighbour $v \in N(u)$ so that $\mathrm{dl}_i(u) \geq \mathrm{dl}_i(v) + d(u, v)$, and we add neighbours $w \in N(u)$ to S_i for which it holds that $\mathrm{dl}_i(w) \geq \mathrm{dl}_i(u) + d(u, w)$. If $\mathrm{dl}_i(u) = 0$ and $k_j \leq k - 1$, then we set the entry $D_i[\mathrm{dl}_i, S_i, k_j + 1, p_j]$ to true. If $\mathrm{dl}_i(u) \in \{1, \ldots, \varrho\}$, then we set the entry $D_i[\mathrm{dl}_i, S_i, k_j, p_j]$ to true. If $\mathrm{dl}_i(u) = \infty$, $u \in V_c$, and $p_j \leq p - 1$, then we set the entry $D_i[\mathrm{dl}_i, S_i, k_j, p_j + 1]$ to true. If $\mathrm{dl}_i(u) = \infty$ and $u \in V_s$, then we set the entry $D_i[\mathrm{dl}_i, S_i, k_j, p_j]$ to true.

We proceed to showing the correctness. Let $\widehat{\mathrm{dl}}$ be a distance labelling function of $G[V_i]$ with cost $\widehat{k}$ and penalty $\widehat{p}$ such that all clients of $V_i \setminus X_i$ with a finite label are satisfied. We denote by $\widehat{S}$ the set of clients in V_i satisfied by $\widehat{\mathrm{dl}}$. We want to show that the algorithm sets the table entry $D_i[\mathrm{dl}_i, S_i, k_i, p_i]$ to 1 where dl_i agrees with $\widehat{\mathrm{dl}}$ on X_i, $S_i = \widehat{S} \cap X_i$, $k_i = \widehat{k}$, and $p_i = \widehat{p}$. By the induction hypothesis this property holds in the child node, thus there exists an entry $D_j[\mathrm{dl}_j, S_j, k_j, p_j]$ with value 1 where dl_j agrees with $\widehat{\mathrm{dl}}$ on X_j, $S_j = \widehat{S} \cap X_j$, and the cost and the penalty of $\widehat{\mathrm{dl}}$ restricted to V_j is k_j and p_j respectively. The algorithm considers a distance labelling function dl_i which agrees with dl_j on X_j and sets $\mathrm{dl}_i(u) = \widehat{\mathrm{dl}}(u)$. We need to show that the algorithm computes the set S_i correctly. Recall that by property (T3) of tree decompositions we have $u \notin V_j$. If a client of $X_i \setminus N[u]$

is satisfied by dl_j, then it is also satisfied by dl_i since the label of the neighbour which satisfies it remains the same. Hence we have $S_j \subseteq S_i$ and $S_i \setminus S_j \subseteq N[u]$. The algorithm handles this by adding the entire set S_j to S_i. If $u \in \widehat{S}$, then there exists a neighbour $v \in N(u) \cap V_i$ such that $\widehat{\mathrm{dl}}(u) \geq \widehat{\mathrm{dl}}(v) + d(u, v)$. In particular this means that there exists a node in a subtree rooted at i whose bag contains v. We want to show that $v \in X_i$. For a vertex $a \in V(G)$, let T_a be the set of nodes of the tree decomposition $\mathcal{T}$ whose bags contain a. For a subset $U \subseteq V(T)$, where $V(T)$ are nodes of the tree decomposition, we denote $\chi(U) = \{X_w : w \in U\}$. By property (T2) of tree decompositions, there must exist a node of $\mathcal{T}$ whose corresponding bag contains both u and v. Thus $\chi(T_u) \cap \chi(T_v) \neq \emptyset$. If it were the case that $v \notin X_i$, then a tree decomposition satisfying properties $u \in X_i$, $u \notin X_j$, $v \in V_i$, and that there exists a node whose bag contains both u and v necessarily violates property (T3). We conclude that $v \in X_i$. Since $\widehat{\mathrm{dl}}$ agrees with dl_i, the algorithm will find v in X_i and add u to S_i. For a client $w \in N(u) \cap X_i$, we have $(N(w) \cap X_i) \setminus (N(w) \cap X_j) = \{u\}$, since i is an introduce node introducing u. Therefore the only reason why w would be satisfied by dl_i but not by dl_j is that it is satisfied by u. Then the set S_j is the set S_i without clients in X_i which are satisfied only by u. The algorithm adds to S_i those vertices of $N(u) \cap X_i$ which are satisfied by u. It remains to describe values k_i and p_i, we distinguish the following cases:

- **Case** $\widehat{\mathrm{dl}}(u) = 0$**.** As $u \notin V_j$, we have $\left|\widehat{\mathrm{dl}}^{-1}(0) \cap V_i\right| = \left|\widehat{\mathrm{dl}}^{-1}(0) \cap V_j\right| + 1$. Since $\widehat{\mathrm{dl}}(u) \neq \infty$, we have $\left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_i \cap V_\mathrm{c}\right| = \left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_j \cap V_\mathrm{c}\right|$. Thus the algorithm sets $k_i = k_j + 1$ and $p_i = p_j$.
- **Case** $\widehat{\mathrm{dl}}(u) \in \{1, \ldots, \varrho\}$**.** Since $\widehat{\mathrm{dl}}(u) \notin \{0, \infty\}$, we have $\left|\widehat{\mathrm{dl}}^{-1}(0) \cap V_i\right| = \left|\widehat{\mathrm{dl}}^{-1}(0) \cap V_j\right|$ and $\left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_i \cap V_\mathrm{c}\right| = \left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_j \cap V_\mathrm{c}\right|$, hence $k_i = k_j$ and $p_i = p_j$ respectively.
- **Case** $\widehat{\mathrm{dl}}(u) = \infty$**.** We have $\left|\widehat{\mathrm{dl}}^{-1}(0) \cap V_i\right| = \left|\widehat{\mathrm{dl}}^{-1}(0) \cap V_j\right|$ and therefore $k_i = k_j$. If $u \in V_\mathrm{c}$, then $\left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_i \cap V_\mathrm{c}\right| = \left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_j \cap V_\mathrm{c}\right| + 1$ and $p_i = p_j + 1$. Otherwise $u \in V_\mathrm{s}$ and then we have $p_i = p_j$.

For the opposite direction, assume that the algorithm sets the value of an entry $D_i[\mathrm{dl}_i, S_i, k_i, p_i]$ to 1. By the description of the algorithm, this means that there exists an entry $D_j[\mathrm{dl}_j, S_j, k_j, p_j]$ in the table of the child node set to 1 where dl_i agrees with dl_j on X_j, $S_j \subseteq S_i$, $k_j \leq k_i$, and $p_j \leq p_i$. By the induction hypothesis there exists a valid distance labelling function $\widehat{\mathrm{dl}}_j$ on V_j which agrees with dl_j on X_j, satisfies clients S_j of X_j, and has cost k_j and penalty p_j. We claim that the function $\widehat{\mathrm{dl}}_i$ which extends $\widehat{\mathrm{dl}}_j$ by setting $\widehat{\mathrm{dl}}_i(u) = \mathrm{dl}_i(u)$ has the required properties. We need to show that all clients of S_i are satisfied by $\widehat{\mathrm{dl}}_i$. Every satisfied client of S_j is still satisfied by $\widehat{\mathrm{dl}}_i$ since the distance label of the neighbour which satisfies it is preserved. From the description of the algorithm it is clear that the added clients $S_i \setminus S_j$ are satisfied by $\widehat{\mathrm{dl}}_i$. It remains to verify that the algorithm computes the cost of $\widehat{\mathrm{dl}}_i$ correctly. Recall that $V_i \setminus V_j = \{u\}$. We distinguish the following cases based on the description of the algorithm:

- **Case** $\mathrm{dl}_i(u) = 0$**.** In this case $\widehat{\mathrm{dl}}_i^{-1}(0) = \widehat{\mathrm{dl}}_j^{-1}(0) \cup \{u\}$ and $\widehat{\mathrm{dl}}_i^{-1}(\infty) \cap V_{\mathrm{c}} = \widehat{\mathrm{dl}}_j^{-1}(\infty) \cap V_{\mathrm{c}}$. Then the cost of $\widehat{\mathrm{dl}}_i$ is $k_j + 1$ and the penalty is p_j.

- **Cases** $\mathrm{dl}_i(u) \in \{1, \ldots, \varrho\}$ **and** $\mathrm{dl}_i(u) = \infty \wedge u \in V_{\mathrm{s}}$**.** In this case $\widehat{\mathrm{dl}}_i^{-1}(0) = \widehat{\mathrm{dl}}_j^{-1}(0)$ and $\widehat{\mathrm{dl}}_i^{-1}(\infty) \cap V_{\mathrm{c}} = \widehat{\mathrm{dl}}_j^{-1}(\infty) \cap V_{\mathrm{c}}$. Then the cost of $\widehat{\mathrm{dl}}_i$ is k_j and the penalty is p_j.

- **Case** $\mathrm{dl}_i(u) = \infty \wedge u \in V_{\mathrm{c}}$**.** In this case $\widehat{\mathrm{dl}}_i^{-1}(0) = \widehat{\mathrm{dl}}_j^{-1}(0)$ and $\widehat{\mathrm{dl}}_i^{-1}(\infty) \cap V_{\mathrm{c}} = \left(\widehat{\mathrm{dl}}_j^{-1}(\infty) \cap V_{\mathrm{c}}\right) \cup \{u\}$. Then the cost of $\widehat{\mathrm{dl}}_i$ is k_j and the penalty is $p_j + 1$.

Forget node. Let i be a forget node where $X_i = X_j \setminus \{u\}$ and $u \in X_j$. For any distance labelling function dl of X_i we denote $L_c(\mathrm{dl})$ the set of functions which extend dl by assigning labels $\{1, \ldots, \varrho\}$ to u and $L_s(\mathrm{dl})$ the set of functions which extend dl by assigning labels $\{0, \ldots, \varrho\} \cup \{\infty\}$ to u. Additionally, let $L_\infty(\mathrm{dl})$ be an extension of dl which assigns label ∞ to u. Let dl_i be any distance labelling function of X_i, S_i be any subset of clients of X_i, $k_i \in \{0, \ldots, k\}$, and $p_i \in \{0, \ldots, p\}$. If u is a supplier, then we set

$$D_i[\mathrm{dl}_i, S_i, k_i, p_i] = \bigvee_{\mathrm{dl}_j \in L_s(\mathrm{dl}_i)} D_j[\mathrm{dl}_j, S_i, k_i, p_i].$$

If u is a client, then we set

$$D_i[\mathrm{dl}_i, S_i, k_i, p_i] = \left(\bigvee_{\mathrm{dl}_j \in L_c(\mathrm{dl}_i)} D_j[\mathrm{dl}_j, S_i \cup \{u\}, k_i, p_i] \right) \vee D_j[L_\infty(\mathrm{dl}), S_i, k_i, p_i].$$

We proceed to showing the correctness. Recall that by properties of nice tree decompositions, we have $V_i = V_j$ for the forget node i. Let $\widehat{\mathrm{dl}}$ be a distance labelling function of $G[V_i]$ of cost $\widehat{k}$ and penalty $\widehat{p}$ such that all clients of $V_i \setminus X_i$ with a finite label are satisfied. Given a node t of the tree decomposition, let $\widehat{\mathrm{dl}}_t$ be the restriction of $\widehat{\mathrm{dl}}$ to X_t, and $\widehat{S}_t$ be the set of satisfied of clients of X_t by $\widehat{\mathrm{dl}}$. Our goal is to show that the algorithm sets the entry $D_i[\widehat{\mathrm{dl}}_i, \widehat{S}_i, \widehat{k}, \widehat{p}]$ to 1. From the induction hypothesis, the entry $D_j[\widehat{\mathrm{dl}}_j, \widehat{S}_j, k_j, p_j]$ is set to 1 for some values k_j and p_j; in fact it is the case that $k_j = \widehat{k}$ and $p_j = \widehat{p}$ as $V_i = V_j$. Since the algorithm inspects all entries of the table D_i, it will eventually consider the entry $D_i[\widehat{\mathrm{dl}}_i, \widehat{S}_i, \widehat{k}, \widehat{p}]$. By trying all possible values for the forgotten vertex u, it will consider the labelling $\widehat{\mathrm{dl}}_j$ of X_j. To finish the proof of this direction, it remains to determine the relationship between $\widehat{S}_i$ and $\widehat{S}_j$.

- **Case** $u \in V_{\mathrm{s}}$**.** We have $X_i \cap V_{\mathrm{c}} = X_j \cap V_{\mathrm{c}}$ and thus $\widehat{S}_i = \widehat{S}_j$.

