Approximation Schemes for the Airport and Railway Problem

by

Lijiangnan Tian

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this thesis, we present approximation schemes for the airport and railway problem (AR) on several classes of graphs. The AR problem, introduced by Adamaszek *et al.* [4], is a combination of the capacitated facility location problem (CFL) and the network design problem. An AR instance comprises a set of points, which are called cities, residing in some metrics, each of which is associated with a non-negative cost and a number k, which represent respectively the cost of establishing an airport in the corresponding city, 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 the strict setting, the network is divided into components, each with an open airport and a maximum of k cities. In a more relaxed setting of the problem, the railway paths heading to different airports are allowed to share edges, either the cost of each city-to-airport path is paid separately (which is the setting we used for AR with a uniform opening cost), or the railways have a universal edge capacity k and we are allowed to use parallel edges (which is called AR$'$ by Adamaszek *et al.* [3]). Adamaszek *et al.* [4] presented a PTAS for AR in the two-dimensional Euclidean metric $\mathbb{R}^2$ with a uniform opening cost in the strict setting. Adamaszek *et al.* [3] presented a bicriteria $\frac{4}{3}\left(2+\frac{1}{\alpha}\right)$-approximation algorithm for AR with non-uniform opening costs in the general metric, with the airport capacity $(1+\alpha)k$ where $0 < \alpha \leq 1$, a $\left(2 + \frac{k}{k-1} + \varepsilon\right)$-approximation algorithm and a bicriteria QPTAS for the same problem in the Euclidean plane $\mathbb{R}^2$. In this work, we give a Quasi-Polynomial-Time Approximation Scheme (QPTAS) for AR with a uniform opening cost in graphs of bounded treewidth, QPTAS for AR$'$ in graphs of bounded doubling dimensions, graphs of bounded highway dimensions and planar graphs, and quasi-polynomial-time $\left(2 + \frac{k}{k-1}\right) \cdot (1+\varepsilon)$-approximation for AR with non-uniform opening costs for the aforementioned metrics. For general metrics, we give a 2-approximation for AR with a uniform opening cost in the strict setting of the problem, and an $O(\log n)$-approximation for AR with non-uniform opening cost.

To my parents

Real knowledge is to know the extent of one's ignorance.

– Confucius (551–479 BC).

Wonder is the beginning of wisdom.

– Socrates (469–399 BC).

Acknowledgements

First and foremost, I would like to extend my gratitude to my supervisor, Professor Mohammad Reza Salavatipour, for his unwavering support and guidance throughout my graduate research journey. Mohammad's professionalism, wealth of knowledge, and patience in addressing my inquiries and revising my thesis write-ups have been invaluable. From my early years as an undergraduate student, I have known him as not only a mentor but also as a source of inspiration, and I am truly fortunate to have had the opportunity to work under his supervision.

I want to thank Assistant Professor Mohsen Rezapour as well, for his contributions to my academic and personal growth. Mohsen's casual chats, engaging discussions on the intricacies of my thesis, and insightful perspectives on the problems I tackled have been instrumental in shaping my understanding of the subject matter in the beginning phase of my research as well as my thesis presentation. Both Mohammad and Mohsen have consistently illuminated the path for me, offering valuable guidance on my academic writing and presenting skills, and pointing me in the right direction when needed.

I am also grateful to all members of our Algorithmics Research Laboratory (also known as the Theory Group) at the Computing Science Centre, along with others, for their enduring friendship and riveting discourses during my graduate program in Alberta. Beyond things related to study, research and teaching assistants, we often discussed regional and cultural matters. Our interactions have been a source of joy, adding depth and diversity to my academic journey and my worldview. I appreciate the intellectual camaraderie and the collective enrichment we've experienced together.

I am thankful to the members of my examining committee—Associate Professor Zachary "Zac" Friggstad, Professor Martin Müller and Professor Osmar R. Zaïane (the chair)—for their time, effort, and constructive feedback, which have significantly contributed to the refinement of this thesis. In particular, I want to thank Zac for his comments that led to a huge improvement in the constant-factor approximation of Section 2.3, in terms of decreasing the approximation factor and simplifying the algorithm. Additionally, I express my gratitude to the Department of Computing Science for their financial support throughout my study. Their commitment to fostering academic excellence has played a crucial role in the successful completion of this thesis, and I am truly thankful for the opportunities they have provided.

Finally, I express my deepest appreciation to my parents, to whom this thesis is dedicated. Their unconditional support, encouragement, and belief in my abilities have been the cornerstone of my academic journey.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problems Considered

In this thesis, we will discuss the Airport and Railway problem (AR), also Airport Routing, which is first defined in [3], [4]. We will first define the problem and then list some of its variants later in this chapter.

1.1.1 Problem Definition

In the problem of Airports and Railways (AR), we are **given** a complete graph $G = (V, E)$ embedded in some metric space (for instance the Euclidean plane), with two cost functions $\alpha : V \to \mathbb{R}_{\geq 0}$ for opening facilities (also known as *airports*) at vertices (also known as *cities*) and $\rho : E \to \mathbb{R}_{\geq 0}$ 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 **task** is to find the set of vertices $A \subseteq V$ and the set of edges $R \subseteq E$, which form a forest (as each airport *will* serve the city it is at), such that each component (also known as a *cluster*) thereof contains only one airport and has at most k vertices therein, including the airport. Our **goal** is to minimise the total cost

$$C = \sum_{v \in A} \alpha_v + \sum_{e \in R} \rho_e.$$

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. In a more relaxed setting, each edge is allowed to be used by multiple clusters each of which needs to pay the cost of the edge separately. To differentiate the relaxed setting from the above definition, we will call the latter one the strict setting when necessary. Note that in the relaxed setting, each connected component in a feasible solution may contain multiple clusters and the total cost that we want to minimise is

$$\sum_{v \in A} \alpha_v + \sum_{e \in R} \rho_e \cdot \phi(e)$$

where $\phi(e)$ is the number of clusters using the edge e.

1.2 Preliminaries

First, we provide a number of essential terminologies, definitions and notations that are indispensable to understanding our problems and this thesis. The definitions listed here are referenced and adapted from [11], [12], [14], [19], [22], [23], [26], [29]–[33].

1.2.1 Graphs

A **graph** G consists of a **vertex set** $V(G)$ and an **edge set** $E(G)$, where each edge is an unordered pair of distinct elements from $V(G)$, which are called its **endpoints**. When u, v are endpoints of edge e, we say u, v are **neighbours** or **adjacent**, and say edge e is an **incident** edge of u and v, or e is incident on u and v.

A **directed graph** (or **digraph**) is a graph where each edge is an ordered pair of vertices. An edge (also known as an arc) from vertex u to vertex v is denoted by (u, v). An **undirected graph** is a graph where each edge is an unordered pair of vertices.

The **degree** of a vertex in an undirected graph is the number of edges incident on it. A vertex whose degree is zero is **isolated**. In a directed graph, the **out-degree** of a vertex is the number of edges leaving it, and the **in-degree** of a vertex is the number of edges entering it. The degree of a vertex in a directed graph is its in-degree plus its out-degree.

A **loop** is an edge whose endpoints are equal. **Multiple edges** (or **parallel edges**) are edges having the same pair of endpoints. This thesis only discusses situations where the input graph is an undirected graph.

A **path** is a simple graph whose vertices can be ordered so that two vertices are adjacent if and only if they are consecutive in the list. A **cycle** is a graph with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are adjacent if and only if they appear consecutively along the circle.

A **bipartite graph** is a graph $G = (V, E)$ in which the vertex set can be partitioned into $V = B \cup W$, where B and W are disjoint and all edges in E go between B and W.

A **planar graph** is an undirected graph that can be drawn in the plane with no edges crossing.

A **subgraph** of graph G is a graph H that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, and the assignment of endpoints to edges in H is the same as in G. We then write $H \subseteq G$ and say that "G **contains** H."

There is a uv-path in graph G if G contains a path starting from u and ending at v. If there is a path P from u to v, we say that v is reachable from u via P. A graph G is **connected** if it has a uv-path for each pair of vertices $u, v \in V(G)$, otherwise, it is **disconnnected**. A directed graph is **strongly connected** if every two vertices are reachable from each other.

A **walk** in a graph is an alternating sequence of vertices and edges, whose first and last elements are vertices, and such that each edge joins the vertices immediately preceding it and succeeding it in the sequence. A **tour** is a closed walk with no repeated edges.

The process of **contracting** an edge e with ends v and u means deleting e and identifying v and u. A graph H is a **minor** of a graph G if it can be obtained from a subgraph of G by contracting edges sequentially.

A **complete graph** is a graph whose vertices are pairwise adjacent. A graph with no cycle is called **acyclic**. A **tree** is a connected acyclic graph. A **rooted tree** has one vertex r chosen as the **root**. For each vertex v, let $P(v)$ be the unique vr-path in tree T. The **parent** of v is its neighbour on $P(v)$; its **children** are its other neighbours in T. Its **ancestors** are the vertices in the set $P(v) \setminus v$. Its **descendants** are the vertices u such that $P(u)$ contains v. Again, the degree of each vertex is the number of vertices adjacent to it. The **leaves** are the vertices with degree equal to one. A **binary tree** is a rooted tree where each vertex has at most two children, and each child of a vertex is designated as its **left child** or **right child**.

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

- (Vertex Coverage) For each vertex v in G, it must be included in at least one bag of T.
- (Edge Coverage) For each edge uv in G, the pair of vertices $u, v \in V$ must be included in at least one bag of T.
- (Coherence) 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 the largest bag therein minus one. The **treewidth** of a graph G is the minimum number Q such that G has a tree decomposition where the cardinality of the largest bag therein is $Q + 1$.

1.2.2 Metrics

A **metric space** (X, d) consists of a set of points X and a distance function $d : X \times X \to \mathbb{R}_{\geq 0}$ which satisfies the following properties:

1. For every $x, y \in X$, $d(x, y) \geq 0$.
2. For every $x \in X$, $d(x, x) = 0$.
3. (Symmetry) For every $x, y \in X$, $d(x, y) = d(y, x)$.

4. (Triangle inequality) For every $x, y, z \in X$, $d(x, z) \leq d(x, y) + d(y, z)$.

A **weighted graph** is a graph with numerical labels on edges, denoted by $w(e)$ for an edge e (and $w(e)$ is call the **weight** of edge e). In this thesis, we only discuss graphs with non-negative edge weights. Any metric (X, d) can be converted into a complete weighted graph G such that $V(G) = X$ and $w(uv) = d(u, v)$ (i.e. the weights represent distances). The length of a path is defined to be the sum of weights of all of its edges. Let $d_G(u, v)$ denote the length of the shortest uv-path in G, and if u, v are not connected in G, we define $d_G(u, v) = +\infty$. Define $d_{\max} = \max_{u,v \in V: u \neq v} d_G(u, v)$, $d_{\min} = \min_{u,v \in V: u \neq v} d_G(u, v)$, and aspect ratio $\Delta = \frac{d_{\max}}{d_{\min}}$.

A metric space (X, d) is called a **tree metric** if there is an edge-weighted tree $T = (V, E)$ such that $X \subseteq V$ and, for every pair of points $x, y \in X$, the distance $d(x, y)$ in the metric space is equal to the distance $d_T(x, y)$ in the tree.

A metric space (X, d) is said to be **doubling** if there exists some constant $\kappa > 0$ such that for any point $x \in X$ and radius $r > 0$, one can cover the ball $B_r(x) = \{x' \in X \mid d(x, x') \leq r\}$ with the union of at most 2^κ balls of radius $\frac{r}{2}$, where κ is referred to as the **doubling dimension** of the metric. [17]

The **highway dimension** of a weighted graph $G = (V, E)$ is the smallest integer ξ such that for some universal constant $c \geq 4$, for every number $r \in \mathbb{R}_{>0}$ and every vertex $v \in V$, there are at most ξ vertices in the ball $B_{cr}(v)$ of radius $c{\cdot}r$ hitting all shortest paths of length more than r that lie completely in $B_{cr}(v)$. Additionally, we define $\lambda = c - 4$ as the **violation**. [1], [2], [13]

1.2.3 Optimization Problems and Approximation Algorithms

The definitions of this section also come from [16], [24], [27], [32].

A **decision problem** is a problem to be answered with either "Yes" or "No". The class **P** consists of those decision problems that are solvable in polynomial time. More specifically, they are problems that can be solved in $O(n^c)$ time for some constant c, where n is the size of the input to the problem. **NP** is the class of decision problems that, given a proposed solution, can be verified in polynomial time.

A problem is **NP**-hard if all problems in **NP** are polynomial time reducible to it. A problem is **NP**-complete it is **NP**-hard and it is in **NP**.

An **NP-optimization problem** Π consists of:

- a set D_Π of valid **instances**, recognisable in polynomial time,
- each instance $I \in D_\Pi$ has a set $S_\Pi(I)$ of **feasible solutions**,
- every solution $s \in S_\Pi(I)$ is of length polynomially bounded in $|I|$,
- there is a polynomial-time algorithm that, given a pair (I, s), decides $s \in S_\Pi(I)$,
- there is a polynomial-time computable **objective function** f_Π that assigns a non-negative rational number to each pair (I, s),
- finally, Π is specified to be either a minimization or a maximization problem.

