Approximation Algorithms for the Airport and Railway Problem

Mohammad R. Salavatipour
Department of Computer Science, University of Alberta, Edmonton, Canada

Lijiangnan Tian
Department of Computer Science, University of Alberta, Edmonton, Canada

Abstract

In this paper, we present approximation algorithms for the airport and railway problem (AR) on several classes of graphs. The AR problem, introduced by [?], is a combination of the Capacitated Facility Location problem (CFL) and the network design problem. An AR instance consists of a set of points (cities) V in a metric $d(.,.)$, each of which is associated with a non-negative cost f_v and a number k, which represent respectively the cost of establishing an airport (facility) in the corresponding point, and the universal airport capacity. A feasible solution is a network of airports and railways providing services to all cities without violating any capacity, where railways are edges connecting pairs of points, with their costs equivalent to the distance between the respective points. The objective is to find such a network with the least cost. In other words, find a forest, each component having at most k points and one open facility, minimizing the total cost of edges and airport opening costs. Adamaszek et al. [?] presented a PTAS for AR in the two-dimensional Euclidean metric $\mathbb{R}^2$ with a uniform opening cost. In subsequent work [?] presented a bicriteria $\frac{4}{3}\left(2+\frac{1}{\alpha}\right)$-approximation algorithm for AR with non-uniform opening costs but violating the airport capacity by a factor of $1+\alpha$, i.e. $(1+\alpha)k$ capacity where $0<\alpha\leq 1$, a $\left(2+\frac{k}{k-1}+\varepsilon\right)$-approximation algorithm and a bicriteria Quasi-Polynomial Time Approximation Scheme (QPTAS) for the same problem in the Euclidean plane $\mathbb{R}^2$. In this work, we give a 2-approximation for AR with a uniform opening cost for general metrics and an $O(\log n)$-approximation for non-uniform opening costs. We also give a QPTAS for AR with a uniform opening cost in graphs of bounded treewidth and a QPTAS for a slightly relaxed version in the non-uniform setting. The latter implies $O(1)$-approximation on graphs of bounded doubling dimensions, graphs of bounded highway dimensions and planar graphs in quasi-polynomial time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Facility Location, Approximation Algorithms, Dynamic Programming

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.14

Funding *Mohammad R. Salavatipour*: Supported by NSERC DG.
Lijiangnan Tian: Supported by first authors NSERC.

1 Introduction

We study a problem that integrates capacitated facility location and network design problems. The problem referred to as *Airport and Railway* problem denoted as AR (introduced by [?] and studied further in [?]) is the following. Suppose we are given a complete weighted graph $G=(V,E)$ embedded in some metric space (for instance the Euclidean plane), with two cost functions $f:V\to\mathbb{R}_{\geq 0}$ for opening facilities (also known as *airports*) at vertices (also known as *cities*) and $c:E\to\mathbb{R}_{\geq 0}$ for installing railways on the edges in order to connect cities to airports. We are also given a positive integer $k\in\mathbb{Z}_+$ as the *capacity* of each airport. The goal is to partition the vertices into a set of clusters each of size at most k, find a set of vertices $A\subseteq V$ at which we open facilities (airports) so that each cluster has exactly one airport, and a set of edges $R\subseteq E$, such that the edges on each cluster induce a connected

19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024).
Editor: Hans L. Bodlaender; Article No. 14; pp. 14:1–14:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

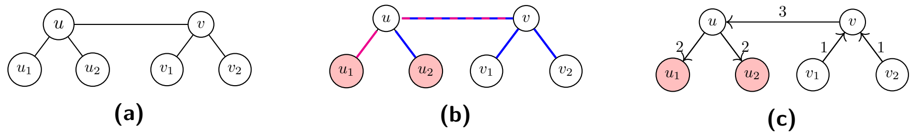


Figure 1 a) An example tree where we assume the airport capacity is 3 and u_1 and u_2 have an opening cost of zero while other vertices have cost infinity; b) The solution to $\widetilde{\text{AR}}$. Pink vertices represent cities with an airport. Each edge is coloured to indicate its cluster. The dashed edge is used by both clusters; c) The solution to AR′. Each directed edge is labelled with its flow value.

graph, while minimising the total cost of the edges plus the opening of selected facilities.

Clearly, the graph induced by each cluster must be a tree. So we have a collection of trees, each of size at most k and each having an open facility. The idea is each open facility serves as an airport that will serve all the cities in the cluster it belongs to (including the city at that vertex). The goal is to minimise the total cost

$$C = \sum_{v \in A} f_v + \sum_{e \in R} c_e.$$

To be more precise, a cluster is an airport and the set of all the cities served by it, together with the set of railways connecting the cities to the airport that forms a tree. Adamaszek et al. [?] also defined a relaxed version of AR (they called AR′) where in a feasible solution a component of the forest might have multiple airports and multiple copies of any edge and each component allows routing one unit of flow from all its cities to the airports so that each airport receives at most k flows and each copy of an edge has capacity k. Note that in this version of the problem, the cities belonging to different airports can share the edges of the network. So an edge might be used by cities from different clusters but no more than k in total; in this case, the cost of the edge occurs only once in the objective.

When considering special metrics (e.g. shortest path metrics induced by trees or other special graph classes) we may not have a feasible solution to AR in the strict setting that clusters need to be disjoint. For this reason, we consider a slightly relaxed version of AR, denoted by $\widetilde{\text{AR}}$ where the clusters do not need to be edge-disjoint but each cluster will pay for the edges it uses separately. In other words, each edge is allowed to be used by multiple clusters but each of them needs to pay the cost of the edges they use separately. Considering this relaxed version becomes useful when we are working on specific metrics e.g. shortest path metrics of certain graph classes such as trees (e.g. see Figure 1). Note that in $\widetilde{\text{AR}}$, each connected component in a feasible solution may contain multiple clusters and the total cost that we want to minimise is $\sum_{v \in A} f_v + \sum_{e \in R} c_e \cdot \phi(e)$ where $\phi(e)$ is the number of clusters using the edge e. We highlight that AR′ is a strictly more relaxed setting vs. $\widetilde{\text{AR}}$. In AR′ the cities sending flows to different airports can share the edges of the network and if the flow over an edge is $\leq k$ (even if used to send flow to different airports) the cost of the edge is paid for only once. This is not the case in $\widetilde{\text{AR}}$. For instance, a feasible solution to $\widetilde{\text{AR}}$ in this Figure 1 has two clusters, one u_1, u, v and the other u_2, v_1, v_2 and has a total cost of 6 whereas a feasible solution to AR′ has one component with cost 5.