- **Case** $\widehat{\mathrm{dl}}(u) \in \{1, \ldots, \varrho\}$ **and** $u \in V_{\mathrm{c}}$**.** If the label of the forgotten client u is finite, then it must be satisfied by $\widehat{\mathrm{dl}}$ from the requirement that $\widehat{\mathrm{dl}}$ satisfies all vertices of $V_i \setminus X_i$. Thus $\widehat{S}_i \cup \{u\} = \widehat{S}_j$.

- **Case** $\widehat{\mathrm{dl}}(u) = \infty$ **and** $u \in V_{\mathrm{c}}$**.** If the label of the forgotten client u is ∞, then it is ignored by $\widehat{\mathrm{dl}}$. In this case we have $\widehat{S}_i = \widehat{S}_j$.

For the opposite direction let $D_i[\mathrm{dl}_i, S_i, k_i, p_i]$ be an entry set to 1 by the algorithm. Then it follows from the description of the algorithm, that there is an entry $D_j[\mathrm{dl}_j, S_j, k_j, p_j]$ set to 1 in the table of the child j where dl_j is an extension of dl_i to X_j, $S_j \supseteq S_i$, $k_i = k_j$ and $p_i = p_j$. By the induction hypothesis, there exists a distance labelling function $\widehat{\mathrm{dl}}_j$ on V_j which agrees with dl_j on X_j, satisfies clients S_j of X_j, has cost k_j and penalty p_j. We claim that $\widehat{\mathrm{dl}}_j$ also agrees with dl_i, satisfies clients S_i of X_i, has cost k_i and penalty p_i. Since dl_j agrees with dl_i on X_i and $X_i \subseteq X_j$, $\widehat{\mathrm{dl}}_j$ agrees with dl_i. The property that every client of S_i is satisfied by $\widehat{\mathrm{dl}}_j$ follows from $S_i \subseteq S_j$, $V_i = V_j$, and the fact that $\widehat{\mathrm{dl}}_j$ satisfies S_j by the induction hypothesis. From $V_i = V_j$ it also follows that $k_i = k_j$ and $p_i = p_j$.

Join node. Let i be a join node with children j_1 and j_2 where $X_i = X_{j_1} = X_{j_2}$. Let dl_i be a distance labelling function on X_i, then for each pair of true entries $D_{j_1}[\mathrm{dl}, S_1, k_1, p_1]$ and $D_{j_2}[\mathrm{dl}, S_2, k_2, p_2]$ we set to 1 the entry

$$D_i\left[\mathrm{dl}_i, S_1 \cup S_2, k_1 + k_2 - \left|\mathrm{dl}_i^{-1}(0)\right|, p_1 + p_2 - \left|\mathrm{dl}_i^{-1}(\infty) \cap V_{\mathrm{c}}\right|\right].$$

We proceed to showing the correctness. It follows from the properties of nice tree decompositions that $V_i = V_{j_1} \cup V_{j_2}$. Let $\widehat{\mathrm{dl}}$ be a valid distance labelling function of $G[V_i]$ of cost $\widehat{k}$ and penalty $\widehat{p}$ such that all clients of $V_i \setminus X_i$ with a finite label are satisfied. We denote by $\widehat{S}_i$ the set of satisfied clients in X_i and by $\widehat{\mathrm{dl}}_i$ the restriction of $\widehat{\mathrm{dl}}$ to X_i. Let $a \in \{1, 2\}$. Then let $\widehat{\mathrm{dl}}_a$ be the restriction of $\widehat{\mathrm{dl}}$ to V_{j_a}, $\widehat{S}_{j_a}$ be the set of clients of X_{j_a} satisfied by $\widehat{\mathrm{dl}}_a$, $\widehat{C}_a$ be the set of opened suppliers in V_{j_a}, and $\widehat{O}_a$ be the set of outliers in V_{j_a}. Note that $\widehat{\mathrm{dl}}, \widehat{\mathrm{dl}}_1$ and $\widehat{\mathrm{dl}}_2$ agree with each other on X_i. From the induction hypothesis the entries $D_{j_a}\left[\widehat{\mathrm{dl}}_i, \widehat{S}_{j_a}, \left|\widehat{C}_{j_a}\right|, \left|\widehat{O}_{j_a}\right|\right]$ are set to 1. To show that $\widehat{S}_i = \widehat{S}_{j_1} \cup \widehat{S}_{j_2}$, we use the standard approach of showing $\widehat{S}_i \supseteq \widehat{S}_{j_1} \cup \widehat{S}_{j_2}$ and $\widehat{S}_i \subseteq \widehat{S}_{j_1} \cup \widehat{S}_{j_2}$. From $V_{j_1} \subseteq V_i$ it trivially follows that $\widehat{S}_{j_a} \subseteq \widehat{S}_i$. For a satisfied client $v \in \widehat{S}_i$, we want to show that $v \in \widehat{S}_{j_1} \cup \widehat{S}_{j_2}$. As $V_i = V_{j_1} \cup V_{j_2}$, the neighbour w which satisfies v lies in $V_{j_1} \cup V_{j_2}$. Hence it must be the case that v is satisfied in at least in one of V_{j_1} or V_{j_2} and thus v is satisfied in at least one of $\widehat{S}_{j_1}$ or $\widehat{S}_{j_2}$. It follows from property (T3) of tree decompositions that if a vertex w is simultaneously contained in a bag of some node of the subtree rooted in j_1 and in a bag of some node of the subtree rooted in j_2, then $w \in X_i$. This means that $V_{j_1} \cap V_{j_2} \subseteq X_i$ and in particular $\widehat{C}_1 \cap \widehat{C}_2 \subseteq X_i$ and $\widehat{O}_1 \cap \widehat{O}_2 \subseteq X_i$. Since $\widehat{\mathrm{dl}}$, $\widehat{\mathrm{dl}}_1$, and $\widehat{\mathrm{dl}}_2$ agree with each other on X_i, $\widehat{C}_1 \cap X_i = \widehat{C}_2 \cap X_i$ and $\widehat{O}_1 \cap X_i = \widehat{O}_2 \cap X_i$. Using these facts we have $\left|\widehat{C}_1\right| + \left|\widehat{C}_2\right| = \widehat{k} + \left|\widehat{\mathrm{dl}}^{-1}(0) \cap X_i\right|$ and $\left|\widehat{O}_1\right| + \left|\widehat{O}_2\right| = \widehat{p} + \left|\widehat{\mathrm{dl}}^{-1}(\infty) \cap X_i \cap V_{\mathrm{c}}\right|$, that is, every opened supplier and every outlier is counted twice in X_i.

For the opposite direction, let $D_i[\mathrm{dl}_i, S_i, k_i, p_i]$ be an entry set to 1 by the algorithm. From the description of the algorithm, this means that there exist entries $D_{j_1}[\mathrm{dl}_i, S_{j_1}, k_{j_1}, p_{j_1}]$ and $D_{j_2}[\mathrm{dl}_i, S_{j_2}, k_{j_2}, p_{j_2}]$ set to 1 where S_{j_1} and S_{j_2} are subsets of X_i (where $X_i = X_{j_1} = X_{j_2}$ for a join node i). By the induction hypothesis, for $a \in \{1, 2\}$ there exists a valid distance labelling function $\widehat{\mathrm{dl}}_a$ on V_{j_a} which agrees with dl_i on X_{j_a}, satisfies clients $S_{j_a} \subseteq X_{j_a}$, and has cost $\widehat{k}_a$ and penalty $\widehat{p}_a$. We claim that a function dl which behaves as $\widehat{\mathrm{dl}}_1$ on V_{j_1} and as $\widehat{\mathrm{dl}}_2$ on V_{j_2} has the desired properties. To verify that dl is a function (and not a multifunction), we use the fact that $V_{j_1} \cap V_{j_2} \subseteq X_i$. The algorithm requires

that $\widehat{\mathrm{dl}}_1$ and $\widehat{\mathrm{dl}}_2$ behave the same on X_i, thus dl is indeed a function. We can prove the remaining required properties, i.e. $S_i = S_{j_1} \cup S_{j_2}$, $k_i = k_{j_1} + k_{j_2} - \left|\mathrm{dl}_i^{-1}(0)\right|$, and $p_i = p_{j_1} + p_{j_2}$, identically as in the proof of the opposite direction.

Running time. The size of the table at each node is at most $(\varrho+2)^{\mathrm{tw}} \cdot 2^{\mathrm{tw}} \cdot (k+1) \cdot (p+1)$. In each introduce node i, we inspect all table entries of its child j and for each such child entry we recalculate the set of satisfied vertices, suppliers with label 0, and ignored clients in time tw. In each forget node i, for each table entry of i we inspect up to $\varrho+2$ entries in the table of the child j. In each join node i, we try to combine all possible entries from its children j_1 and j_2 which takes time $|D_{j_1}| \cdot |D_{j_2}| \leq ((\varrho+2)^{\mathrm{tw}} \cdot 2^{\mathrm{tw}} \cdot (k+1) \cdot (p+1))^2$. Overall, the running time of the algorithm is at most $O^*((4(\varrho+2))^{O(\mathrm{tw})})$. □

The approximation scheme

Now we describe a parameterized approximation scheme based on the algorithm from Theorem 18. We will need the following result from Chatterjee et al.

Theorem 20 ([CIJP14])**.** *Let G be a graph. There exists an algorithm which, given a tree decomposition $\mathcal{T}$ of G such that $\mathcal{T}$ has n nodes and width* tw*, produces a nice tree decomposition of G with width at most* $4\mathrm{tw}+3$ *and height* $O(\mathrm{tw} \cdot \log n)$ *in time* $O(\mathrm{tw} \cdot n)$.

Let us give an approximate version of the distance labelling problem for a fixed error parameter $\varepsilon > 0$. This is a generalization of the approximate distance labelling used in the original algorithm [KLP19]. Let (G, k, p) be an instance of the kSwO problem with edge lengths $d : E \to \mathbb{N}$ and $\delta > 0$ some appropriately chosen secondary parameter (we will eventually set $\delta \approx \frac{\varepsilon}{\log n}$). Let $\Sigma_\varrho = \{(1+\delta)^i : i \in \mathbb{N}, (1+\delta)^i \leq (1+\varepsilon)\varrho\}$, we define a *$\delta$-labelling function* of V as a function $\mathrm{dl}_\delta : V \to \Sigma_\varrho \cup \{0, \infty\}$. Only a supplier can have label 0. A vertex u is *ε-satisfied* if $\mathrm{dl}_\delta(u) = 0$ or if u has a finite label and there exists $v \in N(u)$ such that $\mathrm{dl}_\delta(u) \geq \mathrm{dl}_\delta(v) + \frac{d(u,v)}{1+\varepsilon}$. Given a δ-labelling function dl_δ, if every client is either ε-satisfied or has label ∞, then we say that dl_δ is *valid*. We define the *cost* of such a function dl_δ as $\left|\mathrm{dl}_\delta^{-1}(0)\right|$ and its *penalty* as $\left|\mathrm{dl}_\delta^{-1}(\infty) \cap V_{\mathrm{c}}\right|$. The following lemma shows that given a δ-labelling function of cost k and penalty p, we can produce a solution to the kSwO problem which opens k centers, creates p outliers and has cost $(1+\varepsilon)^2\varrho$.

Lemma 21. *Let $\mathcal{I} = (G, k, p)$ be an instance of the kSwO problem. If there exists a valid δ-labelling function of G with cost at most k and penalty at most p, then $\mathcal{I}$ has a feasible solution of cost $(1+\varepsilon)^2\varrho$.*

Proof. We select vertices with label 0 as the solution S. The number of opened suppliers is the cost of the function, hence we open at most k suppliers. We will show by induction on i that if $\mathrm{dl}_\delta(u) = (1+\delta)^i$, then $\mathrm{dist}(u, S) \leq (1+\varepsilon)\mathrm{dl}_\delta(u)$. In the base case let u be an ε-satisfied vertex with $\mathrm{dl}_\delta(u) = (1+\delta)$. Then there exists a neighbour $v \in N(u)$ with $\mathrm{dl}_\delta(u) \geq \mathrm{dl}_\delta(v) + \frac{d(u,v)}{1+\varepsilon}$. Since $d(u,v) > 0$, it follows that $\mathrm{dl}_\delta(u) > \mathrm{dl}_\delta(v)$ and the only possible δ-label less than $(1+\delta)$ is 0, hence $\mathrm{dl}_\delta(v) = 0$. Then we have $(1+\delta) \geq \frac{d(u,v)}{1+\varepsilon}$ which shows the base case as $v \in S$. For the induction step, let u be an ε-satisfied vertex with $\mathrm{dl}_\delta(u) = (1+\delta)^{i+1}$.

