A QPTAS for Facility Location on Unit Disk Graphs

Zachary Friggstad
Department of Computing Science, University of Alberta

Mohsen Rezapour
Department of Computing Science, University of Alberta

Mohammad R. Salavatipour
Department of Computing Science, University of Alberta

Hao Sun
Department of Computer Science, University of Houston

Abstract

We study the classic (Uncapacitated) Facility Location problem on Unit Disk Graphs (UDGs). For a given point set P in the plane, the unit disk graph UDG(P) on P has vertex set P and an edge between two distinct points $p, q \in P$ if and only if their Euclidean distance $|pq|$ is at most 1. The weight of the edge pq is equal to their distance $|pq|$. An instance of Facility Location on UDG(P) consists of a set $C \subseteq P$ of clients and a set $F \subseteq P$ of facilities, each having an opening cost f_i. The goal is to pick a subset $F' \subseteq F$ to open while minimizing $\sum_{i \in F'} f_i + \sum_{v \in C} d(v, F')$, where $d(v, F')$ is the distance of v to nearest facility in F' through UDG(P).

In this paper, we present the first Quasi-Polynomial Time Approximation Schemes (QPTAS) for the problem. While approximation schemes are well-established for facility location problems on sparse geometric graphs (such as planar graphs), there is a lack of such results for dense graphs. Specifically, prior to this study, to the best of our knowledge, there was no approximation scheme for any facility location problem on UDGs in the general setting.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases Facility Location, Unit Disk Graphs, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.31

Funding *Zachary Friggstad*: Supported by NSERC.
Mohammad R. Salavatipour: Supported by NSERC.

1 Introduction

Unit-disk graphs (UDGs) are a well-studied class of graphs due to their extensive applications in modeling ad-hoc communication and wireless sensor networks; see for example [18, 4, 22, 10, 15, 26, 25]. UDGs are defined as intersection graphs of a collection of unit-diamater disks in the two-dimensional plane. Specifically, each UDG represents a set of n unit disks as vertices, with each vertex corresponding to one unit disk. An edge exists between two vertices/points p, q if and only if their Euclidean distance is at most 1 (equivalently, the unit-diameter balls around p and q intersect) and the weight or length of each such edge is given by their by the Euclidean distance between the corresponding vertices.

Formally, for a given point set P in the plane, the unit disk graph representation of these points, denoted as UDG(P), is a graph $G = (V, E)$ with the vertex set V, where each vertex corresponds to a point in P. The edge set E consists of edges between points p and q if and only if their Euclidean distance, denoted as $|pq|$, is at most 1. The weight of the edge $pq \in E$ is equal to their distance $|pq|$. For a given subset $S \subseteq V$, we define the (weak) diameter of S as $\mathbf{diam}(S) = \max_{x,y} d_G(x, y)$, where $d_G(x, y)$ represents the minimum weight of a path between

vertices x and y in G (we assume G is connected as we may solve facility location for each connected component).

For many optimization problems, approximation schemes are known when the input graph is a UDG (e.g. maximum independent set, minimum dominating set, minimum clique-partition [23, 14, 5, 24]). Some of the techniques for designing PTAS's for optimization problems on UDGs (e.g. maximum independent set) involves partitioning the input into regions of bounded size (at a small loss, e.g. ignoring points that touch the boundary of the partitions) and then solving the problem on such instances using exhaustive search and/or dynamic programming which leverage Euclidean distance properties. This shifting strategy was introduced by [13].

We study the Facility Location problem on UDGs. An instance $I = (G, C, F)$ of Facility Location consists of an edge-weighted graph G, where the edges satisfy the metric property, a set $C \subseteq V$ of clients, and a set $F \subseteq V$ of facilities, each having an opening cost $f_i \in \mathbb{R}^+$. The goal is to pick a subset $F' \subseteq F$ to open to minimize $\sum_{i \in F'} f_i + \sum_{v \in C} d(v, F')$, where $d(v, F')$ is the distance of v to nearest facility in F'. Facility Location has been studied extensively and the best known upper and lower bounds for it are 1.488 [21] and 1.46 [11], respectively. Approximation schemes are known for Facility Location when the metric is Euclidean [2] or when G is a planar graph [8]. Additionally, Cohen et al. [7] developed approximation schemes for the "uniform" (that is all facilities cost 1 to open) facility location problem in minor-free graphs. To the best of our knowledge, no approximation scheme was known for any facility location type problem on UDGs. Our main result of this paper is the following:

▶ **Theorem 1.** *There is an algorithm that, given an instance of Facility Location in UDG and $\epsilon > 0$, finds a $(1+\epsilon)$-approximate solution in time $n^{O_\epsilon(\log n)}$, where the constant in $O_\epsilon(.)$ is $\epsilon^{-O(\epsilon^{-2})}$.*

In order to prove Theorem 1 we combine ideas from [8] with a low-diameter decomposition for UDGs that follows from [19, 20]. We also introduce a new dissection procedure obtained by finding a proper balanced separator for UDGs. This allows us (at a small loss) to break the problem into independent instances and use dynamic programming to combine the solutions to obtain the solution for the original instance. There are many details on how to put these pieces together carefully so as to bound the overall error.

2 Preliminaries

For planar graphs and, more generally, graphs that exclude $K_{r,r}$ as a minor for some fixed r, Klein-Plotkin-Rao [16] showed a decomposition of the input graph into low diameter parts by removing a small fraction of edges. More specifically, given a graph G with n nodes and m edges that excludes $K_{r,r}$ as a minor, one can remove $O(mr/\delta)$ edges so that the (weak) diameter of each remaining component is at most $O(r^2\delta)$. The general idea was based on chopping breadth-first search (BFS) trees (i.e. shortest-path trees in the unweighted version of the graph): suppose one constructs a BFS tree from some root node and then cut the edges at level $i \cdot \delta + r$ for $i \geq 1$ where $r \leq \delta$ is a random offset. Then repeat this procedure on each of the connected components, for $O(r)$ iterations. Then the resulting components have $O(r^2\delta)$ weak diameter. This result was further improved in [9, 1], to show for each graph without K_r as a minor there is a probabilistic decomposition into $O(r\delta)$ (weak) diameter components by removing $O(mr/\delta)$ edges. Lee [19, 20] generalized this by introducing region intersection graphs, which includes UDGs as a special case, and showed that one can obtain similar decomposition results for such graphs. Theorem 4.2 in [19] implies that a similar BFS chopping procedure applied to UDGs for a constant number of iterations results in graphs of bounded (weak) diameter. We describe this chopping procedure a bit more formally.

▶ **Definition 2** (δ-chopping operation). *For any connected graph G and any number $\delta \geq 1$, we define the δ-chopping operation on G as follows. Choose any node x_0 from $V(G)$, select a random integer $0 \leq r_0 \leq \delta$, and then compute a BFS tree from x_0. Partition $V(G)$ into annuli $A_0, A_1, A_2, \ldots$, where $A_0 = \{v \in V(G) : d'(x_0, v) < r_0\}$ and annulus A_j for $j \geq 1$ is defined as: $A_j = \{v \in V(G) : r_0 + (j-1)\delta \leq d'(x_0, v) < r_0 + \delta j\}$, where $d'(x_0, v)$ is the number of edges on the BFS tree path from x_0 to v.*

So there is an offset r_0 that cuts only a $1/\delta$-fraction of edges. Earlier works on minor free graphs [17] imply that if G is K_r-minor free if we perform this chopping procedure on each component of G recursively up to depth $O(r)$ yields components with (weak) diameter at most $O(\text{poly}(r)\delta)$. Corollary 4.3 in [19] immediately implies the following, Appendix A contains the brief argument.

▶ **Theorem 3** ([19]). *$O(1)$ iterations of the δ-chopping iteration applied to a UDG results in a graph of weak diameter $O(\delta)$.*

To prove Theorem 1 we also rely on the following result (which we prove) for the special case when the instance is in a bounded size region.

▶ **Theorem 4.** *There is a PTAS for Facility Location in UDGs when the point set P is contained within a bounding box of constant size $L = O(1)$ in the plane.*

Theorem 4 uses different techniques than discussed above. It follows from a simple reduction to prize-collecting Uncapacitated Facility Location in $\mathbb{R}^2$ for which a PTAS is known [6] after guessing an "ϵ-net" of centers in the optimum solution which is possible in polynomial time since L is bounded. The proof is deferred to Appendix B.

Our algorithm for Theorem 1 starts with some preliminary steps of algorithm of [8] that presented a PTAS for Facility Location on planar metrics. Those preliminary steps in fact are valid for general metrics (they do not use planarity in those initial steps) and reduce the problem to instances with certain structures that serve as our starting point. For this reason, we briefly outline the main steps of their algorithm. These initial steps reduce the problem to instances with certain structural properties and their proof works for general metrics (not just planar ones). Hence, the same initial reductions work in our setting as well.