The AR problem has some characteristics of the Capacitated Facility Location (CFL) problem and network design problem. The instance of AR is the same as CFL with uniform capacities. However, in CFL one has to open a number of facilities and assign each client/city to an open facility (by a direct edge) so that each facility is assigned at most k clients and we minimise the total opening cost and connection cost. The main difference is that in CFL each cluster forms a star (with the facility being the centre) while in AR each cluster is a tree, whose cost might be much cheaper than the star. In AR, the clients might share the same path to be connected to the facility and hence reduce the total cost of forming the railroad network. AR has also similarities to the Capacitated Vehicle Routing Problem (CVRP) and Capacitated Minimum Spanning Tree (CMST). In CMST, the goal is to construct a minimum-cost collection of trees covering all the input vertices, each tree spanning at most k vertices, connected to a single root node. As discussed in [?], AR can be modelled as CMST in general weighted (non-metric) graphs.

The following variants of AR have been studied [?, ?]. For some constant $\beta > 1$, AR_β refers to the bicriteria version of AR, where airport capacity is allowed to be violated by a factor of β (also known as resource augmentation). AR^∞ is a relaxed version where the airport capacity is dropped, or equivalently, set to infinity: $k = +\infty$. When airport opening costs are uniform we refer to it by $\mathbb{1}\mathrm{AR}$. Another special case is AR_P where each component is a path with both endpoints having an airport. AR_P is a relaxation of the capacitated vehicle routing problem (CVRP) since not all the paths need to have a common endpoint (the centralised dépôt in CVRP). The original problem is sometimes denoted as AR_F (or simply AR) where we have a general forest.

1.1 Related Work

As mentioned above, [?, ?] have studied AR and some variants of it defined above. No true (non-trivial) approximation is known for AR in general setting. For the case of uniform airport opening cost, for both 1AR and $1\mathrm{AR}_P$, [?] show that the problems are NP-hard in Euclidean metrics and present PTAS's for them.

In [?] the authors consider bicriteria approximations. They present a $\frac{4}{3}\cdot(2+\frac{1}{p})$-approximate for AR_{1+p}, $p \in (0, 1]$ for general metrics. For Euclidean $\mathbb{R}^2$ they present a QPTAS for $\mathrm{AR}_{1+\mu}$, for arbitrary $\mu > 0$ (i.e. violating the capacities by $1 + \mu$) and a $(2 + \frac{k}{k-1} + \varepsilon)$-approximation in polynomial time. To obtain the latter result they obtain a PTAS for AR' on Euclidean metrics and show that a solution to AR' implies a solution for AR at a loss of factor $2 + \frac{k}{k-1}$.

In CFL, we are given a weighted (metric) graph $G = (V, E)$, a facility opening cost function $f : V \to \mathbb{R}_{\geq 0}$, and edge costs $c : E \to \mathbb{R}_{\geq 0}$, and a capacity u_v. The goal is to open a set of facilities $F \subseteq V$, and assign each point $v \in V$ to an open facility so that each open facility v has at most u_v points assigned to it while minimizing the total opening costs plus the assignment costs of points to open facilities. The only difference between CFL and AR is that in CFL the assignment edges in each cluster form a star whereas in AR it forms a minimum tree spanning the nodes of that cluster. There are constant approximation algorithms for CFL in general as well as uniform settings [?, ?].

For CVRP and its variants there are constant-factor approximations in general settings and QPTAS for special metrics such as Euclidean and doubling metrics and minor-free graphs [?, ?, ?, ?]. Another related problem is the *capacitated cycle cover problem* (CCCP) studied in [?]. In this problem, we are given a weighted graph G and parameters k and γ. The goal is to find a spanning collection of cycles of size at most k while minimizing the cost of the edges of the cycles plus γ times the number of cycles. This problem is related to Min-Max Tree Cover and Bounded Tree Cover studied earlier [?, ?]. In [?] the authors present a $(2 + \frac{2}{7})$-approximation for CCCP. This also implies a $(4 + \frac{4}{7})$-approximation for uniform AR.

For CMST, Jothi and Raghavachari [?] give a 3.15-approximation algorithm for Euclidean CMST and a $(2 + \gamma)$-approximation for metric CMST, where $\gamma \leq 2$ is the ratio of minimum-cost Steiner tree and minimum spanning tree. As pointed out by [?], AR can be reduced to CMST in non-metric setting.

We refer to [?] for discussion of other related works such as capacitated-cable facility location problem (CCFLP) [?] and sink clustering problem [?].

1.2 Contributions

Although AR (and $\widetilde{\mathrm{AR}}$) are similar to both CFL and CVRP, the mix of capacitated facility location and network design components appears to make it significantly more difficult than both. The approximability of AR for general metrics remains uncertain. Even for more restricted settings such as special metrics (e.g. trees) or uniform opening costs, the approximability of the problem is open.

In this paper, we make progress on some special cases. First, we consider AR with uniform opening cost (i.e. $\mathbb{1}$AR) on various metrics. For general metrics, we present a simple 2-approximation algorithm for this.

▶ **Theorem 1.** *There is a* 2*-approximation for uniform* AR *on general metrics.*

We also consider graphs of bounded treewidth and present a QPTAS for $\widetilde{\mathrm{AR}}$ on such metrics.

▶ **Theorem 2.** *There is a* QPTAS *for uniform* $\widetilde{\mathrm{AR}}$ *on graphs of bounded treewidth which runs in time* $n^{O(\omega^{\omega}\cdot\log^3 n/(\varepsilon^2\log^{\omega}\omega))}$*. where* ω *is the treewidth of the input graph.*

Next, we consider AR$'$ in the general setting (i.e. with non-uniform facility opening costs). We propose an exact algorithm for trees and graphs of bounded treewidth.

▶ **Theorem 3.** AR$'$ *can be solved in polynomial time on graphs with bounded treewidth.*

Using embedding results for general metrics into tree metrics with $O(\log n)$ distortion as well as embedding of graphs of bounded doubling dimension, graphs of bounded highway dimension, and minor-free graphs into graphs with polylogarithmic treewidth as well as $O(1)$-reduction from AR to AR$'$ ([?]) we obtain the following corollary.

▶ **Corollary 4.** *There is a polynomial time* $O(\log n)$*-approximation for* AR *on general graphs, a* QPTAS *for* AR$'$ *and therefore a quasi-polynomial* $O(1)$*-approximation for* AR *for graphs with bounded doubling dimension, graphs of bounded highway dimension, and minor-free graphs.*

We also show that at a factor 2 loss, we can reduce the general AR problem to the case that facilities have cost 0 or $+\infty$, we denote this case by $0/+\infty$ AR. In other words, the special case of the problem that all facilities (to be opened) are given to us and we simply have to build clusters of size at most k each of which has one of the open facilities. Even for this special case, a good approximation remains elusive.