There exists a neighbour $v \in N(u)$ such that $\mathrm{dl}_\delta(u) \geq \mathrm{dl}_\delta(v) + \frac{d(u,v)}{1+\varepsilon}$. As edge lengths are positive, we have $\mathrm{dl}_\delta(u) > \mathrm{dl}_\delta(v)$ and using induction hypothesis we receive $\mathrm{dist}(v, S) \leq \mathrm{dl}_\delta(v)$. By triangle inequality we have $\mathrm{dist}(u, S) \leq d(u, v) + \mathrm{dist}(v, S) \leq d(u, v) + \mathrm{dl}_\delta(v) = (1 + \varepsilon)\left(\frac{d(u,v)}{1+\varepsilon} + \mathrm{dl}_\delta(v)\right) \leq (1 + \varepsilon)\mathrm{dl}_\delta(u)$.

Since all satisfied clients have a δ-label at most $(1 + \varepsilon)\varrho$, all satisfied clients are at distance at most $(1 + \varepsilon)^2\varrho$ from S. The number of outliers is at most p as the penalty of dl_δ is at most p. □

The following two lemmas from [KLP19], which they use for their parameterized approximation scheme for the k-Center problem, will be useful for us as well. The first lemma shows that adding all missing edges between vertices of a single bag of length equal to their shortest-path distance does not change the set of solutions. The original formulation is for the k-Center problem, however, their proof generalizes to the kSwO problem without any major modifications.

Lemma 22 ([KLP19, Lemma 29])**.** *Let $\mathcal{I} = (G, k, p)$ be an instance of the kSwO problem, $\mathcal{T}$ a tree decomposition of G and $u, v \in V$ two vertices which appear together in a bag of $\mathcal{T}$ and $(u, v) \notin E$. Let G' be the graph obtained from G by adding the edge (u, v) with length $\mathrm{dist}(u, v)$ and let $\mathcal{I}' = (G', k, p)$ be an instance of the kSwO problem. Then $\mathcal{I}$ has a feasible solution of cost ϱ if and only if $\mathcal{I}'$ does.*

The second lemma shows that an algorithm with a running time in the form $O^*\left((\frac{\log n}{\varepsilon})^{O(k)}\right)$ is still an FPT algorithm.

Lemma 23 ([KLP19, Lemma 1])**.** *Let $\mathcal{A}$ be an algorithm for a parameterized problem with parameter k such that the running time of $\mathcal{A}$ is $O^*\left((\frac{\log n}{\varepsilon})^{O(k)}\right)$. Then the running time of $\mathcal{A}$ can be bounded by $O^*\left((\frac{k}{\varepsilon})^{O(k)}\right)$.*

We are ready to prove Theorem 17. The proof follows the proof of the original algorithm, cf. [KLP19, Theorem 31]. The main obstacle is to bound the accumulated error during the execution of the algorithm. The original proof heavily relies on the properties of the parameterized approximation scheme they present for the k-Center problem. Hence, we will have to modify their proof to work with a parameterized approximation scheme we will give for the kSwO problem.

Proof of Theorem 17. Our algorithm will follow along the same lines as the algorithm in Theorem 18. The major difference is that instead of distance labelling functions we consider δ-labelling functions for some δ we specify later and instead of satisfiability we use ε-satisfiability.

Recall that $\Sigma_\varrho = \{0\} \cup \{(1 + \delta)^i : i \in \mathbb{N}, (1 + \delta)^i \leq (1 + \varepsilon)\varrho\} \cup \{\infty\}$. For each node $i \in V(T)$ we define a table

$$D_i^\delta : \left((X_i \to \Sigma_\varrho) \times 2^{X_i \cap V_c} \times \{0, \ldots, k\} \times \{0, \ldots, p\}\right) \to \{0, 1\}.$$

We may refer to value 1 in the dynamic programming table as *true* and to value 0 as *false*. Recall that our definitions of a δ-labelling allows only suppliers to have label 0. For a node $i \in V(T)$, a δ-labelling function $\mathrm{dl}_\delta : X_i \to \Sigma_\varrho$, a subset of clients $S \subseteq 2^{X_i \cap V_c}$, and integers k_i and p_i where $0 \leq k_i \leq k$ and $0 \leq p_i \leq p$, the value of an entry $D_i^\delta[\mathrm{dl}_\delta, S, k_i, p_i]$ is 1 if and only if there exists a δ-labelling

of $G[V_i]$ which agrees with dl_δ on X_i, satisfies clients $((V_i \setminus X_i) \cup S) \cap V_c$, has cost k_i and penalty p_i.

For the rest of the proof, we denote by n the number of nodes of the tree decomposition $\mathcal{T}$ provided on input. We start by preprocessing the graph using Theorem 20 and Lemma 22. We obtain a tree decomposition $\mathcal{T}'$ of the input graph of width $4\mathrm{tw}+3$ and height H where $H \leq c \cdot \mathrm{tw} \cdot \log n$ for some constant c. For every pair of vertices u, v which appear together in some bag of $\mathcal{T}'$ we have an edge with length at most their shortest-path distance. We define the *height* of a node of $\mathcal{T}'$ inductively where the height of a leaf node is 1 and the height of any inner node is 1 plus the maximum of the heights of its children. Under this definition the root node has height H and all other bags have height less than H. We may refer to a height of a bag by which we mean the height of the node corresponding to the bag.

We set $\delta = \frac{\varepsilon}{2H} = \Omega\left(\frac{\varepsilon}{\mathrm{tw}\cdot\log n}\right)$. Observe that this choice of δ gives for all $h \leq H$ that $(1+\delta)^h \leq \left(1+\frac{\varepsilon}{2H}\right)^H \leq e^{\varepsilon/2} \leq 1+\varepsilon$ for sufficiently small ε (it suffices to assume without loss of generality that $\varepsilon < \frac{1}{4}$). The goal is to return a feasible solution of cost $(1+\varepsilon)^2\varrho$ if a feasible solution of cost ϱ exists by producing a δ-labelling and invoking Lemma 21. The approximation ratio can then be reduced to $1+\varepsilon$ by adjusting ε appropriately.

We now present the dynamic programming procedure. It only differs from the algorithm in Theorem 18 by considering δ-labelling functions instead of distance labelling functions and ε-satisfiability instead of satisfiability.

- **Leaf node.** For a leaf node ℓ we have $V_\ell = \emptyset$. Thus the only true entry is $D^\delta_\ell[\mathrm{dl}, \emptyset, 0, 0]$ where the domain of dl is an empty set.

- **Introduce node.** Let i be an introduce node with a child node j, then $X_i = X_j \cup \{u\}$ where $u \notin X_j$. Let $D^\delta_j[\mathrm{dl}'_\delta, S', k_j, p_j]$ be an entry of the table of the child node j with value 1. We construct a δ-labelling function dl_δ which agrees with dl'_δ on X_j and tries all possible values for u. In particular the constructed δ-labelling functions set $\mathrm{dl}_\delta(u)$ to values $\Sigma_\varrho \setminus \{0\}$ and additionally we try the label 0 for u if u is a supplier. For such a δ-labelling function dl_δ we compute S to be the set of satisfied vertices of X_i as follows. We add the entire set S' to S. We add u to S if there exists a neighbour $v \in N(u)$ so that $\mathrm{dl}_\delta(u) \geq \mathrm{dl}_\delta(v) + \frac{d(u,v)}{1+\varepsilon}$. Finally, we add neighbours $w \in N(u)$ to S for which it holds that $\mathrm{dl}_\delta(w) \geq \mathrm{dl}_\delta(u) + \frac{d(u,w)}{1+\varepsilon}$. If $\mathrm{dl}_\delta(u) = 0$ and $k_j \leq k-1$, then we set the entry $D^\delta_i[\mathrm{dl}_\delta, S, k_j+1, p_j]$ to 1. If $\mathrm{dl}_\delta(u) \in \Sigma_\varrho \setminus \{0, \infty\}$, then we set the entry $D^\delta_i[\mathrm{dl}_\delta, S, k_j, p_j]$ to 1. If $\mathrm{dl}_\delta(u) = \infty$, $u \in V_c$, and $p_j \leq p-1$, then we set the entry $D^\delta_i[\mathrm{dl}_\delta, S, k_j, p_j+1]$ to 1. If $\mathrm{dl}_\delta(u) = \infty$ and $u \in V_s$, then we set the entry $D^\delta_i[\mathrm{dl}_\delta, S, k_j, p_j]$ to 1.

- **Forget node.** Let i be a forget node where $X_i = X_j \setminus \{u\}$ and $u \in X_j$. For any δ-labelling function dl_δ of X_i we denote by $L_c(\mathrm{dl}_\delta)$ the set of functions which extend dl_δ by assigning labels $\Sigma_\varrho \setminus \{0, \infty\}$ to u and by $L_s(\mathrm{dl}_\delta)$ the set of functions which extend dl_δ by assigning labels Σ_ϱ to u. Additionally, let $L_\infty(\mathrm{dl}_\delta)$ be an extension of dl_δ which sets the label ∞ to u. Let S be any subset of clients of X_i, $k_i \in \{0, \ldots, k\}$, and $p_i \in \{0, \ldots, p\}$. If u is a

supplier, then we set

$$D_i^\delta[\mathrm{dl}_\delta, S, k_i, p_i] = \bigvee_{\mathrm{dl}'_\delta \in L_s(\mathrm{dl}_\delta)} D_j^\delta[\mathrm{dl}'_\delta, S, k_i, p_i].$$

If u is a client, then we set

$$D_i^\delta[\mathrm{dl}_\delta, S, k_i, p_i] = \left(\bigvee_{\mathrm{dl}'_\delta \in L_c(\mathrm{dl}_\delta)} D_j^\delta[\mathrm{dl}'_\delta, S \cup \{u\}, k_i, p_i] \right) \vee D_j^\delta[L_\infty(\mathrm{dl}_\delta), S, k_i, p_i].$$

- **Join node.** Let i be a join node with children j_1 and j_2 where $X_i = X_{j_1} = X_{j_2}$. Let dl_δ be a δ-labelling function on X_i. Then for each pair of true entries $D_{j_1}^\delta[\mathrm{dl}_\delta, S_1, k_1, p_1]$ and $D_{j_2}^\delta[\mathrm{dl}_\delta, S_2, k_2, p_2]$ we set to 1 the entry

$$D_i\left[\mathrm{dl}_\delta, S_1 \cup S_2, k_1 + k_2 - \left|\mathrm{dl}_\delta^{-1}(0)\right|, p_1 + p_2 - \left|\mathrm{dl}_\delta^{-1}(\infty) \cap V_\mathrm{c}\right|\right].$$

To establish correctness of the algorithm, there are two tasks we need to accomplish. First, we need to show that for any bag X_i we have $D_i^\delta[\mathrm{dl}_\delta, S, k_i, p_i] = 1$ if and only if there exists a δ-labelling of $G[V_t]$ which agrees with dl_δ on X_i, satisfies clients $((V_i \setminus X_i) \cup S) \cap V_\mathrm{c}$ aside from those clients with label ∞, and has cost k_i and penalty p_i. The proof of this equivalence is done similarly to the proof of correctness of Theorem 18. The only difference is that we need to consider δ-labelling functions and ε-satisfiability instead of distance labelling and satisfiability respectively. Hence we omit this part of the proof. Using Lemma 21, we obtain a solution to the κSwO instance on input of cost $(1+\varepsilon)^2\varrho$.

It is more interesting to prove the following statement: we would like to show that if there exists a solution of cost ϱ, then there exists a δ-labelling which is going to be found by the algorithm. The main difficulty of proving this statement is that the converse of Lemma 21 does not hold for any choice of δ. In the remainder suppose there exists a distance labelling $\mathrm{dl} : V \to \{0, \ldots, \varrho\} \cup \{\infty\}$ which encodes a solution to the instance as in the proof of Lemma 19.

Let X_i be a bag of the decomposition of height h and S the vertices of V_i satisfied by dl including suppliers. We are going to show that there always exists $\mathrm{dl}_\delta : X_i \to \Sigma_\varrho$, $S_\delta \supseteq S$ and values k_i and p_i such that $D_i^\delta[\mathrm{dl}_\delta, S_\delta, k_i, p_i] = 1$, $k_i \leq \left|\mathrm{dl}^{-1}(0)\right|$, $p_i \leq \left|\mathrm{dl}^{-1}(\infty) \cap V_\mathrm{c}\right|$ and for all $u \in X_i$ we have $\mathrm{dl}_\delta(u) \in \left[\frac{\mathrm{dl}(u)}{(1+\delta)^h}, (1+\delta)^h \mathrm{dl}(u)\right]$.