An **optimal solution** for an instance of a minimization (maximization) problem is a feasible solution that achieves the smallest (largest) objective function value. We use $\mathrm{OPT}(I)$ to denote the objective function value of an optimal solution to instance I. We use $\textit{OPT}(I)$ to denote its cost.

With every **NP**-optimization problem, one can naturally associate a decision problem by giving a bound B on the optimal solution. Thus, the decision version of an **NP**-optimization problem Π consists of pairs (I, B) with I being an instance of Π and B being a rational number. Given (I, B), we ask whether there is some $s \in S_\Pi(I)$ such that $f_\Pi(s) \geq B$ for a maximization problem ($f_\Pi(s) \leq B$ for a minimization problem).

An α-**approximation algorithm** for an optimization problem is a polynomial-time algorithm that, for all instances of the problem, produces a solution whose value is within a factor of α of the value of an optimal solution. For an α-approximation algorithm, we will call α the performance guarantee of the algorithm. In the literature, it is also often called the **approximation ratio** or approximation factor of the algorithm. Conventionally, we have $\alpha > 1$ for minimization problems, whereas $\alpha < 1$ for maximization problems.

A **polynomial-time approximation scheme (PTAS)** is a family of polynomial time algorithms $\{A_\varepsilon\}$, where there is an algorithm for each $\varepsilon > 0$, such that A_ε is a $(1+\varepsilon)$-approximation algorithm (for minimization problems) or a $(1-\varepsilon)$-approximation algorithm (for maximization problems). That is, for a fixed $\varepsilon > 0$, the running time is in $n^{O(1)}$.

A **quasi-polynomial-time approximation scheme (QPTAS)** is an approximation scheme such that, given a fixed $\epsilon > 0$, the running time of every A_ϵ is bounded by some **polylogarithmic** function in n, i.e., the runtime is in $n^{\log^{O(1)} n}$.

A maximization problem (or a minimization problem) L is said to be in **APX** (short for *approximable*) if L is an **NP**-optimization problem and there exists a polynomial-time approximation algorithm A such that for all instances I of L, there exists $c > 1$ such that A is a c-approximation for L (i.e. **constant approximation**) [21]:

$$\frac{\mathrm{OPT}(I)}{A(I)} \le c, \ \left(\text{or } \frac{A(I)}{\mathrm{OPT}(I)} \le c\right).$$

If $\mathbf{P} \neq \mathbf{NP}$, then no **APX**-complete problem admits a PTAS.

1.2.4 Metric Embeddings

In this section, we adapt definitions from [9], [22]. The goal of metric embedding is to map graphs into more special and restricted classes of graphs in order to make the problem more tractable.

A mapping $f : X \to Y$, where X is a metric space with a metric ρ and Y is a metric space with a metric σ, is called an **isometric embedding** if it preserves distances, i.e., if $\sigma(f(x), f(y)) = \rho(x, y)$ for all points $x, y \in X$.

A mapping $f : X \to Y$, where X is a metric space with a metric ρ and Y is a metric space with a metric σ, is called a D**-embedding of metric spaces**, where $D \ge 1$ is a real number, if there exists a number $r > 0$ (scaling factor) such that for all points $x, y \in X$,

$$r \cdot \rho(x,y) \leq \sigma(f(x), f(y)) \leq D \cdot r \cdot \rho(x,y).$$

The infimum of the numbers D such that f is a D-embedding is called the **distortion** of f.

1.2.5 Concentration Inequalities

The following theorem is based on [23]. **Chernoff bound** gives exponentially decreasing tail bounds for the sum of independent bounded random variables. Roughly speaking, a tail bound bounds the tail distribution, the probability that the value of a certain random variable is far from its expectation [25].

Theorem 1 (Chernoff bound (from [23])) *Let* $Y = \sum_{i=1}^{n} Y_i$ *where*

$$Y_i = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases}$$

and all variables Y_i's are independent. Denote $\mu = \mathbb{E}[Y]$, we have

$$\Pr[Y > 2\mu] \leq e^{-\frac{\mu}{3}} \quad \textit{and} \quad \Pr\left[Y < \frac{\mu}{2}\right] \leq e^{-\frac{\mu}{8}}.$$

1.2.6 Matroids

These definitions are from [20].

A set system $(E, \mathscr{F})$ is a **matroid** if

(M1) $\varnothing \in \mathscr{F}$;

(M2) If $X \subseteq Y \in \mathscr{F}$ then $X \in \mathscr{F}$.

(M3) If $X, Y \in \mathscr{F}$ and $|X| > |Y|$, then there is an $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathscr{F}$.

A matroid $(E, \mathscr{F})$, where E is the set of edges of some undirected graph G and $\mathscr{F} := \{F \subseteq E \mid (V(G), F) \text{ is a forest}\}$, is called the **cycle matroid** of G and denoted by $\mathscr{M}(G)$. A matroid that is the cycle matroid of some graph, which may contain loops, is called a **graphic matroid**.

1.3 Related Work

We begin by providing the definitions of several related problems.

Definition 1 *Capacitated Facility Location* (CFL)*: The input comprises a metric graph $G = (V, E)$, a set of potential facilities $F \subseteq V$ each having a capacity κ, a set of clients $C \subseteq V$ each of which has a demand to be satisfied, a metric cost function c_{ij} representing the assignment cost of serving client j from facility i, and the opening cost of each facility f_i. The goal is to choose a subset of potential facilities to open and assign clients to these open facilities to minimise the total cost, which is the sum of the cost of all the open facilities and the cost of all assignments, such that no facility serves more than κ clients.*

Definition 2 *(Definition from [18]) Capacitated Vehicle Routing Problem* (CVRP)*: Given a graph $G = (V, E)$ with metric edges costs $w(e) \in \mathbb{Z}_{\geq 0}$, a dépôt $r \in V$, and a vehicle of bounded capacity $Q > 0$. The goal is to find the minimum cost collection of tours, each starting from (and ending at) the dépôt and visiting at most Q customers, whose union covers all the customers.*

We define the concept of a flow network.

Definition 3 *(Definition from [11]) Let $G = (V, E)$ be a **flow network** with a capacity function $c : E \to \mathbb{R}_+$. Let s be the **source** of the network, and let t be the **sink**. A flow in G is a real-valued function $f : E \to \mathbb{R}$ that satisfies the following two properties:*

Capacity constraint: *For all pairs of vertices $u, v \in V$, we require*

$$0 \leq f(u, v) \leq c(u, v).$$

The flow from one vertex to another must be nonnegative and must not exceed the given capacity.

Flow conservation: *For all vertices $u \in V \setminus \{s, t\}$, we require*

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$

The total flow into a vertex (other than the source or sink) must equal the total flow out of that vertex.

When $(u, v) \notin E$, there can be no flow from u to v, and $f(u, v) = 0$.

We also define bicriteria approximations.

Definition 4 *A (ν, ς)-bicriteria approximation algorithm for* AR *finds a solution where each cluster has size at most $\nu \cdot k$, whose cost is at most ς times the optimal cost of a solution (where each cluster has size at most k), where $\nu, \varsigma > 1$.*

We list here a few notations defined in [3], [4]:

- AR_β, for some constant $\beta > 1$, is the bicriteria version of AR, i.e. AR with *resource augmentation*, where the new airport capacity is $\beta \cdot k$.
- AR^∞ is a relaxed version where the airport capacity is dropped, or equivalently, set to infinity: $k = +\infty$.
- 1AR means the airport-opening cost function $\alpha : V \to \mathbb{R}_{\geq 0}$ is constant. In this thesis, 1AR is called AR with a uniform opening cost.
- AR_F, or just AR, is the general version of the problem, where the result forms a forest. AR_F is related to capacitated facility location (CFL) problem.
- AR_P is a special case of AR where each component is a path with both endpoints having an airport. AR_P is a relaxation of the capacitated vehicle routing problem (CVRP) due to lack of a centralised dépôt.

	General metric	Euclidean plane
Nonuniform airport cost	Bicriteria: $\frac{4}{3} \cdot (2 + \frac{1}{p})$-approximate for AR_{1+p}, $p \in (0, 1]$ (in particular, 4-approximate for AR_2)	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
Uniform airport cost		PTAS for both $1\mathrm{AR}_F$ and $1\mathrm{AR}_P$

Figure 1.1: Summary of previous work of AR

1.4 New Results

Currently, the AR problem has not been studied in the tree metrics, graphs of bounded treewidth, graphs of bounded doubling dimensions, graphs of bounded highway dimensions, minor-free graphs, etc. For AR with a uniform facility opening cost, we propose a QPTAS for trees and graphs of bounded treewidth in the relaxed setting of the problem, as well as a constant approximation for the general metric in the strict setting. For AR with non-uniform facility opening costs, we propose a constant approximation for trees and graphs of bounded treewidth, graphs of bounded doubling dimension, graphs of bounded highway dimension, and minor-free graphs in the strict setting of the problem, as well as an $O(\log n)$-approximation for general metric in the strict setting.

1.5 Assumption on Edge Costs

In this section, similar to [4] and [18], we show how we can assume the input graph $G = (V, E)$ has an aspect ratio Δ that is polynomial in $n = |V|$.

Let OPT denote the optimal solution to AR on G. WLOG, we consider the relaxed setting. Note that for the strict setting of the problem, we can simply set $\phi(e) = 1$ for each edge $e \in R$. The cost of OPT is

$$OPT = \sum_{v \in A} \alpha_v + \sum_{e \in R} \rho_e \cdot \phi(e)$$

where α_v is the facility opening cost of vertex $v \in V$, ρ_e is the cost of the edge $e \in E$, A and R are the set of vertices opened as facilities and the set of edges chosen by OPT respectively, and $\phi(e)$ is the number of times the edge e is used by a cluster in OPT.

We construct a new instance G' with all the costs defined as follows. Firstly, we guess the maximum edge weight $W = \max_{e \in R} \rho_e$ in OPT,[1] and eliminate every edge of cost larger than W. Round up the costs of each edge to the maximum

[1]We can enumerate all $|E|$ possibilities for such a guess.

of its cost and $\frac{\varepsilon}{n^3} \cdot W$, that is, set the new edge cost $\tilde{\rho}_e \leftarrow \max\left\{\rho_e, \frac{\varepsilon W}{n^3}\right\}$. Every edge now has cost at least $\frac{\varepsilon W}{n^3}$. Equivalently, the cost of each edge has increased by at most $\frac{\varepsilon W}{n^3}$.

Note that $\phi(e) \leq n$ since there are at most n clusters in OPT. Let $\textsc{opt}'$ denote the cost of the optimal solution to AR on G'. Using the fact that $|E| \leq n^2$ and $W \leq \textsc{opt}$, we have

$$\begin{aligned}
\textsc{opt}' &\leq \sum_{v\in A} \alpha_v + \sum_{e\in R} \tilde{\rho}_e \cdot \phi(e) \\
&= \left(\sum_{v\in A} \alpha_v + \sum_{e\in R} \rho_e \cdot \phi(e)\right) + \sum_{e\in R} (\tilde{\rho}_e - \rho_e) \cdot \phi(e) \\
&\leq \textsc{opt} + \sum_{e\in R} \frac{\varepsilon W}{n^3} \cdot \phi(e) \\
&\leq \textsc{opt} + \sum_{e\in R} \frac{\varepsilon W}{n^3} \cdot n \\
&\leq \textsc{opt} + n^2 \cdot \frac{\varepsilon W}{n^3} \cdot n \\
&= \textsc{opt} + \varepsilon W \\
&\leq \textsc{opt} + \varepsilon \cdot \textsc{opt} \\
&= (1+\varepsilon)\textsc{opt}.
\end{aligned}$$

As mentioned above, for every edge in G', we have

$$\frac{\varepsilon W}{n^3} \leq \tilde{\rho}_e \leq W.$$

Therefore, the aspect ratio of G' satisfies

$$\Delta \leq \frac{n^3}{\varepsilon}.$$

Chapter 2

AR with a Uniform Airport Cost

In this chapter, we first present a Quasi-Polynomial-Time Approximation Scheme (QPTAS) for the AR problem (in the relaxed setting) on trees that have logarithmic heights. Subsequently, we extend this approach to the class of graphs with bounded treewidth. This implies a QPTAS on trees in general, because graphs with bounded treewidth include trees as special cases. We will also give a constant approximation for general metrics for the strict setting of the problem.

2.1 On Trees

Let us consider when the input tree $T = (V, E)$ has a logarithmic height $\delta \log n$, where $n = |V|$ and δ is some positive constant. Each edge in E has a non-negative cost, and facility opening cost is f for all vertices of V, for some positive constant f.

2.1.1 Preliminaries

The thought process here is to ignore the concept of facilities/airports initially in order to utilise the linear grouping procedures of Jayaprakash and Salavatipour [18] to get a set of clusters each of which is essentially a tree, and then assign airports to all the trees in that solution to get a solution to the original problem, without any increase in the cost.

We first describe the notion of *demands* on vertices. We view each vertex in V to initially have one unit of demand which needs to be sent to an airport to be served. Note that 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.

There are two equivalent ways of viewing the addition of dummy vertices at a vertex v. The first one is to simply superimpose some dummy vertices onto the vertex v, and the other is to add more demands to v according to the number of dummies needed. Also 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.