▶ **Theorem 5.** *Given an instance* G *for* AR*, we can build an instance* G' *for* $0/+\infty$ AR *such that any* α*-approximate solution to* $0/+\infty$ AR *implies a* 2α*-approximate solution for* AR *on* G*.*

In the next section, we prove Theorem 1. Then in Section 3 we prove Theorem 2 and in Section 4 we prove Theorem 3 and Corollary 4. We defer the proof of Theorem 5 to the full version.

2 Algorithm for Uniform AR in General Metric

In this section, we prove Theorem 1. Since each facility (airport) is trivially serving its own city, we refer to the remaining capacity $k-1$ (to serve other clients) as k for simplicity. We assume opening a facility at each vertex costs a uniform value f. Given an instance G we first define a modified instance $\tilde{G}$ for each input graph G. The graph $\tilde{G}$ is obtained by adding a dummy node r to G and connecting r to all the vertices $v \in V$ with an edge of cost $c_{vr} = f$. We first define the MST_r^σ problem and prove the following lower bound.

▶ **Definition 6.** *In the MST_r^σ problem, we are given a graph $G = (V, E)$ with a vertex $r \in V$. The task is to find the minimal cost of the spanning tree of the input graph, while ensuring that the degree of vertex r in the solution is at most σ.*

▶ **Lemma 7.** *If σ is the number of components in an optimum solution to AR on G then the cost of an optimal solution to the MST_r^σ problem on $\tilde{G}$ is a lower bound on the optimal solution to AR on G.*

Proof. Consider an optimal solution ξ to AR on G. Say there are σ components in ξ. After adding into ξ a dummy node r and connecting r to the vertices that are open facilities with an edge of cost f, we obtain a spanning tree T for $\tilde{G}$ of the same cost, where the vertex r has a degree of σ. Namely, this is a feasible solution to MST_r^σ. Therefore, an optimal solution to MST_r^σ on $\tilde{G}$ cannot cost more than the optimal solution to AR on G. ◀

Our algorithm first guesses the number of components in the optimal solution. We do this by enumerating all possibilities. Say there are σ components in the optimal solution for some integer $\sigma \leq n$. Note that we know $\sigma \geq \lceil \frac{n}{k} \rceil$ for certain, as otherwise there must exist some cities that are not getting served. Our algorithm is as follows.

Construct the instance $\tilde{G}$. Solve the MST_r^σ problem on instance $\tilde{G}$. After removing the dummy vertex r, we obtain a set $\mathcal{T} = \{T_1, T_2, \dots T_\sigma\}$ of σ connected components (i.e. trees). Note that we can solve the MST_r^σ problem using the technique of matroid intersection [**?**].

Let $M_1 = (\tilde{E}, \mathcal{I}_1)$ represent the graphic matroid of $\tilde{G}$ (also known as the cycle matroid or polygon matroid), where the ground set $\tilde{E}$ is the set of edges in $\tilde{G}$, and the set of independent sets $\mathcal{I}_1$ consists of acyclic subgraphs of $\tilde{G}$. That is to say, each independent set corresponds to the edges of a forest in the underlying graph $\tilde{G}$. Let $M_2 = (\tilde{E}, \mathcal{I}_2)$ denote the partition matroid, where the set of independent sets $\mathcal{I}_2$ is defined as follows, where $N(r)$ represents all the edges incident to the vertex r and $\tilde{V}$ is the vertex set of $\tilde{G}$,

$$\mathcal{I}_2 = \left\{ S \subseteq \tilde{E} \;\middle|\; |S \cap N(r)| \leq \sigma,\; |S \cap (\tilde{E} \setminus N(r))| \leq |\tilde{V}| - 1 - \sigma \right\}.$$

In other words, each independent set of this partitional matroid corresponds to the edge set of a subgraph of $\tilde{G}$ with at most $|\tilde{V}| - 1$ edges, where there are at most σ edges incident to the vertex r and at most $|\tilde{V}| - 1 - \sigma$ edges not incident to r.

Note that a feasible solution to MST_r^σ is an independent set of both matroids. The underlying graph must form a spanning tree, so it is an independent set of M_1. The set of edges must satisfy the degree requirement for vertex r, so it is an independent set of M_2. For each connected component $T_i \in \mathcal{T}$, we obtain a cycle C_i in the following way: double the edges of T_i and trace them while short-cutting whenever we encounter a vertex that has been visited. We cut each cycle C_i into a set of disjoint subpaths of fixed length k, except for at most one subpath per cycle that is strictly shorter than k. Essentially, we have transformed the trees in $\mathcal{T}$ into a set of paths. Let $\mathcal{P}_k$ denote the set of paths with length exactly k. For each path in $\mathcal{P}_k$, we simply open one of its cities as an airport. Note that

$|\mathcal{P}_k| \leq \lfloor \frac{n}{k} \rfloor$ since there are at most n vertices (other than the vertex r) in the graph. In addition, as we know $\sigma \geq \lceil \frac{n}{k} \rceil$, we have $|\mathcal{P}_k| \leq \lfloor \frac{n}{k} \rfloor \leq \lceil \frac{n}{k} \rceil \leq \sigma$. Consequently, the cost of opening these $|\mathcal{P}_k|$ airports is $|\mathcal{P}_k| \cdot f \leq \sigma \cdot f$. For those subpaths of length less than k, we simply open one of its vertices as the facility. Note that since there are $|\mathcal{T}| = \sigma$ trees T_i (hence there are σ corresponding cycles C_i), we have at most σ such short subpaths. The current cost is bounded by twice the edge cost of all the trees in $\mathcal{T}$, as well as the facility cost of all these subpaths, which is at most $f \cdot \sigma + |\mathcal{P}_k| \cdot f \leq 2\sigma \cdot f$. Meanwhile, the cost of the MST_r^σ solution is the edge cost of all the trees in $\mathcal{T}$, plus the cost of incident edges of r in the solution, which is $f \cdot \sigma$. Thus, it is obvious that the cost is no more than twice the cost of the MST_r^σ solution.

From the analysis above, it should be easy to see that Theorem 1 follows.

3 QPTAS for Uniform Case in Graphs of Bounded Treewidth

In this section, our goal is to prove Theorem 2. First, recall the definition of graphs with bounded treewidth.

▶ **Definition 8.** *A* tree decomposition *of a graph $G = (V, E)$ is a tree $T = (V', E')$ and a mapping $\Xi : V' \to 2^V$ where each vertex $\beta \in V'$ (also known as a bag) corresponds to a set of vertices of G, such that*
- *For each vertex v in G, it must be included in at least one bag of T.*
- *For each edge uv in G, the pair of vertices $u, v \in V$ must be included in at least one bag of T.*
- *For each vertex v in G, consider the set of all the bags in T that include v. These bags induce a connected component in T.*

The width of a tree decomposition is defined as the cardinality of its largest bag minus one. The treewidth of a graph G is the smallest w such that G has a tree decomposition with width w. Given a graph $G = (V, E)$ of treewidth ω, there is a tree decomposition $T = (V', E')$ of G where T is binary, with depth $h \in O(\log n)$ (where $n = |V|$) and treewidth not exceeding $\omega' = 3\omega + 2$, according to [?]. For simplicity, denote ω' as ω instead. We assume the tree height $h = \delta \log n$ for some constant $\delta > 0$.