We prove this claim by induction on the height of a bag. This property trivially holds for empty leaf bags in the base case. For the induction step, consider a node at height $h+1$. In the case of forget and join bags, if we assume that the desired property holds for their children, then it follows that it holds for them as well since $\mathrm{dl}_\delta(u) \in \left[\frac{\mathrm{dl}(u)}{(1+\delta)^h}, (1+\delta)^h \mathrm{dl}(u)\right]$ implies $\mathrm{dl}_\delta(u) \in \left[\frac{\mathrm{dl}(u)}{(1+\delta)^{h+1}}, (1+\delta)^{h+1} \mathrm{dl}(u)\right]$.

It remains to prove the property for introduce nodes. Let i be an introduce node with child j where $X_i = X_j \cup \{u\}$, $u \notin X_j$. We cannot use the approach for proving the desired property we used for join and forget nodes. For the introduced vertex u the induction hypothesis $\mathrm{dl}_\delta(u) \in \left[\frac{\mathrm{dl}(u)}{(1+\delta)^h}, (1+\delta)^h \mathrm{dl}(u)\right]$ does not apply when $\mathrm{dl}(u)$ is finite since $u \notin V_j$. Let $S \subseteq X_i$ be the set of vertices (including suppliers) satisfied by dl in V_i and similarly let $S' \subseteq X_j$ be the set of satisfied vertices in V_j. We claim that at least one of the following must be true:

(i) $\mathrm{dl}(u) = 0$,

(ii) $S = S' \cup \{u\}$,

(iii) $u \notin S$ and $\mathrm{dl}(u) < \infty$,

(iv) $\mathrm{dl}(u) = \infty$.

Suppose for contradiction that $\mathrm{dl}(u) \in \{1, \ldots, \varrho\}$ and $S \supseteq S' \cup \{u, v_1\}$ where $v_1 \in X_i \setminus S'$. If v_1 is satisfied in X_i but not in X_j, then the sole cause of this fact is that v_1 is satisfied by u as the labels of its neighbours aside from u remain the same between $X_i \cap N(v_1)$ and $X_j \cap N(v_1)$. Thus we have $\mathrm{dl}(v_1) \geq \mathrm{dl}(u) + d(v_1, u)$. Since u is a satisfied vertex and $\mathrm{dl}(u) \in \{1, \ldots, \varrho\}$, there exists a vertex $v_2 \in X_j$ such that $\mathrm{dl}(u) \geq \mathrm{dl}(v_2) + d(u, v_2)$. Together we have $\mathrm{dl}(v_1) \geq \mathrm{dl}(v_2) + d(v_2, u) + d(u, v_1) \geq \mathrm{dl}(v_2) + d(v_1, v_2)$ where the last inequality holds from the preprocessing using Lemma 22. However, the last inequality shows that $v_1 \in S'$ which is a contradiction.

We therefore need to establish that for each of the four cases above, the algorithm produces an entry $D_i^\delta[\mathrm{dl}_\delta, S_\delta, k_i, p_i]$ with $S \subseteq S_\delta$, $k_i \leq \left|\mathrm{dl}^{-1}(0) \cap V_i\right|$, $p_i \leq \left|\mathrm{dl}^{-1}(\infty) \cap V_i \cap V_{\mathrm{c}}\right|$, and $\mathrm{dl}_\delta(u)$ which is at most a factor $(1+\delta)^h$ apart from $\mathrm{dl}(u)$. Assume by the induction hypothesis that there exists an entry $D_j^\delta[\mathrm{dl}'_\delta, S'_\delta, k_j, p_j]$ with value 1 for some $S'_\delta \supseteq S'$, $k_j \leq \left|\mathrm{dl}^{-1}(0) \cap V_j\right|$, $p_j \leq \left|\mathrm{dl}^{-1}(\infty) \cap V_j \cap V_{\mathrm{c}}\right|$ and dl'_δ which has $(\forall v \in X_j)\left(\mathrm{dl}'_\delta(v) \in \left[\frac{\mathrm{dl}(v)}{(1+\delta)^{h-1}}, (1+\delta)^{h-1}\mathrm{dl}(v)\right]\right)$.

(i) If $\mathrm{dl}(u) = 0$, the algorithm considers a δ-labelling function dl_δ which agrees with dl'_δ on X_j and sets $\mathrm{dl}_\delta(u) = 0$. Since the entry $D_j^\delta[\mathrm{dl}'_\delta, S'_\delta, k_j, p_j]$ has value 1, the algorithm sets the entry $D_i^\delta[\mathrm{dl}_\delta, S_\delta, k_j + 1, p_j]$ to 1 for some S_δ. We claim that $S \subseteq S_\delta$. To see this, let $v \in S \setminus S'$. Then v must be satisfied by u and we have $\mathrm{dl}(v) \geq \mathrm{dl}(u) + d(u, v)$. From the induction hypothesis, $\mathrm{dl}(u) = 0$, and $(1+\delta)^{h-1} \leq 1 + \varepsilon$, we have $\mathrm{dl}_\delta(v) \geq \frac{\mathrm{dl}(v)}{(1+\delta)^{h-1}} \geq \frac{d(u,v)}{1+\varepsilon}$. This shows that every vertex $v \in S \setminus S'$ is satisfied in X_i.

(ii) Assume that $\mathrm{dl}(u) \notin \{0, \infty\}$ and $u \in S$. Then there must exist a vertex v which satisfies u, that is $\mathrm{dl}(u) \geq \mathrm{dl}(v) + d(u, v)$. Let $r = (1+\delta)^{h-1}\mathrm{dl}(u)$. The algorithm considers a δ-labelling function dl_δ which agrees with dl'_δ on X_j and sets $\mathrm{dl}_\delta(u) = (1+\delta)^{\lceil \log_{1+\delta} r \rceil}$. Using $(1+\delta)^{h-1} \leq 1 + \varepsilon$ we have $\mathrm{dl}_\delta(u) \geq (1+\delta)^{h-1}\mathrm{dl}(u) \geq (1+\delta)^{h-1}(\mathrm{dl}(v) + d(u, v)) \geq \mathrm{dl}_\delta(v) + \frac{d(u,v)}{1+\varepsilon}$. Hence the algorithm correctly adds u to S'_δ to obtain $S_\delta \supseteq S$. Moreover we have the required upper bound as well since

$$\mathrm{dl}_\delta(u) = (1+\delta)^{\lceil \log_{1+\delta} r \rceil} \leq (1+\delta)^{\log_{1+\delta}((1+\delta)^{h-1}\mathrm{dl}(u))+1} \leq (1+\delta)^h \mathrm{dl}(u).$$

(iii) Consider the case when $S \setminus S' \neq \emptyset$, otherwise there is nothing to prove. Let $v \in S \setminus S'$, then v must be satisfied by u, that is $\mathrm{dl}(v) \geq \mathrm{dl}(u) + d(u, v)$. Let $r = \frac{\mathrm{dl}(u)}{(1+\delta)^h}$. The algorithm considers a δ-labelling function dl_δ which agrees with dl'_δ on X_j and sets $\mathrm{dl}_\delta(u) = (1+\delta)^{\lceil \log_{1+\delta} r \rceil}$. Using the induction hypothesis and $(1+\delta)^{h-1} \leq 1 + \varepsilon$, we have

$$\mathrm{dl}_\delta(v) \geq \frac{\mathrm{dl}(v)}{(1+\delta)^{h-1}} \geq \frac{\mathrm{dl}(u)}{(1+\delta)^{h-1}} + \frac{d(u, v)}{1+\varepsilon} \geq \mathrm{dl}_\delta(u) + \frac{d(u, v)}{1+\varepsilon}.$$

Hence the algorithm extends S'_δ by adding all elements of $S \setminus S'$ to create the set S_δ. Moreover we have the required upper bound as well since

$$\mathrm{dl}_\delta(u) = (1+\delta)^{\lceil \log_{1+\delta} r \rceil} \leq (1+\delta)^{\log_{1+\delta} \frac{\mathrm{dl}(u)}{(1+\delta)^h} + 1} \leq \frac{\mathrm{dl}(u)}{(1+\delta)^{h-1}}.$$

(iv) The algorithm considers a δ-labelling function dl_δ which agrees with dl'_δ on X_j and sets $\mathrm{dl}_\delta(u) = \infty$. Since a vertex with label ∞ cannot satisfy a neighbour by definition, we have $S_\delta = S'_\delta$. The desired bound on $\mathrm{dl}_\delta(u)$ for $u \in X_i$ follows trivially.

We conclude that whenever a feasible solution of cost ϱ exists to the input instance, we are able to recover from the root bag of the dynamic programming table a solution of cost $(1+\varepsilon)^2\varrho$ with at most k centers and at most p outliers. In particular, there exists an entry in the dynamic programming table of the root bag $D_r[\mathrm{dl}_\delta, X_r \cap V_c, k_r, p_r]$ where dl_δ is a δ-labelling of X_r where for all $u \in X_r$ we have $\mathrm{dl}_\delta(u) \leq (1+\delta)^H \mathrm{dl}(u) \leq (1+\varepsilon)\varrho$, $k_r \leq k$ and $p_r \leq p$.

It remains to bound the running time of the algorithm. We have $|\Sigma_\varrho| = O(\log_{1+\delta} \varrho) = O\left(\frac{\log \varrho}{\log(1+\delta)}\right) = O\left(\frac{\log \varrho}{\delta}\right)$ where we have used the fact that $\ln(1+\delta) \approx \delta$ for sufficiently small δ (that is, sufficiently large n). By setting $\delta = \Omega\left(\frac{\varepsilon}{\mathrm{tw} \cdot \log n}\right)$ and assuming $k, p \leq |V(G)|$, we get a running time $O^*\left((\mathrm{tw} \cdot \frac{\log n}{\varepsilon})^{O(\mathrm{tw})}\right)$. Using Lemma 23, this is an FPT algorithm with running time $O^*\left((\frac{\mathrm{tw}}{\varepsilon})^{O(\mathrm{tw})}\right)$. □

3.2 EPAS for low highway dimension graphs

In this section we present an FPT approximation scheme for the kSwO problem on low highway dimension graphs. For this algorithm, we use Definition 1 of highway dimension where the constant γ in the definition is strictly greater than 4. Let us recall the main result of this section.

Theorem 4. *Let $\mathcal{I} = (G, k, p)$ be an instance of the k-Supplier with Outliers problem where G has highway dimension h. There exists a computable function $f(\cdot,\cdot,\cdot,\cdot)$ and an algorithm such that for any $\varepsilon > 0$ it outputs a solution of cost $(1+\varepsilon)\mathrm{OPT}(\mathcal{I})$ in time $f(h,k,p,\varepsilon) \cdot n^{O(1)}$. Moreover we have $f \in \Omega\left(\left(\frac{1}{\varepsilon}(h+k+p)^2\right)^{O\left((h+k+p)^2 \log(1/\varepsilon)\right)}\right)$.*

An *embedding* of an (undirected) *guest graph* G into a *host graph* H is an injective mapping $\phi : V(G) \to V(H)$. The host graph H can contain vertices which do not appear in G. Our algorithm uses a framework by Becker et al. [BKS18] for embedding low highway dimension graphs into low treewidth graphs. After embedding the input graph into a low treewidth graph, we use the algorithm given by Theorem 17 to obtain an approximate solution. Let us remark that the algorithm we present extends a parameterized approximation scheme for the k-Center problem presented in [BKS18]. The formal statement of the used result is the following theorem.

Theorem 24 ([BKS18, Theorem 4])**.** *There is a function $f(\cdot,\cdot,\cdot)$ such that, for every $\varepsilon > 0$, graph G of highway dimension h and set of vertices $S \subseteq V(G)$, there exists a graph H and an embedding $\phi : V(G) \to V(H)$ such that*

- *H has treewidth $f(h, |S|, \varepsilon)$ and*
- *for all vertices $u, v \in V(G)$*

$$\begin{aligned}&\mathrm{dist}_G(u,v) \leq \mathrm{dist}_H(\phi(u), \phi(v)) \leq \\ &\leq (1+O(\varepsilon))\mathrm{dist}_G(u,v) + \varepsilon \cdot \min\{\mathrm{dist}_G(u,S), \mathrm{dist}_G(v,S)\}.\end{aligned}$$

The algorithm for the kSwO problem proceeds in two steps. First it computes a constant factor approximation to the problem to obtain the set S for Theorem 24. We use a result by Ahmadian and Swamy [AS16] for computing a 5-approximation in polynomial time. Note that their result works for a more general case of the kSwO problem. The approximate solution creates at most p suppliers and we add those into the set S as well. We apply Theorem 24 to the input graph with the set S to obtain a host graph H. In the second step, we use the algorithm given by Theorem 17 on graph H. The host graph H may contain vertices which do not appear in the input graph. It is straightforward to modify the algorithm to not require covering these added vertices. From the statement of Theorem 24, the treewidth of H is bounded by the number of centers k, the number of outliers p, the highway dimension h of the input graph and ε. To finish the proof of Theorem 4, we prove that the obtained solution is indeed a $(1+\varepsilon)$-approximation of the optimum solution in the original graph.

