Approximation Algorithms for the Airport and Railway Problem[*]

Mohammad R. Salavatipour and Lijiangnan Tian

Department of Computer Science University of Alberta.

Contributing authors: mrs@ualberta.ca; lijiangn@ualberta.ca;

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 [1], is a combination of the Capacitated Facility Location problem (CFL) and the Network Design Problem (NDP). 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. [1] presented a PTAS for AR in the two-dimensional Euclidean metric $\mathbb{R}^2$ with a uniform opening cost. In subsequent work [2] 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.

Keywords: Facility Location, Approximation Algorithms, Dynamic Programming

[*]A preliminary version of this paper appeared in Proceedings of SWAT 2024.

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 [1] and studied further in [2]) 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 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. [2] 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

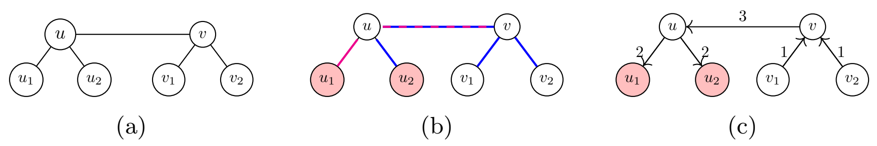


Fig. 1: a) An example tree where we assume the airport capacity is 3 and u_1 and u_2 have an opening cost of zero while other vertices have cost infinity; b) The solution to $\widetilde{\text{AR}}$. Pink vertices represent cities with an airport. Each edge is coloured to indicate its cluster. The dashed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

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 [2], AR can be modelled as CMST in general weighted (non-metric) graphs.

The following variants of AR have been studied [1, 2]. For some constant $\beta > 1$, AR$_\beta$ 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$^\infty$ 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 1AR. 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, [1, 2] 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 1AR$_P$, [1] show that the problems are **NP**-hard in Euclidean metrics and present PTAS's for them.

In [2] 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 [3, 4].

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 [5–8]. Another related problem is the *capacitated cycle cover problem* (CCCP) studied in [9]. 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 [10, 11]. In [9] 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 [12] 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 [2], AR can be reduced to CMST in non-metric setting.

We refer to [2] for discussion of other related works such as capacitated-cable facility location problem (CCFLP) [13] and sink clustering problem [14].

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. 1AR) 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.*

	General metric	Euclidean plane	Bounded treewidth
Nonuniform airport cost	Bicriteria: $\frac{4}{3} \cdot (2 + \frac{1}{p})$-approximate for AR_{1+p}, $p \in (0, 1]$ [2]; $O(\log n)$ (Corollary 1.1)	PTAS for AR_P^∞; exact for AR_F^∞; QPTAS for $\mathrm{AR}_{1+\mu}$, $\mu > 0$; $(2 + \frac{k}{k-1} + \varepsilon)$-approximate for AR [2]	Exact algorithm for AR$'$ (Theorem 3)
Uniform airport cost	2 (Theorem 1)	PTAS for both $1\mathrm{AR}_F$ and $1\mathrm{AR}_P$ [1]	QPTAS for $\widetilde{\mathrm{AR}}$ (Theorem 2)

Fig. 2: Summary of previous work and our results

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$'$ ([2]) we obtain the following corollary.

Corollary 1.1. *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 4. *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.*

A summary of previous and our results appear in Figure 2. 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 1.1. We present the proof of Theorem 4 in Section 5.

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 1. *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* σ*.*

Lemma 2.1. *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. $\square$

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 $\sigma \le n$. Note that we know $\sigma \ge \lceil \frac{n}{k} \rceil$ 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 [15].

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 $\tilde{G}$,

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

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 $|\mathcal{P}_k| \le \lfloor \frac{n}{k} \rfloor$ since there are at most n vertices (other than the vertex r) in the graph. In addition, as we know $\sigma \ge \lceil \frac{n}{k} \rceil$, we have

$|\mathcal{P}_k| \le \lfloor \frac{n}{k} \rfloor \le \lceil \frac{n}{k} \rceil \le \sigma$. Consequently, the cost of opening these $|\mathcal{P}_k|$ airports is $|\mathcal{P}_k| \cdot f \le \sigma \cdot f$. For those subpaths of length less than k, we simply open one of its vertices as the facility. Note that since there are $|\mathcal{T}| = \sigma' \le \sigma$ 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 $f \cdot \sigma + |\mathcal{P}_k| \cdot f \le 2\sigma \cdot f$. 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 at most $f \cdot \sigma$. 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 2. *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 [16]. 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 [8] 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 3. (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 [8]: 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 $(1+O(\varepsilon))$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 [8], 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 [17]). 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 $(1+\varepsilon)^i \dots [(1+\varepsilon)^{i+1}-1]$. 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 [8] 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 [18], 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 4. *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 [8]. 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 5. *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 6. *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 $\alpha \geq \max\{1, 20\delta\}$.*

Definition 7. *For a big bucket $(\beta, b_i, S, \wp)$, define g groups where $g = \frac{2\delta \log n}{\varepsilon}$, denoted as $G^{\beta,S,\wp}_{i,1}, G^{\beta,S,\wp}_{i,2}, \ldots, G^{\beta,S,\wp}_{i,g}$ 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 $\frac{n^{S,\wp}_{\beta,i}}{g}$ clusters into $G^{\beta,S,\wp}_{i,1}$, the second $\frac{n^{S,\wp}_{\beta,i}}{g}$ into $G^{\beta,S,\wp}_{i,2}$, etc. For each group $G^{\beta,S,\wp}_{i,j}$, denote the size of its smallest cluster as $h^{\beta,S,\wp,\min}_{i,j}$ and the size of its biggest cluster as $h^{\beta,S,\wp,\max}_{i,j}$.*

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 [8] 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 [8]. 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 [8] 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 3.1. *The expected cost of the extra clusters sampled is $2\varepsilon \cdot \text{OPT}$.*

Proof. Consider an edge e in the input graph G. Let $\phi(e)$ denote the number of clusters using the edge e in OPT. Let $\phi'(e)$ denote the number of sampled clusters using e. Considering we have duplicated each sampled cluster, $2\phi'(e)$ corresponds to the total

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

number of extra clusters using e in OPT$'$. Let $\mathcal{T}$ and $\mathcal{T}'$ denote the number of clusters in OPT and sampled clusters respectively. We have

$$\text{OPT}' = f \cdot (|\mathcal{T}| + 2|\mathcal{T}'|) + \sum_{e \in E} c(e) \cdot (\phi(e) + 2\phi'(e))$$

where the cost of extra clusters is $f \cdot 2|\mathcal{T}'| + \sum_{e \in E} c(e) \cdot 2\phi'(e)$.

Using the proof in [8], we know $\mathbb{E}[\phi'(e)] = \varepsilon \cdot \phi(e)$.[2]

Additionally, we know

$$\mathbb{E}[|\mathcal{T}'|] = \varepsilon \cdot |\mathcal{T}|$$

Thus we see that the expected total cost of these extra clusters is

$$\begin{aligned}
\mathbb{E}\left[f \cdot 2|\mathcal{T}'| + \sum_{e \in E} c(e) \cdot 2\phi'(e)\right] &= 2f \cdot \mathbb{E}[|\mathcal{T}'|] + \sum_{e \in E} c(e) \cdot 2\mathbb{E}[\phi'(e)] \\
&= 2f \cdot (\varepsilon \cdot |\mathcal{T}|) + \sum_{e \in E} c(e) \cdot 2(\varepsilon \cdot \phi(e)) \\
&= 2\varepsilon \left(f \cdot |\mathcal{T}| + \sum_{e \in E} c(e) \cdot \phi(e)\right) \\
&= 2\varepsilon \cdot \text{OPT}.
\end{aligned}$$

□

We make use of the following modified definitions and lemmata from [8]. 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 3.2. *At any level ℓ, each bag $\beta \in B_\ell$ and its big buckets $(\beta, b_i, S, \wp)$ satisfy, w.h.p.*

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

Proof. The part before the union bound is very similar to the proof in [8]. That is, we know

$$\Pr\left[\left|X_{\beta,i}^{S,\wp}\right| < \frac{1}{2}\mathbb{E}\left[\left|X_{\beta,i}^{S,\wp}\right|\right]\right] \leq \frac{1}{n^5}.$$

[2]For the notations, $\phi(e)$ corresponds to $f^+(e) + f^-(e)$ in [8], and $\phi'(e)$ corresponds to $m^+(e) + m^-(e)$.

where

$$\mathbb{E}\left[\left|X_{\beta,i}^{S,\wp}\right|\right] = \frac{2\varepsilon}{\delta \log n} \cdot n_{\beta,i}^{S,\wp}.$$