Our algorithm for uniform $\widetilde{\mathrm{AR}}$ on bounded treewidth graph relies on the technique developed in [?] for designing QPTAS for CVRP on such metrics. First, we ignore the concept of facilities/airports, we simply pay an extra f for each cluster in our solution (later we designate one vertex in each cluster as the facility to be opened). For that, we define a new version of the problem which we call UAR (meaning AR with *undetermined* airports).

▶ **Definition 9.** (UAR) *The goal is to find a set $\mathcal{F}$ of (not necessarily disjoint) clusters (i.e. trees) using edges in the graph. The size of each cluster must not exceed the capacity constraint k. Each cluster $\gamma \in \mathcal{F}$ has a cost of f and we want to minimise the total cost, which is defined as*

$$|\mathcal{F}| \cdot f + \sum_{\gamma \in \mathcal{F}} \mathrm{cost}(\gamma)$$

where $\mathrm{cost}(\gamma)$ *denotes the railway cost of the cluster γ.*

Since this is a relaxed version of the original problem (as we do not specify the location of the facilities), its cost is a lower bound of that of the original problem. We can think of each vertex in V to have one unit of demand which needs to be sent to an airport to be served. We may add dummy demands to a vertex during the algorithm, so a vertex may end up having more than one unit of demand. The size of a cluster is defined to be the sum of demands on all its vertices, instead of just the number of vertices. Note that a component may not include every vertex that it passes through, as a component may be simply using the edges of a vertex to get to somewhere else, which can also be seen as not picking up the demand of the vertex. Be mindful that, from the perspective of demands, the size of a component is the number of demands it includes, instead of the number of vertices. Therefore the clusters in the solution are not necessarily edge-disjoint or vertex-disjoint, but the total number of demands in each cluster obeys the capacity constraint.

For clarity, we refer to the vertices in T as *bags*, to differentiate them from the vertices in G. For the notation β, we refer to it as the name of the bag $\beta \in V(T)$ as well as the corresponding set of vertices $\beta \subseteq V(G)$. For each bag β, denote the union of vertices in all of the bags in the subtree T_β as C_β. Note that C_β also denotes the set of all bags in T_β.

Each vertex of G may appear in multiple bags of T as tree decomposition generates duplicates. In order to make sure the demand of a vertex does not get duplicated in T, for every vertex $v \in V(G)$, we assume that the copy/instance of v in the bag $\tilde{\beta}$ that is the closest to the root bag (we know there is a unique one and we denote this copy of v as $\tilde{v}$) has a demand of one, and the rest of the copies of v (which resides in other bags) have demand zero.

Given an optimal solution denoted as OPT, we will demonstrate a process for transforming it into a near-optimal solution for UAR and thereby show the existence of such a near-optimal solution. This transformation occurs incrementally on T, moving from the bottom to the top, one level at a time. The solution before modifying level ℓ is denoted as OPT_ℓ, and after the modification as $\mathrm{OPT}_{\ell-1}$.

Overview of the approach and relation to [?]: Our goal is to show the existence of a near-optimum solution with certain structures. Suppose OPT is an optimum solution for UAR and opt is its value. We aim to find a near-optimal solution, of cost $(1 + O(\varepsilon))\mathit{opt}$, where each vertex has at least one unit of demand, and the size of partial clusters in any subtree T_β can only be one of *polylogarithmically* many values. Two concepts are required to describe the following data structures, namely, the notions of partial and complete clusters. We consider a non-root bag $\beta \in V(T)$ and the subtree rooted at β, T_β. A complete cluster in T_β is a cluster that is entirely in the graph C_β, and a partial cluster is one that uses vertices both inside C_β and outside. Similar to [?], we first assume that the number of clusters in OPT is sufficiently large, that is, at least $\lambda \log n$ for some large number λ. Otherwise, if the number of clusters in OPT is upper-bounded by $\Sigma = \lambda \log n$ then a simple DP can solve the problem exactly (see [?]). Given an optimal solution OPT, we will demonstrate a process for transforming it into a near-optimal solution with certain structural properties that help us find one using dynamic programming. This transformation occurs incrementally on T, moving from the bottom to the top, one level at a time. The solution before modifying level ℓ is denoted as OPT_ℓ, and after the modification as $\mathrm{OPT}_{\ell-1}$. Looking at how OPT_ℓ looks like, we would like to "approximately" keep the sizes of partial clusters that extend below β in T_β. A standard approach is to "bucket" the sizes of partial clusters into buckets where each bucket contains all those sizes that are within $(1 + \varepsilon)$ of each other (e.g. bucket i being values in $(1 + \varepsilon)^i \ldots [(1 + \varepsilon)^{i+1} - 1]$. This will reduce the complexity of the DP table to quasi-polynomial: we keep the number of partial clusters of each bucket and try to fill in the DP table bottom-up. The problem is that then when we are combining solutions in the DP table, since we are keeping the sizes approximately (and sacrificing precision), we may violate the capacities unknowingly. The idea developed in [?] was to modify OPT by reducing the sizes of the clusters (at a small increase in the number of clusters) so that even if we scale the sizes of the new clusters by a small number, they are still capacity-respecting. They used a technique that was used later in [?], called *adaptive rounding* that we also use here to round the sizes of partial clusters in T_β for any bag $\beta \in T$. At each bag β, for clusters that are in the same "bucket" we swap parts of them with a net effect of reducing their sizes while having only a poly-logarithmic many possible bucket sizes at the end. We formalize this in the following.

▶ **Definition 10.** *Define the* ***threshold values*** $\{\sigma_1, \dots, \sigma_\tau\}$ *where*

$$\sigma_i = \begin{cases} i & 1 \le i \le \lceil 1/\varepsilon \rceil \\ \lceil \sigma_{i-1} \cdot (1+\varepsilon) \rceil & i > \lceil 1/\varepsilon \rceil \end{cases}$$

in such a way that the last threshold $\sigma_\tau = k$*. So* $\tau \in O(\log k/\varepsilon)$.

We adapted the definitions from [?]. Consider a bag β that is situated at level ℓ. We consider partial clusters that cross β and based on their size in C_β we bucket them. Bucket i contains those partial clusters whose size is in the range $[\sigma_i, \sigma_{i+1})$. Now let's focus on all (partial) clusters that are in bucket i of bag β. Each of these clusters has some vertices in C_β and some vertices outside. For a set $S \subset \beta$ consider all the partial clusters in bucket i that their intersection with β is S. So each of them will form a number of connected components in C_β where each component contains some part of S; this defines a partition of S. We consider all those partial clusters that have the same partition of S together (defined below).