Note that the clusters in the solution are therefore not necessarily edge-disjoint or vertex-disjoint, but the total number of demands in each cluster obeys the capacity constraint.

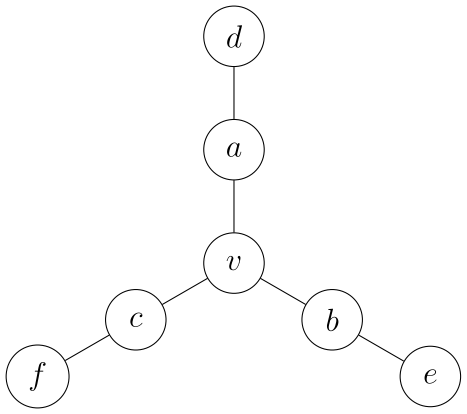
We begin by defining a new version of the problem which we call UAR (meaning AR with *undetermined* airports). In this relaxed version, we do not open airports anywhere in the solution and instead simply charge each component in the solution a cost of f.

Definition 5 (UAR) *The goal is to find a set $\mathscr{T}$ of (not necessarily disjoint) clusters (i.e. trees) using edges in the graph (each costs the corresponding edge's cost). The size of each cluster must not exceed the capacity constraint k. Each cluster $\gamma \in \mathscr{T}$ has a cost of f and we want to minimise the total cost, which is defined as*

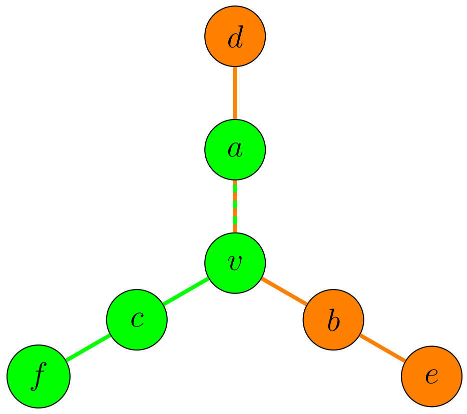
$$|\mathscr{T}| \cdot f + \sum_{\gamma \in \mathscr{T}} \mathrm{cost}(\gamma)$$

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.



(a) The input graph



(b) The optimal solution to UAR

Figure 2.1: This is an example where $k = 4$, $f = 5$ and the cost of every edge is 1. Note how the two clusters (each colour represents a cluster) in the solution share an edge emphasised by the dashed pattern of both colours. Both clusters will pay for the cost of the shared edge separately.

We can view each city as having a demand of 1, and a tree (also called a cluster) does not violate the capacity constraint as long as it does not pick up more than k units of demand. We use a modified version of the method from [18] to solve our problem. The preliminary steps are the same as in that paper.

Two concepts are required to describe the following data structures, namely, the notions of partial and complete clusters. At any non-root vertex $v \in V$ with parent edge e, a complete cluster in the subtree T_v is a cluster that does not use the edge e, and a partial cluster in the subtree T_v is one that does.

Similar to [18], we first assume that the number of clusters in OPT is sufficiently large, that is, at least $\varsigma \log n$ for some large number ς. Otherwise, if the number of clusters in OPT is upper-bounded by $\Sigma = \varsigma \log n$ then we can use the following dynamic programming algorithm. Define a DP table with entries of the form $\mathbf{A}[v, \mathbf{x}_v, \mathbf{a}_v]$ at each vertex v with $\mathbf{x}_v = \left(x_v^1, \ldots, x_v^\Sigma\right)$, and $\mathbf{a}_v = \left(a_v^1, \ldots, a_v^\Sigma\right)$, where $x_v^i \in [0, n]$ and a_v^i is a Boolean variable, for all $1 \leq i \leq \Sigma$. The entry $\mathbf{A}[v, \mathbf{x}_v, \mathbf{a}_v]$ stores the minimum cost of the solution where the i-th cluster is covering x_v^i many vertices in T_v, and it is a complete cluster only if $a_v^i = \textsc{True}$ (otherwise it is a partial one). To obtain the final solution, we calculate $\min_{\mathbf{x}_\rho}\{\mathbf{A}[\rho, \mathbf{x}_\rho, \mathbf{a}_\rho] \mid a_\rho^i = \textsc{True}, 1 \le i \le \Sigma\}$ where ρ is the root vertex. Note that this table has $O(n^\Sigma \cdot 2^\Sigma) = O(n^{\varsigma \log n})$ entries. The time it takes to compute the table from the bottom up is in $O(n^{\varsigma \log n})$.

Assume the optimal solution costs $\textit{OPT}$. We aim to find a near-optimal solution, of cost $(1+O(\varepsilon))\textit{OPT}$, where each vertex has at least one unit of demand, and the size of partial clusters in any subtree T_v can only be one of *polylogarithmically* many values, where $v \in V$.

The following is a summary of the modified data structures. We will use them to help show the existence of the kind of near-optimal solution mentioned above, and then give a dynamic programming algorithm that aims to obtain such a near-optimal solution.

1. Consider the subproblem on T_v (the subtree rooted at v). In T_v, there are **partial clusters** as well as complete ones. Now consider those partial ones as well as their other halves (called the *top*) connecting to them via v from outside the subtree T_v. We define a set of **threshold values** $\{\sigma_1, \ldots, \sigma_\tau\}$ the same way as in [18].
2. Define the i-th **bucket** of city v (denoted as (v, b_i)) to be the set of partial clusters whose size falls into the range $[\sigma_i, \sigma_{i+1})$.
3. For each *big* bucket of v (defined below), we further divide those partial clusters into g **groups** of equal size.

Definition 6 *Define the* ***threshold values*** $\{\sigma_1, \ldots, \sigma_\tau\}$ *where*

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

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

Given an optimal solution OPT, we will show how we can transform it to a near-optimal solution with less complexity in terms of types of cluster sizes in each subproblem, in a manner that is from bottom to top, one level at a time, starting from the bottom level. We denote the solution before modifying the level ℓ as OPT_ℓ and after modifying as $\mathrm{OPT}_{\ell-1}$. We assume the root vertex is at level 1. Suppose v is at level ℓ.

We use the following definitions and notations from [18].

Definition 7 *At a vertex v, we define its i-th bucket, denoted as b_i, to store clusters in OPT_ℓ that have a size between $[\sigma_i, \sigma_{i+1})$ inside T_v, where σ_i is the i-th threshold value. Denote a vertex and its i-th bucket as a pair (v, b_i). Denote the number of clusters in bucket b_i of v as $n_{v,i}$.*

Definition 8 *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 $\alpha \geq \max\{1, 12\delta\}$.*

Within each big bucket, we further divide its partial clusters into g groups of equal size (pad the bucket with the least amount of dummy cluster with size 0 such that all the groups can have an equal size).

Definition 9 *For a big bucket b_i of vertex v, we define g groups, denoted as $G^v_{i,1}, G^v_{i,2}, \ldots, G^v_{i,g}$, where $g = 2\delta \log n/\varepsilon$. Sort the clusters in the padded bucket in non-decreasing order, and put the first $\frac{n_{v,i}}{g}$ clusters into $G^v_{i,1}$, the second $\frac{n_{v,i}}{g}$ into $G^v_{i,2}$, etc. For each group $G^v_{i,j}$, denote the size of its smallest cluster as $h^{v,\min}_{i,j}$ and the size of its biggest cluster as $h^{v,\max}_{i,j}$.*

Suppose we are considering a big bucket of v 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_v) 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**.

We will first obtain a structure theorem and then come up with a dynamic programming algorithm.

2.1.2 Structure Theorem for Trees

Recall we have assumed that the number of clusters in OPT is at least $\varpi \log n$ for some large number ϖ. This assumption is crucial for our proof.

The steps of modifying OPT to a near-optimal solution (denoted as OPT′) are largely the same as the ones in [18]. 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 [18] and we only need to show the part concerning the facility costs.

Lemma 1 *The total cost of the extra clusters sampled is at most $4\varepsilon \cdot \mathit{OPT}$ w.h.p.*

Proof. Consider the parent edge e of some vertex v in the tree. 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 duplicated each sampled cluster, $2\phi'(e)$ corresponds to the total number of extra clusters using e in OPT′. Let $\mathscr{T}$ and $\mathscr{T}'$ denote the number of clusters in OPT and sampled clusters respectively. We have

$$\mathit{OPT}' = f \cdot (|\mathscr{T}| + 2|\mathscr{T}'|) + \sum_{e \in E} w(e) \cdot (\phi(e) + 2\phi'(e))$$

where the cost of extra clusters is $f \cdot 2|\mathscr{T}'| + \sum_{e \in E} w(e) \cdot 2\phi'(e)$. We can assume $\phi(e) \geq \alpha \log^2 n/\varepsilon$ for every edge e that is used by some extra cluster, since otherwise v does not have a big bucket and thus none of the extra clusters needs to go there.

Using the proof in [18], we know $\phi'(e) \leq 2\varepsilon \cdot \phi(e)$ with high probability.

Additionally, we know

$$\mathbb{E}[|\mathscr{T}'|] = \varepsilon \cdot |\mathscr{T}| \geq \varepsilon \cdot \max_{e \in E}\{\phi(e)\} \geq \alpha \log^2 n \geq 6 \log n.$$

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

Using Chernoff bound, we get

$$\Pr[|\mathscr{T}'| > 2\mathbb{E}[|\mathscr{T}'|]] \leq e^{-2\log n} = \frac{1}{n^2}.$$

Thus $|\mathscr{T}'| \leq 2\varepsilon \cdot |\mathscr{T}|$ with high probability. We see that the total cost of these extra clusters is, with high probability, at most

$$\begin{aligned} f \cdot 2|\mathscr{T}'| + \sum_{e\in E} w(e) \cdot 2\phi'(e) &\leq f \cdot 2(2\varepsilon \cdot |\mathscr{T}|) + \sum_{e\in E} w(e) \cdot 2(2\varepsilon \cdot \phi(e)) \\ &= 4\varepsilon \cdot f \cdot |\mathscr{T}| + 4\varepsilon \sum_{e\in E} w(e) \cdot \phi(e) = 4\varepsilon \cdot \mathit{OPT}. \end{aligned}$$

■

We also use the following modified definitions and lemmata from [18]. They apply to our problem as the proofs of the lemmata are almost identical.

Denote the vertices in level ℓ of T as V_ℓ. Define the set X_ℓ to comprise the extra clusters assigned to level ℓ. For every vertex $v \in V_\ell$ and its bucket (v, b_i), let $X_{v,i}$ represent the extra clusters in X_ℓ whose partial clusters inside T_v has a size that falls within the range defined by bucket b_i. For an extra cluster $\gamma \in X_{v,i}$, it covers some partial cluster $\zeta \in G^v_{i,g}$ (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 2 *At any level ℓ, each vertex $v \in V_\ell$ and its big buckets (v, b_i) satisfy, w.h.p.*

$$|X_{v,i}| \geq \frac{\varepsilon}{\delta \log n} \cdot n_{v,i}.$$

Lemma 3 *For all vertices v at level ℓ, their big buckets (v, b_i) and partial clusters in $G^v_{i,g} \subseteq b_i$, we can make adjustments to the extra clusters present in $X_{v,i}$ without incurring any additional cost, and introduce some dummy demands into T_v, if required, so that:*

1. *The partial clusters in $G^v_{i,g}$ are now incorporated into some clusters in $X_{v,i}$. (That is, the cities that were covered by some partial cluster in $G^v_{i,g}$ are picked up by some cluster in $X_{v,i}$.)*

2. *The modified partial clusters that cover the orphans (i.e., vertices in $G^v_{i,g}$) have precisely the size of $h^{v,\max}_{i,g}$ and all clusters remain within the size limit of k.*
3. *For each modified partial cluster of $X_{v,i}$, its partial clusters at a vertex $v' \in V_{\ell'}$ is also of one of $O\left(\log k \log^2 n/\varepsilon^2\right)$ many sizes, where ℓ' is any lower levels $> \ell$.*

Using the previous lemmata and an argument similar to the proof of Theorem 6 in [18], we can obtain a structure theorem for our UAR problem. Again, recall the size of a cluster is the sum of the demands of its vertices.

Theorem 2 *(Structure Theorem) Consider an instance $\mathscr{I}$ for the UAR problem. Denote its optimal solution as* OPT, *with cost* OPT. *We can transform* OPT *to another solution* OPT′ *so that with high probability* OPT′ *is a near-optimal solution of cost at most* $(1+4\varepsilon)$OPT. *Additionally, at every v in* OPT′, *all the partial clusters in subtree T_v have one of $O(\log k \log^2 n/\varepsilon^2)$ possible sizes. Consider a bucket (v, b_i) in* OPT′. *We must have*

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

$$\sigma_i \le h^{v,\max}_{i,1} \le h^{v,\max}_{i,2} \le \cdots \le h^{v,\max}_{i,g} < \sigma_{i+1}$$

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

2.1.3 Dynamic Programming for Trees

The subproblem here corresponds to the subtree T_v rooted at some vertex v, whose cost is the railway cost of all the clusters (partial clusters and complete ones), plus f times the number of complete clusters, and plus the cost of the parent edge of v times the number of partial clusters (which will be using that edge). That is, for a partial cluster spanning down from v, its cost is the sum of the cost of all its edges, together with the cost of the parent edge of v. For a complete cluster that includes v (we also say the cluster is *independent* or stops growing), its cost does not include the cost of the parent edge of v but its cost needs to include the facility opening cost f.

