Proceedings Paper:

Feldmann, A.E. orcid.org/0000-0001-6229-5332 and Vu, T.A. orcid.org/0000-0002-8902-5196 (2022) Generalized k-center: distinguishing doubling and highway dimension. In: Bekos, M. A. and Kaufmann, M., (eds.) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 48th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2022, 22-24 Jun 2022, Tübingen, Germany. Lecture Notes in Computer Science, 13453 . Springer International Publishing , pp. 215-229. ISBN 978-3-031-15914-5

https://doi.org/10.1007/978-3-031-15914-5_16

Generalized k-Center: Distinguishing Doubling and Highway Dimension*

Andreas Emil Feldmann[0000−0001−6229−5332] and
Tung Anh Vu[0000−0002−8902−5196]

Faculty of Mathematics and Physics, Charles University, Czech Republic
feldmann.a.e@gmail.com, tung@kam.mff.cuni.cz

Abstract. We consider generalizations of the k-Center problem in graphs of low doubling and highway dimension. For the Capacitated k-Supplier with Outliers (CkSwO) problem, we show an efficient parameterized approximation scheme (EPAS) when the parameters are k, the number of outliers and the doubling dimension of the supplier set. On the other hand, we show that for the Capacitated k-Center problem, which is a special case of CkSwO, obtaining a parameterized approximation scheme (PAS) is W[1]-hard when the parameters are k, and the highway dimension. This is the first known example of a problem for which it is hard to obtain a PAS for highway dimension, while simultaneously admitting an EPAS for doubling dimension.

Keywords: Capacitated k-Supplier with Outliers · Highway Dimension · Doubling Dimension · Parameterized Approximation

1 Introduction

The well-known k-Center problem and its generalizations has plenty of applications, for example selecting suitable locations for building hospitals to serve households of a municipality (see [3] for a survey of healthcare facility location in practice). In this setting, the number of hospitals we can actually build is limited, e.g. by budgetary constraints. We want to choose the locations so that the quality of the provided service is optimal, and a societally responsible way of measuring the quality of service is to ensure some minimal availability of healthcare to every household. We can quantify this by measuring the distance of a household to its nearest hospital, and then minimize this distance over all households. This strategy, however, does not account for the reality that healthcare providers have (possibly different) limits on the number of patients they can serve, and thus we introduce capacity constraints. Furthermore, as the instances are given by transportation networks, we model them by the titular doubling dimension and highway dimension, which we define later.

* Andreas Emil Feldmann was supported by the project 19-27871X of GA ČR. Tung Anh Vu was supported by the project 22-22997S of GA ČR. An extended abstract of this paper appeared at the 48th International Workshop on Graph-Theoretic Concepts in Computer Science in 2022.

We formalize the problem as follows. In the Capacitated k-Supplier (CkS) problem, the input consists of a graph $G = (V, E)$ with positive edge lengths, a set $V_S \subseteq V$ of *suppliers*, a set $V_C \subseteq V$ of *clients*, a *capacity function* $L \colon V_S \to \mathbb{N}$, and an integer $k \in \mathbb{N}$. A *feasible solution* is an *assignment function* $\phi \colon V_C \to V_S$ such that $|\phi(V_C)| \leq k$ and for every supplier $u \in \phi(V_C)$ we have $|\phi^{-1}(u)| \leq L(u)$. For a pair of vertices $u, v \in V$ we denote by $\mathrm{dist}_G(u, v)$ the *shortest-path distance* between vertices u and v with respect to edge lengths of G. For a subset of vertices $W \subseteq V$ and a vertex $u \in V$, we denote $\mathrm{dist}_G(u, W) = \min_{w \in W} \mathrm{dist}_G(u, w)$. We omit the subscript G if the graph is clear from context. The *cost* of a solution ϕ is defined as $\mathrm{cost}(\phi) = \max_{u \in V_C} \mathrm{dist}(u, \phi(u))$ and we want to find a feasible solution of minimum cost. Let us mention the following special cases of CkS. If $V = V_S = V_C$, then the problem is called Capacitated k-Center (CkC). If $L(u) = \infty$ for every supplier $u \in V_S$, then the problems are called k-Supplier and k-Center respectively. It is known that k-Center is already NP-hard [28].

Two popular approaches of dealing with NP-hard problems are *approximation algorithms* [33,34] and *parameterized algorithms* [14]. Given an instance $\mathcal{I}$ of some minimization problem, a *c-approximation algorithm* computes in polynomial time a solution of cost at most $c \cdot \mathrm{OPT}(\mathcal{I})$ where $\mathrm{OPT}(\mathcal{I})$ is the optimum cost of the instance $\mathcal{I}$, and we say that c is the *approximation ratio* of the algorithm. If the instance is clear from context, we write only OPT. In a *parameterized problem*, the input $\mathcal{I}$ comes with a *parameter* $q \in \mathbb{N}$. If there exists an algorithm which computes the optimum solution in time $f(q) \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ where f is some computable function, then we call such a problem *fixed parameter tractable (FPT)* and the algorithm an *FPT algorithm*. The rationale behind parameterized algorithms is to capture the "difficulty" of the instance by the parameter q and then design an algorithm which is allowed to run in time superpolynomial in q but retains a polynomial running time in the size of the input. In this work, we focus on the superpolynomial part of the running time of FPT algorithms, so we will express $f(q) \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ as $\mathcal{O}^*(f(q))$; in particular the "$\mathcal{O}^*$" notation ignores the polynomial factor in the input size.

It is known that unless P = NP, k-Center and k-Supplier do not admit approximation algorithms with an approximation ratio better than 2 and 3 [29]. It is shown in the same work that these results are tight by giving corresponding approximation algorithms. For Capacitated k-Center, An et al. [4] give a 9-approximation algorithm, and Cygan et al. [15] show a lower bound[1] of $3 - \varepsilon$ for approximation assuming P $\neq$ NP. From the perspective of parameterized algorithms, Feldmann and Marx [23] show that k-Center is W[1]-hard in planar graphs of constant doubling dimension when the parameters are k, highway dimension and pathwidth. Under the standard assumption FPT $\subsetneq$ W[1] $\subsetneq$ W[2], this means that an FPT algorithm for k-Center in planar constant doubling dimension graphs with the aforementioned parameters is unlikely to exist. To overcome these hardness results, we will design *parameterized c-approximation algorithms*, which are algorithms with FPT runtime which output a solution of

[1] To the best of our knowledge, this reduction does not give a straightforward hardness result in the parameterized setting.

cost at most $c \cdot \mathrm{OPT}$. The approach of parameterized approximation algorithms has been studied before, see the survey in [22].

Let us discuss possible choices for a parameter for these problems. An immediate choice would be the size k of the desired solution. Unfortunately, Feldmann [20] has shown that approximating k-Center within a ratio better than 2 when the parameter is k is W[2]-hard. So to design parameterized approximation algorithms, we must explore other parameters. Guided by the introductory example, we focus on parameters which capture properties of transportation networks.

Abraham et al. [2] introduced the *highway dimension* in order to explain fast running times of various shortest-path heuristics in road networks. The definition of highway dimension is motivated by the following empirical observation of Bast et al. [5,6]. Imagine we want to travel from some point A to some sufficiently far point B along the quickest route. Then the observation is that if we travel along the quickest route, we will inevitably pass through a sparse set of "access points". Highway dimension measures the sparsity of this set of access points around any vertex of a graph. We give one of the several formal definitions of highway dimension, see [8,21]. Let (X, dist) be a metric, for a point $u \in X$ and a radius $r \in \mathbb{R}^+$ we call the set $B_u(r) = \{v \in X \mid \mathrm{dist}(u, v) \leq r\}$ the *ball of radius r centered at u*.

Definition 1 ([21]). *The* highway dimension *of a graph G is the smallest integer h such that, for some universal constant*[2] *$\gamma \geq 4$, for every $r \in \mathbb{R}^+$, and every ball $B_v(\gamma r)$ of radius γr where $v \in V(G)$, there are at most h vertices in $B_v(\gamma r)$ hitting all shortest paths of length more than r that lie in $B_v(\gamma r)$.*

We show the following hardness of parameterized approximation for CkC in low highway dimension graphs. Among the definitions of highway dimensions, the one we use gives us the strongest hardness result, cf. [8,21].

Theorem 1. *Consider any universal constant γ in Definition 1. For any $\varepsilon > 0$, there is no parameterized $((1 + \frac{1}{\gamma}) - \varepsilon)$-approximation algorithm for* CkC *with parameters k, treewidth*[3]*, and highway dimension unless* $\mathrm{FPT} = \mathrm{W}[1]$.

Another parameter we consider is *doubling dimension*, defined as follows.

Definition 2. *The* doubling constant *of a metric space (X, dist) is the smallest value λ such for every $x \in X$ and every radius $r \in \mathbb{R}^+$, there exist at most λ points $y_1, \ldots, y_\lambda \in X$ such that $B_u(r) \subseteq \cup_{i=1}^{\lambda} B_{y_i}(\frac{r}{2})$. We say that the ball $B_x(r)$ is* covered *by balls $B_{y_1}(\frac{r}{2}), \ldots, B_{y_\lambda}(\frac{r}{2})$. The* doubling dimension $\Delta(X)$ *of X is defined as $\log_2(\lambda)$. The doubling dimension of a graph is the doubling dimension of its shortest path metric.*

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 $\mathcal{O}(D)$. As a transportation network is embedded on a large sphere (namely the Earth),

[2] See [21, Section 9] for a discussion. In essence, the highway dimension of a given graph can vary depending on the selection of γ.

[3] See Appendix 5 for a formal definition.

a reasonable model is to assume that the shortest-path metric abides to the Euclidean ℓ_2-norm. Buildings in cities form 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 sensible to assume that transportation networks have constant doubling dimension.

Prior results on problems in graphs of low doubling and highway dimension went "hand in hand" in the following sense. For the k-Median problem parameterized by the doubling dimension, Cohen-Addad et al. [13] show an *efficient parameterized approximation scheme (EPAS)*, which is a parameterized algorithm that for some parameter q and any $\varepsilon > 0$ outputs a solution of cost at most $(1+\varepsilon)$OPT and runs in time $\mathcal{O}^*(f(q,\varepsilon))$ where f is a computable function. In graphs of constant highway dimension, Feldmann and Saulpic [24] follow up with a *polynomial time approximation scheme (PTAS)* for k-Median. If we allow k as a parameter as well, then Feldmann and Marx [23] show an EPAS for k-Center in low doubling dimension graphs, while Becker et al. [7] show an EPAS for k-Center in low highway dimension graphs. By using the result of Talwar [32] one can obtain *quasi-polynomial time approximation schemes (QPTAS)* for problems such as TSP, Steiner Tree, and Facility Location in low doubling dimension graphs. Feldmann et al. [21] extend this result to low highway dimension graphs and obtain analogous QPTASs. The takeaway is that approximation schemes for low doubling dimension graphs can be extended to the setting of low highway dimension graphs. In light of Theorem 1, we would then expect that CkC is also hard in graphs of low doubling dimension.