▶ **Definition 11.** *For a bag β at level ℓ in T, for each set $S \subseteq \beta$ and partition $\wp_S$ of S, consider the set $b_S^{\wp_S}$ which contains the clusters that use exactly the set of vertices $S \subseteq \beta$ to span into C_β, where $\wp_S$ denotes a partition of the set S based on connectivity of the of those clusters in C_β. Define the i-th bucket of $b_S^{\wp_S}$, denoted as b_i, to store clusters in OPT_ℓ that have a size between $[\sigma_i, \sigma_{i+1})$ inside C_β, where σ_i is the i-th threshold value. Denote this bucket by a tuple $(\beta, b_i, S, \wp_S)$. Denote the number of clusters in bucket $(\beta, b_i, S, \wp_S)$ as $n_{\beta,i}^{S,\wp_S}$.*

Essentially, the set S represents the interface that the clusters in the bucket $(\beta, b_i, S, \wp_S)$ use to attach to the rest of their parts in C_β, and $\wp_S$ is a set that describes the connectivity between the vertices of S in C_β. That is, each part in the partition $\wp_S$ specifies a subset of vertices of S that need to be connected below. So if $u, v \in S$ and there is some set $P \in \wp_S$ such that $P \supseteq \{u, v\}$, then u and v need to be connected in C_β by some cluster. For simplicity, we just write $\wp_S$ as $\wp$.

▶ **Definition 12.** *A bucket b is said to be* `small` *if it contains no more than $\alpha \log^2 n/\varepsilon$ clusters and is otherwise said to be* `big`*, for some constant $\alpha \ge \max\{1, 20\delta\}$.*

▶ **Definition 13.** *For a big bucket $(\beta, b_i, S, \wp)$, define g groups where $g = \frac{2\delta \log n}{\varepsilon}$, denoted as $G_{i,1}^{\beta,S,\wp}, G_{i,2}^{\beta,S,\wp}, \dots, G_{i,g}^{\beta,S,\wp}$ in the following way (for simplicity assume the size of this bucket is a multiple of g, if not add some empty clusters to achieve this). Sort the clusters in the (padded) bucket in non-decreasing order, and put the first $\frac{n_{\beta,i}^{S,\wp}}{g}$ clusters into $G_{i,1}^{\beta,S,\wp}$, the second $\frac{n_{\beta,i}^{S,\wp}}{g}$ into $G_{i,2}^{\beta,S,\wp}$, etc. For each group $G_{i,j}^{\beta,S,\wp}$, denote the size of its smallest cluster as $h_{i,j}^{\beta,S,\wp,\min}$ and the size of its biggest cluster as $h_{i,j}^{\beta,S,\wp,\max}$.*

Suppose we are considering a big bucket of β and a partial cluster Γ is in the group $j > 1$ of the big bucket. We find its top (that is, the part of the cluster that is outside of T_β) and reassign it to another partial cluster (that is no bigger than Γ) with the same order in the previous group (i.e., group $j-1$) as the order of Γ in group j. The vertices that were originally covered by the partial clusters in the last group are referred to as *orphans*. This is essentially the rounding between groups of a big bucket that was done in [?] for the CVRP on bounded treewidth graphs. The idea is that by this operation, the size of each cluster goes down enough such that if we "approximate" the sizes by the size of the biggest cluster in each group, we are still satisfying the capacity constraints. However, some vertices that were covered by the partial clusters of the last group are now left "uncovered" (or orphan). We will use some extra clusters to pick up (serve) the now orphan vertices.

We come up with a structure theorem that shows the existence of a near-optimal solution with certain structures, and then provide a dynamic programming algorithm for the UAR problem.

3.1 Structure Theorem for Graphs with Bounded Treewidth

The steps of modifying OPT to a near-optimal solution (denoted as OPT′) are largely the same as the ones in [?]. Let's assume we randomly choose clusters from OPT, denoted as C, with a probability of ε. After selecting these clusters, we duplicate each chosen one and assign both duplicates of each chosen cluster to one of the levels ℓ that it visits[1], with equal probability. These duplicated clusters are referred to as the *extra clusters*. We will bound their total cost. The proof is very similar to the one in [?] and we only need to show the part concerning the facility costs.

Recall f is the (uniform) facility opening cost, ε is the probability each cluster γ in OPT is selected as the extra cluster, k is the capacity of each cluster, and ω is the treewidth of G.

▶ **Lemma 14.** *The expected cost of the extra clusters sampled is* $2\varepsilon \cdot \text{OPT}$.

We make use of the following modified definitions and lemmata from [?]. They apply to our problem as the proofs of the lemmata are almost identical.

Denote the bags in level ℓ of T as B_ℓ. Define the set X_ℓ to comprise the extra clusters assigned to bags at level ℓ. For every bag $\beta \in B_\ell$ and its bucket $(\beta, b_i, S, \wp)$, let $X_{\beta,i}^{S,\wp}$ represent the extra clusters (using vertices in S to span into C_β, with $\wp$ depicting connectivity downwards) in X_ℓ whose partial clusters inside C_β has a size that falls within the range defined by bucket b_i. For an extra cluster $\gamma \in X_{\beta,i}^{S,\wp}$, it covers some partial cluster $\zeta \in G_{i,g}^{\beta,S,\wp}$ (which is without its top). That is, the extra cluster γ only picks up demands at the levels $\geq \ell$ and acts as the top of ζ, in particular, this combined cluster picks up only those demands of ζ's vertices (which are all orphans).

▶ **Lemma 15.** *At any level ℓ, each bag $\beta \in B_\ell$ and its big buckets $(\beta, b_i, S, \wp)$ satisfy, w.h.p.*

$$\left|X_{\beta,i}^{S,\wp}\right| \geq \frac{\varepsilon^2}{\delta \log n} \cdot n_{\beta,i}^{S,\wp}.$$

▶ **Lemma 16.** *For all bags β at level ℓ in T, their big buckets $(\beta, b_i, S, \wp)$ and partial clusters in $G_{i,g}^{\beta,S,\wp} \subseteq b_i$, we can make adjustments to the extra clusters present in $X_{\beta,i}^{S,\wp}$ without incurring any additional cost, and introduce some dummy demands within β when necessary, so that:*

1. *The partial clusters in $G_{i,g}^{\beta,S,\wp}$ are now incorporated into some clusters in $X_{\beta,i}^{S,\wp}$. (That is, all the demands that were covered by some partial cluster in $G_{i,g}^{\beta,S,\wp}$ are picked up by some cluster in $X_{\beta,i}^{S,\wp}$.)*

[1] If a cluster γ passes crosses bag of level ℓ, we say γ visits or crosses level ℓ.