2.1 Starting point: the PTAS for Facility Location on planar graphs [8]

Here, we summarize relevant results from [8] needed to prove Theorem 1.

A Well-Structured Instance

The goal of this section is to reduce Uncapacitated Facility Location in UDGs to more well-structured instances that, intuitively speaking, has the clients partitioned into annuli about facilities from some $\tilde{D} \subseteq F$ with bounded aspect ratio and that these facilities are near some facilities opened in an optimal solution. Definition 7 below contains the precise notion of what it means for an instance to be well structured. Corollary 9 is the main result from this section.

Given an instance $I = (G, C, F, f_i)$ of Facility Location, which consists of an edge-weighted graph G, a set of clients C, and a set of facilities F with opening costs f_i (for each $i \in F$), the first step of the algorithm in [8] involves partitioning the instance into separate (independent) sub-instances with specific structural properties. For any solution $D \subseteq F$, we denote by $\text{conn}(D)$ the connection cost of D ($\sum_{c \in C} dist(c, D)$) and by $\text{open}(D)$ the opening cost of facilities in D ($\sum_{i \in D} f_i$), and $\text{cost}(D) = \text{conn}(D) + \text{open}(D)$. We sometimes use $\text{cost}_I(D)$ to denote we refer to the cost of D for instance I. To achieve this, they compute an α-approximation solution $\tilde{D}$ (where $\alpha = O(1)$) to a modified instance $\tilde{I} = (G, C, F, \epsilon f_i)$ where each opening cost is scaled down by a factor of ϵ. In other words, $\tilde{D}$ is an $O(1)$-approximation for $\tilde{I}$. It is not hard to see that

$\tilde{D}$ is also an $O(1/\epsilon)$-approximation for I. For any $i \in \tilde{D}$, let $cluster(i)$ denote the set of clients connected to i in this solution and define $avgcost(i) = \frac{f_i + \sum_{j \in cluster(i)} d_G(j,i)}{|cluster(i)|}$ be the average cost of the cluster by facility i. Suppose D^* is the set of facilities in an optimum solution to I. So the cost of $(\tilde{D})$ is at most $\frac{1}{\epsilon}$ times cost of D^*. They show:

▶ **Lemma 5** (Corollary 5 [8]). $\forall f \in \tilde{D}, \exists g \in D^* : dist(f,g) \leq 2 \cdot avgcost(f)$.

Then they build a modified instance $I' = (G, C', F, f_i)$ where clients of each $cluster(f)$ are not too close or too far away from f compared to $avgcost(f)$. Let opt' be the cost of an optimum solution to I' and opt be the cost of an optimum solution to I. They show that:

▶ **Lemma 6** (Corollary 7 [8]). *For any $R \subseteq F$ if $\text{cost}_{I'}(R) \leq (1+\gamma)\text{opt}' + \delta$ (for some $\gamma, \delta > 0$) then $\text{cost}_I(R) \leq (1 + 2\gamma + 8\alpha\epsilon)\text{opt} + \delta$*

Therefore, a near optimum solution to I' yields a near optimum solution to I. In order to be able to obtain a PTAS they further partition the instance (starting from I) into several instances I_j each of which has certain structural properties.

▶ **Definition 7** (Structured Instance with Bounded Aspect Ratio). *Consider an instance of Facility Location consisting of an edge-weighted graph $G = (V, E)$, a set of clients $C \subseteq V$, and a set of facilities $F \subseteq V$ with opening costs f_i (for each $i \in F$). Suppose we are provided a set $\tilde{D} \subseteq F$ that partitions C into nonempty clusters $\{cluster(i)\}_{i \in \tilde{D}}$. We say that the instance has bounded aspect ratio of the average costs and being structured if the following properties hold:*
i) $\epsilon^2 \cdot avgcost(i) \leq d_G(j,i) \leq \epsilon^{-2} \cdot avgcost(i)$, for each $i \in \tilde{D}$ and each $j \in cluster(i)$,
ii) for each $i \in \tilde{D}$, there exists $i^* \in D^*$ such that $d_G(i, i^*) \leq 2 \cdot avgcost(i)$,
iii) the aspect ratio of the average costs (i.e. $\max_{i,j \in \tilde{D}} \frac{avgcost(i)}{avgcost(j)}$) is bounded by $r = \epsilon^{-O(\epsilon^{-2})}$.
They show that one can partition I into instances $I_j = (G, C_j, F_j, f_i^j)$ such that each instance satisfies the structural properties of Definition 7 and also how to combine solutions for the various I_j to get a good solution for I':

▶ **Lemma 8** (Lemmas 10 and 11 [8]). *Given $D_j \subseteq F$ for I_j, we can build $D \subseteq F$ in polynomial time s.t. $\text{cost}_{I'}(D) \leq \sum_j \text{cost}_{I_j}(D_j) + 10\alpha\epsilon\text{opt}$. Furthermore $\sum_j \text{opt}(\text{I}_\text{j}) \leq (1 + 9\alpha\epsilon)\text{opt}$.*

Recall that all these results only use the metric property of instance I. The preceding lemmas show that to prove Theorem 1 it is sufficient to present a QPTAS for instances satisfying conditions of Definition 7 and this is what we will do. We state this formally.

▶ **Corollary 9.** *Suppose for any constant $\epsilon > 0$ there is a PTAS (resp. QPTAS) for Uncapacitated Facility Location instances in UDGs when we are additionally given $\tilde{D} \subseteq F$ and clusters $\{cluster(i)\}_{i \in \tilde{D}}$ satisfying the properties in Definition 7. Then there is a PTAS (resp. QPTAS) for any Uncapacitated Facility Location instance in UDGs.*

2.2 Overview: A recursive decomposition of UDGs

Adapting the approach from [8]

In order to get a PTAS for well-structured instances in the planar case, [8] uses a Baker-type type layering technique [3] in conjunction with the properties of the instance, and further decompose the instance into instances of constant radius at a small loss. By utilizing balanced separators for planar graphs, they obtain a hierarchical decomposition of the plane embedding of the graph into separate regions (similar to the decomposition of Euclidean instances by Arora [2]). By placing $O(\log n)$ "portals" along each separator they use dynamic programming over this decomposition, while the portals control the interface of different regions.

In our setting, instead of using Baker layering to obtain low diameter instances, we use Theorem 3 to break the instance that satisfies conditions of Definition 7 (at a small loss) into

low diameter instances. We will use a variant of a balanced separator theorem developed by [12] (which improved over a more complex separator by [27]) for UDGs. Roughly speaking, they show that given a UDG, one can find two paths originating from a vertex s to two other vertices x, y that are shortest paths, $P_{s\sim x}$ and $P_{s\sim y}$, such that the removal of these two paths and all the vertices that are within distance 1 of them leaves connected components of size at most $\frac{2}{3}|V(G)|$ (see Theorem 10). We adapt this theorem to get an (almost) balanced separator to obtain a hierarchical decomposition of a low diameter instance into smaller instances. We also place $O(\log n)$ portals at these separators and use Dynamic Programming to combine the solutions. After a logarithmic depth of hierarchical decomposition, we arrive at instances that are easy to solve using other methods (e.g. another PTAS that is described in Theorem 4).

Balanced (partly) Separators for UDGs

To obtain our dynamic program scheme, we would like to be able to break a UDG into (almost) balanced parts by picking a separator. This will act as a "cut" in the disection schema developed by Arora [2] that has been used in designing PTAS's for various optimization problems on Euclidean plane. For this, we utilize the balanced separator theorem for UDGs as presented by Yan et al. [12]. Let $N^i_G[v] = \{u \in V(G) : d_G(v,u) \le i\}$ and $N^i_G[S] = \cup_{v\in S} N^i_G[v]$. A hop-shortest path between two nodes x, y is a path with the minimum number of edges.

▶ **Theorem 10** (Theorem 13 in [12]). *For a UDG G, $X \subset V(G)$ and a root $s \in V(G)$, there exist two nodes x, y of $V(G)$ and hop-shortest paths $P_{s\sim x} = (s, \ldots, x)$ and $P_{s\sim y} = (s, \ldots, y)$ for which the removal of $N^1_G[P_{s\sim x}] \cup N^1_G[P_{s\sim y}]$ from G yields components each having at most $\frac{2}{3}|X|$ vertices from X.*

Theorem 13 in [12] actually only proves it for the case $X = V$. See Appendix A for discussion about why it holds for general $X \subseteq V$. This theorem serves as a counterpart to the well-known balanced shortest paths separator theorem in planar graphs by Lipton and Tarjan. However, it poses a challenge due to the fact that the separator is formed by the 1-neighborhoods of the paths rather than just the paths themselves. This means that we must not only remove the shortest paths but also all nodes within a distance of one from the nodes on the shortest paths. To tackle this challenge, we narrow our focus to cases where the average distance between clients and facilities is relatively large. By doing so, we can assume that clients and facilities, which end up on two different sides of the shortest path, always get connected via nodes in $V(P_{s\sim x} \cup P_{s\sim y})$. Note that, as stated in Theorem 11 (which is a slight modification of this theorem), the only exceptions to this assumption occur when the path between clients and facilities crosses the border using an edge whose endpoints are very close to nodes in $V(P_{s\sim x} \cup P_{s\sim y})$. However, using the fact that we can assume the average distance between clients and facilities is relatively large, we can force those paths to visit nodes in $V(P_{s\sim x} \cup P_{s\sim y})$ with a relatively tiny error.