We will come up with a dynamic programming algorithm that finds a solution with the properties stated in the structure theorem.

We adapt the definitions of the ones in [18]. Recall that τ is the total number of threshold values. Also recall that the bucket (v, b_i) stores the size of all those clusters spanning κ vertices in T_v where $\sigma_i \leq \kappa < \sigma_{i+1}$, and for all $1 \leq i \leq \lceil 1/\varepsilon \rceil$ this interval degenerates to simply $\{i\}$ (by the definition of the threshold values).

Note that the total number of clusters is bounded by n (as each cluster needs to cover at least one unit of demand), so is the number of clusters of some subtree T_v. Throughout the algorithm, although we will add dummy/extra demands, those are added within each cluster (hence already covered) so will not result in an increase in the number of clusters.

For each vertex, we define a vector $\mathbf{s} \in [n]^\tau$ where s_i stores the cardinality of its i-th bucket b_i, for all $1 \leq i \leq \tau$. In particular, s_i is the precise number of clusters of size i, if $1 \leq i \leq \lceil 1/\varepsilon \rceil$.

Define o_v to be the overall number of demands of all the vertices to be covered in T_v. We know $o_v \geq |V(T_v)|$ because we may add dummy demands at v.

Since we do not know if b_i 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 $\mathbf{t}^i \in [n]^\xi$ where t^i_j 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 $h^{\max}_{i,j}$ denotes the size of the maximum partial cluster in group j of b_i.

- $\mathbf{h}^i \in [n]^g$ is a vector that stores all those g cluster sizes, where $h^i_j = h^{\max}_{i,j}$.
- $\mathbf{l}^i \in [n]^g$ is a vector storing the corresponding number of partial clusters that are of one of the g sizes, with l^i_j representing the number of partial

clusters picking up $h^{\max}_{i,j}$ demands (l^i_j is also the cardinality of group j).

For a given triplet $(\mathbf{t}^i_v, \mathbf{h}^i_v, \mathbf{l}^i_v)$, 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 $\mathbf{t}^i_v$ is equal to the zero vector, or the other two vectors are.

Moreover, we use the shorthand $\mathbf{z}_v = \left(\left(\mathbf{t}^1_v, \mathbf{h}^1_v, \mathbf{l}^1_v\right), \left(\mathbf{t}^2_v, \mathbf{h}^2_v, \mathbf{l}^2_v\right), \ldots, \left(\mathbf{t}^\tau_v, \mathbf{h}^\tau_v, \mathbf{l}^\tau_v\right)\right)$. Just like the vector $\mathbf{z}_v$ is used for partial clusters, we define another vector $\tilde{\mathbf{z}}_v$ with the same structure as $\mathbf{z}_v$, except that $\tilde{\mathbf{z}}_v$ is intended for storing information on complete clusters in T_v. We define $\tilde{\mathbf{t}}^j_v, \tilde{\mathbf{h}}^j_v, \tilde{\mathbf{l}}^j_v$ similarly.

Denote $\mathbf{y}_v$ to be a configuration/profile of clusters for v.

$$\mathbf{y}_v = (o_v, \mathbf{s}_v, \mathbf{z}_v, \tilde{\mathbf{z}}_v)\,.$$

Each entry $\mathbf{A}[v, \mathbf{y}]$ stores the cost of the cheapest solution to the subproblem at v having its cluster profile in accordance with $\mathbf{y}$, which consists of the cost of all the partial and complete clusters spanning in T_v as well as the cost from v's parent edge e. Eventually, we compute $\min_{\mathbf{y}_r} \mathbf{A}[r, \mathbf{y}_r]$, where r is the root of the entire tree T and $\mathbf{y}_r$ cannot contain any partial clusters. By the definitions of our data structures, it is easy to see the solution we constructed satisfies the properties described in the structure theorem. Recall that we have shown the existence of a near-optimal solution of cost $\mathit{OPT}' \leq (1 + 4\varepsilon)\mathit{OPT}$ that has such properties. Since our algorithm finds the cheapest solution with such properties, it follows that this result corresponds to a solution of cost bounded by OPT'.

We calculate the table from the bottom up. Base cases: Any leaf v can only have exactly $o_v = 1$ demands as there cannot be any dummies generated at this stage. If the number of demands picked up by the partial and independent clusters in $\mathbf{y}_v$ is $o_v = 1$, then we set

$$\mathbf{A}[v, \mathbf{y}_v] \leftarrow w(e) \cdot \psi_v + f \cdot \varrho_v$$

where ϱ_v and ψ_v are the number of complete and partial clusters in $\mathbf{y}_v$, respectively. Note that it can exclusively be the case that only one of ψ_v and ϱ_v is one, and the other zero. If $\psi_v = 0$ and $\varrho_v = 1$, it means v is independent by its own and does not need to use its parent edge, hence the cost is $\mathbf{A}[v, \mathbf{y}_v] \leftarrow f$. On ther other hand, if $\psi_v = 1$ and $\varrho_v = 0$, it means v needs to use its parent edge and belongs to a partial cluster, hence the cost is $\mathbf{A}[v, \mathbf{y}_v] \leftarrow w(e)$.

For entry $\mathbf{A}[v, \mathbf{y}_v]$ where v is not a leaf, an auxiliary table $\mathbf{B}$ comes in handy. Say v has ζ children $u_1, \ldots, u_\zeta$. Assume the entries $\mathbf{A}[u_j, \mathbf{y}]$ have been computed for all $1 \le j \le \zeta$ and all possible vectors $\mathbf{y}$. An entry in table $\mathbf{B}[v, \mathbf{y}_v^j, j]$ for each $1 \le j \le \zeta$ stores the cost of the cheapest solution covering demands of the vertices in $T_{u_1} \cup \cdots \cup T_{u_j} \cup \{v\}$ if $\mathbf{y}_v^j$ is the cluster profile for the union of the first j subtrees as well as v. Equivalently, $\mathbf{B}[v, \mathbf{y}_v^j, j]$ is what $\mathbf{A}[v, \mathbf{y}_v]$ would be when v has its children $u_{j+1}, \ldots, u_\zeta$ discarded. Therefore, we set

$$\mathbf{A}[v, \mathbf{y}_v] \leftarrow \mathbf{B}[v, \mathbf{y}_v, \zeta].$$

For simplicity, assume the root node has a parent edge of zero cost. Suppose T_{u_i} has o_i demands, then the number of demands in T_v (that is, o_v) would be at least $1 + \sum_{i=1}^{k} o_i$. Now we describe the computation for the auxiliary table.

Consider $j = 1$. Then it is as if v has only the child vertex u_1. Abbreviate $\mathbf{y}_{u_1}$ as $\mathbf{y}'$. Recall that e is the parent edge of v.

$$\mathbf{B}[v, \mathbf{y}_v^j, 1] = \min_{\mathbf{y}'} \left\{ \mathbf{A}[u_1, \mathbf{y}'] + w(e) \cdot \psi_v^j + f \cdot (\varrho_v^j - \varrho') \right\}$$

where ψ_v^j is the number of partial clusters in $\mathbf{y}_v^j$, and ϱ_v^j, ϱ' is the number of complete clusters in $\mathbf{y}_v^j, \mathbf{y}'$ respectively. Note that the difference of ϱ_v^j and ϱ' is the number of newly independent clusters.

In addition, we need the configuration $\mathbf{y}'$ to be consistent with $\mathbf{y}_v^j$. A configuration $\mathbf{y}'$ is consistent with $\mathbf{y}_v^j$ if each cluster γ from $\mathbf{y}'$ is part of some cluster $\tilde{\gamma}$ from $\mathbf{y}_v^j$. That is, either γ is the same as $\tilde{\gamma}$, or $\tilde{\gamma}$ includes γ together with some other dummy demand(s) at v.

For the case where $2 \le j \le \zeta$: Assume the entries $\mathbf{B}[v, \mathbf{y}', j-1]$ and $\mathbf{A}[u_j, \mathbf{y}'']$ have been computed for all possible configurations $\mathbf{y}'$ and $\mathbf{y}''$. We define

$$\mathbf{B}[v, \mathbf{y}_v^j, j] = \min_{\mathbf{y}', \mathbf{y}''} \left\{ \mathbf{B}[v, \mathbf{y}', j-1] + \mathbf{A}[u_j, \mathbf{y}''] + w(e) \cdot (\psi_v^j - \psi') + f \cdot (\varrho_v^j - \varrho') \right\}$$

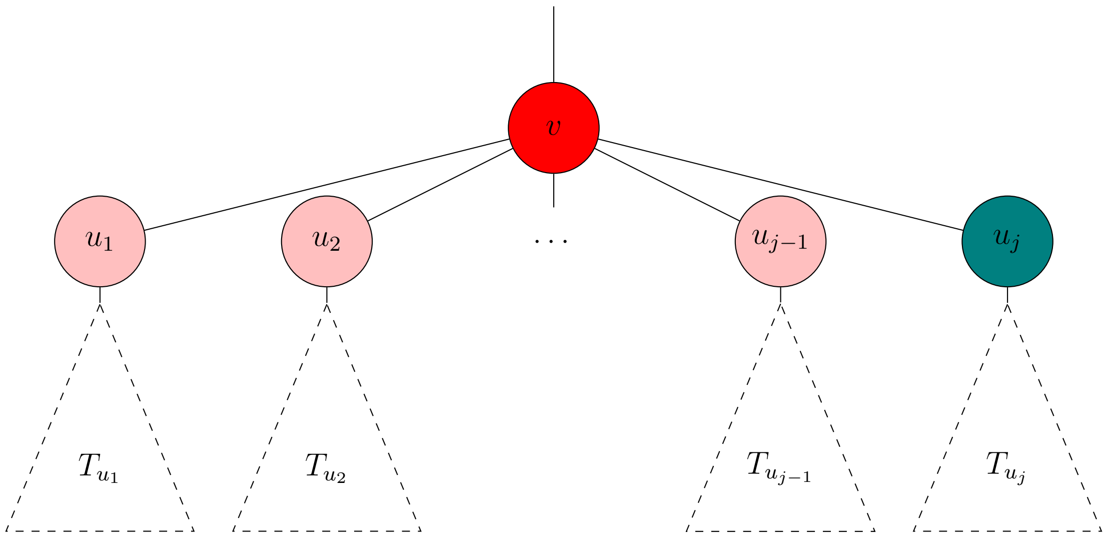


Figure 2.2: The illustration for the inner DP. To be more precise, each subtree T_{u_i} includes its root vertex u_i.

where ϱ_v^j, ϱ' is the number of complete clusters in $\mathbf{y}_v^j, \mathbf{y}'$ respectively, and ψ_v^j, ψ' is the number of partial clusters in $\mathbf{y}_v^j, \mathbf{y}'$ respectively. Note that $\mathbf{y}_v^j, \mathbf{y}'$ and $\mathbf{y}''$ should be *consistent*. Informally, the configurations $\mathbf{y}_v^j, \mathbf{y}'$ and $\mathbf{y}''$ are consistent if the clusters from $\mathbf{y}'$ and $\mathbf{y}''$ can be combined or augmented to form the clusters in $\mathbf{y}_v^j$ (as well as covering the dummy demands at v).

2.1.3.1 Consistency Check

Continue from the previous section, recall that v is a vertex with ζ children $u_1, \ldots, u_\zeta$ and we have three configurations $\mathbf{y}_v^j, \mathbf{y}', \mathbf{y}''$ in question for entries $\mathbf{B}[v, \mathbf{y}_v^j, j]$ where $2 \leq j \leq \zeta$. We adapt the checking procedure in [18] to suit our data structures. The following list is a recap of the meanings of these three configurations.

1. $\mathbf{y}_v^j$ keeps track of clusters covering demands in $\{v\} \cup \bigcup_{i=1}^{j} T_{u_i}$
2. $\mathbf{y}'$ keeps track of clusters covering $\{v\} \cup \bigcup_{i=1}^{j-1} T_{u_i}$
3. $\mathbf{y}''$ keeps track of clusters covering the single subtree T_{u_j}

Given the three numbers of demands, o_v^j being the total demands of T_v with the first j children (and subtrees), $o_\cup$ being the demands for the set of vertices $\{v\} \cup \bigcup_{i=1}^{j-1} T_{u_i}$, and o_{u_j} being the demands for T_{u_j}, it is obvious that the clusters in $\mathbf{y}_v^j$ need to pick up $\hat{o}_v^j = o_v^j - o_\cup - o_{u_j}$ extra dummy demands at v. Each cluster γ_v in $\mathbf{y}_v^j$ that spans down T_v picking up demands in subtrees $T_{u_1}, \ldots, T_{u_j}$ can take one of the following four forms.

1. γ_v only covers demands at v and not the ones from $\bigcup_{i=1}^{j} T_{u_i}$.
2. γ_v covers demands at v as well as the ones from $\bigcup_{i=1}^{j-1} T_{u_i}$.
3. γ_v covers demands at v as well as the ones from the subtree T_{u_j}.
4. γ_v covers demands at v as well as the ones from $\bigcup_{i=1}^{j} T_{u_i}$.