Lemma 25. *A $(1+\varepsilon)$-approximation of kSwO in the host graph H given by Theorem 24 is a $(1+O(\varepsilon))$-approximation of kSwO in the guest graph G.*

Proof. Let OPT_G and OPT_H denote the optimum solution in the input graph G and host graph H respectively. Let V_c^* be the set of clients such that there exist suppliers at distance at most $\mathrm{cost}(\mathrm{OPT}_G)$ from them, i.e. these are the clients covered by the optimum solution. For each client $u \in V_c^*$, we denote by s_u the closest supplier to u in OPT_G. We have

$$\begin{aligned}&\mathrm{cost}_H(\mathrm{OPT}_G) = \max_{u \in V_c^*} \mathrm{dist}_H(u, s_u) \leq \\ &\leq \max_{u \in V_c^*} \{(1+O(\varepsilon))\mathrm{dist}_G(u,s_u) + O(\varepsilon)\min\{\mathrm{dist}_G(u,S), \mathrm{dist}_G(s_u,S)\}\}.\end{aligned} \tag{3.1}$$

We can upper bound the inequality by maximizing over the terms separately. Then we obtain

$$\begin{aligned}&\mathrm{cost}_H(\mathrm{OPT}_G) \leq \\ &\leq (1+O(\varepsilon)) \max_{u \in V_c^*} \mathrm{dist}_G(u,s_u) + O(\varepsilon) \max_{u \in V_c^*} \min\{\mathrm{dist}_G(u,S), \mathrm{dist}_G(s_u,S)\} \leq \\ &\leq (1+O(\varepsilon))\mathrm{cost}_G(\mathrm{OPT}_G) + O(\varepsilon) \max_{u \in V_c^*} \mathrm{dist}_G(u,S).\end{aligned}$$

For the second term, there are two cases to consider. If u is covered by the approximate solution, then we have $\mathrm{dist}_G(u,S) \leq 5 \cdot \mathrm{cost}_G(\mathrm{OPT}_G)$. Otherwise u has to be an outlier in the approximate solution. We added all outliers to the set S, thus $\mathrm{dist}_G(u,S) = 0$. In total we have $\mathrm{cost}_H(\mathrm{OPT}_G) \leq (1+O(\varepsilon))\mathrm{cost}_G(\mathrm{OPT}_G)$. By the definition of OPT_H we have

$$\mathrm{cost}_H(\mathrm{OPT}_H) \leq \mathrm{cost}_H(\mathrm{OPT}_G) \leq (1+O(\varepsilon))\mathrm{cost}_G(\mathrm{OPT}_G).$$

Since the optimum solution in H is an approximate solution in G, an approximate solution in H is also an approximate solution in G. In particular, Theorem 17 gives a $(1+\varepsilon)$-approximate solution therefore we have a $(1+O(\varepsilon))^2$-approximate solution. □

The algorithm we have just shown uses as a subroutine the EPAS given by Theorem 17 for low treewidth graphs, which runs in time $O^*\left((\frac{\mathrm{tw}}{\varepsilon})^{O(\mathrm{tw})}\right)$. By Theorem 24 the host graph H, on which we run the algorithm given by Theorem 17, has treewidth bounded by $f(h,k,p,\varepsilon)$ for some computable function f. While the original authors [BKS18] do not explicitly state the function f, by carefully analysing their proof we can at least give the following upper bound. Recall that we selected the set S in Theorem 24 in such a way that its size is at most $k+p$. From the proof of Theorem 24, see [BKS18, Theorem 4], we have in the worst case that $f \in O\left((1/\varepsilon)^{\theta_S}\right)$ where $\theta_S = O\left(\log((h+|S|)^2 \log(h+|S|)\log(1/\varepsilon)+|S|)\right)$. Hence $f \in O((k+p+h)^2\log(1/\varepsilon))$. This shows that the algorithm given by Theorem 4 has a running time of at least $\Omega\left(\left(\frac{1}{\varepsilon}(k+h+p)^2\right)^{O((k+p+h)^2\log(1/\varepsilon))} \cdot n\right)$. In contrast to the $\frac{3}{2}$-approximation algorithm given by Theorem 3, which runs in time $O^*\left(2^{O(k(h\log h)+p)}\right)$, this algorithm has a significantly worse running time since the base of the exponential function in the running time is not constant but depends on k, p and the highway dimension.

3.3 EPAS for low doubling dimension graphs

In this section we present an FPT approximation scheme for the kSwO problem on low doubling dimension graphs. This algorithm is an extension of an FPT approximation scheme for the k-Center problem by Feldmann and Marx [FM20]. The result is the following theorem.

Theorem 26. *Let $\mathcal{I} = (G,k,p)$ be an instance of the kSwO problem, (V_s, dist) be the shortest-path metric induced by the suppliers V_s, and d be the doubling dimension of (V_s,dist). There exists an algorithm such that for any $\varepsilon > 0$ it outputs a solution of cost $(1+\varepsilon)\mathrm{OPT}(\mathcal{I})$ in time $O^*\left((k+p)^k \cdot \varepsilon^{-O(kd)}\right)$.*

We will present a decision algorithm which can be turned into an approximation scheme; the exact approach is described in the definition of the kCwO problem in Chapter 2. The properties of the decision algorithm are summarized by the following lemma.

Lemma 27. *Let $\mathcal{I} = (G,k,p)$ be an instance of the k-Supplier with Outliers problem, (V_s,dist) be the shortest-path metric induced by the suppliers V_s, and d be the doubling dimension of (V_s,dist). There exists an algorithm which for a cost $\varrho \in \mathbb{R}^+$ and $\varepsilon > 0$ either*

- *computes a feasible solution of cost $(1+\varepsilon)\varrho$ if $\mathcal{I}$ has a feasible solution of cost ϱ or,*
- *correctly determines that $\mathcal{I}$ does not have a solution of cost ϱ,*

running in time $O^*((k+p)^k\varepsilon^{-O(kd)})$.

Let (X, dist) be a metric. By the *aspect ratio* of a set $Y \subseteq X$ we mean the diameter of Y divided by the minimum distance between any two distinct points of Y, that is

$$\frac{\max_{u,v \in Y} \mathrm{dist}(u,v)}{\min_{u,v \in Y, u \neq v} \mathrm{dist}(u,v)}.$$

The following lemma shows that the cardinality of a subset $Y \subseteq X$ can be bounded by its aspect ratio and the doubling dimension of (X, dist). The proof of the lemma can be found in [GKL03].

Lemma 28 ([GKL03])**.** *Let (X, dist) be a metric with doubling dimension d and $Y \subseteq X$ be a subset with aspect ratio α. Then $\mathrm{dd}(Y) \leq 2 \cdot \mathrm{dd}(X)$ and $|Y| \leq 2^{d\lceil \log_2 \alpha \rceil}$.*

For a metric (X, dist), a subset $Y \subseteq X$ is called a *δ-cover* if for every $u \in X$ there is a $v \in Y$ such that $\mathrm{dist}(u,v) \leq \delta$. A *$\delta$-net* is a δ-cover with the additional property that $\mathrm{dist}(u,v) > \delta$ for all distinct points $u, v \in Y$ of the δ-cover. Observe that a δ-net can be computed greedily in polynomial time. We will need the following bound on the size of a δ-net given that there exists a feasible solution to the κSwO problem.

Lemma 29. *Let $\mathcal{I} = (G, k, p)$ be an instance of the κSwO problem, $(V_\mathrm{s}, \mathrm{dist})$ be the shortest-path metric induced by the suppliers V_s, d be the doubling dimension of $(V_\mathrm{s}, \mathrm{dist})$, $\varrho \in \mathbb{R}^+$, and $\varepsilon > 0$. Moreover assume that for each supplier $s \in V_\mathrm{s}$ there exists a client $c \in V_\mathrm{c}$ such that $\mathrm{dist}(s,c) \leq \varrho$. If there exists a feasible solution to the instance $\mathcal{I}$ of cost ϱ, then an $(\varepsilon\varrho)$-net Y of the set V_s has size at most $(k+p)\varepsilon^{-O(d)}$.*

Proof. Let V_s^* be a feasible solution of the instance $\mathcal{I}$ of cost ϱ. Let $\mathcal{B}$ be the vertices covered by the solution V_s^*, i.e. $\mathcal{B} = \bigcup_{s \in V_\mathrm{s}^*} B_s(\varrho)$, and let $V_\mathrm{c}^* = \mathcal{B} \cap V_\mathrm{c}$, that is the set of clients covered by the solution V_s^*. What remains outside the set $\mathcal{B}$ are outliers $V_\mathrm{c} \setminus V_\mathrm{c}^*$ and a (possibly empty) subset of suppliers. Due to the assumption that each supplier has at least one client at distance at most ϱ from it and that such a client is either covered by the solution V_s^* or is an outlier, we have that balls of radius 2ϱ around suppliers V_s^* and balls of radius ϱ around outliers cover the entire metric. In total the entire metric can be covered by $k+p$ balls of diameter 4ϱ. The aspect ratio of Y inside each such a ball is $4\varepsilon^{-1}$. Using Lemma 28 we have that each such a ball contains $\varepsilon^{-O(d)}$ vertices and that in total we have $|Y| \leq (k+p)\varepsilon^{-O(d)}$. □

We are now ready to prove Lemma 27.

Proof of Lemma 27. The algorithm starts by removing all suppliers s which do not have a client at distance at most ϱ from them since these suppliers cannot be in a solution of cost ϱ. Slightly abusing notation, let V_s be the set of remaining suppliers after this preprocessing step.

Then the algorithm computes an $(\varepsilon\varrho)$-net Y of the submetric $(V_\mathrm{s}, \mathrm{dist})$. If the computed net Y has more than $(k+p)\varepsilon^{-O(d)}$ points, then by Lemma 29 we know that there is no feasible solution of cost ϱ.

Assume that our instance has a feasible solution of cost ϱ and let us denote it S^*. From the properties of the net Y we have that for each point $u \in S^*$ there

exists a net point $v \in Y$ such that $\mathrm{dist}(u, v) \leq \varepsilon\varrho$. Hence if we replace each point $u \in S^*$ by its nearest net point, then we obtain a solution of cost $(1 + \varepsilon)\varrho$.

To compute a feasible solution, the algorithm tries each subset of size k of the net Y as the solution. For each such a k-tuple S, we check whether it covers sufficiently many clients, that is at least $|V_{\mathrm{c}}| - p$ of them, by balls of radius $(1+\varepsilon)\varrho$. If a solution of cost ϱ exists, then we have already argued that at least one of the k-tuples of the net will give us a feasible solution of cost $(1 + \varepsilon)\varrho$. On the other hand, if none of k-tuples of Y form a feasible solution, then the instance $\mathcal{I}$ does not have a feasible solution of cost ϱ.

It remains to bound the running time of the algorithm. The preprocessing and computing a $(\varepsilon\varrho)$-net takes polynomial time. Guessing the approximate solution takes time $\binom{|Y|}{k} = (k + p)^k \varepsilon^{-O(kd)}$ and checking feasibility of each such a guess takes polynomial time. □

4. Capacitated k-Supplier with Outliers

Let us begin by recalling the definition of the Capacitated k-Supplier with Outliers (CkSwO) problem. In the CkSwO problem, in addition to the input (G, k, p) of the kSwO problem, we have a *capacity* function $L : V_s \to \mathbb{N}_0$. A feasible solution is a subset of clients $V'_c \subseteq V_c$, a subset of suppliers $V'_s \subseteq V_s$, and a function $\phi : V'_c \to V'_s$ which maps every client of V'_c to some supplier of V'_s such that

- $|V'_c| \geq n - p$,
- $|V'_s| \leq k$,
- $|\phi(u)^{-1}| \leq L(u)$ for each $u \in V'_s$.

Informally speaking, a supplier u is only allowed to serve at most $L(u)$ clients. The cost of such a feasible solution is $\max_{u \in V'_c} \mathrm{dist}(u, \phi(u))$ and the goal is to find a feasible solution of minimum cost. Note that setting the capacity of each supplier to $|V_c|$ gives us the kSwO problem. Similarly to the kSwO problem, we can consider the decision version of this problem where the input contains an additional integer ϱ and our goal is to decide whether there exists a feasible solution of cost at most ϱ.