Note that in a bag β (as a set it satisfies $|\beta| \le \omega + 1$), the number of set of vertices through which the clusters span into C_β (i.e., all possible S) is $O(2^{\omega+1})$, for $S \subseteq \beta$ (that is, $2^\beta \ni S$) has $2^{|\beta|} \in O(2^{\omega+1})$ possibilities. In addition, given a set S with $s = |S|$, the set $\wp$ has $\mathcal{B}_s$ possibilities, where $\mathcal{B}_n$ denotes the n-th Bell number. Note that the Bell numbers $\mathcal{B}_i$ are used to calculate the number of all the possible partitions of a set of size i, defined in [19, 20]. For a specific bag β, set S and partition $\wp$, the number of buckets $(\beta, b_i, S, \wp)$ is by definition the same as the number of threshold values, which is $\tau \in O(\log k/\varepsilon)$. Since $\tau \in O(\log n/\varepsilon)$ and there are in total $O(\omega n)$ bags in T, we know the total number of buckets $(\beta, b_i, S, \wp)$ in T is $O(\omega n \log n/\varepsilon \cdot 2^{\omega+1} \cdot \mathcal{B}_\omega)$. Union bound over every bucket $(\beta, b_i, S, \wp)$, we have

$$\sum_{\text{all } (\beta,\, b_i,\, S,\, \wp)} \Pr\left[\left|X_{\beta,i}^{S,\wp}\right| < \frac{1}{2}\mathbb{E}\left[\left|X_{\beta,i}^{S,\wp}\right|\right]\right] \le O\left(\frac{1}{n}\right).$$

□

Lemma 3.3. *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 $X_{\beta,i}^{S,\wp}$.)*
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 [8], we can obtain a Structure Theorem for our UAR problem in the case of graphs with bounded treewidth.

Theorem 5. (Structure Theorem) *Consider an instance $\mathcal{I}$ for the* UAR *problem. Denote its optimal solution as* OPT, *with cost* $\textsc{opt}$. *We can transform* OPT *to another solution* OPT′ *so that, with high probability,* OPT′ *is a near-optimal solution of cost at most $(1 + 2\varepsilon)\textsc{opt}$. 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*

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

  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 [8] 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 5 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.

3.2 Dynamic Programming for Graphs of Bounded Treewidth

The subproblem here corresponds to the problem for C_β which is rooted at some bag β, whose cost is the railway cost of all the clusters (partial clusters and complete ones), plus f times the number of complete clusters. Again, for a complete cluster, we say it is *independent* or stops growing.

We will come up with a dynamic programming algorithm that finds a solution with the properties stated in the previous Structure Theorem.

We adapt the definitions of the ones in [8]. Recall that τ is the total number of threshold values. Also recall that the bucket $(\beta, b_i, S, \wp)$ stores all those clusters spanning ζ demands in C_β, using vertices in $S \subseteq \beta$ to span into C_β and $\wp$ depicting downwards connectivity, where $\sigma_i \leq \zeta < \sigma_{i+1}$, and in particular, for all $1 \leq i \leq \lceil 1/\varepsilon \rceil$ this interval degenerates to simply the set $\{i\}$, by definition of the threshold values.

Note that initially, we have n demands, so the total number of clusters is bounded by n. Throughout the algorithm, although we will add dummy/extra demands, those are added within each cluster (hence already covered) and thus will not result in any increase in the number of clusters.

For each bag β, set $S \subseteq \beta$ and $\wp$, we define a vector $\mathbf{m}^{\beta,S,\wp} \in [n]^\tau$ where its i-th component $m_i^{\beta,S,\wp}$ stores the cardinality of the bucket $(\beta, b_i, S, \wp)$, for all $1 \leq i \leq \tau$. In particular, $m_i^{\beta,S,\wp}$ is the precise number of clusters of size i, if $1 \leq i \leq \lceil 1/\varepsilon \rceil$.

As each bag β contains at most $\omega + 1$ vertices, define $\mathbf{d}_\beta \in [n]^{\omega+1}$ to be a vector storing the numbers of demands in β, where $d_{\beta,v}$ represents the number of demands to be covered at the vertex $v \in \beta$. Note that $\|\mathbf{d}_\beta\| = \sum_{v\in\beta} d_{\beta,v}$. In addition, let o_β represent the overall demands of every vertex in the bags of C_β. That is, $o_\beta = \sum_{\tilde{\beta}\in C_\beta} \|\mathbf{d}_{\tilde{\beta}}\|$.

Since we do not know if a bucket $(\beta, b_i, S, \wp)$ is small or big in advance, we also need the following three vectors. In case b_i is small (meaning its cardinality is bounded by $\xi = \alpha \log^2 n/\varepsilon$), all the sizes of the partial clusters therein are stored precisely, with the help of a vector $\mathbf{t}^{\beta,S,\wp,i} \in [n]^\xi$ where its j-th element $t_j^{\beta,S,\wp,i}$ stores the size of the j-th partial cluster in b_i. On the other hand, in case b_i is big, there will be $g = 2\delta \log n/\varepsilon$ potential cluster sizes in b_i, and we will store the information in the following two vectors. Recall that $h_{i,j}^{\beta,S,\wp,\max}$ represents the size of the maximum partial cluster in group j of bucket b_i.

- $\mathbf{h}^{\beta,S,\wp,i} \in [n]^g$ is a vector that stores all those g cluster sizes, where $h_j^{\beta,S,\wp,i} = h_{i,j}^{\beta,S,\wp,\max}$.
- $\mathbf{l}^{\beta,S,\wp,i} \in [n]^g$ is a vector storing the corresponding number of partial clusters that are of one of the g sizes, with $l_j^{\beta,S,\wp,i}$ representing the number of partial clusters picking up $h_{i,j}^{\beta,S,\wp,\max}$ demands (which is also the cardinality of $G_{i,j}^{\beta,S,\wp}$, that is, group j of the bucket).

For a given triplet $\left(\mathbf{t}^{\beta,S,\wp,i}, \mathbf{h}^{\beta,S,\wp,i}, \mathbf{l}^{\beta,S,\wp,i}\right)$, it is impossible that none of them are zero vectors. Since b_i is either small or big, it must be the case that either only $\mathbf{t}^{\beta,S,\wp,i}$ is equal to the zero vector, or the other two vectors are.

Moreover, we use the shorthand

$$\mathbf{z}^{\beta,S,\wp} = \left(\left(\mathbf{t}^{\beta,S,\wp,1}, \mathbf{h}^{\beta,S,\wp,1}, \mathbf{l}^{\beta,S,\wp,1}\right), \left(\mathbf{t}^{\beta,S,\wp,2}, \mathbf{h}^{\beta,S,\wp,2}, \mathbf{l}^{\beta,S,\wp,2}\right), \ldots, \left(\mathbf{t}^{\beta,S,\wp,\tau}, \mathbf{h}^{\beta,S,\wp,\tau}, \mathbf{l}^{\beta,S,\wp,\tau}\right)\right).$$

Just like the vector $\mathbf{z}^{\beta,S,\wp}$ is used for partial clusters, we define another vector $\tilde{\mathbf{z}}^{\beta,S,\wp}$ with the same structure as $\mathbf{z}^{\beta,S,\wp}$, except that $\tilde{\mathbf{z}}^{\beta,S,\wp}$ is intended for storing information on complete clusters that pick up demands at β using vertices in S to span into C_β with $\wp$ depicting downwards connectivity. We also define $\tilde{\mathbf{t}}^{\beta,S,\wp,j}, \tilde{\mathbf{h}}^{\beta,S,\wp,j}, \tilde{\mathbf{l}}^{\beta,S,\wp,j}$ in a similar way.

To make the notations more compact, we define another shorthand

$$\mathbf{p}_{\beta,S,\wp} = \left(\mathbf{m}^{\beta,S,\wp}, \mathbf{z}^{\beta,S,\wp}, \tilde{\mathbf{z}}^{\beta,S,\wp}\right).$$

Let $\mathbf{p}_\beta$ be the vector containing all the $\mathbf{p}_{\beta,S,\wp}$, i.e., $\mathbf{p}_{\beta,S,\wp}$ for all possible combination of the set $S \subseteq \beta$ and its partition $\wp$. Note that there are $O(2^{\omega+1} \cdot \mathcal{B}_\omega)$ possibilities for such combinations of S and $\wp$.

Define $\mathbf{y}_\beta$ to be a configuration/profile of clusters in C_β that involves bag β

$$\mathbf{y}_\beta = (o_\beta, \mathbf{d}_\beta, \mathbf{p}_\beta).$$

Each entry $\mathbf{A}[\beta, \mathbf{y}_\beta]$ stores the cost of the cheapest solution to the subproblem at β having its cluster profile in accordance with $\mathbf{y}_\beta$, which consists of the cost of all the partial and complete clusters spanning in C_β. Eventually, we compute $\min_{\mathbf{y}_r} \mathbf{A}[r, \mathbf{y}_r]$ to obtain the final answer, where r is the root bag of the entire tree T and $\mathbf{y}_r$ is a configuration that does not contain any partial clusters. An argument similar to the one for the case of trees in the previous section shows that this result corresponds to a solution of cost bounded by $\textsc{opt}' \leq (1 + 2\varepsilon)\textsc{opt}$ with other properties described in the structure theorem.