2. *The modified partial clusters that cover the orphans (i.e., vertices in $G_{i,g}^{\beta,S,\wp}$) have precisely the size of $h_{i,g}^{\beta,S,\wp,\max}$ and all clusters remain underneath the size limit of k units of demand.*
3. *For each modified partial cluster in the set $X_{\beta,i}^{S,\wp}$, its partial clusters at a bag $\beta' \in B_{\ell'}$ is also of one of $O(\log k \log^2 n/\varepsilon^2)$ many sizes, where ℓ' is any lower levels $> \ell$.*

Note that when we add dummy demands for some cluster γ in some bucket $(\beta, b_i, S, \wp)$, we simply add these dummy demands onto the vertices in S. Using these lemmata and a very similar proof to the one in [?], we can obtain a Structure Theorem for our UAR problem in the case of graphs with bounded treewidth.

▶ **Theorem 17.** (Structure Theorem) *Consider an instance $\mathcal{I}$ for the* UAR *problem. Denote its optimal solution as* OPT*, with cost* OPT*. We can transform* OPT *to another solution* OPT′ *so that, with high probability,* OPT′ *is a near-optimal solution of cost at most $(1+2\varepsilon)$*OPT*. Additionally, at every β in* OPT′*, all the clusters in C_β have one of $O(\log k \log^2 n/\varepsilon^2)$ possible sizes. Consider a bucket $(\beta, b_i, S, \wp)$ in* OPT′*. We must have*

- *If b_i is small, the number of partial clusters in C_β whose size falls within b_i is at most $\alpha \log^2 n/\varepsilon$.*
- *If b_i is big, it has exactly $g = 2\delta \log n/\varepsilon$ group sizes which are denoted as*

$$\sigma_i \le h_{i,1}^{\beta,S,\wp,\max} \le h_{i,2}^{\beta,S,\wp,\max} \le \cdots \le h_{i,g}^{\beta,S,\wp,\max} < \sigma_{i+1}$$

Each cluster in b_i has a size of one of the h-values above.

Having this structure theorem one can design a (relatively complex) DP to compute a near-optimum solution as guaranteed by this structure theorem. This DP builds upon ideas of the DP in [?] but has more complexity as the clusters here do not necessarily have a common point (like the dépôt in the CVRP problem). This will show that we can compute a solution such as OPT′ in Theorem 17 in time $n^{O(\omega^\omega \cdot \log^3 n/(\varepsilon^2 \log^\omega \omega))}$.

We can transform the approximate solution obtained for the UAR problem into a solution to the $\widetilde{\text{AR}}$ problem, without any increase in the cost. All we need to do is to pick a node in each cluster to open a facility at (since we are already paying f for each cluster, this cost is accounted for in the solution to UAR). This can be easily done since in a solution to UAR each vertex is "covered" by a unique cluster.

4 Constant Approximation for Nonuniform-AR

In this section, we prove Theorem 3. For ease of exposition, we present the proof for the case of trees (the extension to graphs with bounded treewidth appears in the full version). Recall that in the relaxation AR′, we are given a graph $G = (V, E)$ where each vertex $v \in V$ has a non-negative opening cost a_v and each edge $e \in E$ has a non-negative weight c_e. Every edge and vertex has capacity $k \in \mathbb{N}_+$. Find a subset of vertices $\Phi \subseteq V$ as facilities (also known as airports), and a multiset Ξ of edges from E to get a transportation network that ensures one unit of flow from each vertex in V can be sent to facilities in Φ, without violating the capacity constraint on any edge or facility. The goal is to find such a network while minimising the total cost $\sum_{v\in\Phi} a_v + \sum_{e\in\Xi} c_e$. First, we prove some properties in an optimum solution to AR′.

▶ **Lemma 18.** *In an optimum solution, we can assume there are not any flows of opposite directions on the same edge, as we can uncross them by redirecting each flow and attain a lower cost.*

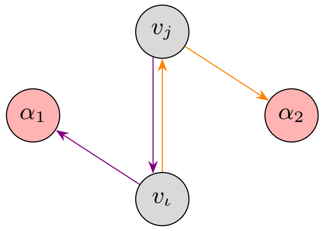


Figure 2 A simplest example of crossing flows in AR′. The red vertices are open facilities.

Note that it is allowed for multiple clients to use the same edge to send their demands in the same direction.

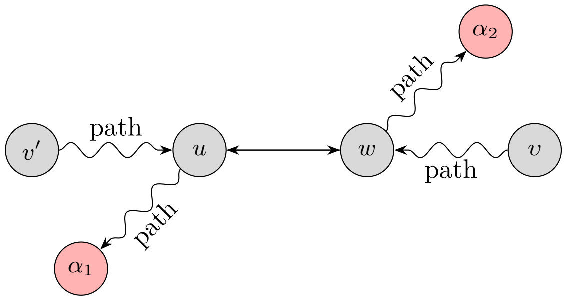


Figure 3 The crossing flow is at the edge uw

Proof. Without loss of generality, assume the vertices v' and v caused crossing flow at edge uw. That is, the demand of v' travels from v' to u, crosses the edge uw from u to w, and from w to a facility α_2; and the demand of v travels from v to w, crosses the edge uw from w to u, and from u to a facility α_1. We can reroute so that the demand of v' travels from v' to u, and then from u to the facility α_1; and similarly, the demand of v travels from v to w, and then from w to the facility α_2. It is easy to see such a rerouting makes the total cost decrease, for the demands of both vertices v' and v now take a shorter path to be served. ◀

Consider a tree T as the input graph. A subproblem here is defined on the subtree T_v for each vertex v. Since we aim to obtain a flow network in T, each vertex v, as the root of the subtree T_v, will be considered a portal in the corresponding subproblem. There is thus a DP cell for each vertex v in T. Note that at each vertex v, the portal configuration ψ_v simplifies to the direction and value of the flow at v

$$\psi_v = \pm f_v$$

where we use $-$ (minus sign) to signify the flow is leaving T_v, and $+$ (plus sign) to signify the flow is entering T_v. f_v is the absolute value of the signed integer ψ_v and denotes the value of the unidirectional (integral) flow passing through the vertex v and satisfies $0 \leq f_v \leq n$, where n is the number of vertices in T. Note that in AR′, if an edge needs to carry a flow f_v, then we need to install $\left\lceil \frac{f_v}{k} \right\rceil$ parallel edges in the solution. At each vertex v, we also consider both of the scenarios where v is an airport or it is not. We use a Boolean variable $\pi_v = \textsc{True}$ (or $\pi_v = 1$) to indicate that the portal v is opened as an airport.