Our main contribution lies in breaking the status quo by showing an EPAS for CkC in low doubling dimension graphs. This is the first example of a problem, for which we provably cannot extend an algorithmic result in low doubling dimension graphs to the setting of low highway dimension graphs.[4] In fact, our algorithm even works in the supplier with outliers regime, where we are allowed to ignore some clients: in the Capacitated k-Supplier with Outliers (CkSwO) problem, in addition to the CkS input (G,k,L), we are given an integer p. A *feasible solution* is an *assignment* $\phi\colon V_C \to V_S \cup \{\perp\}$ which, in addition to the conditions specified in the definition of CkS, satisfies $|\phi^{-1}(\perp)| \le p$. Vertices $\phi^{-1}(\perp)$ are called *outliers*. The goal is to find a solution of minimum cost, which is defined as $\mathrm{cost}(\phi) = \max_{u \in V_C \setminus \phi^{-1}(\perp)} \mathrm{dist}(u, \phi(u))$. Facility location and clustering with outliers were introduced by Charikar et al. [11]. Among other results, they showed a 3-approximation algorithm for k-Center with Outliers and an approximation lower bound of $2-\varepsilon$. Later, Harris et al. [27] and Chakrabarty et al. [10] independently closed this gap and showed a 2-approximation algorithm for the problem. For CkSwO, Cygan and Kociumaka [16] show a 25-approximation al-

[4] We remark that for this distinction to work, one has to be careful of the used definition of highway dimension: a stricter definition of highway dimension from [1] already implies bounded doubling dimension. On the other hand, for certain types of transportation networks, it can be argued that the doubling dimension is large, while the highway dimension is small. See [24, Appendix A] for a detailed discussion.

gorithm. It may be of interest that the algorithm we show requires only that the doubling dimension of the supplier set to be bounded.

Theorem 2. *Let $\mathcal{I} = (G, k, p, L)$ be an instance of* Capacitated k-Supplier with Outliers*. Moreover, let (V_S, dist) be the shortest-path metric induced by V_S and Δ be its doubling dimension. There exists an algorithm which for any $\varepsilon > 0$ outputs a solution of cost $(1+\varepsilon)\mathrm{OPT}(\mathcal{I})$ in time $\mathcal{O}^*((k+p)^k \cdot \varepsilon^{-\mathcal{O}(k\Delta)})$.*

In light of the following results, this algorithm is almost the best we can hope for. We have already justified the necessity of approximation by the result of Feldmann and Marx [23]. An EPAS parameterized only by Δ is unlikely to exist, as Feder and Greene [19] have shown that unless P = NP, approximation algorithms with ratios better than 1.822 and 2 for two-dimensional Euclidean, resp. Manhattan metrics cannot exist. Hence it is necessary to parameterize by both k and Δ. The only improvement we can hope for is a better dependence on the number of outliers in the running time, e.g. by giving an algorithm which is polynomial in p.

Given our hardness of approximation result for CkC on low highway dimension graphs in Theorem 1 and the known EPAS for k-Center given by Becker et al. [7], it is evident that the hardness stems from the introduction of capacities. For low doubling dimension graphs we were able to push the existence of an EPAS further than just introducing capacities, by considering suppliers and outliers. It therefore becomes an interesting question whether we can show an EPAS also for low highway dimension graphs when using suppliers and outliers, but without using capacities. The following theorem shows that this is indeed possible.

Theorem 3. *Let $\mathcal{I} = (G, k, p)$ be an instance of the k-*Supplier with Outliers *problem. There exists an EPAS for this problem with parameters k, p, ε, and highway dimension of G.*

1.1 Used techniques

To prove Theorem 1, we enhance a result of Dom et al. [17] which shows that Capacitated Dominating Set is W[1]-hard in low treewidth graphs.

We prove Theorem 2 by using the concept of a δ-net which is a sparse subset of the input metric such that every input point has a net point near it. In Lemma 7 we show that the size of the net, for $\delta = \varepsilon \cdot \mathrm{OPT}$, can be bounded by parameters k, number of outliers p, doubling dimension, and ε. A naive approach, which ignores capacities of suppliers, would be to guess a k-size subset of the net which is a feasible $(1+\varepsilon)$-approximate solution. This is in fact the approach used by Feldmann and Marx [23] to show an EPAS for k-Center in low doubling dimension graphs. To support the capacities, we need further ideas regarding the structure of the solution with respect to the net. We present these in Lemmas 8 and 9.

To prove Theorem 3, we generalize the EPAS for k-Center in low highway dimension graphs by Becker et al. [7]. A major component of this algorithm is an EPAS for kSwO in low treewidth graphs, cf. Theorem 4, which generalizes an EPAS for k-Center in low treewidth graphs by Katsikarelis et al. [30].

2 Inapproximability in low highway dimension graphs

In this section we are going to prove Theorem 1, i.e. we show that there is no parameterized approximation scheme for Capacitated k-Center in graphs of low highway dimension unless FPT = W[1].

We reduce from Multicolored Clique, which is known to be a W[1]-hard problem [14, Theorem 13.25]. The input of Multicolored Clique consists of a graph G and an integer k. The vertex set of G is partitioned into *color classes* $V_1, \ldots, V_k$ where each color class is an independent set. The goal is to find a k-clique. Note that endpoints of every edge of G have to be in different color classes. Hence if a k-clique exists in G, then it has exactly one vertex in each color class.

To prove Theorem 1, we will need several settings of edge lengths. Namely for every $\lambda \geq 2\gamma \geq 8$, given an instance $\mathcal{I} = (G, k)$ of Multicolored Clique, we produce in polynomial time a CkC instance $\overline{\mathcal{I}}_\lambda = (\overline{G}, \overline{k}, L, d_\lambda)$ where the highway dimension and treewidth of $\overline{G}$ is $\mathcal{O}(k^4)$ and $\overline{k} = 7k(k-1) + 2k$. If we are not interested in a particular setting of λ or we speak generally about all instances for all possible settings of λ, we omit the subscript.

It follows from [14] that we can assume without loss of generality that every color class consists of N vertices and the number of edges between every two color classes is M. For two integers $m \leq n$ by $\langle m, n \rangle$ we mean the set of integers $\{m, m+1, \ldots, n\}$, and $\langle m \rangle = \langle 1, m \rangle$. For distinct $i, j \in \langle k \rangle$ we denote by $E_{i,j}$ the set of ordered pairs of vertices (u, v) such that $u \in V_i$, $v \in V_j$, and $\{u, v\}$ is an edge in G. When we add an (A, B)-*arrow* from vertex u to vertex v, we add A subdivided edges between u and v and additionally we add B unique vertices to the graph and connect them to v, see Fig. 1. When we *mark* a vertex u, we add $\overline{k} + 1$ new vertices to the graph and connect them to u. We denote the set of all marked vertices by Z.

We first describe the structure of $\overline{G}$ and we set the capacities and edge lengths of $\overline{G}$ afterwards. See Fig. 2 for an illustration of the reduction.

Color class gadget. For each color class V_i, we create a gadget as follows. We arbitrarily order vertices of V_i and to the j^{th} vertex $u \in V_i$ we assign numbers $u^\uparrow = j \cdot 2N^2$ and $u^\downarrow = 2N^3 - u^\uparrow$. For each vertex $u \in V_i$ we create a vertex $\overline{u}$ and we denote $\overline{V}_i = \{\overline{u} \mid u \in V_i\}$. We add a marked vertex x_i and connect it to every vertex of $\overline{V}_i$. We add a set S_i of $\overline{k} + 1$ vertices and connect each vertex of S_i to every vertex of $\overline{V}_i$. For every $j \in \langle k \rangle \setminus \{i\}$ we add a pair

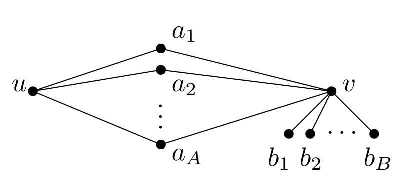

Fig. 1: The result of adding an (A, B)-arrow from u to v.

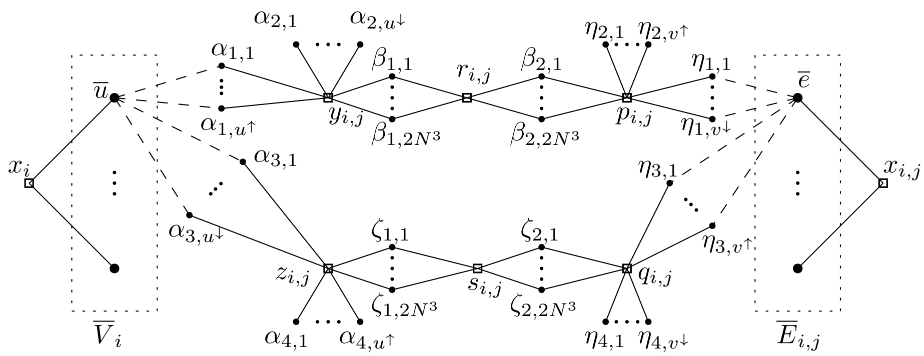

Fig. 2: Part of the reduction for color class V_i and edge set $E_{i,j}$. Vertex $\overline{e}$ represents an edge $(v, w) \in E_{i,j}$ in G. We omit the sets S_i and $S_{i,j}$. Marked vertices are drawn by boxes and we omit their $\overline{k}+1$ "private" neighbors. Edges drawn by a dashed line have length 1 and the remaining edges have length λ. We also omit the appropriate arrows from vertices of $\overline{V}_i \setminus \{\overline{u}\}$ to $y_{i,j}$ and to $z_{i,j}$, and the appropriate arrows from vertices of $\overline{E}_{i,j} \setminus \{\overline{e}\}$ to $p_{i,j}$ and to $q_{i,j}$.

of marked vertices $y_{i,j}$ and $z_{i,j}$. We denote $Y_i = \cup_{j\in\langle k\rangle\setminus\{i\}}\{y_{i,j}, z_{i,j}\}$. For every vertex $\overline{u} \in \overline{V}_i$ we add a $(u^\uparrow, u^\downarrow)$-arrow from u to each vertex of $\cup_{j\in\langle k\rangle\setminus\{i\}} y_{i,j}$ and a $(u^\downarrow, u^\uparrow)$-arrow from u to each vertex of $\cup_{j\in\langle k\rangle\setminus\{i\}} z_{i,j}$.

Edge set gadget. For every $i \in \langle k-1\rangle, j \in \langle i+1, k\rangle$ we create a gadget for the edge set $E_{i,j}$ as follows. For every edge $e \in E_{i,j}$ we create a vertex $\overline{e}$ and we denote $\overline{E}_{i,j} = \{\overline{e} \mid e \in E_{i,j}\}$. We add a marked vertex $x_{i,j}$ and connect it to every vertex of $\overline{E}_{i,j}$. We add a set $S_{i,j}$ of $\overline{k}+1$ vertices and connect each vertex of $S_{i,j}$ to every vertex of $\overline{E}_{i,j}$. We add four marked vertices $p_{i,j}, p_{j,i}, q_{i,j}, q_{j,i}$. Consider an edge $e = (u, v) \in E_{i,j}$, we connect $\overline{e}$ to $p_{i,j}$ with a $(u^\downarrow, u^\uparrow)$-arrow, to $q_{i,j}$ with a $(u^\uparrow, u^\downarrow)$-arrow, to $p_{j,i}$ with a $(v^\downarrow, v^\uparrow)$-arrow, and to $q_{j,i}$ with a $(v^\uparrow, v^\downarrow)$-arrow. We denote $\mathcal{S} = (\cup_{i\in\langle k\rangle} S_i) \cup (\cup_{i\in\langle k-1\rangle, j\in\langle i+1,k\rangle} S_{i,j})$.

Adjacency gadget. To connect the color class gadgets and the edge set gadgets, for every distinct $i, j \in \langle k\rangle$ we add marked vertices $r_{i,j}$ and $s_{i,j}$, and we add $(2N^3, 0)$-arrows from $y_{i,j}$ to $r_{i,j}$, from $p_{i,j}$ to $r_{i,j}$, from $z_{i,j}$ to $s_{i,j}$, and from $q_{i,j}$ to $s_{i,j}$.