In the following, we demonstrate that a slight modification of this theorem yields a balanced, yet partial in some sense, separator for UDGs. More precisely:

▶ **Theorem 11.** *For a UDG G, $X \subset V(G)$ and a source $s \in V(G)$, there exists two nodes x, y of $V(G)$ and hop-shortest paths $P_{s\sim x} = (s, \ldots, x)$ and $P_{s\sim y} = (s, \ldots, y)$ such that removing $V(P_{s\sim x} \cup P_{s\sim y})$ partitions the vertices $V(G \setminus (P_{s\sim x} \cup P_{s\sim y}))$ into two sets G_1, G_2 each having at most $\frac{2}{3}|X|$ vertices from X. Additionally, for any edge $ab \in E(G)$ with $a \in V(G_1)$ and $b \in V(G_2)$, there exists $c \in V(P_{s\sim x} \cup P_{s\sim y})$ such that $d_G(a,c), d_G(b,c) \le 2$.*

This follows easily from Theorem 10 but we provide a proof in Appendix A for the sake of completeness. Note that we say $V(P_{s\sim x} \cup P_{s\sim y})$ is a separator between $V(G_1)$ and (G_2) if there are no edges in G that connect a vertex from set $V(G_1)$ to a vertex from set $V(G_2)$. However, here, we relax this condition and allow for the presence of such edges, provided that their endpoints are in close vicinity to the separator.

3 Proof of Theorem 1

We can now describe our QPTAS for Uncapacitated Facility Location in UDGs. Consider an instance of Facility Location, consisting of an edge-weighted UDG $G = (V, E)$, a set of clients $C \subseteq V$, and a set of facilities $F \subseteq V$ with opening costs f_i (for each $i \in F$). Let D^* denote the set of facilities opened by the optimal solution, with opt representing the cost of this solution. Additionally, let $\tilde{D}$ denote the $O(1/\epsilon)$-approximation solution as described in Section 2.1, and let $\text{cost}(\tilde{D})$ represent the cost of this solution. Note $\text{cost}(\tilde{D}) = \sum_{i \in \tilde{D}} \left(f_i + \sum_{j \in cluster(i)} d_G(j, i) \right) = O(\text{opt}/\epsilon)$. We assume that the instance satisfies the properties mentioned in Definition 7. Let $r = \epsilon^{-O(\epsilon^{-2})}$ and $N > 0$ denote the minimum distance between a client and its facility (cluster center) in $\tilde{D}$. It can be verified using the properties of Definition 7 that the inequalities $N \leq d_G(j, f) \leq rN$ and $avgcost(i) \leq rN$ hold for each $i \in \tilde{D}$ and each $j \in cluster(i)$ (these are essentially the conditions in Lemma 9 of [8] except that we can't do scaling in UDG instances, hence we have the factor N). Moreover, based on property (ii) (Definition 7) of the instance, it follows that $d_G(j, D^*) \leq 3Nr$ holds for each client $j \in C$.

Our next step is to decompose the instance into independent subinstances.

▶ **Lemma 12.** *At a loss of $O(\epsilon \cdot \text{opt})$ we can decompose the instance into a number m of independent instances $H_1, H_2, \ldots, H_m$ where each has diameter at most $O(rN/\epsilon^2)$.*

The full proof is deferred to Appendix A. Intuitively, it uses a BFS from a node s to define layers and partitions V by breaking the graph every $O(rN/\epsilon^2)$ layers using a random offset for this partitioning. Facilities of $\tilde{D}$ in the first $O(rN)$ layers of each part are opened, which has low cost on average over the random choice of offset. This chopping procedure is recursively applied $O(1)$ times, after which each component has edge-hop diameter $O(rN/\epsilon^2)$ by Theorem 3. In the proof, we note BFS distances approximate actual weighted UDG distances within a factor of 2 for pairs of non-adjacent nodes so the actual diameter under weighted UDG distances is also $O(rN/\epsilon^2)$ for each component.

It can also be seen that the sum of optimum solutions of all these instances costs at most opt since for each client in H_ℓ their optimum facility will also be in H_ℓ. The rest of our algorithm approximates solutions in each H_ℓ with the following guarantees.

▶ **Lemma 13.** *Given instance I for Facility Location on a UDG G let $\tilde{D}$ be an approximate solution as described above, consider the instances $H_1, \ldots, H_m$ as in Lemma 12. There is a quasi-polynomial algorithm that produces solutions for H_ℓ's such that the total cost of the solutions is at most $O(\epsilon^2 \cdot \text{cost}(\tilde{D})) + (1 + O(\epsilon)) \sum_\ell \text{opt}(\text{H}_\ell)$.*

From this, we immediately get Theorem 1 since the total error is at most $O(\epsilon \cdot \text{opt})$ since the first term in the expression above is at most $O(\epsilon \cdot \text{opt})$ and $\sum_\ell \text{opt}(\text{H}_\ell) \leq \text{opt}$.

We will treat each individual instance H_ℓ separately and will produce a solution for it of cost $(1 + O(\epsilon))\text{opt}(\text{H}_\ell) + \text{E}_\ell$ such that $\sum_\ell E_\ell \leq O(\epsilon^2 \cdot \text{cost}(\tilde{D}))$. A key observation that we use to bound the sum of the additive error bound E_ℓ above is the following. Suppose that N is sufficiently large, i.e. $N > 1/\epsilon^2$ (we will handle the case of small N separately). Note that (as mentioned in the first paragraph of this Section) since $N \leq d_G(j, f)$, therefore $N \cdot |C| \leq \text{cost}(\tilde{D}) = O(\text{opt}/\epsilon)$; so if N is large ($N > 1/\epsilon^2$) and each client $c \in C$ is moved an extra $O(1)$ distance, then it adds at most $O(|C|) = O(\epsilon \cdot \text{opt})$ to the cost of an optimum solution for the modified instance. Hence, this instance still has a solution of cost at most $(1 + O(\epsilon))\text{opt}$. In our algorithm for each H_ℓ we might consider paying an extra $O(1)$ for each client. Using this argument, it can be seen that the total additive error for each H_ℓ will be $O(|C_\ell|)$; summing over all the instances H_ℓ the additive error is at most $O(\epsilon \cdot \text{opt})$. So from now on we assume our instance is one of the H_ℓ and we prove Lemma 13.

3.1 Hierarchical decomposition with portalization

In this section, we assume the instance is a low diameter instance as obtained by applying our chopping operation from the proof of Lemma 12, i.e. one of the H_ℓ. More specifically, at a loss of $O(\epsilon \cdot \text{opt})$ we assume: $N \le d_G(j,i) \le rN$ and $avgcost(i) \le rN$ hold for each $i \in \tilde{D}$ and each $j \in cluster(i)$, and $d_G(j, D^*) \le 3Nr$ for each client $j \in C$, and the diameter is is at most $3rN/\epsilon^2$.

We obtain a hierarchical decomposition of the instance and describe a Dynamic Programming (DP) algorithm based on this decomposition. This decomposition has a logarithmic depth, and each region within the hierarchy is obtained by applying two shortest paths separators to the parent region (for readers familiar with the standard decomposition obtained by a dissection of the plane used in designing PTASs for problems such as TSP and Facility Location on Euclidean planes: one can think of our shortest paths separators as the "line" that breaks the problem into two balanced instances; we will place "portal" at appropriate locations along the separator). One difference in our schema is that the region at each level might have a "boundary" that is composed of separators of all the ancestor of it; so it is bounded by $O(\log N)$ separators (whereas, for e.g., a region defined in the recursive decomposition for TSP is defined by 4 dissecting lines). This is the only reason our running time becomes quasi-polynomial.[1]

Recall $r = \epsilon^{-O(\epsilon^{-2})}$. Observe that if we have a UDG with diameter at most Δ then all the points must be in a bounding box of $2\Delta \times 2\Delta$. Using this, Theorem 4 provides a PTAS when the value of N is small (i.e. at most $1/\epsilon^2$). Therefore, we can assume that $N \ge 1/\epsilon^2$. We provide a solution that has multiplicative approximation factor $(1+O(\epsilon))$ and additive factor E_ℓ that can be charged to the number of clients in the instance (eventually we will have $\sum_\ell E_\ell \le O(\epsilon^2 \text{cost}(\tilde{D}))$.)