Within each bag β, if no edge is between two vertices $v, w \in \beta$ then we simply add an edge vw between them and assign it a cost that is equal to the cost of the shortest path between v and w in G.

We calculate the table from the bottom up. Base cases: consider each leaf bag β, roughly speaking, we will store the minimum cost of all the clusters' edges, which lies between the vertices within β, together with the cost of all the independent clusters, if any. Note that all these clusters need to pick up o_β demands in total, and in more detail, pick them up in accordance with $\mathbf{d}_\beta$. The formal definition will be given later.

Note that we can ignore the case where β has only one child because we can make all the non-leaf bags ϖ in T with only one child to have two children bags, by adding another child bag which is trivially composed of some vertices in ϖ. Given that T is binary, we can just focus on the case where a non-leaf bag β has two children β_1 and β_2. If C_{β_1} and C_{β_2} have o_{β_1} and o_{β_2} demands respectively, then C_β has $o_\beta = \|\mathbf{d}_\beta\| + o_{\beta_1} + o_{\beta_2}$. Given $\mathbf{A}[\beta_1, \mathbf{y}_{\beta_1}]$ and $\mathbf{A}[\beta_2, \mathbf{y}_{\beta_2}]$, we also need to define the following two auxiliary tables. We explain their functions here and define them formally later.

- The edge-cost table $\mathbf{E}[\mathbf{d}_\beta, \mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}]$ stores the cost of a set of chosen edges which go between vertices from β and β_1, also β and β_2. Note that these chosen edges merge some of the clusters in $\mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$ to form certain clusters in $\mathbf{y}_\beta$.
- The Boolean consistency table $\mathbf{H}[\mathbf{d}_\beta, \mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}]$ checks if $\mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}$ are consistent.

Note that if the configurations $\mathbf{y}_\beta, \mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$ are *consistent*, it means (informally) the clusters from $\mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$ can be combined or augmented to form the clusters in $\mathbf{y}_\beta$ (as well as picking up the dummy demands at the vertices in β).

Recall that $\mathbf{y}_\beta = (o_\beta, \mathbf{d}_\beta, \mathbf{p}_\beta)$ where $\mathbf{p}_{\beta,S,\wp} = \left(\mathbf{m}^{\beta,S,\wp}, \mathbf{z}^{\beta,S,\wp}, \tilde{\mathbf{z}}^{\beta,S,\wp}\right)$. Thus from $\mathbf{y}_\beta$ we can obtain the number of independent clusters in C_β, which we denote as ϱ_β. We define

$$\mathbf{A}[\beta, \mathbf{y}_\beta] = \min_{\substack{\mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}: \\ \mathbf{H}[\mathbf{d}_\beta, \mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}] = \textsc{True}}} \left\{\mathbf{A}[\beta_1, \mathbf{y}_{\beta_1}] + \mathbf{A}[\beta_2, \mathbf{y}_{\beta_2}] + f \cdot (\varrho_\beta - \varrho_{\beta_1} - \varrho_{\beta_2}) + \mathbf{E}[\mathbf{d}_\beta, \mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}]\right\}.$$

The third term in the curly brackets is to account for the change in the cost of independent clusters.

Now we can define the entries in the table for the leaves. For a leaf bag β, it has no children so we set both $\mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}$ to the zero vector. We define

$$\mathbf{A}[\beta, \mathbf{y}_\beta] = f \cdot \varrho_\beta + \mathbf{E}[\mathbf{d}_\beta, \mathbf{y}_\beta, \mathbf{0}, \mathbf{0}].$$

where $\mathbf{E}[\mathbf{d}_\beta, \mathbf{y}_\beta, \mathbf{0}, \mathbf{0}]$ is the minimum railway cost for clusters in $\mathbf{y}_\beta$, which pick up demands in β in accordance with $\mathbf{d}_\beta$.

3.2.1 Consistency Check

Continue the discussion from the previous section, recall that β is a non-leaf bag with children β_1, β_2 and we have three configurations $\mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}$. We adapt the checking procedure in [8] to suit our data structure. The following list is a recap of the meanings of these three configurations.

- $\mathbf{y}_\beta$ keeps track of clusters covering demands of the vertices in $C_\beta = \{\beta\} \cup C_{\beta_1} \cup C_{\beta_2}$.

- $\mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}$ keep track of clusters covering demands of the vertices in C_{β_1}, C_{β_2} respectively.

Denote a cluster from $\mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}$ as $\gamma_\beta, \gamma_{\beta_1}, \gamma_{\beta_2}$ respectively. Recall we define $|\gamma|$ as the number of demands the cluster γ picks up. Each cluster γ in $\mathbf{y}_\beta$ that spans down C_β picking up demands in C_{β_1}, C_{β_2} can take one of the following four forms.

1. γ only covers demands at β and not the ones from $C_{\beta_1} \cup C_{\beta_2}$.
2. γ covers demands at β as well as the ones from C_{β_1}.
3. γ covers demands at β as well as the ones from C_{β_2}.
4. γ covers demands at β as well as the ones from $C_{\beta_1} \cup C_{\beta_2}$.

Definition 8. *We say configurations $\mathbf{y}_\beta, \mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$ are* **consistent** *if the following holds:*

- *Each cluster in $\mathbf{y}_{\beta_1}$ corresponds to a cluster in $\mathbf{y}_\beta$.*
- *Each cluster in $\mathbf{y}_{\beta_2}$ corresponds to a cluster in $\mathbf{y}_\beta$.*
- *Each cluster in $\mathbf{y}_\beta$ has at most $2^{\omega+1}$ clusters that correspond to it, from each of $\mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$.*
- *Each cluster γ_β in $\mathbf{y}_\beta$ has q_{β_1} clusters $\gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \ldots, \gamma^{q_{\beta_1}}_{\beta_1}$ from $\mathbf{y}_{\beta_1}$ and q_{β_2} clusters $\gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \ldots, \gamma^{q_{\beta_2}}_{\beta_2}$ from $\mathbf{y}_{\beta_2}$ that correspond to it, where $0 \le q_{\beta_1}, q_{\beta_2} \le 2^{\omega+1}$, with the help of a set Υ_{γ_β} of edges between vertices in β, β_1 and β_2 to patch them up to get γ_β. Besides, γ_β covers $|\gamma_\beta| - \sum_{i=1}^{q_{\beta_1}} |\gamma^i_{\beta_1}| - \sum_{j=1}^{q_{\beta_2}} |\gamma^j_{\beta_2}|$ dummy demands within β.*
- *Clusters of $\mathbf{y}_\beta$ cover all demands at β in accordance with $\mathbf{d}_\beta$.*

Recall $\mathbf{p}_\beta$ is the vector containing all the $\mathbf{p}_{\beta,S,\wp}$, i.e., $\mathbf{p}_{\beta,S,\wp}$ for all possible combinations of $S \subseteq \beta$ and its partition $\wp$, where $\mathbf{p}_{\beta,S,\wp} = \left(\mathbf{m}^{\beta,S,\wp}, \mathbf{z}^{\beta,S,\wp}, \tilde{\mathbf{z}}^{\beta,S,\wp}\right)$. Recall $|\gamma|$ denotes the total number of demands picked up by cluster γ. We now use $\mathbf{p}_\beta$ as arguments for both of the auxiliary tables, instead of $\mathbf{y}_\beta$. The notation $\mathbf{p}_\beta \setminus \gamma_\beta$ represents a new configuration obtained by removing a cluster of size $|\gamma_\beta|$ from $\mathbf{p}_\beta$. This can be achieved by manipulating vectors $\mathbf{t}$'s and $\mathbf{l}$'s. Now we define the Boolean consistency table $\mathbf{H}$ for each bag β.[3]

$$\mathbf{H}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}] = \begin{cases} \text{TRUE} & \text{if } \mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2} \text{ are consistent} \\ & \text{and } \mathbf{y}_\beta \text{ covers } \mathbf{d}_\beta \text{ demands} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Base case: trivially, $\mathbf{H}[\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}] = $ TRUE, since we can cover zero units of demands with no clusters. Next, it examines all potential ways of merging $\mathbf{y}_{\beta_1}$ with $\mathbf{y}_{\beta_2}$ into $\mathbf{y}_\beta$, including extending some of the clusters in order to cover those $\mathbf{d}_\beta$ dummy demands. Recall $\mathbf{d}_\beta \in [n]^{\omega+1}$ is a vector storing the numbers of demands in β, where $d_{\beta,v}$ represents the number of demands to be covered at $v \in \beta$. Define $\mathbf{d}^{\gamma_\beta}_\beta \in [n]^{\omega+1}$ to be a vector storing the numbers of demands in β covered by γ_β, where $d^{\gamma_\beta}_{\beta,v}$ represents

[3]Recall we have only defined consistency on $\mathbf{y}$. The notion of consistency works on $\mathbf{p}$ as well since the omitted information o_β does not affect anything.

the number of demands covered at the vertex $v \in \beta$ by γ_β. By definition, its L^1-norm $\|\mathbf{d}_\beta^{\gamma_\beta}\| = \sum_{v\in\beta} d_{\beta,v}^{\gamma_\beta}$ is the total number of demands in β covered by γ_β.