We define the DP table $\mathbf{D}$ as follows, for each v in T, let the entry $\mathbf{D}[v, \pi_v, \psi_v]$ store the cost of the optimal solution to AR′ on T_v with the amount of flow going in/out of T_v conforming to ψ_v, with portal v opened as an airport if and only if π_v.

At each node, we also consider its parent edge and see it as part of the subtree T_v. For the root node ϑ, we assume its parent edge has cost 0. The result will be $\min_{\pi_\vartheta}\{\mathbf{D}[\vartheta, \pi_\vartheta, \psi_\vartheta = 0]\}$ as there will be no flow entering or leaving T at the root.

Base cases: At a leaf node v, denote the parent edge of v as e. Recall $f_v = |\psi_v|$.

$$\mathbf{D}[v, \pi_v, \psi_v] = a_v \cdot \pi_v + \begin{cases} c_e & \text{if } \psi_v = -1 \\ c_e \cdot \left\lceil \dfrac{f_v}{k} \right\rceil & \text{if } 0 \le \psi_v < +k \text{ and } \pi_v = 1 \\ +\infty & \text{otherwise} \end{cases}$$

Here $\psi_v = -1$ means there is one unit of flow going out of the leaf v (actually does not need to open a facility at v). If $0 \le \psi_v < +k$, it means v does not emit any flow or it is absorbing flows, then we have to make sure $\pi_v = \text{True}$. Note that in this case, $\left\lceil \frac{f_v}{k} \right\rceil = 1$ when $0 < \psi_v < +k$, and $\left\lceil \frac{f_v}{k} \right\rceil = 0$ when $\psi_v = 0$. If $\psi_v \ge +k$ then we know it is not achievable, since a facility has capacity k and cannot absorb more flows. If $\psi_v < -1$ then it is simply impossible, as a vertex only has one unit of demand and cannot emit more than that. For these cases, we set the entry to $+\infty$.

For a node v with z children $w_1, w_2, \dots, w_z$, similar to the case of uniform facility cost on trees in the previous chapter, we define an inner DP table $\mathbf{B}$. Assume we have computed $\mathbf{D}[w_j, \pi_{w_j}, \psi_{w_j}]$ for all possible π_{w_j} and ψ_{w_j}, for all $1 \le j \le z$. Let $\mathbf{B}[v, \pi_v^j, \psi_v^j, j]$ store the cost of the optimal solution to AR$'$ on T_v as if the portal v only has children $w_1, w_2, \dots, w_j$. Lastly, we define $\mathbf{D}[v, \pi_v, \psi_v] = \mathbf{B}[v, \pi_v, \psi_v, z]$.

Case 1: $j = 1$. Only consider the first child of v.

$$\mathbf{B}[v, \pi_v^1, \psi_v^1, 1] = \min_{\psi_{w_1}} \left\{ \mathbf{D}[w_1, \pi_{w_1}, \psi_{w_1}] + a_v \cdot \pi_v^1 + c_e \cdot \left\lceil \frac{f_v^1}{k} \right\rceil \;\middle|\; \eta(\pi_v^1, \psi_v^1, \psi_{w_1}) = \text{True} \right\}$$

where $\eta(\pi_v^1, \psi_v^1, \psi_{w_1})$ is a Boolean indicator function that takes into account the flow on v's parent edge and the edge vw_1, as well as the decision about whether or not to open the portal v as an airport. It is true if and only if all these parameters are compatible. Recall that f_v is the absolute value of ψ_v.

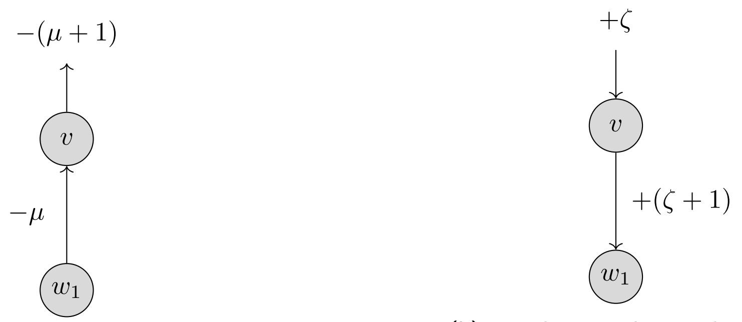
$$\eta(\pi_v^1, \psi_v^1, \psi_{w_1}) = \begin{cases} \text{True} & \text{if } 0 \le \psi_v^1 - \psi_{w_1} < k \;\wedge\; \pi_v^1 = \text{True}, \\ & \text{or if } \psi_{w_1} - \psi_v^1 = 1 \\ \text{False} & \text{otherwise} \end{cases}$$

The case $\psi_{w_1} - \psi_v^1 = 1$ means that v does not act like an airport as it is not absorbing any flow, and is sending its own demand elsewhere (hence unnecessary to open an airport there).

The case $0 \le \psi_v^1 - \psi_{w_1} < k$ means the portal v is absorbing flows and v must be opened as an airport. The other cases are impossible, either because v is absorbing too much flow which violates its capacity limit, or because v is sending out more than one unit of flow.

Case 2: For $2 \le j \le z$. Assume all entries of the form $\mathbf{B}[v, \pi_v^{j-1}, \psi_v^{j-1}, j-1]$ have been computed. We define

$$\mathbf{B}[v, \pi_v^j, \psi_v^j, j] = \min_{\substack{\pi_{w_j}, \pi_v^{j-1}, \psi_{w_j}, \psi_v^{j-1}: \\ \pi_v^j \ge \pi_v^{j-1}, \\ \eta\left(\pi_v^j, \psi_v^j, \psi_v^{j-1}, \psi_{w_j}\right) = \text{True}}} \quad (\Omega)$$



(a) Portal v is sending its demand outside T_v

(b) Portal v is sending its demand into T_{w_1}

Figure 4 Here μ and ζ are non-negative integers. The label on edge vw_1 represents ψ_{w_1} and the label above v stands for ψ_v^1.