Indeed, assuming that N is sufficiently large is vital for our approach, as it enables us to utilize the balanced "partly" separator described in Theorem 11. This separator allows for a hierarchical decomposition of the graph with a logarithmic depth, but it does permit direct interactions between points (within a small distance from the separator path) in the separated regions. By forcing the interaction through the separator nodes, we incur an additive error. However, when the minimum distance between clients and facilities is sufficiently large, this error becomes negligible.

Sparsifying H_ℓ

Let $\Gamma = 3rN/\epsilon^2 = O_\epsilon(N)$ denote the diameter of the graph. Our goal is to obtain a net of size $\text{poly}(\Gamma)$ so that the number of points we deal with is in terms of N (instead of n), while we loose a small factor (compared to opt).

▶ **Lemma 14.** *We can obtain a graph $G'(V', E')$ where $V' \subseteq V(H_\ell)$ with $|V'| = O(\Gamma^4)$, where:*

- *For any two $u, v \in V'$, $d_G(u,v) > 1/8$.*
- *If we move each client and facility in H_ℓ to its nearest point in V', the optimum solution increases by $O(|C_\ell|)$.*
- *Conversely, given a solution to this modified instance we can get a solution to H_ℓ with additional cost $O(|C_\ell|)$.*

We let $B(v)$ for $v \in V'$ denote all clients and facilities of H_ℓ that moved to v.

We can think of $B(v)$ as the set of nodes in a ball of radius $1/8$ around v. The proof is by a fairly standard net construction and is found in Appendix A. Note that the error described above when summed over all H_ℓ's, is at most $O(|C|) = O(\text{cost}(\tilde{D})/N) = O(\epsilon \cdot \text{opt})$ (since $N|C| \le \text{cost}(\tilde{D})$).

Hierarchical decomposition of G'

We obtain a hierarchical decomposition of G' which will be associated with a rooted binary tree

[1] We conjecture a more careful analysis of our scheme and applying the separator could imply a "boundary" that is defined by only a few separators. This would turn the whole algorithm into a PTAS.

T where the root of T is V'. We will have a Dynamic Programming to solve Facility Location using this decomposition. This will be similar to the hierarchical decompositions obtained for PTAS's on the Euclidean instances (e.g. [2]) except that we use the separator in Theorem 11 instead of dissecting lines and that the leaf nodes in our decomposition tree in our case do not correspond to trivial instances (typically a leaf node is a box with only one point in it). In our case, each leaf node is an instance with certain properties which enables us to solve it in quasi-polynomial time at a small loss.

Let us describe the decomposition for G'. Our hierarchical decomposition tree T is associated with a labeling $\psi : V(T) \to 2^{V(G')}$: we set the label of the root of T to be $V(G')$. Each $t \in T$ represents a subgraph $G'[\psi(t)]$ with the property that if t_1, t_2 are children of t, then $\psi(t) = \psi(t_1) \cup \psi(t_2)$. We use $\mathrm{bd}(t)$ to denote the boundary vertices of $\psi(t)$ and the rest of the vertices, $\psi(t) - \mathrm{bd}(t)$, we call them the *core* vertices and are denoted by $X(t)$. The "boundary" is obtained by finding a separator using Theorem 11 and is added to the boundary that is inherited from the parent (details to follow). The boundary of the root node is the empty set (and so all the vertices of G' are core vertices for the root node of T). The overall idea is whenever a current leaf node $t \in T$ has $|X(t)| > 1$ we decompose $G'[\psi(t)]$ into two smaller subgraphs and we obtain children t_1, t_2 for t. More specifically, starting from when T is a single node t corresponding to $V(G')$, iteratively, for each leaf t of T, if $|X(t)| > 1$, apply Theorem 11 over $G'[\psi(t)]$ with $X = X(t)$ to obtain two graphs G'_1 and G'_2, along with the two shortest paths $P_{s\sim x}$ and $P_{s\sim y}$. We use the same vertex s to find the two shortest paths $P_{s\sim x}, P_{s\sim y}$ for each node $t \in T$ and we make sure this vertex s is passed down. Define $\psi(t_i)$ to be $G[V(G'_i) \cup V(P_{s\sim x}) \cup V(P_{s\sim y})]$ and $\mathrm{bd}(t_i) = \mathrm{bd}(t) \cup P_{s\sim x} \cup P_{s\sim y}$. This also defines $X(t_i)$ to be the subset of vertices of $X(t)$ that fall into $V(G'_i)$ and not in the boundary of t_i. Note that our separator $P_{s\sim x} \cup P_{s\sim y}$ separates the vertices $\psi(t)$ of our sub-instance into parts each of which has at most $\frac{2}{3}|X(t)|$ many core vertices.

Every time we find a separator $P_{s\sim x} \cup P_{s\sim y}$ to break the graph $\psi(t)$ (as described) we also designate $m = O_\epsilon(\log N)$ of the vertices of each of these two paths as *portals*. More specifically, let $\delta = O(\epsilon\Gamma/\log N)$ and designate some of the vertices of these paths as portal so that they are δ apart (note that as mentioned before, the hop-distance and UDG distances are within factor 2 of each other). Since these paths have hop-distance length at most Γ, we will have $O_\epsilon(\log N)$ portals per path. Our intention is that if two points $u \in X(t_1)$ and $v \in X(t_2)$ are to be connected they have to go through portals of the boundary. This is illustrated in Figure 1.

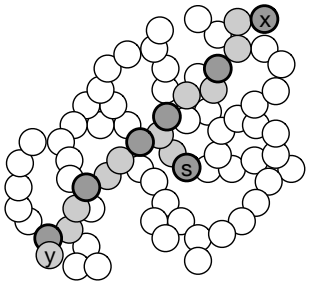

Figure 1 An depiction of two paths $P_{s\sim x}$ and $P_{s\sim y}$ from Theorem 11 in grey at the top level in the hierarchical decomposition. The balls are radius-1/2 balls around points in G', so two intersect if and only if they are adjacent. The darker grey nodes with thicker borders are evenly-spaced portals. Intuitively, at each portal we will keep track of the distance to the nearest facility we will open on either side of the separator.

Observe that the depth of T is $h = O(\log \Gamma) = O_\epsilon(\log N)$ (recall that $|V(G')| = O(\Gamma^4)$). By construction, for each node $t \in T$, the region defined by $\psi(t)$ consists of core nodes in $X(t)$ and the boundary $\mathrm{bd}(t)$ consists of (at most) h separators (which are shortest paths starting from s)

obtained using Theorem 11 each of length proportional to the diameter of the graph, namely Γ. For any t let Π_t be the set of portals on the boundary of region t. Note that each region boundary is composed of at most $h = O_\epsilon(\log N)$ paths and each path has $O_\epsilon(\log N)$ portals, so each t has at most $O_\epsilon(\log^2 N)$ portals.

By Theorem 11 and the way we put the portals (since they are δ apart), it follows that for any two points $u' \in \psi(t_1)$ and $v' \in \psi(t_2)$ (where t_1, t_2 are children of a node $t \in T$), there exists a portal π in the separator that breaks t into t_1, t_2 for which $d_{G'}(u', \pi) + d_{G'}(v', \pi) \leq d_{G'}(u', v') + \delta + 4$. Now if $u \in B(u'), v \in B(v')$ are two vertices of H_ℓ (recall that $B(v')$ is the ball of v' that created net node v' in G'), the distance between u, v to go via their ball centers and then via the portal is bounded as well: $d_G(u, u') + d_{G'}(u', \pi) + d_{G'}(v', \pi) + d_G(v, v') \leq d_{G'}(u', v') + \delta + 4.25 \leq d_G(u, v) + \delta + 5$.

Suppose we require, for each node $t \in T$ with children t_1, t_2, the connection between points in $\psi(t_1), \psi(t_2)$ be via portals in the separator that broke t into t_1, t_2. As argued above, this detour adds at most $\delta + 5$ to the cost for each connection. Since the depth of the recursion tree T is $h = O(\log \Gamma)$, the accumulated error incurred to each client/facility for communicating via portals and centers of their balls (among all the levels of the decomposition) is at most $O((\delta + 5) \log \Gamma)$. Hence, there is a total error of at most $O(|C_\ell| \delta \log \Gamma)$ for all clients connections. By a suitable choice of ϵ' depending on ϵ, and setting $\delta = \frac{\epsilon' \Gamma}{\log \Gamma}$, we get $m = \frac{\log \Gamma}{\epsilon'} = \frac{\log O_\epsilon(N)}{\epsilon'}$, *the total error* for re-routing connections of clients to be via portals (among all levels of decomposition) between regions is bounded by:

$$4|C_\ell|(\epsilon' \Gamma + 5 \log \Gamma) = 4|C_\ell|(\epsilon' O_\epsilon(N) + O_\epsilon(\log N)). \tag{1}$$

This will be our additive error E_ℓ. Recall that we have $N \cdot |C| \leq O(\text{opt}/\epsilon)$, implying $|C| \leq O(\frac{\text{opt}}{\epsilon N})$. This, together with (1) and the fact that $\sum_\ell |C_\ell| \leq |C|$, imply that the total additive error for all H_ℓ's is bounded by $O(\epsilon \cdot \text{opt})$ if ϵ' is sufficiently small.