According to Definition 8, we know it is necessary to check the existence of subclusters $\gamma_{\beta_1}^1, \gamma_{\beta_1}^2, \ldots, \gamma_{\beta_1}^{q_{\beta_1}}$ from $\mathbf{y}_{\beta_1}$ and $\gamma_{\beta_2}^1, \gamma_{\beta_2}^2, \ldots, \gamma_{\beta_2}^{q_{\beta_2}}$ from $\mathbf{y}_{\beta_2}$, as well as a set of edges Υ_{γ_β} for merging them. Besides, for every γ_β, we need to check each part of S (which corresponds to some element in $\wp$) that needs to be connected is actually made into one part thanks to these subclusters. Consider the downwards interface set $S_{\gamma_{\beta_i}^j}$ and the connectivity set $\wp_{\gamma_{\beta_i}^j}$ of the subclusters $\gamma_{\beta_i}^j$, for $i = 1, j \in [1, q_{\beta_1}]$ and $i = 2, j \in [1, q_{\beta_2}]$. We need to check whether the partition

$$\wp' = \left(\bigcup_{1\le j\le q_{\beta_1}} \wp_{\gamma_{\beta_1}^j} \right) \cup \left(\bigcup_{1\le j'\le q_{\beta_2}} \wp_{\gamma_{\beta_2}^{j'}} \right) \tag{$\dagger$}$$

matches $\wp_{\gamma_\beta}$, that is, to check whether the downward connectivity between the vertices of S_{γ_β} can be achieved by these subclusters. To be more precise, if $\wp_{\gamma_\beta} \ni \Psi$ and $\Psi \supseteq \{u, v\}$ (meaning the pair of vertices $u, v \in \beta$ needs to be connected downwards), then there needs to be a set X such that $X \supseteq \{u, v\}$ and $X \in \wp'$.

Define the recurrence relation of $\mathbf{H}$ as follows, where $0 \le q_{\beta_1}, q_{\beta_2} \le 2^{\omega+1}$ and the clusters $\gamma_{\beta_1}^1, \gamma_{\beta_1}^2, \ldots, \gamma_{\beta_1}^{q_{\beta_1}}, \gamma_{\beta_2}^1, \gamma_{\beta_2}^2, \ldots, \gamma_{\beta_2}^{q_{\beta_2}}$ are a part of γ_β.

$$\mathbf{H}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}] = \bigvee_{\substack{\gamma_\beta, \gamma_{\beta_1}^1, \gamma_{\beta_1}^2, \ldots, \gamma_{\beta_1}^{q_{\beta_1}}, \gamma_{\beta_2}^1, \gamma_{\beta_2}^2, \ldots, \gamma_{\beta_2}^{q_{\beta_2}}, \mathbf{d}_\beta^{\gamma_\beta} : \\ |\gamma_\beta| = \sum_{i=1}^{q_{\beta_1}} |\gamma_{\beta_1}^i| + \sum_{j=1}^{q_{\beta_2}} |\gamma_{\beta_2}^j| + \|\mathbf{d}_\beta^{\gamma_\beta}\|, \text{ and } (\dagger)}} \Omega$$

where the expression Ω should be

$$\mathbf{H}\left[\mathbf{d}_\beta - \mathbf{d}_\beta^{\gamma_\beta},\ \ \mathbf{p}_\beta \setminus \gamma_\beta,\ \ \mathbf{p}_{\beta_1} \setminus \gamma_{\beta_1}^1 \setminus \gamma_{\beta_1}^2 \setminus \cdots \setminus \gamma_{\beta_1}^{q_{\beta_1}},\ \ \mathbf{p}_{\beta_2} \setminus \gamma_{\beta_2}^1 \setminus \gamma_{\beta_2}^2 \setminus \cdots \setminus \gamma_{\beta_2}^{q_{\beta_2}}\right].$$

Recall $0 \le q_{\beta_1}, q_{\beta_2} \le 2^{\omega+1}$. This is a shorthand way of expressing it, as it in fact corresponds to the union of 2 entries of $\mathbf{H}$ and considers both the situations where γ_β is independent and where it is partial. That is, $\mathbf{p}_\beta \setminus \gamma_\beta$ means two possibilities where γ_β is deleted from some $\mathbf{z}$, or $\tilde{\mathbf{z}}$ from $\mathbf{p}_\beta$. This expression above essentially examines whether the remaining clusters in the (modified) profile $\mathbf{y}_\beta$ (which is defined by the new $\mathbf{p}_\beta$) cover $\mathbf{d}_\beta - \mathbf{d}_\beta^{\gamma_\beta}$ demands in C_β.

Recall the second auxiliary table $\mathbf{E}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}]$ is used to compute the cheapest cost of edges between vertices from β and β_1, also β and β_2 to merge the clusters at C_{β_1} and C_{β_2} in order to get clusters in $\mathbf{p}_\beta$. Initially, we set every entry of $\mathbf{E}$ to be $+\infty$. An entry $\mathbf{E}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}]$ will be computed only if the three corresponding configurations $\mathbf{y}_\beta, \mathbf{y}_{\beta_1}, \mathbf{y}_{\beta_2}$ are consistent, that is, only if $\mathbf{H}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}] = \textsc{True}$. The base case is to trivially set $\mathbf{E}[\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}] = 0$, for it does not cost anything to conjoin zero cluster. For the recurrence, given a cluster γ_β from $\mathbf{y}_\beta$, recall from Definition 8, we need to consider the clusters $\gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \ldots, \gamma^{q_{\beta_1}}_{\beta_1}$ from $\mathbf{y}_{\beta_1}$ and $\gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \ldots, \gamma^{q_{\beta_2}}_{\beta_2}$ from $\mathbf{y}_{\beta_2}$.

Besides, note that some of the partial clusters $\gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \ldots, \gamma^{q_{\beta_1}}_{\beta_1}$ from $\mathbf{y}_{\beta_1}$ may have been connected in C_{β_1} (the same may happen in C_{β_2} for $\gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \ldots, \gamma^{q_{\beta_2}}_{\beta_2}$ from $\mathbf{y}_{\beta_2}$). Recall S is the set of vertices in bag β used by γ to extend into lower levels of T. Again, we need to consider partitions of S into the form of $\{S_1, S_2, \ldots\}$ where the partial clusters extending below using S_i are all connected for each i. We also consider a set of edges $\Upsilon^{\wp}_{\gamma_\beta}$ between the vertices of bag β and vertices from β_1 as well as vertices from β and vertices from β_2. Set $\Upsilon^{\wp}_{\gamma_\beta}$ is used to patch up the subclusters into a single one. We define $\mathrm{cost}\left(\Upsilon^{\wp}_{\gamma_\beta}\right) = \sum_{e \in \Upsilon^{\wp}_{\gamma_\beta}} \mathrm{cost}(e)$.

We define, only for entries satisfying $\mathbf{H}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}] = \textsc{True}$,

$$\mathbf{E}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}] = \min_{\substack{\gamma_\beta, \gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \ldots, \gamma^{q_{\beta_1}}_{\beta_1}, \gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \ldots, \gamma^{q_{\beta_2}}_{\beta_2}, \mathbf{d}^{\gamma_\beta}_\beta, \Upsilon_{\gamma_\beta} : \\ |\gamma_\beta| = \sum_{i=1}^{q_{\beta_1}} |\gamma^i_{\beta_1}| + \sum_{j=1}^{q_{\beta_2}} |\gamma^j_{\beta_2}| + \|\mathbf{d}^{\gamma_\beta}_\beta\|, \\ \gamma_\beta \text{ consists of } \gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \ldots, \gamma^{q_{\beta_1}}_{\beta_1}, \gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \ldots, \gamma^{q_{\beta_2}}_{\beta_2} \text{ and edges in } \Upsilon_{\gamma_\beta}}} \Theta$$

where Θ should be the expression

$$\left\{\mathrm{cost}(\Upsilon^{\wp}_{\gamma_\beta}) + \mathbf{E}\left[\mathbf{d}_\beta - \mathbf{d}^{\gamma_\beta}_\beta,\ \mathbf{p}_\beta \setminus \gamma_\beta,\ \mathbf{p}_{\beta_1} \setminus \gamma^1_{\beta_1} \setminus \gamma^2_{\beta_1} \setminus \cdots \setminus \gamma^{q_{\beta_1}}_{\beta_1},\ \mathbf{p}_{\beta_2} \setminus \gamma^1_{\beta_2} \setminus \gamma^2_{\beta_2} \setminus \cdots \setminus \gamma^{q_{\beta_2}}_{\beta_2}\right]\right\}.$$