Capacities. We now describe the capacities $L\colon V(\overline{G}) \to \mathbb{N}$. To streamline the exposition, we will assume that each vertex of $\phi(V(\overline{G}))$ covers itself at no "cost" with respect to the capacity. For every two distinct $i, j \in \langle k\rangle$, the vertex x_i has capacity $N - 1 + \overline{k} + 1$, the vertex $x_{i,j}$ has capacity $M - 1 + \overline{k} + 1$, vertices $y_{i,j}$ and $z_{i,j}$ have capacity $2N^4 + \overline{k} + 1$, vertices $p_{i,j}$ and $q_{i,j}$ have capacity $2MN^3 + \overline{k} + 1$, vertices $r_{i,j}$ and $s_{i,j}$ have capacity $2N^3 + \overline{k} + 1$, and the remaining vertices have capacity equal to their degree.

Edge lengths. Given $\lambda \in \mathbb{R}^+$ we set edge lengths $d_\lambda \colon E(\overline{G}) \to \mathbb{R}$ as follows. For every $i \in \langle k \rangle$ and every vertex $\overline{u} \in \overline{V}_i$, we assign length 1 to edges between $\overline{u}$ and $N(\overline{u}) \setminus (\{x_i\} \cup S_i)$, i.e. the set of vertices originating from subdivided edges of arrows between $\overline{u}$ and Y_i. Similarly for every $i \in \langle k-1 \rangle$, $j \in \langle i+1, k \rangle$ and every vertex $\overline{v} \in \overline{E}_{i,j}$, we assign length 1 to edges between $\overline{v}$ and $N(\overline{v}) \setminus (\{x_{i,j}\} \cup S_{i,j})$. To the remaining edges we assign length λ.

From the way we assign edge lengths d_λ, in a solution ϕ of cost $\lambda + 1$ such that $|\phi(V(\overline{G}))| \le \overline{k}$, it must be the case that $Z \subseteq \phi(V(\overline{G}))$, since Z is the set of marked vertices with $\overline{k}+1$ private neighbors.

Dom et al. [17, Observation 1] observe that $\mathrm{tw}(\overline{G}) = \mathcal{O}(k^4)$: The size of $\mathcal{S}$ is $\mathcal{O}(k^4)$ and the size of Z is $\frac{1}{2}(13k^2 - 11k)$. Removing $\mathcal{S} \cup Z$ leaves us with a forest and the observation follows.

We prove that $\overline{G}$ has bounded highway dimension. For an interval of reals I, we will consider shortest paths of $\overline{G}$ such that their length belongs to I, thus we accordingly denote this set by $\mathcal{P}_I$. We also denote $\mathcal{V} = (\cup_{i=1}^{k} \overline{V_i}) \cup (\cup_{1 \le i < j \le k} \overline{E}_{i,j})$.

Lemma 1. *For any $\lambda \ge 2\gamma$, where γ is the universal constant in Definition 1, graph $\overline{G}$ with edge lengths d_λ has highway dimension $h(\overline{G}) \in \mathcal{O}(k^4)$.*

Proof. Consider a scale $r \ge 2$ and any shortest path $\pi \in \mathcal{P}_{(r,\infty)}$. Let $H = \mathcal{S} \cup Z$. From the way we constructed $\overline{G}$, the maximal length of a shortest path which does not contain an edge of length λ is 2. Thus π has to contain an edge of length λ. Every edge of length λ is adjacent to a vertex of H, which means that H forms a hitting set for $\mathcal{P}_{(r,\infty)}$. We have $|\mathcal{S} \cup Z| = \mathcal{O}(k^4)$, and this trivially bounds $|B_u(\gamma r) \cap H| = \mathcal{O}(k^4)$ for any vertex $u \in V(\overline{G})$.

Now consider the case $r < 2$. Let $H = \mathcal{S} \cup \mathcal{V} \cup Z$. We have already seen that every edge of length λ is incident to $\mathcal{S} \cup Z$. Every edge of length 1 is incident to a vertex of $\mathcal{V}$, hence the set H hits all shortest paths $\mathcal{P}_{(r,\infty)}$. We have $\gamma r < \lambda$, as $\lambda \ge 2\gamma$. This means that if $B_u(\gamma r)$ contains a vertex v other than u, then $\mathrm{dist}(u,v) < \lambda$. From the construction of the graph, it then must be the case that $\mathrm{dist}(u,v) = 1$, and in fact $\{u,v\} \in E(\overline{G})$ and $d_\lambda(u,v) = 1$. Let $F = \{e \in E(\overline{G}) \mid d_\lambda(e) = 1\}$. We have to bound the size of $B_u(\gamma r) \cap H$ where u is an endpoint of an edge in F. We will show that the bound is 1.

For every edge of F, one of its endpoints is a vertex $u \in \mathcal{V}$ and the other is a vertex which belongs to a vertex originating from a subdivided edge of an arrow incident to u. In the first case, a shortest path between any pair of vertices of $\mathcal{V}$ has to contain two edges of length λ. Thus the distance between them is at least 2λ. Any shortest path from a vertex of $\mathcal{V}$ to a vertex of $\mathcal{S} \cup Z$ contains an edge of length λ. Thus for a vertex $u \in \mathcal{V}$, we have $B_u(\gamma r) \cap H = \{u\}$ as $\gamma r < \lambda$. In the second case, a vertex v belonging to a subdivided edge has degree two and exactly one of the edges incident to it has length 1; the other has length λ, which is not consequential in the current case as $\gamma r < \lambda$. The neighbor w of v such that $d_\lambda(v,w) = 1$ belongs to $\mathcal{V}$. This means that $|B_v(\gamma r) \cap H| = 1$ since $B_v(\gamma r) = \{v,w\}$, i.e. we have considered all the possible cases. □

Now we prove that $\mathcal{I}$ contains a k-clique if and only if $\overline{\mathcal{I}}_\lambda$ contains a solution of cost at most $\lambda+1$. As d_λ assigns edge lengths 1 and λ, this will imply that if $\mathcal{I}$

does not contain a k-clique, then any solution of $\overline{\mathcal{I}}$ has to have cost at least $\lambda+2$ and vice versa.

The forward implication follows implicitly from the original result of Dom et al. [17], since we can interpret a capacitated dominating set as a solution of cost λ, hence we omit the proof.

Lemma 2 ([17, Lemma 1]). *If $\mathcal{I}$ contains a k-clique, then $\overline{\mathcal{I}}_\lambda$ contains a solution of cost λ.*

To prove the backward implication, we start need to show that a solution of cost $\lambda+1$ has to open a vertex in every $\overline{V}_i$ and every $\overline{E}_{i,j}$. In contrast to Lemma 2, the backward implication does not simply follow from the original result since we have added edge lengths to the graph.

Lemma 3. *Let ϕ be a solution of $\overline{\mathcal{I}}_\lambda$ of cost $\lambda+1$, and $D=\phi(V(\overline{G}))$. Then for each $i \in \langle k\rangle$ we have $|\overline{V}_i \cap D| = 1$ and for each $i \in \langle k-1\rangle, j \in \langle i+1, k\rangle$ we have and $|\overline{E}_{i,j} \cap D| = 1$.*

Proof. We prove this statement by contradiction. Suppose that there exists i such that $|\overline{V}_i \cap D| \neq 1$. We first consider the case when $|\overline{V}_i \cap D| = 0$. Vertices of S_i have to be covered by vertices at distance at most $\lambda+1$ from them excluding those belonging to $\overline{V}_i$. Let W be the set of vertices originating from subdivided edges of arrows between $\overline{V}_i$ and Y_i. Then it must be the case that $\phi^{-1}(S_i) \subseteq S_i \cup W$ as $(\cup_{u\in S_i} B_u(\lambda+1)) \setminus \overline{V_i} = S_i \cup W$. The distance between a pair of vertices of S_i is 2λ, which is strictly greater than $\lambda+1$ since $\lambda \geq 2\gamma \geq 8$. Hence for a vertex $u \in (D \cap S_i)$ we have $|\phi^{-1}(u) \cap S_i| \leq 1$. For a vertex $w \in W$ we have $L(w) = 2$, and so $|\phi^{-1}(w) \cap S_i| \leq 2$. In total, we would have to pick at least $\frac{k+1}{2}$ vertices of $S_i \cup W$ to cover S_i. We have $|Z| > \overline{k} - \frac{k+1}{2}$ and D must contain Z. As $Z \cap (S_i \cup W) = \emptyset$, this contradicts the fact that $|D| \leq \overline{k}$ and thus $|\overline{V}_i \cap D| \geq 1$.

In the case that there exist i and j such that $|\overline{E}_{i,j} \cap D| = 0$, we can apply a similar argument to show that $|\overline{E}_{i,j} \cap D| \geq 1$

We have shown that every $\overline{V}_i$ and every $\overline{E}_{i,j}$ contains at least one vertex of D. By calculating $\overline{k} - |Z| = k + \binom{k}{2}$, which is equal to the number of color classes plus the number of edge sets, it follows that each of them must contain exactly 1 vertex of D as $Z \subset D$. ◻

Now we show that if $\overline{\mathcal{I}}_\lambda$ contains a solution of cost $\lambda+1$, then $\mathcal{I}$ contains a k-clique.

Lemma 4. *If $\overline{\mathcal{I}}_\lambda$ has a solution ϕ of cost $\lambda+1$, then $\mathcal{I}$ contains a k-clique.*

Proof. Let $D = \phi(V(\overline{G}))$. For $i \in \langle k\rangle$ let $\overline{u}_i$ be the vertex of $\overline{V}_i \cap D$ and for $i \in \langle k-1\rangle, j \in \langle i+1, k\rangle$ let $\overline{e}_{i,j}$ be the vertex of $\overline{E}_{i,j} \cap D$. These vertices are well-defined by Lemma 3. To prove that these vertices encode a k-clique in G, we want to show for every $i \in \langle k-1\rangle, j \in \langle i+1, k\rangle$ that vertices u_i and u_j, which correspond to $\overline{u}_i$ and $\overline{u}_j$ respectively, are incident to the edge $e_{i,j}$ corresponding to the vertex $\overline{e}_{i,j}$. We will only present the proof of incidence for u_i and $e_{i,j}$, for u_j we can proceed analogously. Let v be the vertex of the edge $e_{i,j}$ which belongs to V_i in G. Before we prove that u_i and $e_{i,j}$ are incident, we first argue that $u_i^{\uparrow} + v^{\downarrow} = 2N^3$.