Denote a cluster exclusively covering demands from $\{v\} \cup \bigcup_{i=1}^{j-1} T_{u_i}$ as $\gamma_\cup$, and a cluster exclusively covering demands from T_{u_j} as γ_{u_j}.

Let $|\gamma|$ denote the number of demands the cluster γ picks up, that is, the sum of demands on all vertices of γ.

Definition 10 *We say configurations $\mathbf{y}_v^j, \mathbf{y}'$ and $\mathbf{y}''$ are* ***consistent*** *if the following holds:*

- *Each cluster in $\mathbf{y}'$ corresponds to a cluster in $\mathbf{y}_v^j$.*
- *Each cluster in $\mathbf{y}''$ corresponds to a cluster in $\mathbf{y}_v^j$.*
- *Each cluster in $\mathbf{y}_v^j$ has zero, one or two clusters that correspond to it. If it has two then these two clusters must be one from $\mathbf{y}'$, and the other from $\mathbf{y}''$.*
- *If a cluster γ_v in $\mathbf{y}_v^j$ has only one cluster $\gamma_\cup$ that corresponds to it, then γ_v covers $|\gamma_v| - |\gamma_\cup|$ dummy demands at v.*
- *If a cluster γ_v in $\mathbf{y}_v^j$ has two clusters $\gamma_\cup$ from $\mathbf{y}'$ and γ_{u_j} from $\mathbf{y}''$ that correspond to it, then γ_v covers $|\gamma_v| - |\gamma_\cup| - |\gamma_{u_j}|$ dummy demands at v.*
- *Clusters in $\mathbf{y}_v^j$ cover those $\hat{o}_v^j = o_v^j - o_\cup - o_{u_j}$ dummy demands at v.*

By verifying consistency at each entry of the table, the clusters in the current subtree will contain all the clusters of the corresponding subproblems as well as cover all the dummy demands at the root of the current subtree. To achieve this, we will define the consistency table $\mathbf{C}$. Recall that v's cluster configuration is $\mathbf{y}_v = (o_v, \mathbf{s}_v, \mathbf{z}_v, \tilde{\mathbf{z}}_v)$ where the core information is $\mathbf{z}_v = \left((\mathbf{t}_v^1, \mathbf{h}_v^1, \mathbf{l}_v^1), (\mathbf{t}_v^2, \mathbf{h}_v^2, \mathbf{l}_v^2), \dots, (\mathbf{t}_v^\tau, \mathbf{h}_v^\tau, \mathbf{l}_v^\tau)\right)$ as well as $\tilde{\mathbf{z}}_v$, since all other variables in $\mathbf{y}_v$ can be obtained through these two. Recall $|\gamma|$ denotes the total number of demands picked up by cluster γ. Define[2]

$$\mathbf{C}[\hat{o}_v^j, (\mathbf{z}_v^j, \mathbf{z}', \mathbf{z}''), (\tilde{\mathbf{z}}_v^j, \tilde{\mathbf{z}}', \tilde{\mathbf{z}}'')] = \begin{cases} \textsc{True} & \text{if } \mathbf{y}_v^j, \mathbf{y}', \mathbf{y}'' \text{ are consistent and } \mathbf{y}_v^j \\ & \text{covers } \hat{o}_v^j \text{ dummy demands at } v \\ \textsc{False} & \text{otherwise} \end{cases}$$

for each vertex v. Base case: trivially, $\mathbf{C}[0, (\mathbf{0}, \mathbf{0}, \mathbf{0}), (\mathbf{0}, \mathbf{0}, \mathbf{0})] = \textsc{True}$, since in T_v we can cover zero units of demands with no clusters. Next, it examines all potential ways of merging $\mathbf{y}'$ with $\mathbf{y}''$ into $\mathbf{y}_v^j$, including extending some of the clusters in order to cover those $\hat{o}_v^j$ dummy demands. In the following expression, the notation $\mathbf{z} \setminus \gamma$ represents a new configuration that is obtained by removing a cluster of size $|\gamma|$ from $\mathbf{z}$. This can be achieved by manipulating vectors $\mathbf{t}$'s and $\mathbf{l}$'s. Define[3]

$$\mathbf{C}\left[\hat{o}_v^j, (\mathbf{z}_v^j, \mathbf{z}', \mathbf{z}''), (\tilde{\mathbf{z}}_v^j, \tilde{\mathbf{z}}', \tilde{\mathbf{z}}'')\right] = \bigvee_{\substack{\gamma_v \in \mathbf{y}_v^j, \gamma_\cup \in \mathbf{z}', \gamma_{u_j} \in \mathbf{z}'' : \\ |\gamma_v| = |\gamma_\cup| + |\gamma_{u_j}| + \hat{o}}} (\Omega)$$

where the inside of the parentheses (Ω) should be

$$\mathbf{C}\left[\hat{o}_v^j - \hat{o}, (\mathbf{z}_v^j \setminus \gamma_v, \mathbf{z}' \setminus \gamma_\cup, \mathbf{z}'' \setminus \gamma_{u_j}), (\tilde{\mathbf{z}}_v^j, \tilde{\mathbf{z}}', \tilde{\mathbf{z}}'')\right] \vee \mathbf{C}\left[\hat{o}_v^j - \hat{o}, (\mathbf{z}_v^j, \mathbf{z}' \setminus \gamma_\cup, \mathbf{z}'' \setminus \gamma_{u_j}), (\tilde{\mathbf{z}}_v^j \setminus \gamma_v, \tilde{\mathbf{z}}', \tilde{\mathbf{z}}'')\right].$$

The two entries above consider both of the scenarios where the cluster γ_v is either partial or complete. Each examines whether the remaining clusters in the (modified) profile $\mathbf{y}_v^j$ (that is defined by $\mathbf{z}_v^j$ and $\tilde{\mathbf{z}}_v^j$) cover $\hat{o}_v^j - \hat{o}$ dummy demands at v or not.

[2] Recall we have only defined consistency on $\mathbf{y}$. Besides, given $\mathbf{z}$ and $\tilde{\mathbf{z}}$, we can always compute the corresponding $\mathbf{y}$.

[3] According to the third point in Definition 10, we know $|\gamma_\cup|$ and $|\gamma_{u_j}|$ are potentially zero.

2.1.3.2 Algorithm Efficiency

Recall that we have assumed the height of the input tree to be logarithmic. Following a very similar analysis as the one in [18], we know for each vertex $v \in V$, there are

- $O(n^\tau) = n^{O(\log k/\varepsilon)} = n^{O(\log n/\varepsilon)}$ different choices for $\mathbf{s}_v$.
- $O(n^{\alpha \log^2 n/\varepsilon}) = n^{O(\log^2 n/\varepsilon)}$ different choices for $\mathbf{t}_v^i$.
- $O(n^g) = O(n^{2\delta \log n/\varepsilon}) = n^{O(\log n/\varepsilon)}$ different choices for $\mathbf{h}_v^i$ and $\mathbf{l}_v^i$.
- $n^{O(\log^2 n/\varepsilon)+O(\log n/\varepsilon)} = n^{O(\log^2 n/\varepsilon)}$ different choices for $(\mathbf{t}_v^i, \mathbf{h}_v^i, \mathbf{l}_v^i)$.
- $\prod_{i=1}^{\tau} n^{O(\log^2 n/\varepsilon)} = n^{\tau \cdot O(\log^2 n/\varepsilon)} = n^{O(\log k \log^2 n/\varepsilon^2)}$ choices for $\mathbf{z}_v$ and $\tilde{\mathbf{z}}_v$.
- Thus, $n^{O(\log k \log^2 n/\varepsilon^2)}$ choices for $\mathbf{y}_v$.

By definition, to calculate an entry for table B, we generally need to consider all possible $\mathbf{y}'$ and $\mathbf{y}''$, which have the same number of different choices as that of $\mathbf{y}_v$. Assuming the computation is from the bottom up, each entry of $\mathbf{B}$ takes $n^{O(\log k \log^2 n/\varepsilon^2)}$ to compute. To fill the entire table of $\mathbf{A}$ and $\mathbf{B}$, we need to consider all vertex $v \in V$ as well as its cluster profile $\mathbf{y}_v$, so it takes $n^{O(\log k \log^2 n/\varepsilon^2)}$ in total.

Consider the consistency table $\mathbf{C}$. A triplet $(\mathbf{z}, \mathbf{z}', \mathbf{z}'')$ has $n^{O(\log k \log^2 n/\varepsilon^2)}$ possibilities, so the table have $n^{O(\log k \log^2 n/\varepsilon^2)}$ entries. To compute each entry, we need to consider all possible combinations of three clusters from $\mathbf{y}, \mathbf{y}', \mathbf{y}''$ respectively. This has $n^{O(\log k \log^2 n/\varepsilon^2)}$ possibilities, since each cluster is either from $\mathbf{z}$ or $\tilde{\mathbf{z}}$. Therefore, the runtime to fill the entire table is $n^{O(\log k \log^2 n/\varepsilon^2)}$.

Given the time for calculating both the DP table and the consistency table is in $n^{O(\log k \log^2 n/\varepsilon^2)}$, our algorithm has runtime $n^{O(\log k \log^2 n/\varepsilon^2)}$. Moreover, the runtime is in $n^{O(\log^3 n/\varepsilon^2)}$ as the capacity k satisfies $k \leq n$. Thus the algorithm is a QPTAS.

2.1.4 Solution for the Original Problem

Now we discuss how to get a solution for the UAR problem, and then how to get a solution for the AR problem based on that. Using the DP from the previous section, it is easy to adapt the algorithm so that we obtain a set $\mathscr{T}$ of clusters/trees, which is essentially depicted by one of the cluster configurations of the root vertex.

Consider a bipartite graph, denoted as G', with two sets of vertices: B and W. The set B consists of vertices representing trees (clusters) γ in $\mathscr{T}$, while the set W consists of vertices representing individual vertices v in the input tree $T = (V, E)$. In G', each vertex $\gamma \in B$ is connected by an edge to a vertex $v \in W$ if the vertex $v \in V$ belongs to the cluster $\gamma \in \mathscr{T}$.

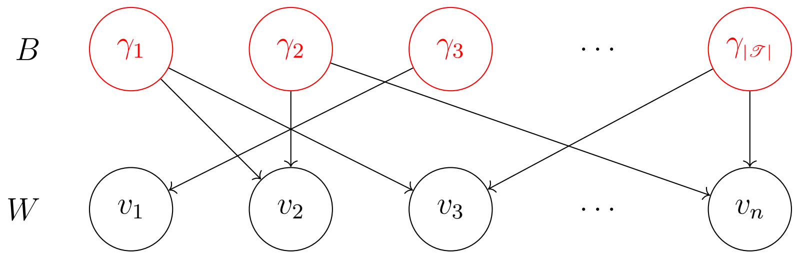


Figure 2.3: An example bipartite graph G' showing affiliations

Observe that, for every subset Λ of B, we have $|\Lambda| \leq |\mathscr{N}_{G'}(\Lambda)|$ where $\mathscr{N}$ denotes the set of neighbours. This is because each cluster $\gamma \in \mathscr{T}$ must possess at least one vertex, and therefore the cluster(s) represented by $\Lambda \subseteq B$ must possess at least $|\Lambda|$ vertices (i.e. picking up their demands). It follows that the number of neighbours of the set Λ in G' must be at least $|\Lambda|$. Thus, by Hall's marriage theorem [15], we have a B-perfect/saturating matching. This implies that we can assign an airport to every cluster in $\mathscr{T}$ such that every cluster has a distinct airport, without increasing the cost (since the facility cost f is already paid when every cluster becomes independent). This implies that the solution $\mathscr{T}$ to the relaxed problem (UAR) can be transformed into a feasible solution to the original problem with the same cost, which is at most $(1 + 4\varepsilon)\mathit{OPT}$.

2.2 On Graphs with Bounded Treewidth

2.2.1 Preliminaries

Given a graph $G = (V, E)$ of treewidth ω, there is a tree decomposition[4] $T = (V', E')$ 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 Bodlaender and Hagerup [8]. For simplicity, denote ω' as ω instead. We assume the tree height $h = \delta \log n$ for some constant $\delta > 0$. 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 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 $\mathrm{OPT}_{\ell-1}$.

To generalise the method we used previously for the case of trees, we generalise the notations first. We adapted the definitions from [18]. Consider a bag β that is situated at level ℓ.

Definition 11 *For a bag β at level ℓ in T, 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. Define the i-th bucket of $b_S^{\wp_S}$, denoted*

[4]See section 1.2.1 for its definition.

as b_i, to store clusters in OPT_ℓ *that have a size between* $[\sigma_i, \sigma_{i+1})$ *inside* C_β, *where* σ_i *is the i-th threshold value. Denote this bucket by a tuple* $(\beta, b_i, S, \wp_S)$. *Denote the number of clusters in bucket* $(\beta, b_i, S, \wp_S)$ *as* $n_{\beta,i}^{S,\wp_S}$.