The expression Ω should be

$$\left\{\mathbf{D}[w_j, \pi_{w_j}, \psi_{w_j}] + \mathbf{B}[v, \pi_v^{j-1}, \psi_v^{j-1}, j-1] + a_v \cdot \left(\pi_v^j - \pi_v^{j-1}\right) + c_e \cdot \left\lceil \frac{f_v^j - f_v^{j-1}}{k} \right\rceil \right\}$$

where we define the indicator function η as follows:

$$\eta(\pi_v^j, \psi_v^j, \psi_v^{j-1}, \psi_{w_j}) = \begin{cases} \text{TRUE} & \text{if } 0 < \psi_v^j - (\psi_v^{j-1} + \psi_{w_j}) \le k \ \wedge \ \pi_v^j = \text{TRUE}, \\ & \text{or if } \psi_v^{j-1} + \psi_{w_j} = \psi_v^j \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Let e denote v's parent edge. The case $\psi_v^{j-1} + \psi_{w_j} = \psi_v^j$ means that after taking w_j (the j-th child of v) into consideration, the flow on e whilst only considering the first $j-1$ children (which is ψ_v^{j-1}), and the flow on the edge vw_j adds up to the flow on e while considering all the j children (which is ψ_v^j). This means the portal v is not absorbing any of the flow from T_{w_j}, and thus there is no need to open it as an airport if it has not been opened. The case $0 < \psi_v^j - (\psi_v^{j-1} + \psi_{w_j}) \le k$ means after taking w_j into consideration, the portal v is absorbing flows and needs to be opened, if it has not been opened. Note that $\left\lceil \frac{f_v^j - f_v^{j-1}}{k} \right\rceil$ can be negative if $f_v^j < f_v^{j-1}$, which means the "load" on the parent edge of v has decreased and we pay less on the edge cost. This exact algorithm on trees suggests we have an $O(\log n)$-approximation algorithm for the general metric (using metric approximation, also known as embeddings by tree metrics).

4.1 Algorithm Efficiency

We will use a bottom-up approach, assuming that the relevant entries for subproblems have already been pre-computed. At any step, checking the value for the indicator function η takes $O(1)$ time. To compute $\mathbf{B}[v, \pi_v^j, \psi_v^j, j]$, we need to consider all possible ψ_{w_j} and ψ_v^{j-1}, which is in total $O(n^2)$ possibilities. Since there are n nodes in the tree, the time for computing the table $\mathbf{D}$ is in $O(n^4)$.

4.2 Generalisation for AR with Steiner Vertices

In this section, we describe how the algorithm above can be generalised for AR' with Steiner vertices with a few modifications. More generally, this algorithm can apply to the case where the set of facilities or the set of clients is not the same as the entire vertex set of the input graph. If a vertex v is not part of the set of facilities, it should not be opened as a facility (after all, no facility cost has been defined for it). So the Π-vector should not allow any copy of v to be opened. If a vertex v is not part of the set of clients, it carries no demand, and so does any of its copies in the tree decomposition.

Note that this will be useful when we try to embed a graph into a graph with bounded treewidth where the host graph of the input graph (via graph embedding) may have Steiner vertices. If Δ is the aspect ratio of G (ratio of largest to smallest edge cost) then by standard scaling (see for e.g. [?]) one can assume that Δ is bounded by polynomial in n at a loss of $(1+\epsilon)$ on optimum solution.

We use the following lemma by [?] about embedding graphs of doubling dimension D into a graph with treewidth $\omega \le 2^{O(D)} \left\lceil \left(\frac{4D\log\Delta}{\varepsilon}\right)^D \right\rceil$.

▶ **Lemma 19.** *(Theorem 9 in [?]) Let (X,d) be a metric with doubling dimension D and aspect ratio Δ. Given any $\varepsilon > 0$, the metric (X,d) can be $(1+\varepsilon)$ probabilistically approximated by a family of treewidth ω-metrics for*

$$\omega \le 2^{O(D)} \left\lceil \left(\frac{4D\log\Delta}{\varepsilon}\right)^D \right\rceil.$$

We adapt Theorem 8 and its proof from [?] to get the following result.

▶ **Theorem 20.** *For any $\varepsilon > 0$ and $D > 0$, given an input graph G of the AR' problem where G has doubling dimension D, there is an algorithm that finds a $(1+\varepsilon)$-approximate solution in time $n^{O\left(D^D \log^D n/\varepsilon^D\right)}$.*

We introduce the following lemma proposed by [?] about embedding graphs of highway dimension W into a graph with treewidth $\omega \in (\log\Delta)^{O\left(\log^2\left(\frac{W}{\varepsilon\lambda}\right)/\lambda\right)}$.

▶ **Lemma 21.** *(Theorem 1.3 in [?]) Let G be a graph with highway dimension W of violation $\lambda > 0$, and aspect ratio Δ. For any $\epsilon > 0$, there is a polynomial-time computable probabilistic embedding H of G with expected distortion $1+\varepsilon$ and treewidth ω where*

$$\omega \in (\log\Delta)^{O\left(\log^2\left(\frac{W}{\varepsilon\lambda}\right)/\lambda\right)}.$$

We adapt Theorem 9 and its proof from [?] to get the following result.

▶ **Theorem 22.** *For any $\varepsilon > 0$, $\lambda > 0$ and $W > 0$, given an input graph G of the AR' problem where G has highway dimension W and violation λ, there is an algorithm that finds a $(1+\varepsilon)$-approximate solution in time $n^{O\left(\log^{\log^2\left(\frac{W}{\varepsilon\lambda}\right)\cdot\frac{1}{\lambda}} n\right)}$.*

We introduce the following lemma proposed by [?] about embedding minor-free graphs (including planar graphs, which is a kind of K-minor-free graphs) into a graph with treewidth $O_K\left((\ell+\ln n)^6/\varepsilon\cdot\ln^2 n\cdot(\ln n+\ln\ell+\ln(1/\varepsilon))^5\right)$ where ℓ is the logarithm of the aspect ratio of the input graph.

▶ **Lemma 23.** *(Theorem 1.1 in [?]) For every fixed graph K, there exists a randomised polynomial-time algorithm that, given an edge-weighted K-minor-free graph $G=(V,E)$ and an accuracy parameter $\varepsilon > 0$, constructs a probabilistic metric embedding of G with expected distortion $(1+\varepsilon)$ into a graph of treedepth (the treedepth of a graph is an upper bound on its treewidth)*

$$O_K\left((\ell+\ln n)^6/\varepsilon\cdot\ln^2 n\cdot(\ln n+\ln\ell+\ln(1/\varepsilon))^5\right)$$

where $n = |V|$ *and* $\ell = \log \Delta$ *is the logarithm of the aspect ratio* Δ *of the metric induced by* G*.*

▶ **Theorem 24.** *For any* $\varepsilon > 0$*, given an input graph* G *of the* AR' *problem where* G *is a minor-free graph, there exists an algorithm that finds a* $(1+\varepsilon)$*-approximate solution in time* $n^{O_K\left(\log^8 n \cdot (\log n + \log(1/\varepsilon))^5/\varepsilon\right)}$*.*

Theorems 20, 22, and 24 imply Corollary 4.

5 Concluding Remarks

The special case of $0/{+\infty}$ AR (at a factor 2 loss) is equivalent to the following variant of CCCP: given a collection R of dépôts in a metric, find a collection of cycles of size $\leq k$ each containing a unique dépôt that together covers all the non-dépôt nodes. Although there are constant-factor approximations for CVRP, we do not know of a good approximation for this version.

Acknowledgements. We want to thank Zachary Friggstad for his comments that improved and simplified Theorem 1. We also talked to Mohsen Rezapour for some initial discussions.