Note: The observation above (that if we re-route clients connections to be via portals adds only $O(\epsilon \cdot \text{opt})$ to the total cost) will be crucially used in our DP. In other words, if for every client/facility connection distance, we have a rounding error of δ in every level of T, then the total error across all H_ℓ's is bounded by $O(\epsilon \cdot \text{opt})$. In particular, imagine for each leaf node $t \in T$ we have moved all the points (clients/facilities) in the ball of each node in $\psi(t)$ to the nearest portal in Π_t. This adds an extra cost of $O(\epsilon \cdot \text{opt})$ over all levels of decomposition for all H_ℓ's. This simplifies our instance significantly and allows us to find a near optimum solution using Dynamic Programming. We present an overview of the DP procedure before going into details.

Overview of Dynamic Program based on T:
Consider an arbitrary node $t \in T$ and portals $\pi_1, \ldots, \pi_{m'}$ (where $m' = O_\epsilon(\log^2 N)$) on the paths that define $\text{bd}(t)$. For each portal π_i we will have two values $in(\pi_i), out(\pi_i)$: $in(\pi_i)$ indicates (approximately) the distance to the nearest facility (to π_i) that is supposed to be open in the instance defined by $\psi(t)$, and $out(\pi_i)$ indicates (approximately) the distance to the nearest facility (to π_i) that will be open outside the region defined by $\psi(t)$ (and clients inside the balls of vertices of $\psi(t)$ can be connected to them via π_i by paying an additional distance cost of $out(\pi_i)$). These distances are "approximate" since we only keep multiples of δ, since having a precision parameter of δ (as argued above) results in total error $O(\epsilon \cdot \text{opt})$.

DP Table:
For any $t \in T$ and any two vectors $\vec{in}, \vec{out}$ of dimensions m' (for the portals of t) we will have a DP table entry $A[t, \vec{in}, \vec{out}]$. This entry is supposed to store the (approximate) cost of an optimum solution to the Facility Location instance defined by the balls $B(\psi(t))$ (where $B(S) = \cup_{v \in S} B(v)$) subject to the following conditions:

- for each portal π_i there is an open facility in the solution with distance to π_i at most as specified by $in(\pi_i)$,

- for each client either it is served by an open facility inside, or is paying connection cost to go to a portal π_i and if $n(\pi_i)$ is the number of clients that go to portal π_i then they pay $n(\pi_i) \times out(\pi_i)$ to get serviced by a facility outside of $\psi(t)$ at distance $out(\pi_i)$. This will be part of the cost for the solution.

We say vector $\vec{in}$ (and similarly $\vec{out}$) is *valid* if it satisfies the following condition: for any two portals π, π', if their distance (rounded to the nearest multiple of δ) is z then $|in(\pi) - in(\pi')| \leq z + \delta$. This condition clearly must hold as if there is a facility with distance $in(\pi)$ from π then the nearest facility to π' cannot be further than $\in (\pi) + z + \delta$ away from it. Our DP table is computed only for valid vectors for each node $t \in T$.

Recurrence Overview:
Suppose we have computed (approximate) solutions for each problem defined for each leaf node of T and each (valid) vectors $\vec{in}, \vec{out}$ for the portals. Our DP will compute the solution for the instance of root of T (i.e. $V(G')$) in a bottom up manner from the leaf nodes to the root. For each internal node t with children t_1, t_2 and vectors $\vec{in}, \vec{out}$ (for t) and $\vec{in}_1, \vec{out}_2, \vec{in}_2, \vec{out}_2$ (for t_1, t_2, respectively) it will see if the three solutions are "consistent". We will define this formally in the next section but at a high level this checks whether the solutions for t_1, t_2 (given the portal vectors) can be combined so that we get a solution for $\psi(t_1) \cup \psi(t_2)$ where it is consistent with the vectors of portals specified by t.

If one keeps the distances (stored in portal vectors) as as multiples of δ, since distances are bounded by $\Gamma = O_\epsilon(N)$, there will be $O_\epsilon(\log N)$ choices for each portal, which leads to a $(\log N)^{O_\epsilon(\log^2 N)}$ table size.[2] The base case will be solved using an exhaustive search by guessing a subset of portals and opening the cheapest facility on that portal followed by a check if the distance requirements for facilities are satisfied: the best consistent solution will be kept and we will store ∞ if there is no such solution.

3.2 Dynamic Program

In this section we describe the details of our dynamic program based on the decomposition T. As mentioned earlier, each node $t \in T$ corresponds to a set of vertices $\psi(t) \subset G'$ with boundary nodes $\text{bd}(t)$ compose of at most h separators, each of which is two paths initiating from s (using Theorem 11) and each containing m portals that are δ apart (for a total of $m' = O_\epsilon(\log^2 N)$ portals). Note that the diameter of the instance was Γ, so if we round a distance to the nearest multiple of δ, we get an integer of value at most $O(\Gamma/\delta) = O_\epsilon(\log N)$. As mentioned in the overview, for each pair of m'-dimension vectors $\vec{in}, \vec{out}$, where each entry $in(\pi_i), out(\pi_i)$ (for portal π_i) is an integer at most $O_\epsilon(\log N)$, we have an entry in our DP table $A[t, \vec{in}, \vec{out}]$. As mentioned before, we only consider valid vectors $\vec{in}, \vec{out}$.

The goal of this subproblem is to identify a set of facilities to open in $B(\psi(t))$ and to assign each client in $B(\psi(t))$ to either an open facility or bring to a portal π_i such that minimizes the total opening cost plus connection cost such that: (i) For each portal π_i, there is an open facility in $B(\psi(t))$ of distance at most $in(\pi_i)$ (rounded to the nearest multiple of δ); and (ii) For any portal

[2] We can reduce this using a trick that is also used earlier (e.g. see [2]) to show how to reduce the size of the portal vectors by storing only "smoothed" vectors. Informally, the observation that helps is that if we keep distances of the nearest facility (inside or outside) for a portal π_i as the multiple of δ then if the value for portal π_i is σ then the value for portal just before or after π_i on the separator path that π_i belongs to is in $\{\sigma - 1, \sigma, \sigma + 1\}$. Thus, the total number of vectors we need to consider for a node $t \in T$ is $O_\epsilon(\log^2 N \times 3^{m'}) = 2^{O_\epsilon(\log^2 N)}$, where there are $O_\epsilon(\log^2 N)$ choices for the first portal values and then for the subsequent portals there are only 3 choices for each distance.

π_i suppose $n(\pi_i)$ is the number of clients in $B(\psi(t))$ that are connected to π_i by the solution. The connection cost for these clients is the cost they pay to be connected to π_i plus $n(\pi_i) \times out(\pi_i)$.

A well-structured near-optimum solution

We first show the existence of a near optimum solution with certain properties such that our DP actually finds such a near optimum solution. Starting from an optimum solution D^*_ℓ on H_ℓ we make some changes to it to satisfy the properties we want, while increasing the cost a little only. First for each node $v \in V'$ with $D^*_\ell \cap B(v) \neq \emptyset$ we consider keeping the cheapest open facility in $B(v)$ and closing all the others and re-routing all the clients that were served by other facilities in $B(v)$ to that single open facility via v. This adds at most $1/4$ to the distance each client has to travel (via v) as nodes in $B(v)$ have distance at most $1/8$ to v. This increases the cost by at most $O(|C_\ell|)$ over all clients. We can assume the open facilities are on the centers of the balls now. Let's call this new (near optimum) solution D'_ℓ. Next we modify the solution further by the following process. We say a solution is t-adapted for a node $t \in T$ if (i) for each of its descendant t' the solution is t'-adapted, and (ii) for each client $c \in B(\psi(t))$, if c is connected to a facility outside of $B(\psi(t))$ then it is connected via a portal $\pi \in \Pi$, where Π is the set of portals of t.