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

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

Definition 13 *For a big bucket* $(\beta, b_i, S, \wp)$*, define g groups where* $g = \frac{2\delta \log n}{\varepsilon}$*, denoted as* $G_{i,1}^{\beta,S,\wp}, G_{i,2}^{\beta,S,\wp}, \ldots, G_{i,g}^{\beta,S,\wp}$*. Sort the clusters in the padded bucket in non-decreasing order, and put the first* $\frac{n_{\beta,i}^{S,\wp}}{g}$ *clusters into* $G_{i,1}^{\beta,S,\wp}$*, the second* $\frac{n_{\beta,i}^{S,\wp}}{g}$ *into* $G_{i,2}^{\beta,S,\wp}$*, etc. For each group* $G_{i,j}^{\beta,S,\wp}$*, denote the size of its smallest cluster as* $h_{i,j}^{\beta,S,\wp,\min}$ *and the size of its biggest cluster as* $h_{i,j}^{\beta,S,\wp,\max}$.

We will use the approach for trees as in the previous sections, that is, we first 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.

2.2.2 Structure Theorem for Graphs with Bounded Treewidth

Using the same way of picking and assigning extra clusters, we now prove the following lemma. 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 4 *The expected cost of the extra clusters sampled is* $2\varepsilon \cdot \textsc{opt}$.

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

$$\textsc{opt}' = f \cdot (|\mathscr{T}| + 2|\mathscr{T}'|) + \sum_{e \in E} w(e) \cdot (\phi(e) + 2\phi'(e))$$

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

Using the proof in [18], we know $\mathbb{E}[\phi'(e)] = \varepsilon \cdot \phi(e)$.[5]

Additionally, we know

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

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

$$\begin{aligned}
\mathbb{E}\left[f \cdot 2|\mathscr{T}'| + \sum_{e \in E} w(e) \cdot 2\phi'(e) \right] &= 2f \cdot \mathbb{E}[|\mathscr{T}'|] + \sum_{e \in E} w(e) \cdot 2\mathbb{E}[\phi'(e)] \\
&= 2f \cdot (\varepsilon \cdot |\mathscr{T}|) + \sum_{e \in E} w(e) \cdot 2(\varepsilon \cdot \phi(e)) \\
&= 2\varepsilon \left(f \cdot |\mathscr{T}| + \sum_{e \in E} w(e) \cdot \phi(e) \right) \\
&= 2\varepsilon \cdot \textsc{opt}.
\end{aligned}$$

■

We make use of the following modified definitions and lemmata from [18]. 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. Similar to what we did in the last section where we discussed about the case of trees, 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

[5]For the notations, $\phi(e)$ corresponds to $f^+(e) + f^-(e)$ in [18], and $\phi'(e)$ corresponds to $m^+(e) + m^-(e)$.

$(\beta, b_i, S, \wp)$, 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 5 *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 [18]. That is, we know

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

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 $|\beta| \leq \omega + 1$), 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 $S \subseteq \beta$ (i.e. $2^\beta \ni S$) has $2^{|\beta|} \in O(2^{\omega+1})$ possibilities. In addition, given a set S with $s = |S|$, the set $\wp$ has $\mathscr{B}_s$ possibilities, where $\mathscr{B}_n$ denotes the n-th Bell number. Note that the Bell numbers $\mathscr{B}_i$ are used to calculate the number of all the possible partitions of a set of size i, defined in [5], [6]. 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 $O(\omega n \log n/\varepsilon \cdot 2^{\omega+1} \cdot \mathscr{B}_\omega)$. Union bound over every bucket $(\beta, b_i, S, \wp)$, we have

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

■

Lemma 6 *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 $O\left(\log k \log^2 n/\varepsilon^2\right)$ 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 [18], we can obtain a Structure Theorem for our UAR problem in the case of graphs with bounded treewidth.

Theorem 3 *(Structure Theorem) Consider an instance $\mathscr{I}$ for the* UAR *problem. Denote its optimal solution as* OPT*, with cost* OPT*. We can transform* OPT *to another solution* OPT$'$ *so that, with high probability,* OPT$'$ *is a near-optimal solution of cost at most $(1+2\varepsilon)$*OPT*. Additionally, at every β in* OPT$'$*, all the clusters in C_β has 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.

2.2.3 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 [18]. 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 $O(2^{\omega+1} \cdot \mathscr{B}_\omega)$ 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

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

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 $\mathbf{d}_\beta$.

2.2.3.1 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 [18] 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 $C_{\beta_1} \cup C_{\beta_2}$.

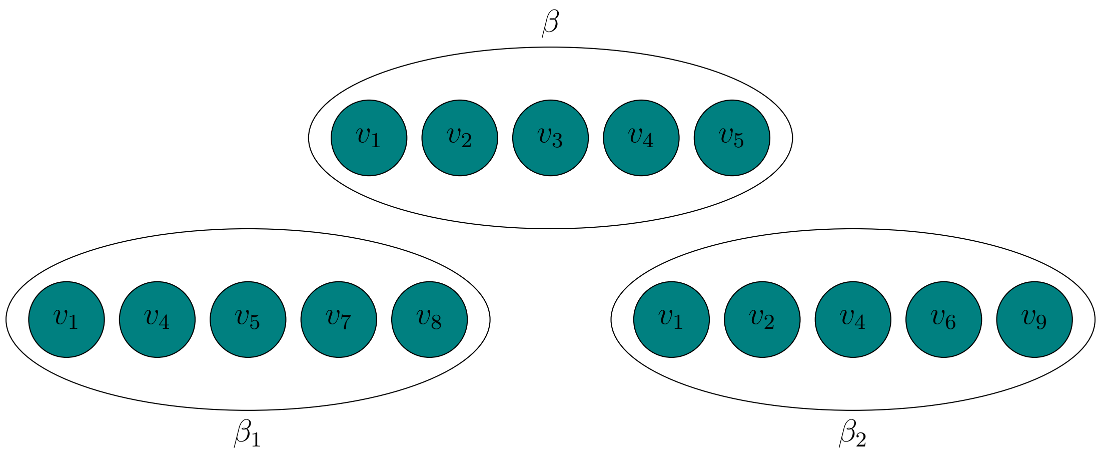


Figure 2.4: An example showing a bag β with its two children bags β_1 and β_2

Definition 14 *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 $0 \leq q_{\beta_1}, q_{\beta_2} \leq 2^{\omega+1}$, 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.[6]

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

Base case: trivially, $\mathbf{H}[\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}] = \text{TRUE}$, 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 $\mathbf{d}_\beta^{\gamma_\beta} \in [n]^{\omega+1}$ to be a vector storing the numbers of demands in β covered by γ_β, where $d_{\beta,v}^{\gamma_\beta}$ represents 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 14, 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

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

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

[6]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.

connected downwards), then there needs to be a set X such that $X \supseteq \{u, v\}$ and $X \in \wp'$. We denote this test as (†).

Define the recurrence relation of $\mathbf{H}$ as follows, where $0 \leq q_{\beta_1}, q_{\beta_2} \leq 2^{\omega+1}$ 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 $0 \leq q_{\beta_1}, q_{\beta_2} \leq 2^{\omega+1}$. 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 $\mathbf{H}[\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_2}] = \text{TRUE}$. 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 14, we need to consider the clusters $\gamma_{\beta_1}^1, \gamma_{\beta_1}^2, \ldots, \gamma_{\beta_1}^{q_{\beta_1}}$ from $\mathbf{y}_{\beta_1}$ and $\gamma_{\beta_2}^1, \gamma_{\beta_2}^2, \ldots, \gamma_{\beta_2}^{q_{\beta_2}}$ from $\mathbf{y}_{\beta_2}$.

Besides, note that some of the partial clusters $\gamma_{\beta_1}^1, \gamma_{\beta_1}^2, \ldots, \gamma_{\beta_1}^{q_{\beta_1}}$ from $\mathbf{y}_{\beta_1}$ may have been connected in C_{β_1} (the same may happen in C_{β_2} for $\gamma_{\beta_2}^1, \gamma_{\beta_2}^2, \ldots, \gamma_{\beta_2}^{q_{\beta_2}}$ 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 $\{S_1, S_2, \dots\}$ 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 $\text{cost}\left(\Upsilon^{\wp}_{\gamma_\beta}\right) = \sum_{e \in \Upsilon^{\wp}_{\gamma_\beta}} \text{cost}(e)$.

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

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

where Θ should be the expression

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

2.2.3.2 Algorithm Efficiency

We first discuss the runtime for table $\mathbf{A}$ and then we talk about $\mathbf{E}$ and $\mathbf{H}$. The runtime analysis is very similar to that in [18], 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 $O(2^{\omega+1} \cdot \mathscr{B}_\omega)$ many $\mathbf{p}^{\beta,S,\wp}$, we know $\mathbf{p}_\beta$ has $n^{O(2^{\omega+1} \cdot \mathscr{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ possibilities. By definition of $\mathbf{y}_\beta$, it therefore also has $n^{O(2^{\omega+1} \cdot \mathscr{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ possibilities. The number of possibilities is the same for $\mathbf{y}_{\beta_1}$ and $\mathbf{y}_{\beta_2}$, so it takes $n^{O(2^{\omega+1} \cdot \mathscr{B}_\omega \cdot \log k \log^2 n/\varepsilon^2)}$ to calculate the entries of table $\mathbf{A}[\beta, \cdot]$ for a single bag β (according to $\mathbf{A}$'s recurrence), assuming the other tables $\mathbf{E}$ and $\mathbf{H}$ 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 $\left(\gamma_\beta, \gamma^1_{\beta_1}, \gamma^2_{\beta_1}, \dots, \gamma^{q_{\beta_1}}_{\beta_1}, \gamma^1_{\beta_2}, \gamma^2_{\beta_2}, \dots, \gamma^{q_{\beta_2}}_{\beta_2}, \mathbf{d}^{\gamma_\beta}_\beta, \Upsilon_{\gamma_\beta}\right)$, 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 $O(2^{\omega^2})$ possibilities by definition. Since there are $n^{O(2^{\omega+1}\cdot\mathscr{B}_\omega\cdot\log k\log^2 n/\varepsilon^2)}$ possibilities for $(\mathbf{d}_\beta, \mathbf{p}_\beta, \mathbf{p}_{\beta_1}, \mathbf{p}_{\beta_1})$, we know it takes $n^{O(2^{\omega+1}\cdot\mathscr{B}_\omega\cdot\log k\log^2 n/\varepsilon^2)}$ to compute the table $\mathbf{E}$. From a similar analysis, we conclude the same runtime for the consistency table $\mathbf{H}$. Therefore, our algorithm has a runtime of $n^{O(2^{\omega+1}\cdot\mathscr{B}_\omega\cdot\log k\log^2 n/\varepsilon^2)} = n^{O(2^{\omega}\cdot\mathscr{B}_\omega\cdot\log^3 n/\varepsilon^2)}$, since the facility capacity k satisfies $k < n$. According to the following lemma by Berend and Tassa [7], the runtime is thus in $n^{O((1.584\omega)^{\omega}\cdot\log^3 n/(\varepsilon^2\log^{\omega}\omega))}$.

Lemma 7 *(Theorem 2.1 in [7]) The Bell numbers satisfy*

$$\mathscr{B}_n < \left(\frac{0.792n}{\ln(n+1)}\right)^n, \quad \forall n \in \mathbb{N}_+.$$

2.2.4 Solution for the Original Problem

We use the same method as the one described in the previous section (where we talked about the case of trees) to transform the solution obtained (that is, to the UAR problem) into a solution to the AR problem, without any increase in the cost.

2.2.5 Generalisation for AR with Steiner Vertices

Assume the input graph is $G = (V, E)$. In this section, we describe how we can easily generalise the algorithm above to suit the case where Steiner nodes may exist, and more generally, the set of cities/clients or the set of potential airports/facilities may not be V. This is useful for future extensions of this algorithm to other graph metrics. First observe that a Steiner vertex (or more generally, if this vertex is not in the set of cities) should not have a demand initially (but can be assigned with dummy demands), and the same applies to all of its copies in the tree decomposition. Additionally, a Steiner vertex (or more generally, if this vertex is not in the set of potential airports) should not be opened as an airport. We add these requirements into the consistency check of the algorithm. The runtime of the algorithm is asymptotically the same as the one above.

2.3 Constant-factor for General Metric

We assume we are given a complete metric graph $G = (V, E)$. Note that for simplicity, we write $k - 1$ as k (the capacity of each airport) below. We first define a modified instance $\tilde{G}$ for each input graph G. The graph $\tilde{G}$ is obtained by adding a dummy node δ to G and connecting δ to all the vertices $v \in V$ with an edge of cost $c_{v\delta} = f$ (the facility opening cost). We first define the $\mathrm{MST}_{\delta}^{\chi}$ problem and prove the following lower bound.

Definition 15 *In the* $\mathrm{MST}_{\delta}^{\chi}$ *problem, we are given a graph* $G = (V, E)$ *where* $V \ni \delta$. *The task is to find the minimal cost of the spanning tree of the input graph, while ensuring that the degree of the vertex* δ *in the solution is* χ.

Lemma 8 *The cost of an optimal solution to the* $\mathrm{MST}_{\delta}^{\chi}$ *problem on* $\tilde{G}$ *is a lower bound on the optimal solution to* AR *on* G.