3.2.2 Algorithm Efficiency

We first discuss the runtime for table **A** and then we talk about **E** and **H**. The runtime analysis is very similar to that in [8], so from the analysis there, we see for a triplet of the form $(\mathbf{t}^{\beta,S,\wp,i}, \mathbf{h}^{\beta,S,\wp,i}, \mathbf{l}^{\beta,S,\wp,i})$, there exists $n^{O(\log^2 n/\varepsilon)}$ possibilities. Since $\mathbf{z}^{\beta,S,\wp}$ contains $\tau \in O(\log k/\varepsilon)$ such triplets, it has $n^{O(\tau \log^2 n/\varepsilon)} = n^{O(\log k \log^2 n/\varepsilon^2)}$ possibilities. By definition of $\mathbf{p}^{\beta,S,\wp}$, we see it also has $n^{O(\log k \log^2 n/\varepsilon^2)}$ possibilities. Given $\mathbf{p}_\beta$ contains $O(2^{\omega+1} \cdot \mathcal{B}_\omega)$ many $\mathbf{p}^{\beta,S,\wp}$, we know $\mathbf{p}_\beta$ has $n^{O(2^{\omega+1} \cdot \mathcal{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ possibilities. By definition of $\mathbf{y}_\beta$, it therefore also has $n^{O(2^{\omega+1} \cdot \mathcal{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ possibilities. The number of possibilities is the same for $\mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$, so it takes $n^{O(2^{\omega+1} \cdot \mathcal{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ to calculate the entries of table $\mathbf{A}[\beta, \cdot]$ for a single bag β (according to **A**'s recurrence), assuming the other tables **E** and **H** are already computed. The time it takes to calculate the entries across all the bags in T is the same.

For a single entry of the table $\mathbf{E}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_1}]$, we need to consider every possible combinations of $\left(\gamma_\beta, \gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \ldots, \gamma^{q_{\beta_1}}_{\beta_1}, \gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \ldots, \gamma^{q_{\beta_2}}_{\beta_2}, \mathbf{d}^{\gamma_\beta}_\beta, \Upsilon_{\gamma_\beta}\right)$, which is $n^{O(2^{\omega+1})}$, because the possibilities for all the clusters are $n^{O(2^{\omega+1})}$, and $\mathbf{d}_\beta$ has $n^{O(\omega)}$ possibilities, also we know Υ_{γ_β} has $O\left(2^{\omega^2}\right)$ possibilities by definition. Since there are $n^{O(2^{\omega+1} \cdot \mathcal{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ possibilities for $(\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_1})$, we know it takes $n^{O(2^{\omega+1} \cdot \mathcal{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ to compute the table **E**. From a similar analysis, we conclude the same runtime for the consistency table $\mathbf{H}$. Therefore, our algorithm has a runtime of $n^{O(2^{\omega+1} \cdot \mathcal{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)} = n^{O(2^\omega \cdot \mathcal{B}_\omega \cdot \log^3 n/\varepsilon^2)}$, since the facility capacity k satisfies $k < n$. Since $\mathcal{B}_n \leq O((n/\ln n)^n)$ the runtime is thus in $n^{O(\omega^\omega \cdot \log^3 n/(\varepsilon^2 \log^\omega \omega))}$.

4 Constant Approximation for Nonuniform-AR

4.1 AR′ in Trees

For ease of exposition, we first present the proof for the case of trees (the extension to graphs with bounded treewidth appears in the full version). In the next section, we will prove Theorem 3. 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 $\sum\limits_{v \in \Phi} a_v + \sum\limits_{e \in \Xi} c_e$. First, we prove some properties in an optimum solution to AR′.

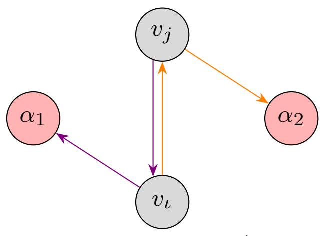


Fig. 3: A simplest example of crossing flows in AR′. The red vertices are open facilities.

Lemma 4.1. *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.*

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

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. $\square$

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

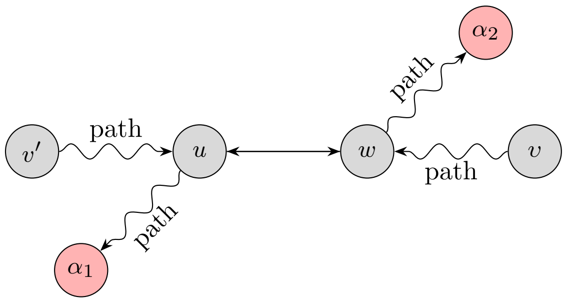


Fig. 4: The crossing flow is at the edge uw

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 $0 \le f_v \le n$, 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 $\pi_v = \textsc{True}$. 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 $w_1, w_2, \ldots, w_z$, 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 $w_1, w_2, \ldots, w_j$. Lastly, we define $\mathbf{D}[v, \pi_v, \psi_v] = \mathbf{B}[v, \pi_v, \psi_v, z]$.

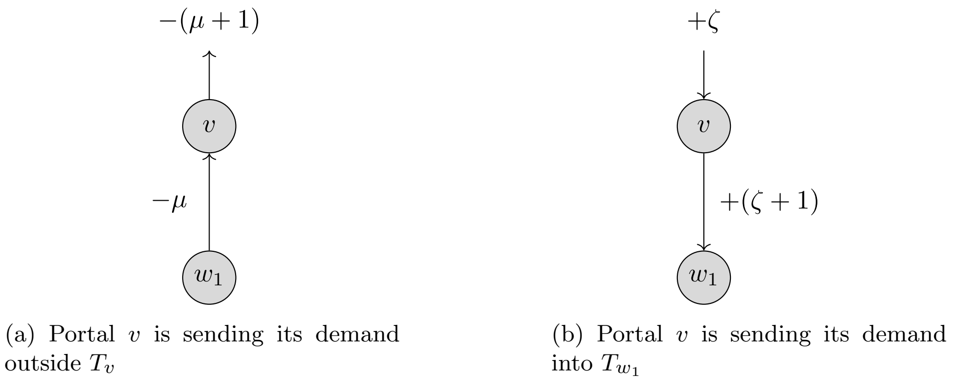
Case 1: $j = 1$. Only consider the first child of v.

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

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.

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

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).



(a) Portal v is sending its demand outside T_v

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

Fig. 5: 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 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 $2 \leq j \leq z$. Assume all entries of the form $\mathbf{B}[v, \pi_v^{j-1}, \psi_v^{j-1}, j-1]$ have been computed. We define

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

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:

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

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 $0 < \psi_v^j - (\psi_v^{j-1} + \psi_{w_j}) \leq k$ 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).

4.1.1 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 AR$'$ in Graphs with Bounded Treewidth

In this section, we prove Theorem 3. As in the case of uniform opening cost, given an input graph $G = (V, E)$ of treewidth ω, we can assume there is a tree decomposition $T = (V', E')$ of G that is binary, with depth $O(\log |V|)$ and treewidth no bigger than

$\tilde{\omega} = 3\omega + 2$, according to [16]. For simplicity, we write $\tilde{\omega}$ as ω. Recall that we refer to the vertices in T as bags, to differentiate them from the vertices in G. For each bag β, denote the union of bags in the subtree T_β as C_β. For the notation bag β, we refer to it as the name of the bag in T as well as the corresponding set of vertices in G.

Consider any two adjacent bags β_1, β_2 in T. The coherent set of the two bags is defined as the intersection of the two sets, $\beta_1 \cap \beta_2$, a nonempty set containing the vertices of a vertex-cutset in G. Thus, while considering any bag β (other than the root bag) in T, we can first obtain the intersection of β and β's parent bag, and use the vertices in the intersection as portals. So when a flow in T is "jumping" to another bag, it would cost nothing as it is essentially at the same portal, before the flow goes elsewhere. This way we can ensure the railway cost would only arise inside each bag.

Recall that a vertex of G can potentially show up in many bags of T, we thus assume that for any vertex $v \in V(G)$, the copy of v in the bag $\tilde{\beta}$ that is the closest to the root bag (we know there exists such a unique one) has a demand of one, and the rest of the copies of v (in other bags) have demand zero. We denote the vertex v in its corresponding closest-to-the-root bag $\tilde{\beta}$ as $\tilde{v}$. Note that, in the perspective of the flow network, if $\tilde{v}$ is not a facility then it is a source emitting one unit of demand. In addition, if v is to be opened as a facility, only the copy of v in the bag $\tilde{\beta}$ (that is, $\tilde{v}$) will be opened. In this case, $\tilde{v}$ is a sink of capacity k. Note that if a portal p is not $\tilde{p}$, then it can be neither a source nor a sink.