We prove this statement by contradiction. First suppose $u_i^{\uparrow} + v^{\downarrow} < 2N^3$. Then $u_i^{\uparrow} + v^{\downarrow} \leq 2N^3 - 2N^2$ as for every two distinct vertices w_1 and w_2 of a color class we have $|w_1^{\uparrow} - w_2^{\uparrow}| \geq 2N^2$ and $|w_1^{\downarrow} - w_2^{\downarrow}| \geq 2N^2$. Consider the set

$$\mathcal{T} = (N(y_{i,j}) \cup N(r_{i,j}) \cup N(p_{i,j})) \setminus \phi^{-1}(\{x_i, u_i, e_{i,j}, x_{i,j}\}). \tag{1}$$

It follows that in a solution of cost $\lambda + 1$, vertices of $\mathcal{T}$ must be covered by $y_{i,j}$, $r_{i,j}$ or $p_{i,j}$ as edges of $\overline{G}$ have length 1 or λ. We have $L(y_{i,j}) + L(r_{i,j}) + L(p_{i,j}) = 2N^4 + 2MN^3 + 2N^3 + 3(\overline{k} + 1)$. However,

$$\begin{aligned} |\mathcal{T}| &\geq 2N^4 + 2MN^3 + 4N^3 + 3(\overline{k} + 1) - ((2N^3 - 2N^2) + (N - 1) + (M - 1)) \\ &> 2N^4 + 2MN^3 + 2N^3 + 3(\overline{k} + 1), \end{aligned} \tag{2}$$

where we used that $M \leq N^2$ and $N^2 > N$. Thus $y_{i,j}$, $r_{i,j}$, and $p_{i,j}$ cannot cover $\mathcal{T}$. This contradicts the fact that ϕ is a solution of cost $\lambda + 1$.

If $u_i^{\uparrow} + v^{\downarrow} > 2N^3$, then $u_i^{\downarrow} + v^{\uparrow} < 2N^3$ and we can apply the identical argument for vertices $z_{i,j}, s_{i,j}, q_{i,j}$.

It remains to prove that u_i is incident to $e_{i,j}$. Again, let v be the vertex of $e_{i,j}$ which lies in V_i of G. We know from the preceding argument that $u_i^{\uparrow} + v^{\downarrow} = 2N^3$. However, the only vertex $w \in V_i$ such that $w^{\downarrow} = 2N^3 - u_i^{\uparrow}$ is u_i itself; for any $w \in V_i \setminus \{u_i\}$ we would have $|(u_i^{\uparrow} + w^{\downarrow}) - 2N^3| \geq 2N^2$. Hence $v = u_i$ and so $e_{i,j}$ is incident to u_i. This concludes the proof. □

We are ready to prove Theorem 1.

Proof (of Theorem 1). Let $\gamma \geq 4$ be the universal constant in Definition 1, and $\lambda = 2\gamma$. For contradiction, suppose that there exists an algorithm $\mathcal{A}$ with an approximation ratio $c = \left((1 + \frac{1}{\gamma}) - \varepsilon\right)$ for some fixed $\varepsilon > 0$ parameterized by k, treewidth and highway dimension.

Given a Multicolored Clique instance $\mathcal{I} = (G, k)$, we produce a CkC instance $\overline{\mathcal{I}}_\lambda = (\overline{G}, \overline{k}, L)$ with edge lengths d_λ using the reduction from Section 2. We run algorithm $\mathcal{A}$ on the instance $\overline{\mathcal{I}}$, which takes time $\mathcal{O}^*(f(\overline{k}, \mathrm{tw}(\overline{G}), h(\overline{G})))$ for some computable function f. We have $\overline{k} = 7k(k-1) + 2k$. By the observation of Dom et al. [17], graph $\overline{G}$ has treewidth $\mathcal{O}(k^4)$. By Lemma 1, graph $\overline{G}$ has highway dimension $\mathcal{O}(k^4)$. Thus there exists a computable function g such that $f(\overline{k}, \mathrm{tw}(\overline{G}), h(\overline{G})) \leq g(k)$. If $\mathcal{I}$ is a YES-instance, then $\overline{\mathcal{I}}$ has a solution of cost λ by Lemma 2. On the other hand, if $\mathcal{I}$ is a NO-instance, then any solution of $\overline{\mathcal{I}}$ has to have cost at least $\lambda + 2$, which follows from Lemma 4. Algorithm $\mathcal{A}$ is able to distinguish between these two cases in time $\mathcal{O}^*(f(\overline{k}, \mathrm{tw}(\overline{G}), h(\overline{G}))) = \mathcal{O}^*(g(k))$, since $c < \frac{\lambda+2}{\lambda}$. This is an FPT algorithm for a W[1]-hard problem, which contradicts FPT $\neq$ W[1]. □

3 EPAS on graphs of bounded doubling dimension

In this section we prove Theorem 2, i.e. we show the existence of an EPAS for CkSwO on instances where the supplier set has bounded doubling dimension. To be more precise, we develop a decision algorithm which, given a cost $\varrho \in \mathbb{R}^+$, and $\varepsilon > 0$, computes a solution of cost $(1+\varepsilon)\varrho$ in FPT time with parameters k, p, doubling dimension and ε. Formally, the result is the following lemma.

Lemma 5. *Let $\mathcal{I} = (G, k, p, L)$ be a* CkSwO *instance. Moreover, let (V_S, dist) be the shortest-path metric induced by V_S and Δ be its doubling dimension. There exists an algorithm which, given a cost $\varrho \in \mathbb{R}^+$ and $\varepsilon > 0$, either*

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

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

Using Lemma 5, we can obtain the algorithm of Theorem 2 as follows. We can first assume without loss of generality, that $V_C \cup V_S = V(G)$. Suppose that we can guess the optimum cost OPT of any CkSwO instance. By using OPT as ϱ in Lemma 5, we can output a solution of cost $(1+\varepsilon)$OPT. To guess the optimum cost OPT, observe that OPT must be one of the inter-vertex distances. Hence the minimum inter-vertex distance ϱ for which the algorithm outputs a solution has the property that $\varrho \leq \text{OPT}$ and consequently $(1+\varepsilon)\varrho \leq (1+\varepsilon)\text{OPT}$.

The main ingredient of the algorithm is the notion of a δ-net. For a metric (X, dist), a subset $Y \subseteq X$ is called a *δ-cover* if for every $u \in X$ there exists a $v \in Y$ such that $\text{dist}(u, v) \leq \delta$. If a δ-cover Y has an additional property that for every two distinct $u, v \in Y$ we have $\text{dist}(u, v) > \delta$, then we say that Y is a *δ-net*. Observe that a δ-net can be computed greedily in polynomial time.

Let us give the main idea behind the algorithm. Given an instance of the problem and $\varepsilon > 0$, let ϕ^* be an optimum solution of cost OPT, V_C^* be clients that are not outliers according to ϕ^*, i.e. $V_C^* = \{u \in V_C \mid \phi^*(u) \neq \bot\}$, and Y be an $(\varepsilon \cdot \text{OPT})$-net of the metric (V_S, dist). Consider an assignment function ϕ constructed as follows. For each client $u \in V_C^*$ we set $\phi(u)$ to the nearest point of Y to $\phi^*(u)$, and for the remaining clients we set the value of ϕ to $\bot$. If for every selected supplier $s \in (\phi(V_C) \setminus \{\bot\})$ we have $|\phi^{-1}(s)| \leq L(s)$, then ϕ is a feasible solution. Since Y is an $(\varepsilon \cdot \text{OPT})$-net, the cost of ϕ is at most $(1+\varepsilon)$OPT.

The main obstacle to implementing an algorithm from this idea is that we do not know the optimum solution ϕ^*. However, by the definition of the net Y, we know that each selected supplier $\phi^*(V_C^*)$ is near some point of Y. If Y was not too large, we could guess which k of its points are near to every supplier of $\phi^*(V_C^*)$. Later, we will also show how to ensure that the solution we create respects capacities of suppliers we pick.

We now show how to bound the size of the net. Let (X, dist) be a metric of doubling dimension Δ, by the *aspect ratio* of a set $X' \subseteq X$, we mean the diameter of X' divided by the minimum distance between any two distinct points of X', that is $\frac{\max_{u,v \in X'} \text{dist}(u,v)}{\min_{u,v \in X', u \neq v} \text{dist}(u,v)}$. The following lemma by Gupta et al. [26] shows that the cardinality of a subset $X' \subseteq X$ can be bounded by its aspect ratio and Δ.

Lemma 6 **([26]).** *Let* (X, dist) *be a metric and* Δ *its doubling dimension. Consider a subset* $X' \subseteq X$ *of aspect ratio* α *and doubling dimension* Δ'*. Then it holds that* $\Delta' = \mathcal{O}(\Delta)$ *and* $|X'| \leq 2^{\mathcal{O}(\Delta \lceil \log_2 \alpha \rceil)}$*.*

Using Lemma 6, we bound the size of Y.

Lemma 7. *Let* $\mathcal{I} = (G, k, p, L)$ *be an instance of the* CkSwO *problem,* $\varepsilon > 0$*, and* $\varrho \in \mathbb{R}^+$ *a cost. Moreover let* (V_S, dist) *be the shortest-path metric induced by* V_S *and* Δ *its doubling dimension. Assume that for each supplier* $s \in V_S$ *there exists a client* $c \in V_C$ *such that* $\mathrm{dist}(s, c) \leq \varrho$*. If* $\mathcal{I}$ *has a feasible solution* ϕ *with* $\mathrm{cost}(\phi) \leq \varrho$*, then an* $(\varepsilon\varrho)$*-net* Y *of* V_S *has size at most* $(k+p)\varepsilon^{-\mathcal{O}(\Delta)}$*.*

Before we prove the lemma, let us make a few comments the statement of the lemma. We do not know the cost of the optimum solution and we are merely guessing it. Hence we need to also consider the case when our guess on the cost ϱ is wrong, i.e. it is less than the cost of the optimum solution. The requirement that every supplier has a client nearby is a natural one: if we assume that our solution has cost ϱ and a supplier s has $\mathrm{dist}(s, V_C) > \varrho$, then it will never be picked in a solution. Thus we can without loss of generality remove all such suppliers from the input.

Proof (of Lemma 7). Let $V_C' = \{u \in V_C \mid \phi(u) \neq \bot\}$ and $V_S' = \phi(V_C')$. Since ϕ is a solution of cost ϱ, all clients of V_C' can be covered by balls of radius ϱ around vertices of V_S' and there are k such balls. To cover the outliers $V_C \setminus V_C'$, we use the fact that there are at most p of them. Thus we place a ball of radius ϱ centered at each outlier. Since every supplier is at distance at most ϱ from some client and every client lies in one of the balls of radius ϱ we have placed, by increasing the radius of every placed ball to 2ϱ we cover all suppliers as well.

In total, $V_C \cup V_S$ can be covered by $k + p$ balls of diameter at most 4ϱ. For each such ball, let us consider the aspect ratio of points of the net Y which lie in it. Since the diameter is at most 4ϱ and the distance of every two points of a net is more than $\varepsilon\varrho$, the aspect ratio is at most $\frac{4}{\varepsilon}$. By Lemma 6 each of the balls contains at most $\varepsilon^{-\mathcal{O}(\Delta)}$ points of Y. Thus $|Y| \leq (k+p)\varepsilon^{-\mathcal{O}(\Delta)}$. □

When we gave the intuition behind the algorithm, we assumed that the derived solution ϕ, which replaces every optimum supplier of $\phi^*(V_C) \setminus \{\bot\}$ by its nearest net point, does not violate the capacity of any selected net point, i.e. for every $s \in \phi(V_C) \setminus \{\bot\}$ we have $|\phi^{-1}(s)| \leq L(s)$. This does not have to be the case, so instead of replacing every optimum supplier by its nearest net point, we need to select the replacement net point in a more sophisticated manner, in particular to avoid violating the capacity of the replacement net point.