Cygan et al. [CHK12] have shown that under the assumption P $\neq$ NP, the CkSwO problem does not have a $(3-\varepsilon)$-approximation for any constant $\varepsilon > 0$. Cygan and Kociumaka [CK14] have shown that there is a 25-approximation algorithm for CkSwO.

4.1 EPAS for low doubling dimension graphs

In this section, we will develop an FPT approximation scheme for the CkSwO problem for low doubling dimension graphs. It follows similar steps as the algorithm given by Theorem 26 but it requires some additional modifications to support capacities. Let us restate the main result of this section.

Theorem 5. *Let $\mathcal{I} = (G, k, p, L)$ be an instance of the Capacitated k-Supplier with Outliers problem, (V_s, dist) be the shortest-path metric induced by the supplier set V_s, and d be the doubling dimension of (V_s, dist). There exists an algorithm such that for any $\varepsilon > 0$ it outputs a solution of cost $(1+\varepsilon)\mathrm{OPT}(\mathcal{I})$ in time $(k+p)^k \cdot \varepsilon^{-O(kd)} \cdot n^{O(1)}$.*

We will present a decision algorithm which can be turned into an approximation scheme; the exact approach is described in the definition of the kCwO problem in Chapter 2. The properties of the decision algorithm are summarized by the following lemma.

Lemma 30. *Let $\mathcal{I} = (G, k, p, L)$ be an instance of the Capacitated k-Supplier with Outliers problem, (V_s, dist) be the shortest-path metric induced by the supplier set V_s, and d be the doubling dimension of (V_s, dist). There exists an algorithm which for a cost $\varrho \in \mathbb{R}^+$ and $\varepsilon > 0$ either*

- *computes a feasible solution of cost* $(1+\varepsilon)\varrho$ *if* $\mathcal{I}$ *has a feasible solution of cost* ϱ*, or*
- *correctly decides that* $\mathcal{I}$ *has no solution of cost at most* ϱ*,*

running in time $O^*\left((k+p)^k \varepsilon^{-O(kd)}\right)$.

Given a set of suppliers $S \subseteq V_\mathrm{s}$ of size at most k, we cannot spontaneously assign every client to its nearest supplier in S since such an assignment might exceed the capacity of some supplier $s \in S$. However, in the next lemma we will see a polynomial algorithm which, given a set of suppliers $S \subseteq V_\mathrm{s}$ and a cost ϱ, computes a feasible solution $\phi : V_\mathrm{c} \to S$ of cost ϱ if it exists. In other words the algorithm finds a feasible solution of cost ϱ which opens the given set S as the set of suppliers if such a solution exists.

Lemma 31. *Given an instance* $\mathcal{I} = (G, k, p, L)$ *of the* CkSwO *problem, a cost* $\varrho \in \mathbb{R}^+$*, and a subset of suppliers* $S \subseteq V_\mathrm{s}$ *of size at most* k*, we can determine in polynomial time whether there exists subset of clients* $V'_\mathrm{c} \subseteq V_\mathrm{c}$ *of size at least* $n - p$*, and an assignment* $\phi : V'_\mathrm{c} \to S$ *such that* $|\phi(u)^{-1}| \leq L(u)$ *for each* $u \in S$*.*

Proof. We build a network $N = (G' = (V', E'), a, b, c)$ where $a \in V'$ is a source, $b \in V'$ is a sink, and $c : E' \to \mathbb{R}_0^+$ assigns capacities to edges of the directed graph $G' = (V', E')$. Refer to Figure 4.1 for an example of such a network. For each vertex of $S \cup V_\mathrm{c}$ we are going to create a unique vertex in V', we denote this bijection by m. For a subset of vertices $U \subseteq V_\mathrm{c} \cup S$ we denote $m(U) = \{m(u) : u \in U\}$. For each client $u \in V_\mathrm{c}$ we add a vertex v_u to V', assign $m(u) = v_u$, and add an edge between the source a and v_u with capacity 1. For each supplier $s \in S$ we add a vertex v_s to V' and assign $m(s) = v_s$. We add edges from each vertex of the set $\{m(u) : u \in V_\mathrm{c} \cap B_s(\varrho)\}$ to v_s with capacity 1. That is, we add an edge from a client vertex gadget to a supplier vertex gadget if the client is at distance at most ϱ from the supplier. We add an edge from v_s to the sink b with capacity $L(s)$. Finally we add a vertex o and add edges from each client vertex gadget to o with capacity 1 and an edge from o to the sink b with capacity p.

We claim that the desired set of clients $V'_\mathrm{c} \subseteq V_\mathrm{c}$ and the assignment $\phi : V'_\mathrm{c} \to S$ exist if and only if the maximum flow of network N is $|V_\mathrm{c}|$. That is, the size of V'_c is at least $n - p$, the assignment ϕ maps every client in V'_c to some supplier in S, and for each $u \in S$ we have $|\phi(u)^{-1}| \leq L(u)$. Then, the sets V'_c, S, and the assignment ϕ form a feasible solution of cost ϱ of the instance $\mathcal{I}$.

To prove the forward implication, let ϕ be a feasible solution of the instance $\mathcal{I}$. For each client u which is not an outlier according to ϕ, we send a unit flow on the path $(a, u, \phi(u), b)$. Since ϕ is a feasible solution, for each supplier $s \in V_\mathrm{s}$ there is at most $L(s)$ units flowing through the edge (v_s, b). For each outlier v of the solution ϕ we send a unit of flow on the path (a, v, o, b). The solution ϕ creates at most p outliers, thus the flow through the edge (o, b) is at most p, which does not exceed the capacity of the edge (o, b). There is one unit of flow going through each client, hence the flow has value $|V_\mathrm{c}|$. All edges leaving the source a are saturated and there are no incoming edges to a hence this flow is also maximum.

To prove the backward implication, let the maximum flow of the network N be $|V_\mathrm{c}|$ and let f be the corresponding flow. Assume that f assigns an integral flow

to each edge, this is without loss of generality since we can compute such a flow using the Edmonds–Karp [EK72] algorithm, and it is a well known fact that if the capacities are integral, then the flow through each edge will be integral as well. To simplify the exposition, let us remove edges through which there is no flow. From the construction of network N and from the preceding assumptions, each vertex of $m(V_c)$ has one outgoing edge and this edge has its other endpoint in $m(S) \cup \{o\}$. We construct a feasible solution ϕ to the instance $\mathcal{I}$ as follows. If the outgoing edge of the client gadget $m(u)$ of a client $u \in V_c$ ends in vertex o, then we designate u an outlier. Otherwise the outgoing edge from $m(u)$ ends in a supplier gadget $m(s)$ of some supplier $s \in V_s$, then we let $\phi(u) = s$. Since the capacity of the edge (o, b) is p there can be at most p outliers in our solution ϕ. From the construction of the network we have that the distance between a client and its assigned supplier does not exceed ϱ. Finally, the capacity of the edge $(m(s), b)$ for a supplier $s \in S$ is set to $L(s)$, hence there cannot be more than $L(s)$ clients assigned to s.

Clearly, the construction of the network N can be done in polynomial time. The aforementioned Edmonds–Karp algorithm [EK72] has a polynomial running time. Note that we have also described how to compute from a maximal flow in N a feasible solution to $\mathcal{I}$ of cost ϱ such that S is the set of opened suppliers. □

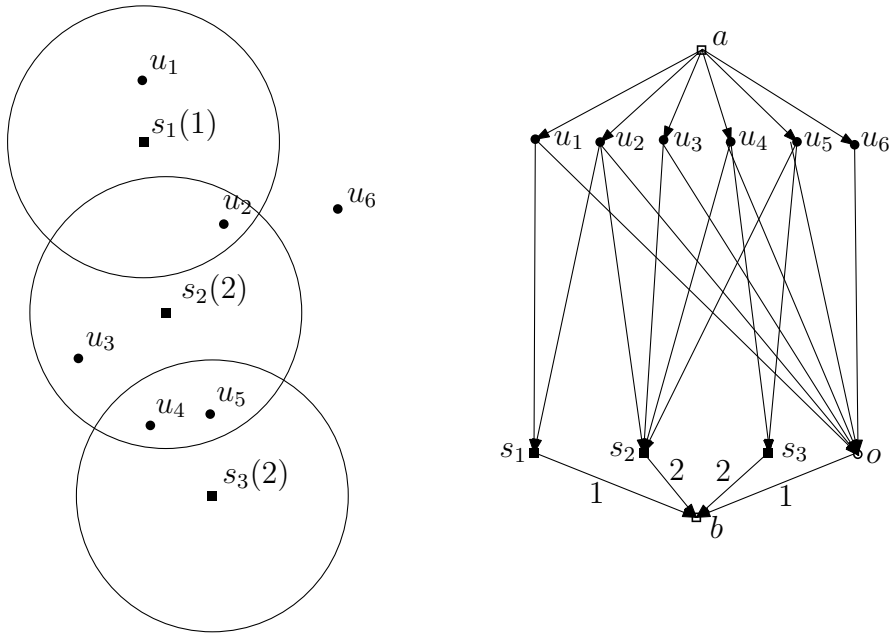


Figure 4.1: An example of a network created by Lemma 31. Vertices s_1, s_2, s_3 are suppliers and circles centered at them have radius ϱ. The numbers at edges are the capacity of the corresponding edge. Edges with unspecified capacity have capacity 1. We have capacities $L(s_1) = 1, L(s_2) = 2, L(s_3) = 2$ and $p = 1$.

Contrary to the approach used in the proof of Theorem 26, we cannot simply replace each point of a solution with its nearest net point as such an assignment might violate the capacity of some selected net point. We present an alternative approach in the following lemma.

Lemma 32. *Let $\mathcal{I} = (G, k, p, L)$ be an instance of the* CkSwO *problem such that there exists a solution ϕ^* of cost ϱ and for each supplier there exists a client at distance at most ϱ from it. Given such an instance $\mathcal{I}$, an $(\varepsilon\varrho)$-net Y of the shortest-path metric induced by $(V_\mathrm{s}, \mathrm{dist})$, and $\varepsilon > 0$, we can compute a solution of cost $(1+2\varepsilon)\varrho$ in time $O^*\left(\binom{|Y|}{k}k^k\right)$.*

Proof. Let us fix an arbitrary linear order $\preceq$ on the net points Y. For two net points $v_1, v_2 \in Y$, if $v_1 \preceq v_2$ and $v_1 \neq v_2$, then we write $v_1 \prec v_2$. For a net point $v \in Y$ we denote $P(v) = \{v' \in Y : v' \prec v\}$ and $M(v) = B_v(\varepsilon\varrho) \setminus (\bigcup_{v' \in P(v)} B_{v'}(\varepsilon\varrho))$. Let $S^* = \{\phi^*(u) : u \in V'_\mathrm{c}\}$, that is the set of suppliers opened by the solution ϕ^*. For each net point $v \in Y$, let $D(v) = |M(v) \cap S^*|$ and $R(v)$ be the set of $D(v)$ suppliers in $M(v)$ with the highest capacities. It is easy to see that for each net point $v \in Y$, the sum of capacities of $R(v)$ is at least the sum of capacities of $M(v) \cap S^*$. The sets $\{R(v) : v \in Y\}$ are disjoint by the way we defined them.

The idea is that instead of replacing each solution point $u \in S^*$ with its nearest net point $v \in Y$ like in the proof of Theorem 26, we replace it with an arbitrary point from $R(v)$ whose capacity is not yet saturated. Such a replacement will increase the cost of the solution by at most $2\varepsilon\varrho$ and since sets $\{R(v) : v \in Y\}$ are disjoint, the resulting solution will not violate the capacity of any opened supplier.

We guess a k-tuple $Y' = (y_1, \ldots, y_k)$ of elements of Y such that $y_1 \prec y_2 \prec \cdots \prec y_k$ and $S^* \subseteq \bigcup_{v \in Y'} B_v(\varepsilon\varrho)$. We also guess $D(v)$ for each $v \in Y'$. Let ϕ be our solution created by picking $D(v)$ distinct suppliers with the highest capacities among those in $R(v)$ for each $v \in Y'$. To verify whether the computed solution ϕ is feasible, we first check whether the guessed values $D(v)$ are at most $|R(v)|$ and then we invoke the algorithm given by Lemma 31 for cost $(1+2\varepsilon)\varrho$. If none of our guesses pass the feasibility check, then the instance $\mathcal{I}$ by contrapositive has no solution of cost ϱ.