For each bag β, we define a vector Π_β where its v-th component Π_β^v is a Boolean variable saying whether the vertex $v \in \beta$ is opened as an airport. The portal configuration Ψ_β of a bag β will include three signed integers $\left(\phi_{p_{\beta_0}}, \phi_{p_{\beta_1}}, \phi_{p_{\beta_2}}\right)$ for each portal $p \in \beta$, indicating the amounts and the directions of the demands going in or out of it. Note that we need three such values because the portal p may send flows to (or receive flows from) some other copies of the vertex p in β's parent bag β_0 (denote this copy of p as p_{β_0}) or in β's two children bags β_1 and β_2 (denote these copies of p as p_{β_1} and p_{β_2}). If p does not have some of these copies (p_{β_0}, p_{β_1} and p_{β_2}) then the corresponding directed values of it are simply set to 0. For a directed value $\phi_{p_{\beta_i}} = \pm f_{p_{\beta_i}}$ of a portal $p \in \beta$, $f_{p_{\beta_i}}$ represents the value of the flow, and "+" means going into $p \in \beta$ from the copy $p_{\beta_i} \in \beta_i$ whereas "−" means coming out of p to the copy p_{β_i}.

We have a DP cell $\mathbf{D}[\beta, \Pi_\beta, \Psi_\beta]$ for each bag β, its portal configuration Ψ_β, and the facilities to be opened within β given by Π_β. We let the entry $\mathbf{D}[\beta, \Pi_\beta, \Psi_\beta]$ store the cost of the optimal solution to AR$'$ on the induced subgraph $G[C_\beta]$, where the vertices in bag β conform to the portal configuration Ψ_β, and are opened as airports according to Π_β. The notation $G[C_\beta]$ denotes the subgraph of G on the vertex set C_β. Essentially, the copy of a vertex v in bag β has signed values the same as $\Psi_\beta^v = \left(\phi_{v_{\beta_0}}, \phi_{v_{\beta_1}}, \phi_{v_{\beta_2}}\right)$, that is, the copy $v \in \beta$ sends/receives $\phi_{v_{\beta_i}} \in [-n, +n]$ units of flow to the copy $v_{\beta_i} \in \beta_i$ for $i = 0, 1, 2$. A vertex $v \in \beta$ is opened as an airport only if $\Pi_\beta^v = \textsc{True}$. We will discuss the consistency check later.

The final result would be $\min_{\Pi_\theta, \Psi_\theta}\{\mathbf{D}[\theta, \Pi_\theta, \Psi_\theta]\}$ where θ is the root bag of T, with $\phi_{p_{\theta_0}}$ set to 0 for every vertex $p \in \theta$ as the parent bag θ_0 does not exist.

In the base case, we consider the leaf bags of T. For a leaf bag β, $\phi_{p_{\beta_1}}$ and $\phi_{p_{\beta_2}}$ need to be 0 for every vertex $p \in \beta$, as the children bags β_1 and β_2 do not exist. For a leaf bag β, its portals are simply the vertices in the coherent set of β and β_0 (β's parent bag). So if there is a vertex $v \in \beta$ such that $\phi_{v_{\beta_0}} \neq 0$ then the entry $\mathbf{D}[\beta, \Pi_\beta, \Psi_\beta]$ should be set to infinity and we call the entry invalid, for v does not have a copy in bag b_0 and thus is not a portal.

Furthermore, let us consider the edge cost. To do this, we need to define a flow network. If a vertex $v \in \beta$ is actually $\tilde{v}$, then it has a demand 1 and is a potential facility. On the other hand, if a vertex $v \in \beta$ is not $\tilde{v}$ and $\Pi_\beta^v = \text{TRUE}$, then the entry $\mathbf{D}[\beta, \Pi_\beta, \Psi_\beta]$ is invalid, as we can only open the copy $\tilde{v}$ as an airport. In the perspective within bag β, each portal p can be a source or sink depending on $\Psi_\beta^p = \big(\phi_{p_{\beta_0}}, \phi_{p_{\beta_1}}, \phi_{p_{\beta_2}}\big)$. For instance, if vertex $v \in \beta$ receives x units of flow outside of leaf bag β (we have $\phi_{v_{\beta_0}} = +x$), then inside bag β, it must be a source of $x + 1$ units of flow, where the extra one unit is its own demand. This is called a source portal for bag β. A sink portal is defined similarly.

As in the case of trees, whether or not $\tilde{v}$ is opened as a facility, it has one unit of flow to be sent to an airport or a sink portal. When $\tilde{v}$ is opened, it is capable of absorbing k units of flow. On the other hand, if v is not $\tilde{v}$ or a sink/source portal, then it is neither a source nor a sink inside β. For each of these $\tilde{v}$'s in the bag β, the Boolean variable Π_β^v depicts whether it is opened or not opened. From a portal configuration Ψ_β, we are given the portals together with the information of the flows going through them, we can compute the min-cost flow $\mathcal{F}_\beta(\Pi_\beta, \Psi_\beta)$ in β. For each Π_β and Ψ_β, we store the cost of the flow $\mathcal{F}_\beta(\Pi_\beta, \Psi_\beta)$, plus the cost of the opened facilities into $\mathbf{D}[\beta, \Pi_\beta, \Psi_\beta]$. If there is no valid flow given Ψ_β, or some copy $v \neq \tilde{v}$ is opened by vector Π_β, then we set $\mathbf{D}[\beta, \Pi_\beta, \Psi_\beta]$ to $+\infty$. The situations leading to an invalid flow are similar to the case of trees discussed in the previous section. Since we assume that T is binary, we can make sure that a non-leaf bag β has two children β_1 and β_2. The portals of β are the union of the coherent sets of β and β's parent bag β_0, that of β and β_1, and that of β and β_2. We calculate the min-cost flow $\mathcal{F}_\beta(\Pi_\beta, \Psi_\beta)$ in β according to Π_β and Ψ_β, the same way as the base case. Let $\lambda(\beta, \Pi_\beta, \Psi_\beta)$ denote the sum of the cost of the flow network $\mathcal{F}_\beta(\Pi_\beta, \Psi_\beta)$, and the cost of the airports opened by Π_β. Define

$$\mathbf{D}[\beta, \Pi_\beta, \psi_\beta] = \lambda(\beta, \Pi_\beta, \Psi_\beta) + \mathbf{D}[\beta_1, \Pi_{\beta_1}, \Psi_{\beta_1}] + \mathbf{D}[\beta_2, \Pi_{\beta_2}, \Psi_{\beta_2}]$$

where Ψ_β, Ψ_{β_1} and Ψ_{β_2} are *compatible*. We say a pair of portal configurations Ψ_β and Ψ_{β_i} are compatible if the configurations of all the portals (i.e., all the vertices in the coherent set of β and β_i) therein can match up with those of their copies. For instance, if the vertex $p \in \beta$ has $\phi_{p_{\beta_1}} = +f$ then at the copy $p_{\beta_1} \in \beta_1$ we must have $\phi_p = -f$.

For consistency check, as mentioned above, at each bag, we check if the flow is valid and if the portal configurations between the current bag and its child (or two children) match.

Note that we do not need to worry about the decisions on opening airports at each bag contradicting each other, since only one copy of each vertex $v \in V$, namely the copy $\tilde{v}$, can be opened as an airport, and such a copy only exists in one of the bags.

4.2.1 Algorithm Efficiency

For each bag β, there are $O(2^{\omega+1})$ possibilities for Π_β and $n^{O(\omega)}$ possibilities for Ψ_β. Assume we calculate the table from the bottom up. For each entry of $\mathbf{D}$, it needs to do the consistency check and then find the min-cost flow, which takes $O(\omega^3)$. Thus, the runtime to fill the table is in $n^{O(\omega)}2^{O(\omega)} = (2n)^{O(\omega)}$.

4.3 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. [8]) one can assume that Δ is bounded by polynomial in n at a loss of $(1+\epsilon)$ on optimum solution.

4.4 Extensions to Other Graph Metrics

We use the following lemma by [21] about embedding graphs of doubling dimension D into a graph with treewidth $\omega \leq 2^{O(D)} \left\lceil \left(\frac{4D\log\Delta}{\varepsilon}\right)^D \right\rceil$.

Lemma 4.2. *(Theorem 9 in [21]) 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*

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

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

Theorem 6. *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 $n^{O\left(D^D \log^D n/\varepsilon^D\right)}$.*

Proof. We use the algorithm A of AR' for graphs with bounded treewidth as a subroutine. We see this follows from Lemma 4.2. Essentially, we embed G into a host graph H with treewidth $\omega \leq 2^{O(D)} \left\lceil \left(\frac{4D\log\Delta}{\varepsilon}\right)^D \right\rceil$ and obtain a solution OPT_H by solving AR' on H using algorithm A. Then we lift OPT_H back to G. Thus, simply plug $\omega \leq 2^{O(D)} \left\lceil \left(\frac{4D\log\Delta}{\varepsilon}\right)^D \right\rceil$ into the runtime of the algorithm A, which is $(2n)^{O(\omega)}$, and we obtain an algorithm of runtime $(2n)^{O\left(2^{O(D)}\left\lceil\left(\frac{4D\log\Delta}{\varepsilon}\right)^D\right\rceil\right)}$. Since we have assumed that the aspect ratio Δ is polynomial in n, the runtime is $n^{O\left(D^D\log^D n/\varepsilon^D\right)}$, which means the algorithm is therefore a QPTAS. □

We introduce the following lemma proposed by Feldmann et al. [22] 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 4.3. *(Theorem 1.3 in [22]) 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 [8] to get the following result.