We start with D'_ℓ and at leaf nodes of T and going up the tree, we make the solution t-adapted for each $t \in T$ at small increase in cost. Let us consider a leaf node $t \in T$ with $X(t) = w_t$ (the case when $X(t) = \emptyset$ is even easier) and boundary $\text{bd}(t)$ corresponds to paths with a total of m' portals $\Pi = \pi_1, \ldots, \pi_{m'}$. We consider w_t as a portal as well and add it to Π. Suppose $\Pi' = \pi_{a_1}, \pi_{a_2}, \ldots, \pi_{a_\sigma}$ is the subset of Π, where there is a facility in distance at most δ of π_{a_i} in $D' \cap \psi(t)$. For each such π_{a_i}, we assume we have kept open the cheapest facility within δ of it. For any client $c \in B(\psi(t))$ if c was connected to a facility in $B(\psi(t))$ in D' (note that all of those are within distance δ of some portal in Π') we consider routing c to the nearest portal in Π' (and then from there to the single cheap facility we kept open). Note that this increases the connection cost for each client by at most 2δ. If c was connected to a facility outside of $B(\psi(t))$ we re-route it first to a nearest portal π along its way and then from there to be connected to the facility it was connected to outside of $B(\psi(t))$ (i.e. we make the connection of c to go through a portal). This increases the connection cost for each c by at most $\delta + 5$ as argued earlier in the overview and the solution becomes t-adapted.

Now suppose $t \in T$ is a non-leaf node with children t_1, t_2 and supposed D'_ℓ is t_1-adapted and t_2-adapted. We make it t-adapted with small increase in the cost. For any client $c \in B(\psi(t))$, if c is connected to a facility in $\psi(t)$ we don't need to make any further changes. Otherwise we make a detour for the connection of c to go through one of the portals Π. This detour increases the connection cost of a client by at most 2δ.

One last change we do is in the calculation of the cost of the adapted solution to make it **portal vector** adapted: for each node $t \in T$, the t-adapted solution also induces vectors $\tilde{in}, \tilde{out}$ for portals of t in the following way. The clients that are served by facilities outside $\psi(t)$ are first going to a portal π (of t). The distance from that portal to the nearest open facility (outside $\psi(t)$) rounded up to the nearest multiple of δ is what induces a value for $\tilde{out}(\pi)$ at node t. We use this rounded (up) value instead of the actual distance in the calculation of the cost of the t-adapted solution. Similarly, for each portal π, the distance to the nearest open facility in $\psi(t)$ rounded to the nearest multiple of δ induces a value $\tilde{in}(\pi)$ for node t. Let $\tilde{in}, \tilde{out}$ correspond to the vectors induced by the t-adapted optimum solution. We assume the cost a client pays to go out of $\psi(t)$ to be connected to a facility via a portal π (from π) is $\tilde{out}(\pi)$. Note that our estimate of the nearest to π open facility inside or outside $\psi(t)$, described by $\tilde{in}(\pi), \tilde{out}(\pi)$ has an additive error of at most δ. Thus, the connection costs for each client at each node t can be larger by δ again. We call this new cost, portal vector adapted.

It is easy to see that if we make the solution t_0-adapted, where t_0 is the root of T, and consider

the portal adapted cost (as described), then the increase in the connection cost for each client over all the nodes of T is at most $O(\delta \log \Gamma)$ (since the height of T is $O(\log \Gamma)$); summed over all the clients this was shown to be $O(\epsilon' |C_\ell| \Gamma)$. Thus, the best t_0-adapted and portal adapted solution has cost $\text{opt}(\mathrm{H}_\ell) + \mathrm{O}(\epsilon' |\mathrm{C}_\ell| \Gamma)$, and for a suitable choice of ϵ' the additive error over all H_ℓ's will be add to at most $O(\epsilon \cdot \text{opt})$.

Recurrence details:
Let $\tilde{in}, \tilde{out}$ correspond to the vectors induced by the t_0-adapted near optimum solution. By this argument it is enough to find compute the entries $A[t_0, \vec{0}, \vec{0}]$ to obtain a $(1 + O(\epsilon))$-approximate solution.

Base case: Let us consider a leaf node $t \in T$ with $X(t) = w_t$ (the case when $X(t) = \emptyset$ is even easier) and boundary $\text{bd}(t)$, which corresponds to paths with a total of m' portals $\Pi = \pi_1, \ldots, \pi_{m'}$. We consider w_t as a portal as well and add it to Π. For any subset $\Pi' = \pi_{a_1}, \pi_{a_2}, \ldots, \pi_{a_\sigma}$ of Π, where there is a facility in $B(\pi_{a_i})$, we consider opening the cheapest facility in $B(\pi_{a_i})$; let's call that facility $f(a_i)$. For any client $c \in B(\psi(t))$ we consider routing c to (i) nearest portal with an open facility, or (ii) to a portal π_{a_i} to be connected outside at a total cost $d_G(c, \pi_{a_i}) + out(\pi_i)$ if this is less than the distance to the nearest portal with an open facility. This will be considered a feasible solution if: for each portal π_i there is a portal $\pi_{a_j} \in \Pi'$ with an open facility such that $d_G(\pi_i, \pi_{a_j}) \leq in(\pi_i)$. The cost for $A[t, \vec{in}, \vec{out}]$ will be the cost of the cheapest feasible solution over all choices of Π' as described above. If there is no such solution (consistent with vectors $\vec{in}, \vec{out}$, we set $A[t, \vec{in}, \vec{out}] = \infty$. It is easy to see that we obtain the best t-adapted portal adapted solution.

Filling in the rest of DP table: Now consider an arbitrary (non-leaf) node $t \in T$ and vectors $\vec{in}, \vec{out}$ and suppose t has children t_1, t_2. Suppose all the entries of t_1, t_2 for all vectors of portals are computed. Let Π, Π_1, Π_2 be the set of portals of t, t_1, t_2, respectively. For vectors $\vec{in}_1, \vec{out}_1$ for portals of t_1, and vectors $\vec{in}_2, \vec{out}_2$ for portals of t_2 we say subproblems $(t, \vec{in}, \vec{out})$, $(t_1, \vec{in}_1, \vec{out}_1)$, $(t_2, \vec{in}_2, \vec{out}_2)$ are consistent if the following hold:

- For each portal $\pi \in \Pi$, either $\vec{in}_1(\pi) = \vec{in}(\pi)$ or $\vec{in}_2(\pi) = \vec{in}(\pi)$.
- For each portal $\pi \in (\Pi_1 \cap \Pi_2) - \Pi$, i.e. a portal that is on the separator of t that creates t_1, t_2: $\vec{in}_1(\pi) = \vec{out}_2(\pi)$ and $\vec{in}_2(\pi) = \vec{out}_1(\pi)$.
- for each $\pi \in (\Pi_1 \cap \Pi_2 \cap \Pi)$ we must have $\vec{out}_1(\pi) = \vec{out}_2(\pi) = \vec{out}(\pi)$.

First observe that checking consistency for the three subproblems can be done in time $\text{poly}(m)$. Then

$$A[t, \vec{in}, \vec{out}] = \min\{A[t_1, \vec{in}_1, \vec{out}_1] + A[t_2, \vec{in}_2, \vec{out}_2]\},$$

where the min is over all vectors $\vec{in}_1, \vec{out}_1, \vec{in}_2, \vec{out}_2$ such that $(t, \vec{in}, \vec{out})$, $(t_1, \vec{in}_1, \vec{out}_1)$, $(t_2, \vec{in}_2, \vec{out}_2)$ are consistent. By induction, assuming that $\vec{in}, \vec{out}, \vec{in}_1, \vec{out}_1, \vec{in}_2, \vec{out}_2$ are induced portal vectors for a t-adapted near optimum solution D' and that $A[t_1, \vec{in}_1, \vec{out}_1]$ and $A[t_2, \vec{in}_2, \vec{out}_2]$ are computed correctly, one can see that we get that the cost of a solution at $A[t, \vec{in}, \vec{out}]$ that is no more than that of optimum t-adapted solution at induced on $B(\psi(t))$.

Run time analysis: Note that diameter of a (connected) UDG is at most n (and we assume the graph is connected since we can run the algorithm on each connected component). Thus $N = O(n)$ and hence the size of the DP is $2^{O_\epsilon(\log^2 N)} = 2^{O_\epsilon(\log^2 n)}$. To compute each base case it takes $2^{O_\epsilon(\log^2 n)}$ time and for each non-leaf node of t which children t_1, t_2 and vectors $\vec{in}, \vec{out}$, the solution $A[t, \vec{in}, \vec{out}]$ can be computed by comparing all triples of valid solutions for t, t_1, t_2; this also takes $2^{O_\epsilon(\log^2 n)}$. Overall the runtime is therefore $n^{O_\epsilon(\log n)}$, where the constant in $O_\epsilon(.)$ is $\epsilon^{-O(\epsilon^{-2})}$.