The running time of our algorithm is dominated by the time required to guess Y' and the cardinalities $D(v)$ for each $v \in Y'$. From $|Y'| \leq k$, the time required to guess the k-tuple Y' is $O\left(\binom{|Y|}{k}\right)$. Since $D(v) \leq k$ for every $v \in Y$, the time required to guess $D(v)$ for each $v \in Y'$ is k^k. In total, the running time of the algorithm is $O^*\left(\binom{|Y|}{k}k^k\right)$. □

Lastly, we need a bound on the net size analogous to Lemma 29. We can view a solution ϕ^* of cost ϱ as covering the graph up to the outliers with balls of radius ϱ around opened suppliers. Hence the same lemma can be proven for the CkSwO problem as well.

We are now ready to prove Lemma 30.

Proof of Lemma 30. Analogously to the algorithm from Theorem 26 the algorithm starts by removing all suppliers which do not have a client at distance at most ϱ from them and let V_s be the set of remaining suppliers. The algorithm then computes a $(\varepsilon\varrho)$-net Y of the metric $(V_\mathrm{s}, \mathrm{dist})$. Using Lemma 29, if the net Y has more than $(k+p)\varepsilon^{-O(d)}$ points, then the algorithm concludes that the instance has no solution of cost ϱ. To compute a solution of cost $(1+2\varepsilon)\varrho$ or to show that there exists no solution of cost ϱ, we apply the algorithm given by Lemma 32.

Since the net Y has size at most $(k+p)\varepsilon^{-O(d)}$, the running time of the algorithm is $O^*\left((k+p)^k\varepsilon^{-O(kd)}\right)$. □

4.2 Hardness of parameterized approximation on low treewidth graphs

In this section we show a result about hardness of parameterized approximation of the CkSwO problem on low treewidth graphs. This result excludes using the approach we used to obtain an EPAS for the kSwO problem (Theorem 17) for low treewidth graphs through the framework of Becker et al. [BKS18]. Let us restate the main result of this section.

Theorem 6. *For any $\varepsilon > 0$ it is* W[1]*-hard to $(2-\varepsilon)$-approximate the* Capacitated k-Supplier with Outliers *problem for parameters k and treewidth.*

We will reduce from the Capacitated Dominating Set (CDS) problem. The input of CDS consists of a graph G, a capacity function $c : V(G) \to \mathbb{N}$ such that $1 \leq c(v) \leq \deg(v)$ for every vertex $v \in V$ and an integer k. A subset $D \subseteq V$ is a *capacitated dominating set* of graph G if there exists a mapping $f : (V \setminus D) \to D$ which maps every vertex in $V \setminus D$ to one of its neighbours such that the total number of vertices mapped by f to any vertex $v \in D$ does not exceed $c(v)$. Our goal is to determine whether there exists a capacitated dominating set D for G containing at most k vertices. The CDS problem was proven to be W[1]-hard parameterized by k and the treewidth of the input graph by Dom et al. [DLSV08].

Instead of directly reducing to the CkSwO problem, we will reduce to a special case of the CkSwO problem called Capacitated k-Center (CkC). In the CkC problem, the input consists of a graph $G = (V, E)$ with edge lengths $d : E \to \mathbb{R}^+$, a capacity function $L : V \to \mathbb{N}$ and an integer k. Given a subset of vertices $C \subseteq V$, a feasible solution is a function $\phi : (V \setminus C) \to C$ such that the total number of vertices mapped by ϕ to any opened center $c \in C$ does not exceed $L(c)$. The cost of a feasible solution is $\max_{u \in V} \operatorname{dist}(u, C)$ and the objective is to find a solution of minimum cost. As the CkC problem is a special case of the CkSwO problem, the W[1]-hardness of CkC that we are going to prove applies to CkSwO as well.

The reduction from CDS to CkC we present is analogous to the standard reduction (cf. [HS86, WS11]) of their uncapacitated counterparts.

Lemma 33. *For any $\varepsilon > 0$, it is* W[1]*-hard to $(2-\varepsilon)$-approximate the* CkC *problem for parameters k and treewidth.*

Proof. Let $\mathcal{I} = (G, c, k)$ be an instance of the CDS problem. We produce an instance $\mathcal{I}' = (G', L, k')$ of the CkC problem where $G' = G$ and we set the length of each edge of $E(G')$ to 1. We set $L(u) = c(u)$ for every vertex $u \in V(G)$ and $k' = k$.

Assume that $\mathcal{I}$ has a solution $D \subseteq V$ with an assignment $f : (V \setminus D) \to D$ where $|D| \leq k$. Then the set D with an assignment $\phi : (V \setminus D) \to D$ where $\phi(u) = f(u)$ for every $u \in V(G)$ is clearly a solution of $\mathcal{I}'$ of cost 1 and size $|D| \leq k'$ since $\bigcup_{u \in D} N[u] = V(G)$.

On the other hand, if $\mathcal{I}$ does not have a solution of size at most k, then we claim that any feasible solution of $\mathcal{I}'$ has cost at least 2. For contradiction, suppose $S' \subseteq V(G')$ where $|S'| \leq k'$ is a solution of $\mathcal{I}'$ of cost less than 2 with an assignment function $\phi' : (V \setminus S') \to S'$. For every pair of vertices $u, v \in V(G')$ such that $\{u, v\} \notin V(E')$ we have $\text{dist}_{G'}(u, v) \geq 2$ from the manner we assigned the edge lengths. Then for each vertex $u \in V(G')$ we have $\phi'(u) \in N(u)$ since the solution has cost less than 2. However, this implies that S' is a dominating set of $V(G)$ which is a contradiction.

Hence a $(2 - \varepsilon)$-approximation algorithm for any $\varepsilon > 0$ parameterized by k and $\text{tw}(G)$ would contradict the W[1]-hardness of CDS. □

5. Non-Uniform k-Supplier

Let us recall the definition of the Non-Uniform k-Supplier (NUkS) problem. The input is a graph G and k reals $r_1 \geq r_2 \geq \cdots \geq r_k$. A feasible solution is a tuple of k suppliers $S = (s_1, s_2, \ldots, s_k)$. The cost of a solution S is the minimum *dilation parameter* α such that

$$(B_{s_1}(\alpha \cdot r_1) \cup B_{s_2}(\alpha \cdot r_2) \cup \cdots \cup B_{s_k}(\alpha \cdot r_k)) \supseteq V_c.$$

The goal is to find a solution of a minimum cost. An alternative way of viewing the goal is to find a k-tuple of suppliers $(s_1, \ldots, s_k)$ which minimizes

$$\max_{v \in V_c} \min_{i=1}^{k} \frac{\text{dist}(v, s_i)}{r_i}.$$

When the set of clients and the set of suppliers coincide, then we have an instance of the Non-Uniform k-Center (NUkC) problem, which was first introduced by Chakrabarty et al. [CGK16]. The NUkC problem is a generalization of k-Center with Outliers: we can express an instance of kCwO by using $k + p$ centers where we set $r_1 = \cdots = r_k = 1$ and $r_{k+1} = \cdots = r_{k+p} = 0$, the optimum dilation α is the optimum cost of kCwO, and vertices which have a center of radius 0 opened in them are outliers. To the best of our knowledge, the NUkS problem has not been studied before.

5.1 Hardness results

Chakrabarty et al. [CGK16] have shown that for any constant $b \geq 1$ it is NP-hard to b-approximate the NUkC problem, even when the underlying metric is a tree metric. Their reduction uses a problem called Resource Minimization for Fire Containment on Trees (RMFC-T) defined as follows. The input is an unweighted rooted tree T with all leaves at the same distance from the root. Let L_t be the set of vertices of T which are at distance exactly t from their nearest leaf. The set L_0 are the set of leaves of T. If the height of T is h, then T_h consists only of the root of T. The goal is to select a collection of non-root vertices $N \subseteq V(T)$ such that

a) every path from the root to any leaf has at least one vertex from N, and

b) $\max_t |N \cap L_t|$ is minimized.

King and MacGillivray [KM10] show that it is NP-hard to decide whether the optimum is 1 or not. Moreover they show that this NP-hardness result holds already for trees of maximum degree 3. We extend their result to show the following hardness result for the NUkS problem.

Theorem 7. *It is NP-hard to b-approximate the Non-Uniform k-Supplier problem for any constant $b \geq 1$, even on instances with the following properties:*

- *the graph G is a rooted tree where every vertex has maximum degree three, and every leaf is at the same distance from the root,*

- *the highway dimension and the doubling dimension of G are both* 2,
- *both the supplier set V_s and the client set V_c are a subset of the leaves of G.*

We give a brief description of the reduction from RMFC-T to NUkC presented in [CGK16]. Suppose we want to exclude a b-approximation for a constant $b \geq 1$. Let T be an instance of RMFC-T such that all leaves are at the same height, and h be the height of T. We denote by L_t be the set of vertices at height t, then in particular L_0 is exactly the set of leaves of T. Our goal is to produce a NUkC instance $(G = (V, E, d), \{r_1, \ldots, r_k\})$ where d are the edge lengths of G. We set $V(G)$ to be the set of leaves L_0 of T. We use the tree T to define the edge lengths d as follows. Let $e = \{x, y\}$ be an edge of T such that $x \in L_{i-1}$ and $y \in L_i$. We assign to e the length $\ell(e) = (2b+1)^i$. For two vertices $u, v \in V(G)$, we set the length $d(u, v)$ to be the length of the shortest path between u and v in T according to edge lengths ℓ. The ball radii are defined inductively as follows: let $r_k = 0$ and $r_{k-1} = (2b+1) \cdot r_i + 2(2b+1)$.

We define $L(t)$ as the distance between a vertex at level t and a leaf in the subtree rooted at such a vertex. This is a well defined notion as all leaves are at the same distance from the root vertex. By a straightforward calculation we have $L(t) = \sum_{i=1}^{t-1}(2b+1)^i = \frac{2b+1}{2b}((2b+1)^t - 1)$. The following observation is going to be useful for future calculations.

Observation 35. *Let T be the tree created by the above reduction, e be an edge of the tree T and its endpoint with larger height be at height t. Then for any $b \geq 1$ we have $\ell(e) \geq 2b \cdot L(t-1)$.*

Proof. The following calculation proves the observation

$$\frac{\ell(e)}{L(t-1)} = \frac{2b(2b+1)^{t-1}}{(2b+1)^{t-1} - 1} \geq 2b. \qquad \square$$

To prove Theorem 7 we show that the tree T produced by the reduction has low highway dimension and doubling dimension for some appropriate constant b. We denote by $\mathcal{Q}_t$ the set of paths of non-zero length in T which have their (unique) highest vertex at height t. The following statement shows a relation between the length of a path in $\mathcal{Q}_t$ and the length of a path in $\mathcal{Q}_{t'}$ where $t' \leq t-1$.

Observation 36. *Let t and t' be two constants such that $t' \leq t-1$, $\mathcal{Q}_t \neq \emptyset$, and $\mathcal{Q}_{t'} \neq \emptyset$. Then for any $P \in \mathcal{Q}_t$ and $P' \in \mathcal{Q}_{t'}$ we have $\frac{\ell(P)}{\ell(P')} \geq b$.*

Proof. Let $P = \arg\min_{Q \in \mathcal{Q}_t} \ell(Q)$, that is the shortest path among those of $\mathcal{Q}_t$. Such a path P is a single edge which has one end at height t and the other end at height $t-1$. Let $P' = \arg\max_{Q \in \mathcal{Q}_{t'}} \ell(Q)$, that is the longest path among those of $\mathcal{Q}_{t'}$. Such a path P' has two leaves as its end vertices. Hence the length of P' is at most $2 \cdot L(t-1)$. From Observation 35 we obtain the required relation. $\square$

Lemma 37. *The tree produced by the above reduction has highway dimension* 1 *(according to Definition 8) for any $b \geq 2$.*

Proof. Consider any scale $r > 0$. Recall that $\mathcal{P}_{(r,2r]}$ is the set of shortest paths of length more that r and at most $2r$. We want to show that there exists a locally 1-sparse shortest path cover for $\mathcal{P}_{(r,2r]}$. Assume that $\mathcal{P}_{(r,2r]} \neq \emptyset$, otherwise

there is nothing to prove. Then there exists a path $P \in \mathcal{P}_{(r,2r]}$. This path has a unique lowest common ancestor at some height t. Note that $P \in \mathcal{Q}_t$.

We claim that $\mathcal{Q}_{t'} \cap \mathcal{P}_{(r,2r]} = \emptyset$ for any $t' \leq t-1$. To prove this claim, we need to show that the length of any path of $\mathcal{Q}_{t'}$ is at most r. Let $P' \in \mathcal{Q}_{t'}$, we want to show that $r \geq \ell(P')$. Using Observation 36, we have $\frac{\ell(P)}{\ell(P')} \geq b$. After rearranging, we get $\frac{\ell(P)}{b} \geq \ell(P')$. As $2r \geq \ell(P)$, we get $\frac{2r}{b} \geq \ell(P')$. We receive $r \geq \ell(P')$ for any $b \geq 2$.