Theorem 7. *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 $n^{O\left(\log^{\log^2\left(\frac{W}{\varepsilon\lambda}\right)\cdot\frac{1}{\lambda}} n\right)}$.*

Proof. Again, we reduce the problem to graphs with bounded treewidth (via embedding) and use the algorithm A of AR$'$ as a subroutine. This follows from Lemma 4.3. Essentially, we embed G into a host graph H with treewidth $\omega \in (\log\Delta)^{O\left(\log^2\left(\frac{W}{\varepsilon\lambda}\right)/\lambda\right)}$ and obtain a solution OPT_H by solving AR$'$ on H using algorithm A. Then we lift OPT_H back to G. Thus, simply plug $\omega \in (\log\Delta)^{O\left(\log^2\left(\frac{W}{\varepsilon\lambda}\right)/\lambda\right)}$ into the runtime of the algorithm A, which is $(2n)^{O(\omega)}$. Since we have assumed that the aspect ratio Δ is polynomial in n, the runtime is $n^{O\left(\log^{\log^2\left(\frac{W}{\varepsilon\lambda}\right)\cdot\frac{1}{\lambda}} n\right)}$, which means the algorithm is therefore a QPTAS. □

We introduce the following lemma proposed by [23] 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 4.4. *(Theorem 1.1 in [23]) 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 8. *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 $n^{O_K\left(\log^8 n\cdot(\log n+\log(1/\varepsilon))^5/\varepsilon\right)}$.*

Proof. Embed the given graph G using Lemma 4.4 into host graph H with treewidth

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

and obtain a solution OPT_H by solving AR' on H using the algorithm for bounded treewidth. Then we lift OPT_H back to G. Thus, simply plug this bound on the treewidth into the runtime of the algorithm A, which is $(2n)^{O(\omega)}$, and we obtain an algorithm of runtime $n^{O_K\left(\log^8 n \cdot (\log n + \log(1/\varepsilon))^5/\varepsilon\right)}$, as we have assumed that the aspect ratio Δ is polynomial in n. The algorithm is therefore a QPTAS. $\square$

Theorems 6, 7, and 8 imply Corollary 1.1.

5 Proof of Theorem 4

Given an instance G for AR, without loss of generality, we assume the set of cities that need to be connected to an airport $\mathcal{C}$ and potential airports (facilities) $\mathcal{F}$ are disjoint.[4]

First note that any optimum solution OPT can be turned into a solution of cost at most $2 \cdot \mathrm{OPT}$ where each cluster is a cycle by simply doubling the edges of each cluster and short-cutting the resulting Eulerian walk into a cycle. So if we define CYCLE-AR to be the variant of AR where the goal is to find a min-cost set of cycles each having at most k vertices, using edges in the graph, such that each cycle C has one opened facility, then the optimum solution of CYCLE-AR is within factor 2 of optimum of AR on the same graph.

Given an instance of the general metric AR problem, say graph G, we can transform it into another instance G' for the $0/+\infty$ CYCLE-AR problem in the following way: For any potential facility v, move one half of its opening cost f_v onto each of its incident edges, that is, we update the cost of each edge e incident to v from c_e to $c_e + \frac{1}{2} f_v$ and set v's opening cost to zero.

Lemma 5.1. *The above transformation preserves the triangle inequality.*

Proof. Consider any triangle in the graph G, there are three types as depicted below. Say there are three vertices A, B and C in the triangle, and the cost of the edges are $c_{AB} = c, c_{AC} = b$ and $c_{BC} = a$. Denote their respective opening cost as f_A, f_B and f_C. Since G is metric, we know the following inequalities are true

$$a + b \geq c$$
$$a + c \geq b$$
$$b + c \geq a$$

There are three scenarios that the triangle can become in the transformed graph G' (via the transformation mentioned above), which depends on how many vertices in the triangle are facilities.

[4]If they are not, say a vertex v is a city as well as a potential airport with opening cost f, then we make v only a city (i.e. $v \in \mathcal{C}$ and $v \notin \mathcal{F}$), and add another vertex v' to the set $\mathcal{F}$ that overlaps/superimposes on v, where v' has opening cost f.

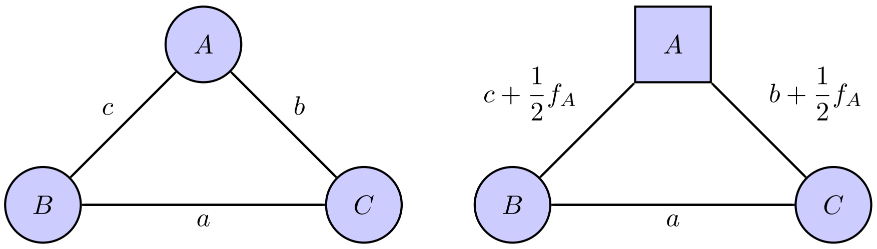


Fig. 6: On the left is the original triangle in G. On the right is the scenario in G' where one of the vertex is a facility.

Say only one of the vertices is a facility. Without loss of generality, assume A is the facility. It is easy to see the following inequalities hold.

$$\begin{aligned}
a+\left(b+\frac{1}{2}f_A\right) &\geq c+\frac{1}{2}f_A\\
a+\left(c+\frac{1}{2}f_A\right) &\geq b+\frac{1}{2}f_A\\
\left(b+\frac{1}{2}f_A\right)+\left(c+\frac{1}{2}f_A\right) &\geq a
\end{aligned}$$

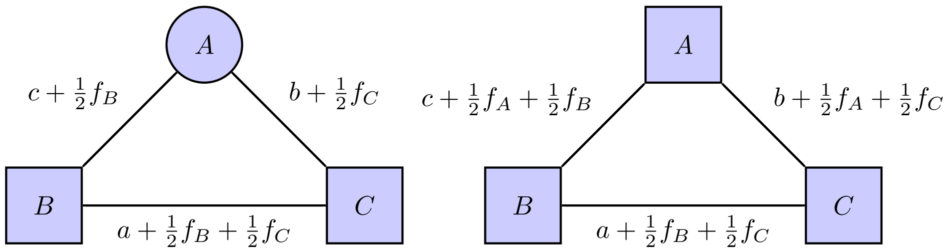

A
$c+\frac{1}{2}f_B$
$b+\frac{1}{2}f_C$
B
C
$a+\frac{1}{2}f_B+\frac{1}{2}f_C$
A
$c+\frac{1}{2}f_A+\frac{1}{2}f_B$
$b+\frac{1}{2}f_A+\frac{1}{2}f_C$
B
C
$a+\frac{1}{2}f_B+\frac{1}{2}f_C$

Fig. 7: On the left is the scenario in G' where two of the vertex is a facility. On the right is the scenario in G' where all of the vertex is a facility.

Say two of the vertices are facilities. Without loss of generality, assume B and C are the facilities. It is easy to see the following inequalities hold.

$$\begin{aligned}
\left(a+\frac{1}{2}f_B+\frac{1}{2}f_C\right)+\left(b+\frac{1}{2}f_C\right) &\geq c+\frac{1}{2}f_B\\
\left(a+\frac{1}{2}f_B+\frac{1}{2}f_C\right)+\left(c+\frac{1}{2}f_B\right) &\geq b+\frac{1}{2}f_C\\
\left(b+\frac{1}{2}f_C\right)+\left(c+\frac{1}{2}f_B\right) &\geq a+\frac{1}{2}f_B+\frac{1}{2}f_C
\end{aligned}$$

Say all of the vertices are facilities. It is easy to see the following inequalities hold.

$$\left(a+\frac{1}{2}f_B+\frac{1}{2}f_C\right)+\left(b+\frac{1}{2}f_A+\frac{1}{2}f_C\right)\geq c+\frac{1}{2}f_A+\frac{1}{2}f_B$$
$$\left(a+\frac{1}{2}f_B+\frac{1}{2}f_C\right)+\left(c+\frac{1}{2}f_A+\frac{1}{2}f_B\right)\geq b+\frac{1}{2}f_A+\frac{1}{2}f_C$$
$$\left(b+\frac{1}{2}f_A+\frac{1}{2}f_C\right)+\left(c+\frac{1}{2}f_A+\frac{1}{2}f_B\right)\geq a+\frac{1}{2}f_B+\frac{1}{2}f_C$$

We have shown any arbitrary triangle in the transformed graph G' still obeys the triangle inequality. Thus the transformation does not break metric. $\square$

As mentioned above, we know we can transform any solution to the general metric AR problem on G into a solution to the $0/+\infty$ CYCLE-AR problem on G' with no more than twice the original cost. Let OPT and OPT$'$ denote the cost of the optimal solution to the general metric AR problem on G and the $0/+\infty$ CYCLE-AR problem on G' respectively. It follows that OPT$' \leq 2$OPT.