References

1. Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 79–88, 2014.
2. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 106–113, 1998.
3. Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, jan 1994. doi:10.1145/174644.174650.
4. Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015.
5. Xiuzhen Cheng, Xiao Huang, Deying Li, Weili Wu, and Ding-Zhu Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.10097, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.10097, doi:10.1002/net.10097.
6. Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. Near-linear time approximation schemes for clustering in doubling metrics. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.
7. Vincent Cohen-Addad, Philip N Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. *SIAM Journal on Computing*, 48(2):644–667, 2019.
8. Vincent Cohen-Addad, Michał Pilipczuk, and Marcin Pilipczuk. A polynomial-time approximation scheme for facility location on planar graphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 560–581. IEEE, 2019.
9. Jittat Fakcharoenphol and Kunal Talwar. An improved decomposition theorem for graphs excluding a fixed minor. In *RANDOM-APPROX*, pages 36–46, 2003.
10. Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 483–492, 2003.
11. Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of algorithms*, 31(1):228–248, 1999.
12. Elfarouk Harb, Zhengcheng Huang, and Da Wei Zheng. Shortest path separators in unit disk graphs. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 66:1–66:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.ESA.2024.66, doi:10.4230/LIPICS.ESA.2024.66.
13. Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM*, 32(1):130–136, jan 1985. doi:10.1145/2455.214106.
14. Harry B Hunt, Madhav V Marathe, Venkatesh Radhakrishnan, S.S Ravi, Daniel J Rosenkrantz, and Richard E Stearns. Nc-approximation schemes for np- and pspace-hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238–274, 1998. URL: https://www.sciencedirect.com/science/article/pii/S0196677497909032, doi:10.1006/jagm.1997.0903.
15. Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80:830–848, 2018.
16. Philip Klein, Serge A Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 682–690, 1993.
17. Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 682–690, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/167088.167261.

18 Fabian Kuhn, Tim Nieberg, Thomas Moscibroda, and Rogert Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 97–103, 2005.

19 James R. Lee. Separators in region intersection graphs, 2016. arXiv:1608.01612.

20 James R. Lee. Separators in Region Intersection Graphs. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:8, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2017/8197, doi:10.4230/LIPIcs.ITCS.2017.1.

21 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

22 Xiang-Yang Li, Wen-Zhan Song, and Yu Wang. Efficient topology control for ad-hoc wireless networks with non-uniform transmission ranges. *Wireless Networks*, 11(3):255–264, 2005.

23 Tomomi Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In Jin Akiyama, Mikio Kano, and Masatsugu Urabe, editors, *Discrete and Computational Geometry*, pages 194–200, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

24 Imran A. Pirwani and Mohammad R. Salavatipour. A weakly robust PTAS for minimum clique partition in unit disk graphs. *Algorithmica*, 62(3-4):1050–1072, 2012. doi:10.1007/s00453-011-9503-8.

25 Erik Jan van Leeuwen. Approximation algorithms for unit disk graphs. In *Graph-Theoretic Concepts in Computer Science: 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers 31*, pages 351–361. Springer, 2005.

26 Chenyu Yan, Yang Xiang, and Feodor F Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry*, 45(7):305–325, 2012.

27 Chenyu Yan, Yang Xiang, and Feodor F. Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry*, 45(7):305–325, 2012. URL: https://www.sciencedirect.com/science/article/pii/S0925772112000387, doi:10.1016/j.comgeo.2012.01.015.

A Missing Proofs

Proof of Theorem 3. Unit disk graphs are *string graphs* - intersection graphs of continuous arcs in the plane. Lemma 1.4 in [19] then implies each unit disk graphs is a so-called *region intersection graphs* over a planar graph (see the introduction in [19] for a definition). Since planar graphs exclude the complete graph K_5 as a minor, then Corollary 4.3 in [19] immediately implies the result. ◀

Proof of Theorem 10. We simply describe how to adapt the proof in [12] to the more general setting with $X \subseteq V$. Intuitively, [12] reduces their separators to those of finding separators in planar graphs. In fact, they use a vertex-weighted separator theorem in planar graphs in their proof (see Theorem 2 in [12]) to distinguish nodes of V from auxiliary nodes created in their reduction.

Precisely, in the proof of Lemma 11 of [12] one can make the following modification to the start of the second paragraph: for vertices $u_1, \ldots, u_{\ell(u)} \in V'$ that correspond to some vertex $u \in V$, if $u \in X$ then we give a weight of 1 to u_1 and a weight of 0 to $u_2, \ldots, u_{\ell(u)}$ and if $u \in V - X$ then give all $u_1, \ldots, u_{\ell(u)}$ a weight of 0. The subsequent appeal to finding balanced separating cycles vertex-weighted planar graphs ensures the final UDG separator leaves each component having at most $\frac{2}{3} \cdot |X|$ nodes from X. ◀

Proof of Theorem 11. Consider shortest paths $P_{s\sim x} = (r, \ldots, x)$ and $P_{s\sim y} = (r, \ldots, y)$ by Theorem 10, partition the components of $G \backslash (N^1_G[P_{s\sim x}] \cup N^1_G[P_{s\sim y}])$ into two graphs G'_1 and G'_2 each containing at most $\frac{2}{3}|X|$ vertices from X.

Partition $(N^1_G[P_{s\sim x}] \cup N^1_G[P_{s\sim y}]) \backslash (P_{s\sim x} \cup P_{s\sim y})$ into two sets S_1, S_2 such that each $G'_i \cup S_i$ (for $i = 1, 2$) has size at most $\frac{2}{3}|X|$ vertices from X: such a partition can be obtained greedily by starting with $S_1 = S_2 = \emptyset$ and then iteratively adding nodes from $(N^1_G[P_{s\sim x}] \cup N^1_G[P_{s\sim y}]) \backslash (P_{s\sim x} \cup P_{s\sim y})$ to either S_1 or S_2 while maintaining $|(V(G'_i) \cup S) \cap X| \leq \frac{2}{3} \cdot |X|$. Observe this property is true initially when $S_1 = S_2 = \emptyset$ and it is trivial to maintain by adding each new node to part with the fewest nodes from X.

We claim that $G_i = G'_i \cup S_i$ satisfies the required property. Let $ab \in E(G)$ be such that $a \in V(G_1), b \in V(G_2)$, then it cannot be that $a \in G'_1$ and $b \in G'_2$ by Theorem 10. Without loss of generality, say $b \notin G'_2$. Then there exists $c \in V(P_{s\sim x} \cup P_{s\sim y})$ such that $cb \in E(G)$. Then a, b, c is a path of length 2 from a to c. ◀

Proof of Lemma 12. We begin with the following observation: If T is a BFS tree from a vertex s in a UDG G, then for any two vertices u, v at levels $i, i+2$ of the tree respectively (for any i) we have their (weighted) distance in G is strictly larger than 1 (or else they would be adjacent and hence cannot be at two levels $i, i+2$ of the BFS tree) and no more than 2 (as any two adjacent vertices have distance at most 1), i.e. $1 < d_G(u, v) \leq 2$. Thus, the BFS will 2-approximate actual (weighted) shortest paths for any node that is not a neighbour of s.

Now suppose we run a BFS from an arbitrary node s and group the vertices into layers where layer i consists of all the nodes of BFS at levels between $(i-1)14Nr, \ldots, 14iNr - 1$. Note that the "thickness" of a layer is $14Nr$ levels of BFS, so for any two vertices u, v in layers $i, i+2$: $d_G(u, v) \geq 7Nr$. Since the distance of any client to their facilities in the optimum is at most $3Nr$, no path from a client to a facility (in the optimum) would cross an entire layer. Now we group consecutive layers into bundles of $\lceil \frac{1}{\epsilon^2} \rceil$ layers, with a random off-set chosen from $0, \ldots \lceil \frac{1}{\epsilon^2} \rceil$. Suppose we call the first layer of each bundle a "red" layer and all the other layers of a bundle are blue; so between every two "red" layers we have $\lceil \frac{1}{\epsilon^2} \rceil - 1$ blue layers. We open all the facilities of $\tilde{D}$ in the red layers and serve the clients in their cluster. Based on the random shift and the fact that $\tilde{D}$ was a $O(1/\epsilon)$-approximate solution, the total cost incurred to open these facilities and serve their clients is at most $O(\epsilon^2 \cdot \text{cost}(\tilde{D})) = O(\epsilon \cdot \text{opt})$. So we can assume all these facilities are open (i.e. have zero opening cost) and we delete the clients they have served. For any other client left in the red layer, they can be partitioned into two parts: those in the top $7Nr$ levels are called *top* red clients and those in the bottom $7Nr$ levels are called *bottom* clients. Since for each client j, $d_G(j, D^*) \leq 3Nr$, the top clients cannot cross over the bottom $7Nr$ levels to be served by a facility. Similarly, the bottom clients cannot be crossing the top $7Nr$ levels to be served by a facility. We show how this breaks the instance into independent instances.