Let V_S^* be the optimum supplier set corresponding to the optimum assignment function ϕ^*. Suppose that we are able to guess a subset $S^* \subseteq Y$ of size k such that for every supplier $u \in V_S^*$ we have $\mathrm{dist}(u, S^*) \leq \varepsilon\varrho$. Let $A\colon V_S^* \to S^*$ map each optimum supplier to its nearest net point. As we have discussed, we cannot just replace each supplier $u \in V_S^*$ by $A(u)$ since it may happen that $|(\phi^*)^{-1}(u)| > L(A(u))$. However, there is a supplier in the ball $B_{A(u)}(\varepsilon\varrho)$

which is guaranteed to have capacity at least $L(u)$ since $u \in B_{A(u)}(\varepsilon\varrho)$ Thus we can implement the "replacement step" by replacing each optimum supplier by the supplier of highest capacity in $B_{A(u)}(\varepsilon\varrho)$ and this increases the cost of the optimum solution by at most $2\varepsilon\varrho$, i.e. the diameter of the ball.

We must also consider the case when $|A^{-1}(v)| > 1$ for some net point $v \in S^*$. Generalizing the previous idea, we replace suppliers $A^{-1}(v)$ by $|A^{-1}(v)|$ suppliers of $B_v(\varepsilon\varrho)$ with the highest capacities. As we do not know the optimum solution, we do not know $|A^{-1}(v)|$ either. Nevertheless, we know that $|S^*| \le k$, and so we can afford to guess these values after guessing the set S^*.

The final ingredient we need is the ability to verify our guesses. That is, given a set of at most k suppliers, we need to check if there exists a feasible solution of a given cost which assigns clients to a prescribed set of suppliers. We do so by a standard reduction to network flows.

Lemma 8. *Given a* CkSwO *instance* $\mathcal{I} = (G, k, p, L)$*, a cost* $\varrho \in \mathbb{R}^+$*, and a subset* $S \subseteq V_S$*, we can determine in polynomial time whether there exists an assignment* $\phi\colon V_C \to S$ *such that* $|\phi(u)^{-1}| \le L(u)$ *for each* $u \in S$*,* $|\phi^{-1}(\bot)| \le p$*, and* $\operatorname{cost}(\phi) \le \varrho$*.*

Proof. We build a network $N = (G' = (V', E'), a, b, c)$ where $a \in V'$ is the source, $b \in V'$ is the sink, and $c : E' \to \mathbb{R}_0^+$ assigns capacities to edges of the directed graph $G' = (V', E')$. Refer to Fig. 3 for an example of such a network. For each vertex of $S \cup V_C$ we are going to create a unique vertex in V', we denote this bijection by m. For a subset $U \subseteq S \cup V_C$ we denote $m(U) = \{m(u) \mid u \in U\}$. For each client $u \in V_C$ we add a vertex v_u to V', assign $m(u) = v_u$, and add an edge from a to 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 $\{m(u) \mid u \in V_C \cap B_s(\varrho)\}$ to v_s with capacity 1. We add an edge from v_s to b with capacity $L(s)$. Finally we add a vertex o and add edges from each $m(V_C)$ to o with capacity 1 and an edge from o to b with capacity p.

We claim that the desired assignment $\phi : V'_C \to S$ exists if and only if the maximum flow of N is $|V_C|$.

To prove the forward implication, let ϕ be a feasible solution of the instance $\mathcal{I}$. For each client u such that $\phi(u) \ne \bot$, we send a unit flow on the path $(a, m(u), m(\phi(u)), b)$. As ϕ is a feasible solution, for each $s \in V_S$ there are at most $L(s)$ units flowing through the edge $(m(s), b)$. For each outlier $v \in \phi^{-1}(\bot)$ we send a unit of flow on the path $(a, m(v), o, b)$. 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 $|V_C|$. 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 $|V_C|$ 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 [18] algorithm, and it is a well known fact that if the capacities are integral, then the flow through each edge will be integral

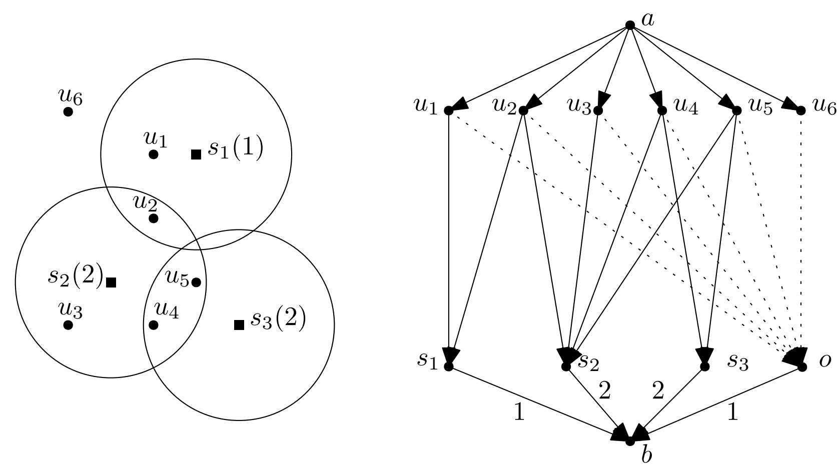

Fig. 3: On the left, we have a CkSwO instance with one allowed outlier. Vertices s_1, s_2, and s_3 are suppliers and circles centered at them have radius ϱ. The remaining vertices are clients. The capacities are $L(s_1) = 1$, $L(s_2) = 2$, $L(s_3) = 2$. On the right, we have the result of applying Lemma 8 on the instance in the left figure. Edges with unspecified capacity have capacity 1.

as well. To simplify the exposition, let $E' = E' \setminus \{e \in E' \mid f(e) = 0\}$, that is we remove all edges through which there is no flow. From the construction of network N and 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 from $m(u)$ of a client $u \in V_C$ ends in o, then we set $\phi(u) = \perp$. Otherwise the outgoing edge from $m(u)$ ends in $m(s)$ of some supplier $s \in V_S$, then we set $\phi(u) = s$. 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, the distance between a client and its assigned supplier is at most 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 [18] has a polynomial running time. Note that we have also described how to compute a feasible solution to $\mathcal{I}$ of cost ϱ such that S is the set of opened suppliers from a maximal flow in N. $\square$

We now prove the correctness of the replacement strategy.

Lemma 9. *Let $\mathcal{I} = (G, k, p, L)$ be a* CkSwO *instance 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 an $(\varepsilon\varrho)$-net Y of the shortest-path metric induced by V_S, and $\varepsilon > 0$, we can compute a solution of cost $(1 + 2\varepsilon)\varrho$ in time $\mathcal{O}^*\left(\binom{|Y|}{k} k^k\right)$.*

Proof. Let $V_S^* = \phi^*(V_C) \setminus \{\perp\}$. For an optimum supplier $u \in V_S^*$ it may happen that $|B_u(\varepsilon\varrho) \cap Y| > 1$, i.e. it is close to more than one net point. This may cause issues when we guess for each net point $v \in Y$ the size of $B_v(\varepsilon\varrho) \cap V_S^*$. To circumvent this problem, we fix a linear order $\preceq$ on the set of net points Y and we assign each optimum supplier to the first close net point. Formally, we define for a net point $v \in Y$

- $P(v) = \{v' \in Y \mid v' \prec v\}$ (note that $v \notin P(v)$),
- $M(v) = B_v(\varepsilon\varrho) \setminus (\cup_{v' \in P(v)} B_{v'}(\varepsilon\varrho))$,
- $D(v) = |M(v) \cap V_S^*|$, and
- $R(v)$ to be the set of $D(v)$ suppliers in $M(v)$ with the highest capacities.

For a net point $v \in Y$, it is easy to see that $\sum_{s \in R(v)} L(s) \geq \sum_{t \in M(v) \cap V_S^*} L(t)$. The sets $\{R(v) \mid v \in Y\}$ are disjoint by the way we defined $M(v)$.

We guess a subset $Y' \subseteq Y$ of size k such that $V_S^* \subseteq \bigcup_{v \in Y'} B_v(\varepsilon\varrho)$. For each $v \in Y'$ we guess $D(v)$ and select $S = \bigcup_{v \in Y'} R(v)$. We apply the algorithm from Lemma 8 with the set S and cost $(1+2\varepsilon)\varrho$. If this check passes, then the solution we obtain is in fact a solution of cost $(1+2\varepsilon)\varrho$ since we replaced each optimum supplier by a supplier at distance at most $2\varepsilon\varrho$ from it. Conversely, if none of our guesses pass this check, then the instance $\mathcal{I}$ 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 Y' is $\mathcal{O}\left(\binom{|Y|}{k}\right)$. Since $D(v) \leq k$ for every $v \in Y$, the time required to guess $D(v)$ for each $v \in Y'$ is $\mathcal{O}(k^k)$. In total, the running time of the algorithm is $\mathcal{O}^*\left(\binom{|Y|}{k} k^k\right)$. ◻

We are ready to prove Lemma 5.

Proof (of Lemma 5). To satisfy the requirements of Lemma 7 and Lemma 9, we remove all suppliers which do not have a client at distance ϱ from them. The algorithm then computes a $(\varepsilon\varrho)$-net Y of the metric (V_S, dist). By Lemma 7, if the net Y has more than $(k+p)\varepsilon^{-\mathcal{O}(\Delta)}$ 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 9. Since the net Y has size at most $(k+p)\varepsilon^{-\mathcal{O}(\Delta)}$, the running time of the algorithm is $\mathcal{O}^*\left((k+p)^k \varepsilon^{-\mathcal{O}(k\Delta)}\right)$. ◻

4 EPAS for kSwO in Low Highway Dimension Graphs

In this section we prove Theorem 3, i.e. we show an EPAS for k-Supplier with Outliers in low highway dimension graphs. We assume without loss of generality that edge lengths are integral. In our algorithm, we will require the constant γ in Definition 1 to be strictly greater than 4.

In contrast to CkSwO, we can simply specify the solution of kSwO for a given cost ϱ by a subset of suppliers $S \subseteq V_S$. Then a client c can be assigned to any supplier s such that $\mathrm{dist}(c,s) \le \varrho$; if such a supplier does not exist, then c is an outlier. For such a solution S, we denote the set of outliers by $S^{\perp}$.

Let us first give an overview of our algorithm. We use a framework by Becker et al. [7] for embedding low highway dimension graphs into low treewidth graphs. An *embedding* of an (undirected) *guest graph* G into a *host graph* H is a distance-preserving injective mapping $\psi\colon V(G) \to V(H)$. For the purpose of this overview, suppose that for every two vertices $u, v \in V(G)$ we have $\mathrm{dist}_G(u,v) \le \mathrm{dist}_H(\psi(u), \psi(v)) \le c \cdot \mathrm{dist}_G(u,v)$ for some constant $c \in \mathbb{R}^+$. Ideally, we would like to solve the problem optimally in the host graph and then translate the solution in the host graph back into the input guest graph. The resulting solution in the guest graph would be a c-approximate solution of the input instance.

Our situation is complicated by the fact that optimally solving k-Center, which is a special case of kSwO, is already W[1]-hard in low treewidth graphs, as was shown by Katsikarelis et al. [30]. On the bright side, the authors of the same paper show that it is possible to design an EPAS for k-Center in low treewidth graphs. We generalize their algorithm to kSwO. The following theorem summarizes the properties of the algorithm we obtain.