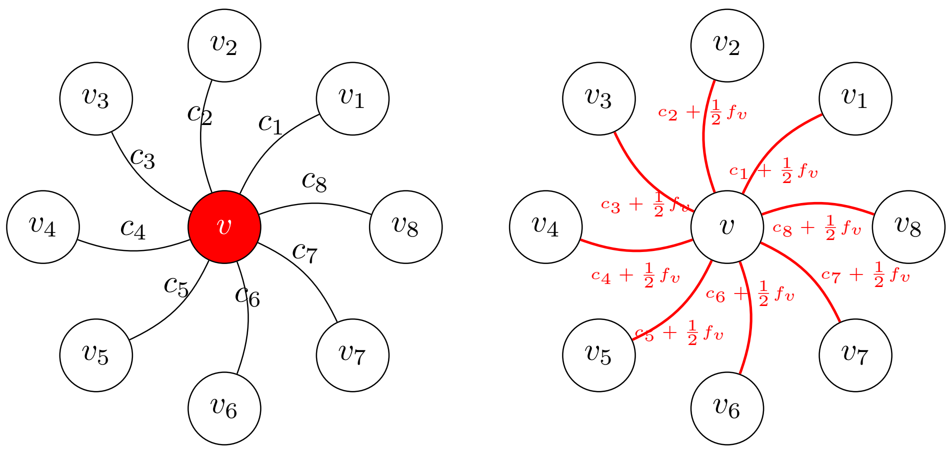


Fig. 8: Incident edges of each potential facility before and after changes

In summary, for every input graph G to the general metric AR problem, we transform the graph to G' (in the manner mentioned above) to make it a $0/+\infty$ AR instance, and then solve the CYCLE-AR problem on G' to obtain a solution Ψ. It is easy to see that, if Ψ is a solution to CYCLE-AR on graph G' with cost αOPT$'$, then Ψ is a solution to CYCLE-AR on graph G with the same cost, i.e. $2\alpha \cdot$ OPT, as OPT$' \leq 2$OPT. Since a solution to CYCLE-AR on graph G can be easily transformed into a solution to general metric AR on the same graph (simply delete an edge from each cycle in the solution), we know Ψ can be transformed into a solution to general metric AR on graph G with cost $2\alpha \cdot$ OPT.

6 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 have also talked with Mohsen Rezapour for some initial discussions.

References

[1] Adamaszek, A., Antoniadis, A., Mömke, T.: Airports and Railways: Facility Location Meets Network Design. In: Ollinger, N., Vollmer, H. (eds.) 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016). Leibniz International Proceedings in Informatics (LIPIcs), vol. 47, pp. 6–1614. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2016). https://doi.org/10.4230/LIPIcs.STACS.2016.6 . http://drops.dagstuhl.de/opus/volltexte/2016/5707

[2] Adamaszek, A., Antoniadis, A., Kumar, A., Mömke, T.: Approximating Airports and Railways. In: Niedermeier, R., Vallée, B. (eds.) 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 96, pp. 5–1513. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.STACS.2018.5 . http://drops.dagstuhl.de/opus/volltexte/2018/8518

[3] Kao, M.-J.: Improved LP-based approximation algorithms for facility location with hard capacities. arXiv preprint arXiv:2102.06613 (2021)

[4] Miao, R., Yuan, J.: A note on LP-based approximation algorithms for capacitated facility location problem. Theoretical Computer Science **932**, 31–40 (2022) https://doi.org/10.1016/j.tcs.2022.08.002

[5] Blauth, J., Traub, V., Vygen, J.: Improving the approximation ratio for capacitated vehicle routing. Mathematical Programming **197**(2), 451–497 (2023) https://doi.org/10.1007/S10107-022-01841-4

[6] Das, A., Mathieu, C.: A Quasi-polynomial Time Approximation Scheme for Euclidean Capacitated Vehicle Routing. In: Charikar, M. (ed.) Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010, pp. 390–403. SIAM, ??? (2010). https://doi.org/10.1137/1.9781611973075.33 . https://doi.org/10.1137/1.9781611973075.33

[7] Friggstad, Z., Mousavi, R., Rahgoshay, M., Salavatipour, M.R.: Improved Approximations for Capacitated Vehicle Routing with Unsplittable Client Demands. In: Aardal, K.I., Sanità, L. (eds.) Integer Programming and Combinatorial Optimization - 23rd International Conference, IPCO 2022, Eindhoven, The Netherlands, June 27-29, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13265, pp. 251–261. Springer, ??? (2022). https://doi.org/10.1007/978-3-031-06901-7_19 . https://doi.org/10.1007/978-3-031-06901-7_19

[8] Jayaprakash, A., Salavatipour, M.R.: Approximation Schemes for Capacitated Vehicle Routing on Graphs of Bounded Treewidth, Bounded Doubling, or Highway Dimension. ACM Transactions on Algorithms **19**(2) (2023) https://doi.org/10.1145/3582500

[9] Traub, V., Tröbst, T.: A Fast $(2 + \frac{2}{7})$-Approximation Algorithm for Capacitated Cycle Covering. Mathematical Programming **192**(1), 497–518 (2022) https://doi.org/10.1007/S10107-021-01678-3

[10] Khani, M.R., Salavatipour, M.R.: Improved Approximation Algorithms for the Min-max Tree Cover and Bounded Tree Cover Problems. Algorithmica **69**(2), 443–460 (2014) https://doi.org/10.1007/S00453-012-9740-5

[11] Yu, W., Liu, Z., Bao, X.: New Approximation Algorithms for the Minimum Cycle Cover Problem. Theoretical Computer Science **793**, 44–58 (2019) https://doi.org/10.1016/J.TCS.2019.04.009

[12] Jothi, R., Raghavachari, B.: Approximation Algorithms for the Capacitated Minimum Spanning Tree Problem and Its Variants in Network Design. ACM Transactions on Algorithms **1**(2), 265–282 (2005) https://doi.org/10.1145/1103963.1103967

[13] Ravi, R., Sinha, A.: Approximation Algorithms for Problems Combining Facility Location and Network Design. Operations Research **54**(1), 73–81 (2006)

[14] Maßberg, J., Vygen, J.: Approximation Algorithms for a Facility Location Problem with Service Capacities. ACM Transactions on Algorithms **4**(4) (2008) https://doi.org/10.1145/1383369.1383381

[15] Edmonds, J.: Matroid Intersection. In: Hammer, P.L., Johnson, E.L., Korte, B.H. (eds.) Discrete Optimization I. Annals of Discrete Mathematics, vol. 4, pp. 39–49. Elsevier, ??? (1979). https://doi.org/10.1016/S0167-5060(08)70817-3 . https://www.sciencedirect.com/science/article/pii/S0167506008708173

[16] Bodlaender, H.L., Hagerup, T.: Parallel Algorithms with Optimal Speedup for Bounded Treewidth. SIAM Journal on Computing **27**(6), 1725–1746 (1998) https://doi.org/10.1137/S0097539795289859 https://doi.org/10.1137/S0097539795289859

[17] Tian, L.: Approximation Schemes for the Airport and Railway Problem. Master's thesis, Department of Computing Science, Faculty of Science, University of Alberta (Fall 2023)

[18] Mathieu, C., Zhou, H.: A PTAS for Capacitated Vehicle Routing on Trees. ACM Transactions on Algorithms **19**(2), 17–11728 (2023) https://doi.org/10.1145/3575799

[19] Bell, E.T.: Exponential polynomials. Annals of Mathematics, 258–277 (1934)

[20] Bell, E.T.: The iterated exponential integers. Annals of Mathematics, 539–557 (1938)

[21] Talwar, K.: Bypassing the Embedding: Algorithms for Low Dimensional Metrics. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing. STOC '04, pp. 281–290. Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/1007352.1007399 . https://doi.org/10.1145/1007352.1007399

[22] Feldmann, A.E., Fung, W.S., Könemann, J., Post, I.: A $(1+\epsilon)$-Embedding of Low Highway Dimension Graphs into Bounded Treewidth Graphs. SIAM Journal on Computing **47**(4), 1667–1704 (2018) https://doi.org/10.1137/16m1067196

[23] Cohen-Addad, V., Le, H., Pilipczuk, M., Pilipczuk, M.: Planar and Minor-Free Metrics Embed into Metrics of Polylogarithmic Treewidth with Expected Multiplicative Distortion Arbitrarily Close to 1 (2023)