For the remaining (blue) layers in every bundle we consider the clients and facilities in those layers, together with the facilities and remaining clients in the nearest $7Nr$ levels of the two red layers above and below them. Note that these instances are now independent since for every client j, $d_G(j, D^*) \leq 3Nr$, so no client in a blue layer would need to pass beyond $7Nr$ levels into a red layer to reach its facility in optimum. Similarly, the remaining red clients will only need the facilities in the blue layers they are grouped with. This means we can solve the blue layers of each bundle (together with the facilities and clients in a strip of $7Nr$ layers above and below) independently. So we consider the blue layers of each bundle plus the $7Nr$ (red) layers above and below as one instance (recall the facilities in the red layers are open). This means we can assume we have deleted any connections (edges) between these independent instances. This is similar to one round of chopping in the proof of Theorem 3. We perform a sequence of $O(1)$ chopping rounds as above on the graph and utilizing Theorem 3, we can assume that the weak diameter of

each independent instance generated is bounded by rN/ϵ^2 and the total cost paid for the facilities in the layers chopped is $O(\epsilon \cdot \text{opt})$; those facilities now have opening cost zero.

So from now on (at a loss of $O(\epsilon \cdot \text{opt})$) we focus on each independent instance where the weak diameter is bounded by rN/ϵ^2. Let's call these instances $H_1, H_2, \ldots$. For each such instance H_ℓ we use C_ℓ to denote the clients that belong to H_ℓ. It is easy to see that the C_ℓ's are disjoint. Next, we modify H_ℓ's so that they have bounded diameter (not just weak diameter): we add all the vertices of G that are within distance rN/ϵ^2 of some vertex of H_ℓ to H_ℓ. Now for each H_ℓ we have that the diameter (not just weak diameter) of H_ℓ is bounded by $3rN/\epsilon^2$. Note that the set of clients and facilities of H_ℓ is the same as before (we do not bring in the clients and facilities that were outside of H_ℓ when adding vertices to bound the diameter). ◀

Proof of Lemma 14. We construct a new graph G' by selecting a subset of vertices from $V(H_\ell)$, denoted as V' (V' will be a "net"): start by adding an arbitrary vertex of $V(H_\ell)$ to V' and then iteratively include nodes from $V(H_\ell)$ that have a minimum Euclidean distance of at least $1/8$ from nodes in V' (alternatively we can think of picking a node from $V(H_\ell)$ to be added to V' and then deleting all the nodes within distance $1/8$ of it from $V(H_\ell)$ iteratively). Note that after this round is done, $|V'|$ is $O(\Gamma^2)$. Furthermore, for each pair of vertices $u, v \in V'$, if there is a vertex u' within distance $1/8$ of u in $V(H_\ell) - V'$ and a vertex v' within distance $1/8$ of v in $V(H_\ell) - V'$ where $u'v' \in E(G)$ then add both u', v' to V'. Note that in this case, $uu', vv', u'v'$ all are edges in G'. This augmentation results in the graph G' (over V') with a total number of vertices of $O(\Gamma^4)$. We arbitrarily order the nodes of G' and for each node $v' \in G'$, we define $B(v')$ as the set of nodes in H_ℓ that lie inside the Euclidean ball of radius $1/8$ centered at v' but outside the balls of previously processed nodes. We focus on the UDG induced by G'.

To see why we can focus on the net G' and how it helps: each client and facility in the instance H_ℓ is moved to its nearest point in V' (keeping only the cheapest facility moved to a point $v \in V'$ if multiple move there). This instance has a solution of cost $O(|C_\ell|)$ more than with H_ℓ since each client moves at most two extra steps of distance $\leq 1/8$. Conversely, given any solution to this new instance G' we obtain a solution in H_ℓ by opening the same set of facilities except at their original locations. Again, clients move an additional $O(1)$ each when translating from the solution in G' to the solution in H_ℓ.

So the total error will be at most $O(|C_\ell|)$ (whereas the cost of optimum for H_ℓ was at least $O(N|C_\ell|)$). Also, it is easy to see that the size of the graph G', is in terms of N now ($O(\Gamma^4)$). ◀

B PTAS for Facility Location on UDG in Bounded Regions

In this section we present a PTAS for Facility Location in UDG in the special case that the point set P is contained within a bounding box of size $L \times L$ in the plane where L can be regarded as a constant, i.e. prove Theorem 4.

Consider an instance of Facility Location consisting of an edge-weighted graph $G = UDG(P)$, a set of clients $C \subseteq P$, and a set of facilities $F \subseteq P$ with opening costs f_i (for each $i \in F$). Let $D^* \subseteq F$ be the facilities in an optimum solution and let $i^*_j \in D^*$ denote the facility that serves the client j in D^*. Suppose we know D^* (this assumption will be removed) and let $\epsilon > 0$ be the error parameter. We greedily form an *ϵ-net* F' as follows:

- Sort the facilities in D^* by their opening costs in non-decreasing order.
- In this order, while there is some $i \in D^*$ such that $d_G(i, F') > \epsilon$, add i to F'.

This procedure ensures that the resulting F' forms an ϵ-net, where no facility in D^* is at a distance greater than ϵ from F'.

▷ Claim 15. $|F'| \leq O(L/\epsilon^2)$.

Proof. The balls of radius $\epsilon/2$ centered at each point in F' are interior-disjoint. These balls collectively occupy a total area of $\Omega(\epsilon^2 \cdot |F'|)$. The proof follows from the fact that the balls are entirely contained within a square of side length $L + \epsilon$. ◀

Now we divide the instance into sub-instances using a random grid of size $1/2$ that splits the bounding box into squares of size at most $1/2 \times 1/2$. Note $d_G(i,j) \leq 1$ for any two points i and j lying in the same cell. As a result, the metric between points within a cell can be treated as Euclidean distance. For any client $j \in P$, we say j is **cut** if j and i^*_j lie in different grid cells. Let $c^*_j = d_G(j, i^*_j)$.

▷ Claim 16. For any point $j \in P$, $\mathbf{Pr}[j \text{ is cut}] \leq 2c^*_j$.

Proof. This is obvious if $c^*_j \geq 1/2$, so let's assume $c^*_j < 1/2$. The probability that a horizontal line in the grid separates points p, q is at most $|pq|$. Same when considering vertical lines in the random grid. Since $c^*_j < 1/2$, then the centre serving j has a direct connection with j so $c^*_j = d_G(j, i^*)$ as well. ◀

For each grid cell c, let X_c be the restriction of the points in the input to cell c. Recall that, in the prize-collecting version of the facility location problem (Prize Collecting Facility Location), in addition to the input for facility location, each client j is associated with a penalty cost π_j. This penalty cost can be paid instead of the connection cost. The goal is to find an optimal solution that minimizes the total cost, including opening costs and both connection costs and penalties. Define an Euclidean Prize Collecting Facility Location instance for each cell c. For $j \in X_c$, its penalty is $\pi_j := d_G(j, F')$. Let $D^*_c := (D^* - F') \cap X_c$ be the optimum facilities in cell c that are not part of the net.

▷ Claim 17. The optimum Prize Collecting Facility Location solution for this instance has cost at most

$$\sum_{i \in D^*_c} f_i + \sum_{j \in X_c} c^*_j + \sum_{j \in X_c : j \text{ cut}} \epsilon.$$

Proof. Consider the solution that opens D^*_c. If a point j is not cut, we can directly connect it to its centre in D^*_c paying a cost of c^*_j (since the cell c has dimensions $1/2 \times 1/2$ then the direct connection is possible).

Otherwise, we can pay the penalty for j. Note this is upper bounded by moving j to its optimum centre in D^* (paying c^*_j) and then from there to the nearest net point (paying an additional ϵ). ◀

Proof of Theorem 4. Consider the following algorithm:

1. For all possible choices of $F' \subset F$ with $|F'| \leq O(L/\epsilon^2)$ do
   - Partition the instance into sub-instances using a random grid of size $1/2$. Let $\mathcal{C}$ be the corresponding cells.
   - For each $c \in C$ run a PTAS on the corresponding Euclidean Prize Collecting Facility Location instance [6].
   - Obtain a solution for the facility location instance: open all facilities in set F', as well as the facilities opened by PTASs in each cell. For every $j \in P$ that paid a penalty in its corresponding Prize Collecting Facility Location instance, assign j to its nearest (in the UDG metric) facility in F'.
2. Output a minimum cost solution, among the solutions obtained.

It is sufficient to show that ϵ-*net* F' satisfies the claim. Using the previous claims, we obtain the following. The cost of opening all facilities in ϵ-*net* F' plus expected total cost of all FACILITY LOCATION solutions for all cells c is at most:

$$\sum_{i \in D^*} f_i + \sum_{j \in X_c} (1 + 2 \cdot \epsilon) \cdot c_j^* \leq (1 + O(\epsilon)) \Big(\sum_{i \in D^*} f_i + \sum_{j \in P} c_j^* \Big)$$

using the fact that for each client j, we always pay c_j^* and, perhaps, an additional ϵ if j is cut. But j is cut with probability at most $2 \cdot c_j^*$. ◀