Theorem 4. *Let $\mathcal{I} = (G, k, p)$ be a* kSwO *instance, $\varrho \in \mathbb{N}$ a cost, and $\varepsilon > 0$. Suppose that we receive a nice tree decomposition*[5] *of G of width $\mathrm{tw}(G)$ on input. There exists an algorithm which either*

- *returns a solution of cost $(1+\varepsilon)\varrho$ if $(1+\varepsilon)\varrho \ge \mathrm{OPT}(\mathcal{I})$, or*
- *correctly decides that $\mathcal{I}$ does not have a feasible solution of cost ϱ,*

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

To ease the presentation, we defer the full proof of Theorem 4 to Section 5, however, let us at least give the idea behind the algorithm. We use the standard approach of dynamic programming on nice tree decompositions. We use a technique of Lampis [31] for designing approximation schemes for problems hard in low treewidth graphs. The idea behind his technique is that if the cause of the hardness of a problem is due to the necessity of storing large numbers in the dynamic programming table, which in our case are integers from 0 to ϱ, then instead of storing exact values we store powers of $(1+\delta)$ for an appropriately selected δ. This decreases the size of the dynamic programming table from ϱ^{tw} to roughly $(\log \varrho)^{\mathcal{O}(\mathrm{tw})}$, which is sufficient for us. This approach is not cost-free though, since storing rounded approximate values may create errors which can accumulate during the execution of the algorithm. We will show how to bound this error in the height of the given tree decomposition. Together with a result of Chatterjee et al. [12], which rebalances a given tree decomposition into a nice tree decomposition with a logarithmic height and while only increasing the width by a constant factor, we will obtain an EPAS parameterized by treewidth of the input graph.

The main result by Becker et al. [7] that we use is the following theorem. Note that this result requires that the universal constant γ in Definition 1 is strictly greater than γ.

[5] See Section 5 for a formal definition.

Theorem 5 ([7, 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 $U \subseteq V(G)$, there exists a graph H and an embedding $\psi \colon V(G) \to V(H)$ such that*

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

$$\begin{aligned} \mathrm{dist}_G(u,v) &\le \mathrm{dist}_H(\psi(u),\psi(v)) \\ &\le (1+\mathcal{O}(\varepsilon))\mathrm{dist}_G(u,v) + \varepsilon \cdot \min\{\mathrm{dist}_G(u,U), \mathrm{dist}_G(v,U)\}. \end{aligned} \tag{3}$$

Our algorithm starts by computing a constant factor approximation solution S to the input instance, we can compute such solution using the result of Cygan and Kociumaka [16]. We select $S \cup S^{\perp}$ as the set U in Theorem 5. Since S is a feasible solution, we have $|U| \le k + p$. We apply the algorithm of Theorem 5 on graph G to obtain a host graph H with an embedding $\psi \colon V(G) \to V(H)$. For a subset of suppliers $S \subseteq V_S$ let $\mathrm{cost}_G(S)$ be the cost S in graph G with its edge lengths.

Let OPT_G and OPT_H be the optimum cost of the instance G and H respectively. We apply the algorithm of Theorem 4 on graph H with values ϱ and ε to obtain a solution S such that $\mathrm{cost}_H(S) \le (1+\varepsilon)\varrho$. As ψ is an injective function, it is correct to refer to S as a solution in G. Our goal is to bound the cost of the obtained solution in the input graph G, which we do in the following lemma.

Lemma 10. *Let S be a $(1+\varepsilon)$-approximate solution in the host graph H, i.e. $\mathrm{cost}_H(S) \le (1+\varepsilon)\mathrm{OPT}_H$. Then $\mathrm{cost}_G(S) \le (1+\mathcal{O}(\varepsilon))\mathrm{OPT}_G$.*

Proof. Let S^* be a solution of G of cost OPT_G and $V_C^* = (V_C \setminus (S^*)^{\perp})$. For a client $u \in V_C^*$ we denote by $S^*(u)$ the nearest supplier of S^* to u. By definition of κSwO we have $\mathrm{cost}_H(S^*) = \max_{u \in V_C^*} \mathrm{dist}_H(u, S)$. From (3) we get

$$\begin{aligned} &\mathrm{cost}_H(S^*) \le \\ &\max_{u \in V_C^*} \{(1+\mathcal{O}(\varepsilon))\mathrm{dist}_G(u, S^*(u)) + \varepsilon \cdot \min\{\mathrm{dist}_G(u,U), \mathrm{dist}_G(S^*(u),U)\}\}. \end{aligned} \tag{4}$$

By maximizing over each term of (4) and using the fact that $\min\{a,b\} \le a$ for any $a, b \in \mathbb{R}_0^+$ in the second term we get

$$\mathrm{cost}_H(S^*) \le (1+\mathcal{O}(\varepsilon))\mathrm{OPT}_G + \varepsilon \cdot \max_{u \in V_C^*} \mathrm{dist}_G(u,U). \tag{5}$$

We are going to give an upper bound for the second term of (5) using the first term. There are two cases to consider. If u is covered by the constant-approximate solution, then $\mathrm{dist}_G(u,U) = \mathcal{O}(\mathrm{OPT}_G)$. Otherwise u has to be an outlier in the approximate solution. We added all outliers to the set u, so we have $\mathrm{dist}_G(u,U) = 0$ in this case. In total, we have

$$\mathrm{cost}_H(S^*) \le (1+\mathcal{O}(\varepsilon))\mathrm{OPT}_G. \tag{6}$$

As S^* is an optimal solution in G but not necessarily in H we get from (6)

$$\mathrm{OPT}_H \le \mathrm{cost}_H(S^*) \le (1+\mathcal{O}(\varepsilon))\mathrm{OPT}_G \tag{7}$$

The approximate solution has cost $(1+\varepsilon)\mathrm{OPT}_H$. By multiplying both sides of (7), we get the desired bound. □

We are ready to prove Theorem 3.

Proof (of Theorem 3.). Lemma 10 shows the correctness of the algorithm, it remains to bound its running time. The algorithm consists of three steps, first we compute a constant-factor approximation to the instance in polynomial time. Then we run the algorithm from Theorem 5 on the input to obtain a host graph H. This step runs in time $\mathcal{O}^*(g(k,p,h,\varepsilon))$ for some computable function g. Finally, we run the algorithm from Theorem 4 on H. This step takes time $\mathcal{O}((\mathrm{tw}(H)/\varepsilon)^{\mathcal{O}(\mathrm{tw}(H))})$ where $\mathrm{tw}(H) = \mathcal{O}^*(g'(k,p,h,\varepsilon))$ for some computable function g'. Thus the algorithm runs in FPT time with parameters k, p, ε, and the highway dimension of the input.

5 EPAS for *k*-Supplier with Outliers on Low Treewidth Graphs

The following definition of treewidth is given in [14]. 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 properties hold.

Property 1. Every vertex of G is in at least one bag, i.e. $\cup_{t\in V(T)} X_t = V(G)$.

Property 2. For every edge $e \in E(G)$, there exists a node $t \in T$ such that the corresponding bag X_t contains both endpoints of e.

Property 3. For every $v \in V(G)$, the set $T_u = \{t \in V(T) \mid u \in X_t\}$, 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 Property 3 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.

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' where $X_t = X_{t'} \cup \{u\}$ 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' where $X_t = X_{t'} \setminus \{v\}$ 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 where $X_t = X_{t_1} = X_{t_2}$.

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 $\mathcal{O}(k^2 \cdot \max(|V(T)|, |V(G)|))$ time compute a nice tree decomposition of G of width at most k that has at most $\mathcal{O}(k \cdot |V(G)|)$ nodes, for more details, see [14, 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.

In this section we develop an EPAS for kSwO parameterized by k, p, ε, and the treewidth of the input graph to prove Theorem 4. To simplify the exposition, we restrict ourselves to instances with integral edge lengths, and we also assume that $V_S \cap V_C = \emptyset$. We start by giving an exact algorithm for the problem, which we later turn into an approximation scheme by the technique of Lampis [31]. The overall approach generalizes the approach for obtaining an EPAS for k-Center by Katsikarelis et al. [30]. In the rest of this section, we denote the treewidth of the input graph by tw. The properties of the exact algorithm are summarized by the following theorem.

Theorem 6. *There exists an algorithm which given a* kSwO *instance $\mathcal{I}$ and cost $\varrho \in \mathbb{N}$ decides whether $\mathcal{I}$ has a feasible solution of cost ϱ running in time $\mathcal{O}^*\left(\varrho^{\mathcal{O}(\mathrm{tw})}\right)$.*

We give an equivalent formulation of kSwO 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 $d\colon E \to \mathbb{N}$ and vertices are partitioned into clients V_C and suppliers V_S. A *distance labelling* function dl of G is a function $\mathrm{dl}\colon V \to \{0, \ldots, \varrho\} \cup \{\infty\}$. We require that only suppliers 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 neighbor $v \in N(u)$ 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 $|\mathrm{dl}^{-1}(0)|$ and the *penalty* as $|\mathrm{dl}^{-1}(\infty) \cap V_C|$. The following lemma shows the equivalence between the two formulations. A special case of this statement for k-Center is proved in the original algorithm, cf. [30, Lemma 26].

Lemma 11. *A* kSwO *instance $\mathcal{I} = (G, k, p)$ admits a feasible solution of cost ϱ if and only if it admits a valid distance labelling function* $\mathrm{dl}\colon V \to \{0, \ldots, \varrho\} \cup \{\infty\}$ *of cost k and penalty p.*

Proof. Let $S \subseteq V_S$ 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 kSwO 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 neighbor of u on 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 neighbor $v \in N(u)$ 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

$$\mathrm{dist}(u, S) \leq \mathrm{dist}(u, v) + \mathrm{dist}(v, S) \leq \mathrm{dist}(u, v) + \mathrm{dl}(v) \leq \mathrm{dl}(u) \tag{8}$$

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 6. 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 is given as a part of the input.

Proof (Theorem 6). 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

$$D_i \colon \big((X_i \to \{0, \dots, \varrho\} \cup \{\infty\}) \times 2^{X_i \cap V_C} \times \{0, \dots, k\} \times \{0, \dots, p\}\big) \to \{0, 1\}. \tag{9}$$

We may refer to the value 0 in the dynamic programming table as *false* and to 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 where $i \in V(T)$, 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 $|\mathrm{dl}_*^{-1}(0)| = k_i$,
- the penalty of dl_* is p_i, that is $|\mathrm{dl}_*^{-1}(\infty) \cap V_C| = p_i$,