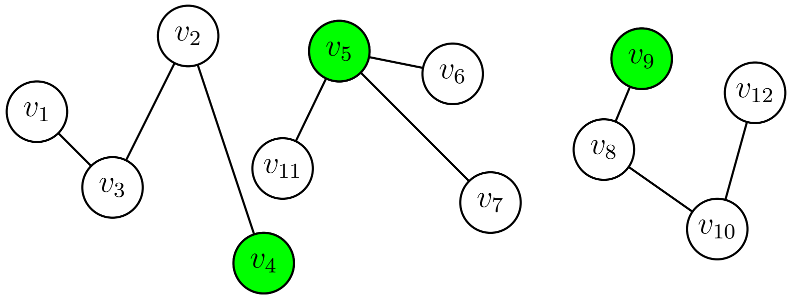
Proof. Consider an optimal solution ξ to AR on G. Say there are χ components in ξ. After adding into ξ a dummy node δ and connecting δ to vertices that are open facilities with an edge of cost f, we obtain a spanning tree T for $\tilde{G}$ of the same cost, where the vertex δ has a degree of χ. That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

2.3.1 Algorithm for AR in General Metric

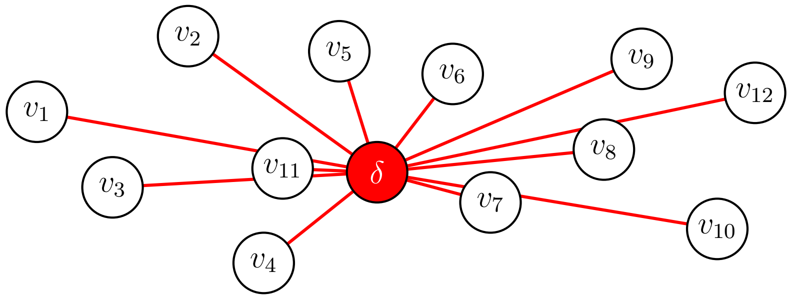
First we guess the number of components in the optimal solution. We do this by enumerating all possibilities. Say there are χ components in the optimal solution for some integer $\chi \leq n$. Note that we know $\chi \geq \lceil \frac{n}{k} \rceil$ for certain, as otherwise there must exist some cities that are not getting served.

Our algorithm is as follows:

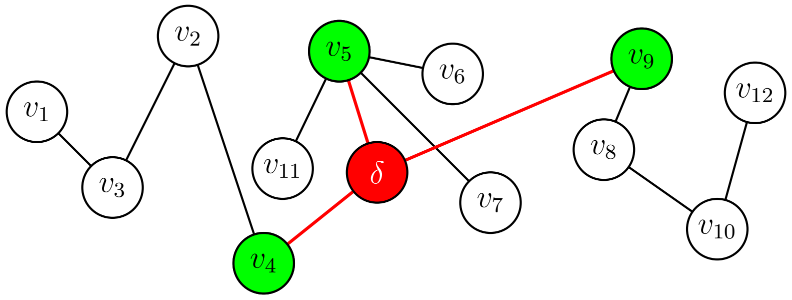
- Construct the instance $\tilde{G}$. Solve the $\mathrm{MST}_{\delta}^{\chi}$ problem on instance $\tilde{G}$. After removing the dummy vertex δ, we obtain a set $\mathscr{T} = \{T_1, T_2, \ldots T_{\chi}\}$ of χ connected components (i.e. trees).



(a) An example showing three trees each having a facility drawn in green. This is the solution to AR on G.



(b) This shows in red the changes added to G in order to generate $\tilde{G}$. The dummy node δ (red) connects to every vertex in G. Each of these red edges costs f.



(c) This shows the spanning tree T in $\tilde{G}$ created based on the aforementioned solution to AR on G.

Figure 2.5: Illustrations for proof of Lemma 8

  - Note that we solve the MST_δ^χ problem using the technique of matroid intersection. Let $M_1 = (\tilde{E}, I_1)$ represent the graphic matroid of $\tilde{G}$, where the ground set $\tilde{E}$ is the set of edges in $\tilde{G}$, and the set of independent sets I_1 consists of acyclic subgraphs of $\tilde{G}$. Let $M_2 = (\tilde{E}, I_2)$ denote the partition matroid, where the set of independent sets I_2 is defined as follows, where Δ represents all the edges incident to the vertex δ and $\tilde{V}$ is the vertex set of $\tilde{G}$,

$$I_2 = \left\{ S \subseteq \tilde{E} \;\middle|\; |S \cap \Delta| \leq \chi,\ |S \cap (\tilde{E} \setminus \Delta)| \leq |\tilde{V}| - 1 - \chi \right\}.$$

    Note that a feasible solution to MST_δ^χ 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 δ, so it is an independent set of M_2.

- For each connected component $T_i \in \mathscr{T}$, 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 $\mathscr{T}$ into a set of paths. Let $\mathscr{P}_k$ denote the set of paths with length exactly k. For each path in $\mathscr{P}_k$, we simply open one of its cities as an airport.
  - Note that $|\mathscr{P}_k| \leq \lfloor \frac{n}{k} \rfloor$ since there are at most n vertices (other than the vertex δ) in the graph. In addition, as we know $\chi \geq \lceil \frac{n}{k} \rceil$, we have $|\mathscr{P}_k| \leq \lfloor \frac{n}{k} \rfloor \leq \lceil \frac{n}{k} \rceil \leq \chi$. Consequently, the cost of opening these $|\mathscr{P}_k|$ airports is $|\mathscr{P}_k| \cdot f \leq \chi \cdot f$.
  - For those subpaths of length less than k, we simply open one of its vertices as the facility. Note that since there are $|\mathscr{T}| = \chi$ trees T_i (hence there are χ corresponding cycles C_i), we have at most χ such short subpaths. The current cost is bounded by twice the edge cost of all the trees in $\mathscr{T}$, as well as the facility cost of all these subpaths, which is at most $f \cdot \chi + |\mathscr{P}_k| \cdot f \leq 2\chi \cdot f$. Meanwhile, the cost of the MST_δ^χ solution is the edge cost of all the trees in $\mathscr{T}$, plus the cost of incident edges of δ in the solution, which is $f \cdot \chi$.

Thus, it is obvious that the cost is no more than twice the cost of the MST_δ^χ solution.

From the analysis above, we conclude with the following theorem.

Theorem 4 *This algorithm is a 2-approximation.*

Chapter 3

AR **with Nonuniform Airport Costs**

3.1 Preliminaries

In AR with non-uniform airport cost, unlike the previous chapter, each vertex ν has a potentially distinct opening cost a_ν.

A relaxation of the problem, called RELAXED AR (denoted as AR$'$, also known as Edge-Capacitated Capacitated Facility Location problem (EDGE-CAPACITATED CFL)), has been defined by Adamaszek *et al.* [3]. Informally, AR and AR$'$ differ in that each connected component χ in a feasible solution to AR$'$ is allowed to have more than one airport (denote Φ_χ as the set of airports in χ), and each edge in the input graph can be chosen multiple times, that is, the solution is allowed to contain numerous copies of any of the edges. Additionally, each connected component χ ensures the routing of demands of all its cities/clients to the airports in Φ_χ, with each airport having a capacity of k cities, and each copy of an edge also having a capacity of k demands.

As pointed out by Adamaszek *et al.* [3], it is obvious that, in each connected component χ, we can build a flow network $\mathscr{F}_\chi$ (aka transportation network) that depicts how the demands of the cities are transported to Φ_χ. That is, consider taking an instance $\mathscr{I}$ of AR$'$, along with its solution Δ. There exists a feasible flow denoted as $\mathscr{F}_\Delta$ that corresponds to the solution Δ. The flow $\mathscr{F}_\Delta$ is essentially an assignment/mapping of all the cities to all the airports, including the paths that connect each city to the airport that serves it, in accordance with the solution Δ.

Definition 16 (Relaxed AR (*i.e.* AR$'$) from [3]) *Given a graph $G = (V, E)$ where each vertex $v \in V$ has a non-negative opening cost a_v and each edge $e \in E$ has a non-negative weight c_e. Every edge and vertex has capacity $k \in \mathbb{N}_+$. Find a subset of vertices $\Phi \subseteq V$ as facilities (also known as airports), and a multiset Ξ of edges from E to get a transportation network that ensures one unit of flow from each vertex in V can be sent to facilities in Φ, without violating the capacity constraint on any edge or facility. The goal is to find such a network while minimising the total cost*

$$\sum_{v\in\Phi} a_v + \sum_{e\in\Xi} c_e.$$

Adamaszek *et al.* [3] have shown that if one has an α-approximation for the Relaxed AR, then one can obtain a $\left(2 + \frac{k}{k-1}\right) \cdot \alpha$-approximation for AR, through the procedure described in [3].

Adamaszek *et al.* [3] have proposed a DP-based QPTAS algorithm for the case of Euclidean planes ($\mathbb{R}^2$). We modify their DP algorithm for the cases of trees and graphs of bounded treewidth. In each case, we use the algorithm to find the optimal solution to AR$'$, and then use the aforementioned procedure to obtain a solution to AR. The following assumption of uncrossing flows holds in these cases.

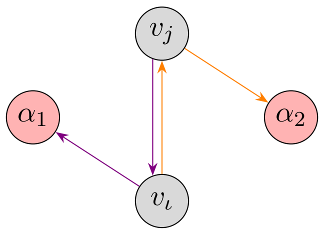


Figure 3.1: A simplest example of crossing flows in AR$'$. The red vertices are open facilities.

Lemma 9 (**Uncrossing Flows**) *One 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.

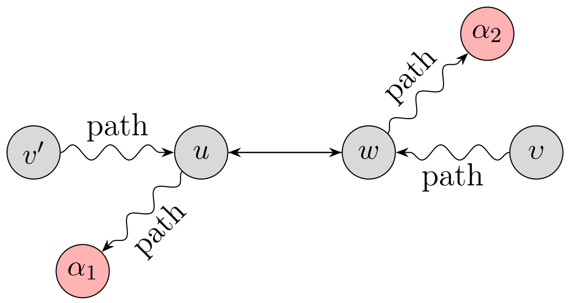


Figure 3.2: The crossing flow is at the edge uw

WLOG, 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. ■

3.2 Exact Algorithm for AR$'$ on Trees

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

$$\psi_v = \pm f_v$$

where we use $-$ (minus sign) to signify the flow is leaving T_v, and $+$ (plus sign) to signify the flow is entering T_v. f_v is the absolute value of the signed integer

ψ_v and denotes the value of the unidirectional (integral) flow passing through the vertex v and satisfies $0 \leq f_v \leq n$, where n is the number of vertices in T. Note that in AR$'$, if an edge needs to carry a flow f_v, then we need to install $\lceil \frac{f_v}{k} \rceil$ 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 $f_v = |\psi_v|$.

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

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 $0 \leq \psi_v < +k$, 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, $\lceil \frac{f_v}{k} \rceil = 1$ when $0 < \psi_v < +k$, and $\lceil \frac{f_v}{k} \rceil = 0$ when $\psi_v = 0$. If $\psi_v \geq +k$ 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 $1 \leq j \leq z$. 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 $\mathbf{D}[v, \pi_v, \psi_v] = \mathbf{B}[v, \pi_v, \psi_v, z]$.

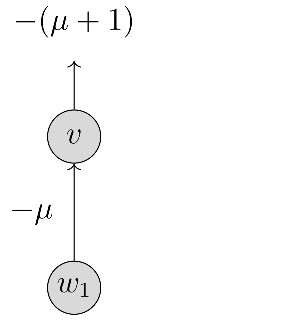
Case 1: $j = 1$. Only consider the first child of v.

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

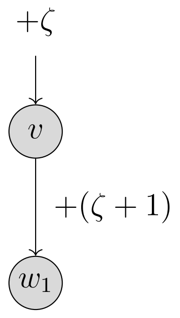
where $\eta(\pi_v^1, \psi_v^1, \psi_{w_1})$ is a Boolean indicator function that takes into account the flow on v's parent edge and the edge vw_1, as well as the decision about whether or not to open the portal v as an airport. It is true if and only if all these parameters are compatible. Recall that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

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

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

Figure 3.3: 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 we have computed all entries of the form $\mathbf{B}[v, \pi_v^{j-1}, \psi_v^{j-1}, j-1]$.

We define

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

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

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

Let e denote v's parent edge. The case $\psi_v^{j-1} + \psi_{w_j} = \psi_v^j$ means that after taking w_j (the j-th child of v) into consideration, the flow on e whilst only considering the first $j-1$ children (which is ψ_v^{j-1}), and the flow on the edge vw_j adds up to the flow on e while considering all the j children (which is ψ_v^j). This means the portal v is not absorbing any of the flow from T_{w_j}, and thus there is no need to open it as an airport if it has not been opened. The case $0 < \psi_v^j - (\psi_v^{j-1} + \psi_{w_j}) \leq k$ 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).

3.2.1 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 $O(n^4)$.

3.3 AR' on Graphs with Bounded Treewidth

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 Bodlaender and Hagerup [8]. 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 (i.e. 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}.