We also claim that $\mathcal{Q}_{\bar{t}} \cap \mathcal{P}_{(r,2r]} = \emptyset$ for any $\bar{t} \geq t+1$. To prove this claim, we need to show that the length of any path of $\mathcal{Q}_{\bar{t}}$ is more than $2r$. Let $\bar{P} \in \mathcal{Q}_{\bar{t}}$, we want to show that $\ell(\bar{P}) > 2r$. Using Observation 36, we have $\frac{\ell(\bar{P})}{\ell(P)} \geq b$. After rearranging, we get $\ell(\bar{P}) \geq b \cdot \ell(P)$. As $\ell(P) > r$, we get $\ell(\bar{P}) > b \cdot r$. We receive $\ell(\bar{P}) > 2r$ for any $b \geq 2$.

We conclude that for any $t' \neq t$, we have $\mathcal{Q}_{t'} \cap \mathcal{P}_{(r,2r]} = \emptyset$. As $\mathcal{P}_{(r,2r]} \neq \emptyset$, this means that $\mathcal{P}_{(r,2r]} \subseteq \mathcal{Q}_t$. Thus the set L_t is a hitting set of $\mathcal{P}_{(r,2r]}$. It remains to show that it is also a sparse hitting set.

Consider any vertex $u \in V(T)$ and suppose that $|B_u(2r) \cap L_t| > 1$. Let v_1 and v_2 be a pair of vertices of $B_u(2r) \cap L_t$. In particular we have an upper bound of $2r$ on $\mathrm{dist}(u, v_1)$ and $\mathrm{dist}(u, v_2)$. Using triangle inequality, we have $\mathrm{dist}(v_1, v_2) \leq \mathrm{dist}(v_1, u) + \mathrm{dist}(u, v_2) \leq 4r$. What this shows is that the necessary condition for $|B_u(2r) \cap L_t| > 1$ is that the distance between a pair of vertices of L_t is at most $4r$.

Consider a pair of distinct vertices $u, v \in L_t$. Any path between them must pass through a vertex of L_{t+1}. Hence $\mathrm{dist}(u, v) \geq 2 \cdot (2b+1)^{i+1}$. A longest path $\widehat{P} = \arg\max_{Q \in \mathcal{Q}_t} \ell(Q)$ has two leaves as its endpoints hence its length is at most $2 \cdot L(t)$. By a similar calculation as in Observation 35, we have just shown that $\frac{\mathrm{dist}(u,v)}{\ell(\widehat{P})} \geq 2b$. From $\mathcal{P}_{(r,2r]} \subseteq \mathcal{Q}_t$, we have $2r \leq 2 \cdot \ell(\widehat{P})$. Therefore we have $\mathrm{dist}(u, v) \geq 4br$. For any $b > 1$ we have $\mathrm{dist}(u, v) > 4r$ which violates the necessary condition for $|B_u(2r) \cap L_t| > 1$. Thus by setting $b > 1$ the set L_t is a locally 1-sparse shortest path cover for scale r. □

Next we show an upper bound on the doubling dimension of the tree T. We need to show that for every $u \in V(T)$ and any $R \in \mathbb{R}^+$, the ball $B_u(2R)$ is a union of a bounded number of balls of radius R.

Lemma 38. *The tree produced by the above reduction has doubling dimension* 2 *for* $b \geq 2$.

Proof. Let $R \in \mathbb{R}^+$ and consider the ball $B_u(2R)$ for any vertex $u \in V(T)$. We denote by t the vertex with the largest height in $B_u(2R)$. For a vertex $v \in V(T)$, we denote by T_v the subtree of T rooted in vertex v.

If $t = u$, then we will show how to cover $B_u(2R)$ with four balls of radius R. Let h be the height of u and v be any vertex of $B_u(2R) \setminus \{u\}$ and $P = (u, v_1, v_2, \ldots, v)$ be a path from u to v. Note that v is a vertex of a subtree of T_u. Thus the length of the path segment $(v_1, v_2, \ldots, v)$ is at most $L(h-1)$. By Observation 35 with $b \geq 1$, the edge $\{u, v_1\}$ has length at least $\frac{1}{2}\ell(P)$ and the path segment $(v_1, v_2, \ldots, v)$ has length at most $\frac{1}{2}\ell(P)$. Since $\ell(P) \leq 2R$ and $\frac{1}{2}\ell(P) \leq R$, the path P can be covered by $B_u(R) \cup B_{v_1}(R)$ where $B_u(R) = \{u\}$. Vertex u has at most three children, let us denote them by w_1, w_2, w_3. Note

that u has three children if and only if u is the root of T. The union of four balls $B_u(R) \cup (\bigcup_{i\in\{1,2,3\}} B_{w_i}(R))$ covers $B_u(2R)$.

It remains to solve the case when $t \neq u$. Let h be the height of t and v be any vertex of $B_u(2R) \setminus \{t\}$. As $t \in B_u(2R)$, we have $\mathrm{dist}(t, v) \leq 4R$ and $v \in T_t$. We will use a similar approach as in the case $t = u$. Let $P = (t, w, \ldots, v)$ be a path from t to v. As $v \in T_t$, the length of the path segment of P between w and v is at most $L(h-1)$. By Observation 35 with $b \geq 2$, the edge $\{t, w\}$ has length at least $\frac{3}{4}\ell(P)$ and the path segment of P between w and v has length at most $\frac{1}{4}\ell(P)$. Since $\ell(P) \leq 4R$ and $\frac{1}{4}\ell(P) \leq R$, the path P can be covered by $B_t(R) \cup B_w(R)$ where $B_t(R) = \{t\}$. Vertex t has at most three children, let us denote them by w_1, w_2, w_3. The union of four balls $B_t(R) \cup (\bigcup_{i\in\{1,2,3\}} B_{w_i}(R))$ covers $B_u(2R)$. This shows that the doubling dimension of T is at most $\log_2(4) = 2$. □

Conclusion

In this work we designed efficient parameterized approximation schemes for several generalizations of the k-Supplier problem. Due to the hardness results we have summarized in the introduction, it may seem that getting an approximation scheme for parameters k and either doubling dimension or highway dimension is the "next best thing". However, these results can still be further improved. In the remainder, we outline several aspects in which one may attempt to do so.

Our $\frac{3}{2}$-approximation algorithm (Theorem 3) for the k-Center with Outliers problem runs in time exponential in the number of outliers. This result could be improved by devising a fixed-parameter approximation algorithm which is polynomial in the number of outliers. Or on the other hand, this result can be complemented by showing that it is W[1]-hard to $(2 - \varepsilon)$-approximate this problem when the number of outliers is not a parameter.

In a similar vein, the efficient parameterized approximation schemes for the k-Supplier with Outliers problem on low highway dimension graphs (Theorem 4) and low doubling dimension graphs (Theorem 5) also have running times which are FPT in the number of outliers. To improve these results, one can either devise efficient parameterized approximation schemes, parameterized by either doubling dimension or highway dimension, with a running time polynomial in the number of outliers, or show that the existence of such an algorithm would imply FPT = W[1]. An important distinction between these algorithms is their dependence on the number of outliers in the running time. The algorithm for low highway dimension graphs (Theorem 4) is exponential in the number of outliers, while the algorithm for low doubling dimension graphs (Theorem 5) has the number of outliers only in the base of the exponent. Can the algorithm for low highway dimension graphs be improved to have the number of outliers in the base of the exponent as well? Or would such an algorithm imply that FPT = W[1]?

For the Capacitated k-Supplier with Outliers problem, we have shown a parameterized approximation scheme for low doubling dimension graphs (Theorem 5). We were not able to show a parameterized approximation scheme for low highway dimension graphs. We showed that this problem does not admit an approximation scheme for low treewidth graphs (Theorem 6). This shows that the approach we used, based on the framework of Becker et al. [BKS18], to obtain a parameterized approximation scheme for (uncapacitated) k-Supplier with Outliers for low highway dimension graphs cannot be used to obtain an analogous algorithm for Capacitated k-Supplier with Outliers. It would be interesting to show that it is indeed W[1]-hard to obtain a parameterized approximation scheme when parametrizing by k and highway dimension. Such a result would show that the Capacitated k-Supplier with Outliers problem is the first example of a problem which admits an efficient parameterized approximation scheme for low doubling dimension graphs while simultaneously not admitting such an algorithm for low highway dimension graphs.

We have shown a hardness of parameterized approximation for Non-Uniform k-Supplier problem (Theorem 7). To improve this result, one may attempt to prove that the special case of Non-Uniform k-Center problem is already hard.

Bibliography

[ADF+11] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. VC-dimension and shortest path algorithms. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 690–699, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[ADF+16] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension and provably efficient shortest path algorithms. *Journal of the ACM (JACM)*, 63(5):1–26, 2016.

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 782–793, 2010.

[AS16] Sara Ahmadian and Chaitanya Swamy. Approximation Algorithms for Clustering Problems with Lower Bounds and Outliers. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69:1–69:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BFM06] Holger Bast, Stefan Funke, and Domagoj Matijevic. Transit ultrafast shortest-path queries with linear-time preprocessing. *9th DIMACS Implementation Challenge [1]*, 2006.

[BFM+07] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes. In transit to constant time shortest-path queries in road networks. In *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 46–59. Society for Industrial and Applied Mathematics, January 2007.

[BH98] Hans L Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM Journal on Computing*, 27(6):1725–1746, 1998.

[BKS18] Amariah Becker, Philip N. Klein, and David Saulpic. Polynomial-Time Approximation Schemes for k-center, k-median, and Capacitated Vehicle Routing in Bounded Highway Dimension. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Blu19] Johannes Blum. Hierarchy of transportation network parameters and hardness results. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[CFK+15] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[CGG06] Yijia Chen, Martin Grohe, and Magdalena Grüber. On parameterized approximability. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation*, pages 109–120, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[CGK16] Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The Non-Uniform k-Center Problem. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CHK12] Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 273–282. IEEE, 2012.

[CIJP14] Krishnendu Chatterjee, Rasmus Ibsen-Jensen, and Andreas Pavlogiannis. Optimal tree-decomposition balancing and reachability on low treewidth graphs. `https://research-explorer.app.ist.ac.at/record/5427`, 2014. Accessed: 2021-05-13.

[CK14] Marek Cygan and Tomasz Kociumaka. Constant Factor Approximation for Capacitated k-Center with Outliers. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 251–262, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CKMN01] Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 642–651. Society for Industrial and Applied Mathematics, 2001.

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

[DFHT05] Erik D. Demaine, Fedor V. Fomin, MohammadTaghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Trans. Algorithms*, 1(1):33–47, July 2005.

[DLSV08] Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In Martin Grohe and Rolf Niedermeier, editors, *Parameterized and Exact Computation*, pages 78–90, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[Fel19] Andreas Emil Feldmann. Fixed-parameter approximations for k-center problems in low highway dimension graphs. *Algorithmica*, 81(3):1031–1052, 2019.

[FFKP18] Andreas Emil Feldmann, Wai Shing Fung, Jochen Könemann, and Ian Post. A $(1+\varepsilon)$-embedding of low highway dimension graphs into bounded treewidth graphs. *SIAM Journal on Computing*, 47(4):1667–1704, 2018.

[FG88] Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444, 1988.

[FG01] Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM Journal on Computing*, 31(1):113–145, 2001.

[FKW04] Fedor V. Fomin, Dieter Kratsch, and Gerhard J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 245–256. Springer, 2004.

[FM20] Andreas Emil Feldmann and Dániel Marx. The parameterized hardness of the k-center problem in transportation networks. *Algorithmica*, pages 1–17, 2020.

[FSLM20] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6), 2020.

[GKL03] Anupam Gupta, Robert Krauthgamer, and James R Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 534–543. IEEE, 2003.

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

[HN79] Wen-Lian Hsu and George L Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.

[HPST19] David G Harris, Thomas Pensyl, Aravind Srinivasan, and Khoa Trinh. A lottery model for center-type problems with outliers. *ACM Transactions on Algorithms (TALG)*, 15(3):1–25, 2019.

[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.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[KLP19] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th Paschos. Structural parameters, tight bounds, and approximation for (k, r)-center. *Discrete Applied Mathematics*, 264:90–117, 2019.

[KM10] Andrew King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

[Lam14] Michael Lampis. Parameterized approximation schemes using graph widths. In *International Colloquium on Automata, Languages, and Programming*, pages 775–786. Springer, 2014.

[Mar05] Dániel Marx. Efficient approximation schemes for geometric problems? In *European Symposium on Algorithms*, pages 448–459. Springer, 2005.

[Mar08] Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[Vaz13] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

List of Figures