– 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 11, the input admits a feasible solution of cost ϱ if there exists a distance labelling dl_r of the root node r, and constants $k_r \le k$ and $p_r \le p$ 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 $\{1, \dots, \varrho\} \cup \{\infty\}$ 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 neighbor $v \in N(u)$ so that $\mathrm{dl}_i(u) \ge \mathrm{dl}_i(v) + d(u, v)$, and we add neighbors $w \in N(u)$ to S_i for which it holds that $\mathrm{dl}_i(w) \ge \mathrm{dl}_i(u) + d(u, w)$. If $\mathrm{dl}_i(u) = 0$ and $k_j \le k-1$, then we set the entry $D_i[\mathrm{dl}_i, S_i, k_j + 1, p_j]$ to true. If $\mathrm{dl}_i(u) \in \{1, \dots, \varrho\}$, 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 $p_j \le p-1$, 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 3 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 neighbor 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 neighbor $v \in N(u) \cap V_i$ such that $\widehat{\mathrm{dl}}(u) \ge \widehat{\mathrm{dl}}(v) + d(u, v)$. In particular this means that there exists a node in a subtree rooted at i whose bag contains v. We need to show that $v \in X_i$ as well. 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 $U \subseteq V(T)$ we denote $X_U = \{X_w \mid w \in U\}$. By Property 2 of tree decompositions, there must exist a node of $\mathcal{T}$ whose corresponding bag contains both u

and v. Thus $X_{T_u} \cap X_{T_v} \neq \emptyset$. 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 3. 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 $|\widehat{\mathrm{dl}}^{-1}(0) \cap V_i| = |\widehat{\mathrm{dl}}^{-1}(0) \cap V_j| + 1$. Since $\widehat{\mathrm{dl}}(u) \neq \infty$, we have $|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_i \cap V_C| = |\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_j \cap V_C|$. Thus the algorithm sets $k_i = k_j + 1$ and $p_i = p_j$.
- **Case** $\widehat{\mathrm{dl}}(u) \in \{1, \dots, \varrho\}$**.** Since $\widehat{\mathrm{dl}}(u) \notin \{0, \infty\}$, we have $|\widehat{\mathrm{dl}}^{-1}(0) \cap V_i| = |\widehat{\mathrm{dl}}^{-1}(0) \cap V_j|$ and $|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_i \cap V_C| = |\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_j \cap V_C|$, hence $k_i = k_j$ and $p_i = p_j$ respectively.
- **Case** $\widehat{\mathrm{dl}}(u) = \infty$**.** We have $|\widehat{\mathrm{dl}}^{-1}(0) \cap V_i| = |\widehat{\mathrm{dl}}^{-1}(0) \cap V_j|$ and therefore $k_i = k_j$. If $u \in V_C$, then $|\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_i \cap V_C| = |\widehat{\mathrm{dl}}^{-1}(\infty) \cap V_j \cap V_C| + 1$ and $p_i = p_j + 1$. Otherwise $u \in V_S$ 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 neighbor 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 $\widehat{\mathrm{dl}}_i^{-1}(\infty) \cap V_C = \widehat{\mathrm{dl}}_j^{-1}(\infty) \cap V_C$. Then the cost of $\widehat{\mathrm{dl}}_i$ is $k_j + 1$ and the penalty is p_j.
- **Cases** $\mathrm{dl}_i(u) \in \{1, \dots, \varrho\}$ **and** $\mathrm{dl}_i(u) = \infty \wedge u \in V_S$**.** In this case $\widehat{\mathrm{dl}}_i^{-1}(0) = \widehat{\mathrm{dl}}_j^{-1}(0)$ and $\widehat{\mathrm{dl}}_i^{-1}(\infty) \cap V_C = \widehat{\mathrm{dl}}_j^{-1}(\infty) \cap V_C$. Then the cost of $\widehat{\mathrm{dl}}_i$ is k_j and the penalty is p_j.
- **Case** $\mathrm{dl}_i(u) = \infty \wedge u \in V_C$**.** In this case $\widehat{\mathrm{dl}}_i^{-1}(0) = \widehat{\mathrm{dl}}_j^{-1}(0)$ and $\widehat{\mathrm{dl}}_i^{-1}(\infty) \cap V_C = \left(\widehat{\mathrm{dl}}_j^{-1}(\infty) \cap V_C\right) \cup \{u\}$. 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, $L_s(\mathrm{dl})$ the set of functions which extend dl by assigning labels $\{0, \ldots, \varrho\} \cup \{\infty\}$ to u, and $L_\infty(\mathrm{dl})$ the 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

$$D_i[\mathrm{dl}_i, S_i, k_i, p_i] = \bigvee_{\mathrm{dl}_j \in L_s(\mathrm{dl}_i)} D_j[\mathrm{dl}_j, S_i, k_i, p_i]. \tag{10}$$

If u is a client, then we set

$$D_i[\mathrm{dl}_i, S_i, k_i, p_i] = \left(\bigvee_{\mathrm{dl}_j \in L_c(\mathrm{dl}_i)} D_j[\mathrm{dl}_j, S_i \cup \{u\}, k_i, p_i] \right) \vee D_j[L_\infty(\mathrm{dl}), S_i, k_i, p_i]. \tag{11}$$

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** $u \in V_S$**.** We have $X_i \cap V_C = X_j \cap V_C$ and thus $\widehat{S}_i = \widehat{S}_j$.
- **Case** $\widehat{\mathrm{dl}}(u) \in \{1, \ldots, \varrho\}$ **and** $u \in V_C$**.** 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** $u \in V_C$**.** 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

$$D_i\left[\mathrm{dl}_i, S_1 \cup S_2, k_1 + k_2 - |\mathrm{dl}_i^{-1}(0)|, p_1 + p_2 - |\mathrm{dl}_i^{-1}(\infty) \cap V_C|\right]. \quad (12)$$

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 $a \in \{1,2\}$, $\widehat{\mathrm{dl}}_a$ 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 $D_{j_a}\left[\widehat{\mathrm{dl}}_i, \widehat{S}_{j_a}, |\widehat{C}_{j_a}|, |\widehat{O}_{j_a}|\right]$ 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 neighbor 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 3 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 $|\widehat{C}_1| + |\widehat{C}_2| = \widehat{k} + |\widehat{\mathrm{dl}}^{-1}(0) \cap X_i|$ and $|\widehat{O}_1| + |\widehat{O}_2| = \widehat{p} + |\widehat{\mathrm{dl}}^{-1}(\infty) \cap X_i \cap V_C|$, as 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. $S_i = S_{j_1} \cup S_{j_2}$, $k_i = k_{j_1} + k_{j_2} - |\mathrm{dl}_i^{-1}(0)|$, 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 $\mathcal{O}^*((4(\varrho+2))^{\mathcal{O}(\mathrm{tw})})$. □

The approximation scheme. Now we describe a parameterized approximation scheme based on the algorithm from Theorem 6. We will need the following result by Chatterjee et al. [12].

Theorem 7 ([12]). *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 $\mathcal{O}(\mathrm{tw} \cdot \log n)$ in time $\mathcal{O}(\mathrm{tw} \cdot n)$.*

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 [30]. Let (G, k, p) be a κSwO instance 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 $\mathrm{dl}_\delta : V \to \Sigma_\varrho \cup \{0, \infty\}$. We require that only suppliers 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 $|\mathrm{dl}_\delta^{-1}(0)|$ and its *penalty* as $|\mathrm{dl}_\delta^{-1}(\infty) \cap V_C|$. The following lemma shows that given a δ-labelling function of cost k and penalty p, we can produce a solution to the κSwO problem which opens k centers, creates p outliers and has cost $(1+\varepsilon)^2\varrho$.

Lemma 12. *Let $\mathcal{I} = (G, k, p)$ be an instance of the* κSwO *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 neighbor $v \in N(u)$ 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 neighbor $v \in N(u)$ 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 [30], which they use for their EPAS for k-Center, 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 k-Center, however, their proof generalizes to kSwO without any major modifications.

Lemma 13 ([30, Lemma 29]). *Let $\mathcal{I} = (G, k, p)$ be a kSwO instance, $\mathcal{T}$ 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 a kSwO instance. 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 $\mathcal{O}^*\left((\frac{\log n}{\varepsilon})^{\mathcal{O}(k)}\right)$ is still an FPT algorithm.

Lemma 14 ([30, Lemma 1]). *Let $\mathcal{A}$ be an algorithm for a parameterized problem with parameter k such that the running time of $\mathcal{A}$ is $\mathcal{O}^*\left((\frac{\log n}{\varepsilon})^{\mathcal{O}(k)}\right)$. Then the running time of $\mathcal{A}$ can be bounded by $\mathcal{O}^*\left((\frac{k}{\varepsilon})^{\mathcal{O}(k)}\right)$.*

We are ready to prove Theorem 4. The proof follows the proof of the original algorithm, cf. [30, Theorem 31]. The main obstacle lies in bounding the accumulated error during the execution of the algorithm. The original proof heavily relies on the properties of the algorithm they present for k-Center. Hence, we will have to modify their proof to work with the algorithm we give for kSwO.

Proof (Theorem 4). Our algorithm will follow along the same lines as the algorithm in Theorem 6. 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

$$D_i^\delta \colon \left((X_i \to \Sigma_\varrho) \times 2^{X_i \cap V_C} \times \{0, \ldots, k\} \times \{0, \ldots, p\}\right) \to \{0, 1\}. \qquad (13)$$

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 $\mathrm{dl}_\delta \colon X_i \to \Sigma_\varrho$, 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 7 and Lemma 13. We obtain a tree decomposition $\mathcal{T}'$ of the input graph of width $4\text{tw} + 3$ and height H where $H \in \mathcal{O}(\text{tw} \cdot \log n)$. 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 $\delta = \frac{\varepsilon}{2H} = \Omega\left(\frac{\varepsilon}{\text{tw} \cdot \log n}\right)$. Observe that this choice of δ gives for all $h \le H$ that $(1+\delta)^h \le \left(1 + \frac{\varepsilon}{2H}\right)^H \le e^{\varepsilon/2} \le 1 + \varepsilon$ 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 12. 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 6 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 $D^\delta_\ell[\text{dl}, \emptyset, 0, 0]$ 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 $D^\delta_j[\text{dl}'_\delta, S', k_j, p_j]$ 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 $\text{dl}_\delta(u)$ 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 neighbor $v \in N(u)$ so that $\text{dl}_\delta(u) \ge \text{dl}_\delta(v) + \frac{d(u,v)}{1+\varepsilon}$. Finally, we add neighbors $w \in N(u)$ to S for which it holds that $\text{dl}_\delta(w) \ge \text{dl}_\delta(u) + \frac{d(u,w)}{1+\varepsilon}$. If $\text{dl}_\delta(u) = 0$ and $k_j \le k-1$, then we set the entry $D^\delta_i[\text{dl}_\delta, S, k_j+1, p_j]$ to 1. If $\text{dl}_\delta(u) \in \Sigma_\varrho \setminus \{0, \infty\}$, then we set the entry $D^\delta_i[\text{dl}_\delta, S, k_j, p_j]$ to 1. If $\text{dl}_\delta(u) = \infty$, $u \in V_C$, and $p_j \le p-1$, then we set the entry $D^\delta_i[\text{dl}_\delta, S, k_j, p_j+1]$ to 1. If $\text{dl}_\delta(u) = \infty$ and $u \in V_S$, then we set the entry $D^\delta_i[\text{dl}_\delta, S, k_j, p_j]$ 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 $L_c(\text{dl}_\delta)$ the set of functions which extend dl_δ by assigning labels $\Sigma_\varrho \setminus \{0, \infty\}$ to u, by $L_s(\text{dl}_\delta)$ the set of functions which extend dl_δ by assigning labels Σ_ϱ to u, and by $L_\infty(\text{dl}_\delta)$ the extension of dl_δ which assigns label ∞ to u. Let S be any subset of clients of X_i, $k_i \in \{0, \dots, k\}$, and $p_i \in \{0, \dots, p\}$. If u is a supplier, then we set

$$D^\delta_i[\text{dl}_\delta, S, k_i, p_i] = \bigvee_{\text{dl}'_\delta \in L_s(\text{dl}_\delta)} D^\delta_j[\text{dl}'_\delta, S, k_i, p_i]. \tag{14}$$

If u is a client, then we set

$$D_i^\delta[\mathrm{dl}_\delta, S, k_i, p_i] = \left(\bigvee_{\mathrm{dl}'_\delta \in L_c(\mathrm{dl}_\delta)} D_j^\delta[\mathrm{dl}'_\delta, S \cup \{u\}, k_i, p_i] \right) \vee \vee D_j^\delta[L_\infty(\mathrm{dl}_\delta), S, k_i, p_i]. \tag{15}$$

– **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

$$D_i \left[\mathrm{dl}_\delta, S_1 \cup S_2, k_1 + k_2 - |\mathrm{dl}_\delta^{-1}(0)|, p_1 + p_2 - |\mathrm{dl}_\delta^{-1}(\infty) \cap V_C|\right]. \tag{16}$$

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 $((V_i \setminus X_i) \cup S) \cap V_C$ 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 6. 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 12, we obtain a solution to the input kSwO instance 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 12 does not hold for any choice of δ. In the remainder suppose there exists a distance labelling $\mathrm{dl}\colon V \to \{0, \dots, \varrho\} \cup \{\infty\}$ which encodes a solution to the instance as in the proof of Lemma 11.

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 $\mathrm{dl}_\delta\colon X_i \to \Sigma_\varrho$, $S_\delta \supseteq S$ and values k_i and p_i such that $D_i^\delta[\mathrm{dl}_\delta, S_\delta, k_i, p_i] = 1$, $k_i \le |\mathrm{dl}^{-1}(0)|$, $p_i \le |\mathrm{dl}^{-1}(\infty) \cap V_C|$ 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:

Case 1. $\mathrm{dl}(u) = 0$.

Case 2. $S = S' \cup \{u\}$.

Case 3. $u \notin S$ and $\mathrm{dl}(u) < \infty$.

Case 4. $\mathrm{dl}(u) = \infty$.

Suppose for contradiction that $\mathrm{dl}(u) \in \{1, \dots, \varrho\}$ 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 neighbors 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 $\mathrm{dl}(u) \in \{1, \dots, \varrho\}$, 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 13. 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 $S \subseteq S_\delta$, $k_i \leq |\mathrm{dl}^{-1}(0) \cap V_i|$, $p_i \leq |\mathrm{dl}^{-1}(\infty) \cap V_i \cap V_C|$, 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 $S'_\delta \supseteq S'$, $k_j \leq |\mathrm{dl}^{-1}(0) \cap V_j|$, $p_j \leq |\mathrm{dl}^{-1}(\infty) \cap V_j \cap V_C|$ and dl'_δ which has $(\forall v \in X_j)\left(\mathrm{dl}'_\delta(v) \in \left[\frac{\mathrm{dl}(v)}{(1+\delta)^{h-1}}, (1+\delta)^{h-1}\mathrm{dl}(v)\right]\right)$.

Case 1 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.

Case 2 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

$$\mathrm{dl}_\delta(u) = (1+\delta)^{\lceil \log_{1+\delta} r \rceil} \leq (1+\delta)^{\log_{1+\delta}((1+\delta)^{h-1}\mathrm{dl}(u))+1} \leq (1+\delta)^h \mathrm{dl}(u). \quad (17)$$

Case 3 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 $(1+\delta)^{h-1} \le 1+\varepsilon$, we have

$$\mathrm{dl}_\delta(v) \ge \frac{\mathrm{dl}(v)}{(1+\delta)^{h-1}} \ge \frac{\mathrm{dl}(u)}{(1+\delta)^{h-1}} + \frac{d(u,v)}{1+\varepsilon} \ge \mathrm{dl}_\delta(u) + \frac{d(u,v)}{1+\varepsilon}. \tag{18}$$

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

$$\mathrm{dl}_\delta(u) = (1+\delta)^{\lceil \log_{1+\delta} r \rceil} \le (1+\delta)^{\log_{1+\delta} \frac{\mathrm{dl}(u)}{(1+\delta)^h} + 1} \le \frac{\mathrm{dl}(u)}{(1+\delta)^{h-1}}. \tag{19}$$

Case 4 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 neighbor 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 $\mathrm{dl}_\delta(u) \le (1+\delta)^H \mathrm{dl}(u) \le (1+\varepsilon)\varrho$, $k_r \le k$ and $p_r \le p$.

It remains to bound the running time of the algorithm. We have $|\varSigma_\varrho| = \mathcal{O}(\log_{1+\delta} \varrho) = \mathcal{O}\left(\frac{\log \varrho}{\log(1+\delta)}\right) = \mathcal{O}\left(\frac{\log \varrho}{\delta}\right)$ where we use the fact that $\ln(1+\delta) \approx \delta$ for sufficiently small δ (that is, sufficiently large n). By setting $\delta = \varOmega\left(\frac{\varepsilon}{\mathrm{tw} \cdot \log n}\right)$ and assuming $k, p \le |V(G)|$, we get a running time $\mathcal{O}^*\left((\mathrm{tw} \cdot \frac{\log n}{\varepsilon})^{\mathcal{O}(\mathrm{tw})}\right)$. Using Lemma 14, this is an FPT algorithm with running time $\mathcal{O}^*\left((\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})}\right)$. □

6 Open Problems

We conclude with the following open problems. The algorithms given by Theorems 2 and 3 have the number of outliers in the base of the exponent. Is it possible to remove the outliers from the set of parameters? An improvement of Theorem 1 would be to show that the hardness is preserved in the case of planar graphs. It may be of interest that Planar Capacitated Dominating Set is W[1]-hard when parameterized by solution size [9]. Goyal and Jaiswal [25] have shown that it is possible to 2-approximate CkC when the parameter is only k, and that this result is tight. An improvement of Theorem 1 would be to show that this lower bound is tight in low highway dimension graphs. Finally, we ask whether there exists a problem which admits an EPAS in low highway dimension graphs but we cannot approximate in low doubling dimension graphs, i.e. the converse of Theorems 1 and 2.

References

1. Abraham, I., Delling, D., Fiat, A., Goldberg, A.V., Werneck, R.F.: Highway Dimension and Provably Efficient Shortest Path Algorithms. Journal of the ACM (JACM) **63**(5), 41:1–41:26 (2016)
2. Abraham, I., Fiat, A., Goldberg, A.V., Werneck, R.F.F.: Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In: Charikar, M. (ed.) Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010. pp. 782–793. SIAM (2010)
3. Ahmadi-Javid, A., Seyedi, P., Syam, S.S.: A survey of healthcare facility location. Computers & Operations Research **79**, 223–263 (2017)
4. An, H., Bhaskara, A., Chekuri, C., Gupta, S., Madan, V., Svensson, O.: Centrality of trees for capacitated k-center. Mathematical Programming **154**(1-2), 29–53 (2015)
5. Bast, H., Funke, S., Matijevic, D.: Transit ultrafast shortest-path queries with linear-time preprocessing. 9th DIMACS Implementation Challenge (2006)
6. Bast, H., Funke, S., Matijevic, D., Sanders, P., Schultes, D.: In Transit to Constant Time Shortest-Path Queries in Road Networks. In: Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX 2007, New Orleans, Louisiana, USA, January 6, 2007. SIAM (2007)
7. Becker, A., Klein, P.N., Saulpic, D.: Polynomial-Time Approximation Schemes for k-center, k-median, and Capacitated Vehicle Routing in Bounded Highway Dimension. In: Azar, Y., Bast, H., Herman, G. (eds.) 26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland. LIPIcs, vol. 112, pp. 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018)
8. Blum, J.: Hierarchy of Transportation Network Parameters and Hardness Results. In: Jansen, B.M.P., Telle, J.A. (eds.) 14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany. LIPIcs, vol. 148, pp. 4:1–4:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
9. Bodlaender, H.L., Lokshtanov, D., Penninkx, E.: Planar Capacitated Dominating Set Is $W[1]$-Hard. In: Chen, J., Fomin, F.V. (eds.) Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5917, pp. 50–60. Springer (2009)
10. Chakrabarty, D., Goyal, P., Krishnaswamy, R.: The Non-Uniform k-Center Problem. ACM Trans. Algorithms **16**(4), 46:1–46:19 (2020)
11. Charikar, M., Khuller, S., Mount, D.M., Narasimhan, G.: Algorithms for facility location problems with outliers. In: Kosaraju, S.R. (ed.) Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA. pp. 642–651. ACM/SIAM (2001)
12. Chatterjee, K., Ibsen-Jensen, R., Pavlogiannis, A.: Optimal tree-decomposition balancing and reachability on low treewidth graphs. https://research-explorer.app.ist.ac.at/record/5427 (2014), accessed: 2021-05-13
13. Cohen-Addad, V., Feldmann, A.E., Saulpic, D.: Near-linear Time Approximation Schemes for Clustering in Doubling Metrics. J. ACM **68**(6), 44:1–44:34 (2021)
14. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)

15. Cygan, M., Hajiaghayi, M., Khuller, S.: LP Rounding for k-Centers with Non-uniform Hard Capacities . In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012. pp. 273–282. IEEE Computer Society (2012)
16. Cygan, M., Kociumaka, T.: Constant Factor Approximation for Capacitated k-Center with Outliers . In: Mayr, E.W., Portier, N. (eds.) 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France. LIPIcs, vol. 25, pp. 251–262. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2014)
17. Dom, M., Lokshtanov, D., Saurabh, S., Villanger, Y.: Capacitated Domination and Covering: A Parameterized Perspective. In: Grohe, M., Niedermeier, R. (eds.) Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5018, pp. 78–90. Springer (2008)
18. Edmonds, J., Karp, R.M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. J. ACM **19**(2), 248–264 (1972)
19. Feder, T., Greene, D.H.: Optimal Algorithms for Approximate Clustering. In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 434–444. ACM (1988)
20. Feldmann, A.E.: Fixed-Parameter Approximations for k-Center Problems in Low Highway Dimension Graphs. Algorithmica **81**(3), 1031–1052 (2019)
21. Feldmann, A.E., Fung, W.S., Könemann, J., Post, I.: A $(1+\epsilon)$-Embedding of Low Highway Dimension Graphs into Bounded Treewidth Graphs. SIAM J. Comput. **47**(4), 1667–1704 (2018)
22. Feldmann, A.E., Karthik C. S., Lee, E., Manurangsi, P.: A Survey on Approximation in Parameterized Complexity: Hardness and Algorithms. Algorithms **13**(6), 146 (2020)
23. Feldmann, A.E., Marx, D.: The Parameterized Hardness of the k-Center Problem in Transportation Networks. Algorithmica **82**(7), 1989–2005 (2020)
24. Feldmann, A.E., Saulpic, D.: Polynomial time approximation schemes for clustering in low highway dimension graphs. J. Comput. Syst. Sci. **122**, 72–93 (2021)
25. Goyal, D., Jaiswal, R.: Tight FPT Approximation for Constrained k-Center and k-Supplier (2021), https://arxiv.org/abs/2110.14242
26. Gupta, A., Krauthgamer, R., Lee, J.R.: Bounded Geometries, Fractals, and Low-Distortion Embeddings. In: 44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings. pp. 534–543. IEEE Computer Society (2003)
27. Harris, D.G., Pensyl, T.W., Srinivasan, A., Trinh, K.: A Lottery Model for Center-Type Problems With Outliers . ACM Trans. Algorithms **15**(3), 36:1–36:25 (2019)
28. Hochbaum, D.S., Shmoys, D.B.: A Best Possible Heuristic for the k-Center Problem. Math. Oper. Res. **10**(2), 180–184 (1985)
29. Hochbaum, D.S., Shmoys, D.B.: A unified approach to approximation algorithms for bottleneck problems. J. ACM **33**(3), 533–550 (1986)
30. Katsikarelis, I., Lampis, M., Paschos, V.T.: Structural parameters, tight bounds, and approximation for (k, r)-center. Discret. Appl. Math. **264**, 90–117 (2019)
31. Lampis, M.: Parameterized Approximation Schemes Using Graph Widths. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8572, pp. 775–786. Springer (2014)

32. Talwar, K.: Bypassing the Embedding: Algorithms for Low Dimensional Metrics. In: Babai, L. (ed.) Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004. pp. 281–290. ACM (2004)
33. Vazirani, V.V.: Approximation algorithms. Springer (2001)
34. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press (2011)