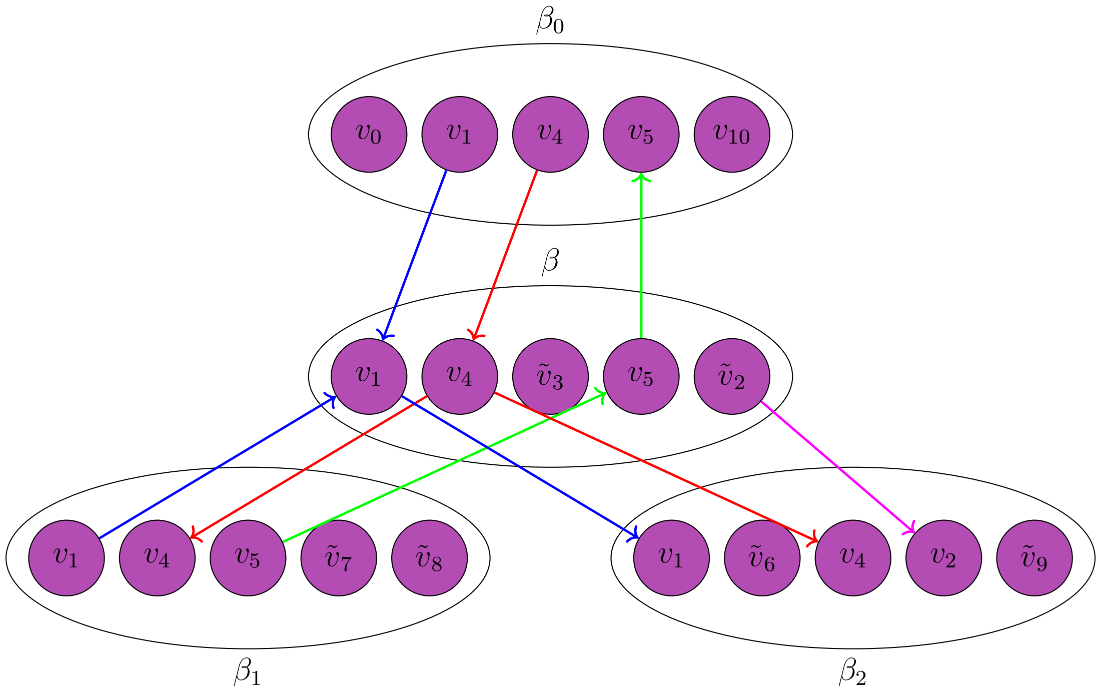


Figure 3.4: An example showing a bag β with its two children bags β_1 and β_2, and its parent bag β_0. These are the three adjacent bags of β in the tree T. The inter-bag edges in T only connect between copies of the same vertices.

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 $\Psi^v_\beta = \left(\phi_{v_{\beta_0}}, \phi_{v_{\beta_1}}, \phi_{v_{\beta_2}}\right)$, 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 $\Pi^v_\beta = \textsc{True}$. We will discuss about 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, i.e. 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 $\Pi^v_\beta = \textsc{True}$, 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 $\Psi^p_\beta = \left(\phi_{p_{\beta_0}}, \phi_{p_{\beta_1}}, \phi_{p_{\beta_2}}\right)$. For instance, if vertex $v \in \beta$ receives x units of flow outside of leaf bag β (i.e. 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 $\mathscr{F}_\beta(\Pi_\beta, \Psi_\beta)$ in β. For each Π_β and Ψ_β, we store the cost of the flow $\mathscr{F}_\beta(\Pi_\beta, \Psi_\beta)$, 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 $\mathscr{F}_\beta(\Pi_\beta, \Psi_\beta)$ 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 $\mathscr{F}_\beta(\Pi_\beta, \Psi_\beta)$, 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 $\phi_p = -f$.

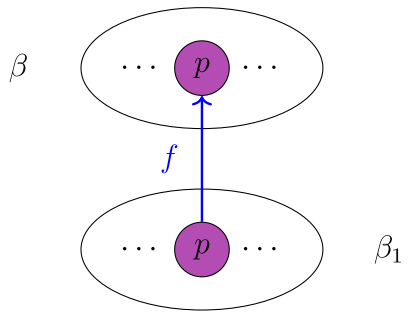


Figure 3.5: We need to check if the number of units being sent and received matches up on each of the inter-bag edges. The illustration shows an inter-bag edge "carrying" f units of demands coming from the copy $p_{\beta_1} \in \beta_1$ to $p \in \beta$.

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.

3.3.1 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 $n^{O(\omega)}2^{O(\omega)} = (2n)^{O(\omega)}$.

3.3.2 Generalisation for AR with Steiner Vertices

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

Note that this will be useful for the following sections where we extend the algorithm to other graph metrics where the host graph of the input graph (via graph embedding) may have Steiner vertices.

3.4 Extension to Other Metrics

For the AR$'$ problem, we follow a similar procedure as the one used by Jayaprakash and Salavatipour [18] to show we can make use of our algorithm for graphs with bounded treewidth to obtain a QPTAS for graphs of bounded doubling metrics, graphs of bounded highway dimension, and minor-free graphs. In each of these graph metrics, the QPTAS obtained for AR$'$ implies a constant quasi-polynomial-time approximation algorithm for AR with non-uniform airport cost.

3.4.1 Embedding Lemma for AR$'$

First, we introduce the concept of a probabilistic embedding $\phi : G \to H$ with distortion $(1+\varepsilon)$. This means for any pair of vertices v_1, v_2 in the input graph G, we have, on expectation, the distance between them in H will be at least their original distance in G, but will be upper-bounded by a factor of $(1+\varepsilon)$, i.e.

$$d_G(v_1, v_2) \leq d_H(v_1, v_2) \leq (1+\varepsilon) d_G(v_1, v_2).$$

The idea is to embed the input graph G into a host graph H through such a probabilistic embedding $\phi : G \to H$, and solve the AR$'$ problem on the host graph H. If we can obtain a near-optimal solution S_H on H, we can lift the solution S_H back into a solution for G, with a factor-ε extra cost.

Consider a probabilistic embedding $\phi : G \to H$ with distortion $(1+\varepsilon)$ for an arbitrary positive number ε and G is an input graph. Let OPT_G denote the optimal solution to AR$'$ on G and OPT_G denote its cost. We use ϕ to embed G into a host graph H and solve AR$'$ on H to obtain the solution OPT_H. We obtain a feasible solution to AR$'$ on G by lifting the solution OPT_H from H to G, where $\mathrm{cost}_G(\mathrm{OPT}_H) \leq (1+\varepsilon)\mathit{OPT}_G$. This is easy to see.

In the following sections, we will prove a QPTAS for AR$'$ for different graph metrics, which then proves a constant quasi-polynomial-time approximation algorithm for AR. See the previous chapter for the relationship between AR$'$ and AR.

3.4.2 Constant Quasi-Polynomial-Time Approximation for Graphs of Bounded Doubling Dimension

We achieve this by proving a QPTAS for AR$'$. We begin by introducing the following lemma proposed by Talwar [28] 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 10 *(Theorem 9 in [28]) 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 [18] to get the following result.

Theorem 5 *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 (see Section 3.3). We see this follows from Lemma 10. 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. ■

3.4.3 Constant Quasi-Polynomial-Time Approximation for Graphs of Bounded Highway Dimension

We achieve this by proving a QPTAS for AR$'$. We begin by introducing the following lemma proposed by Feldmann *et al.* [13] 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 11 *(Theorem 1.3 in [13]) 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 [18] to get the following result.

Theorem 6 *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. We use the algorithm A of AR' for graphs with bounded treewidth as a subroutine. We see this follows from Lemma 11. 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. ■

3.4.4 Constant Quasi-Polynomial-Time Approximation for Minor-Free Graphs

We achieve this by proving a QPTAS for AR'. We begin by introducing the following lemma proposed by Cohen-Addad *et al.* [10] 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 12 *(Theorem 1.1 in [10]) 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 7 *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. We use the algorithm A of AR$'$ for graphs with bounded treewidth as a subroutine. We see this follows from Lemma 12. Essentially, we embed the minor-free graph G into a 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 algorithm A. 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. ■

As Cohen-Addad *et al.* [10] pointed out, in terms of planar graphs, which exclude K_5 as a minor, with edge weights in $[1, n^c]$ for any constant c, the upper bound of the treewidth ω obtained from the theorem is polylogarithmic in n, namely we have $\omega \in O\left(\log^{13} n/\varepsilon\right)$, as $\ell \in \log(n^c) = O(\log n)$.[1]

Corollary 1 *For any $\varepsilon > 0$, given an input graph G of the* AR$'$ *problem where G is a planar graph, there exists an algorithm that finds a $(1+\varepsilon)$-approximate solution in time $n^{O\left(\log^{13} n/\varepsilon\right)}$.*

[1] ℓ is in $O(\log n)$ assuming the aspect ratio Δ is polynomial in n.

Chapter 4

Conclusion and Future Problems

Throughout the thesis, we have studied AR with uniform and non-uniform facility opening costs, in both trees and graphs of bounded treewidth. We have also extended the algorithm for AR with non-uniform facility opening costs to graphs of bounded doubling metric, graphs of bounded highway metric and minor-free graphs. In addition, for the general metrics, we have obtained a constant approximation for AR with a uniform facility opening cost, and a logarithmic approximation for AR with non-uniform facility opening costs. In view of all the quasi-polynomial time approximation schemes (QPTAS) in our current results, an important point for consideration involves improving these QPTAS to achieve polynomial-time approximation schemes (PTAS).

A crucial open problem is to get a constant approximation for the general metric. Below we will show that, if you can get a constant approximation for the case of AR where the facilities are given free, then you can get a constant approximation for AR with general airport costs.

4.1 The $0/+\infty$ Case of AR

The $0/+\infty$ case of AR is a special setting of the AR problem where the facilities are predetermined and given as free, which is equivalent to being defined as a subset of the vertices having a zero opening cost, and the rest of the vertices having opening cost $+\infty$.

Getting an $O(1)$-approximation algorithm for this problem has remained open. This manifests the intractability of the AR problem in more general cases. In the following, we show how to reduce the metric AR problem with general facility opening costs to the $0/+\infty$ case with an $O(1)$-approximation algorithm.

We will prove that we can transform the general cost case of AR in general metric into the $0/+\infty$ case and show that the metric properties are not violated during the process. Without loss of generality, we assume the set of clients $\mathscr{C}$ and potential facilities $\mathscr{F}$ are disjoint.[1]

We first note that there exists a near-optimum factor-2 solution where the degree of each opened facility is two. To see this, let us consider each component in the result of AR, which is simply a tree. We double all the edges of the tree and trace along the new edge set and shortcut if necessary. This way we can transform the original result, which is a forest, into a set of cycles, with the cost increasing by a factor of at most 2. That is, for every solution Σ to AR on the general metric graph G, we can convert Σ into a solution Σ' of cost at most $2\text{cost}(\Sigma)$ through this procedure. We define the following CYCLE-AR problem.

Definition 17 (CYCLE-AR) *The goal is to find a min-cost set of cycles each having at most k vertices, using edges in the graph. Each cycle C has one opened facility.*

Note that since we have assumed the set of potential facilities and the set of clients are disjoint, and considering we only need to cover the clients, we might have some potential facilities not used at all in the solution. We will consider, in particular, the CYCLE-AR problem in the $0/+\infty$ setting, i.e. $0/+\infty$ CYCLE-AR. This means the input graph is also an instance to the $0/+\infty$ AR problem.

Theorem 8 *If there is a c-approximation algorithm for the* $0/+\infty$ CYCLE-AR *problem for some constant $c > 1$, then there exists a $2c$-approximation algorithm for the general metric* AR *problem.*

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

Proof. 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 a_v onto each of its incident edges, that is, we update the cost of each edge e incident to v from c_e to $c_e + \frac{1}{2}a_v$ and set v's opening cost to zero.

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 $\textsc{opt}$ and $\textsc{opt}'$ 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 $\textsc{opt}' \leq 2\textsc{opt}$.

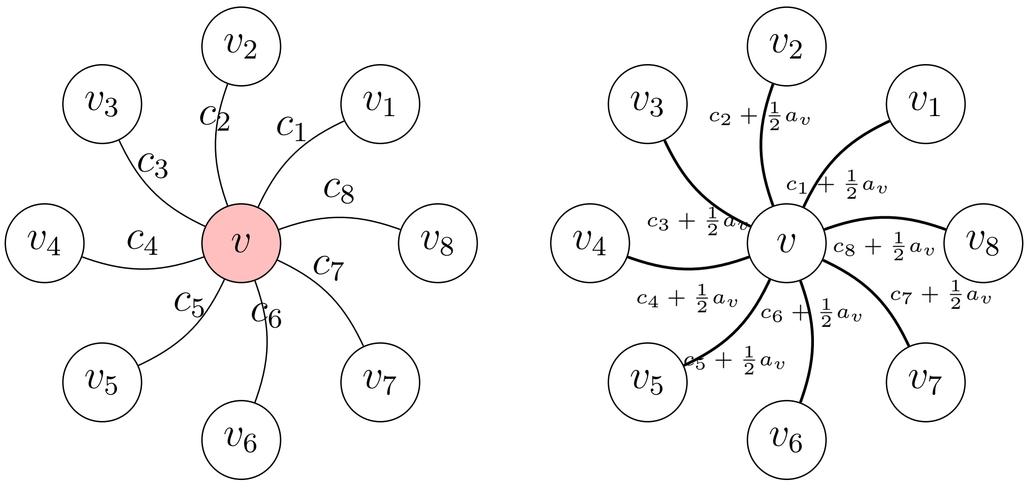


Figure 4.1: Incident edges of each potential facility before and after changes

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 $c\textsc{opt}'$, then Ψ is a solution to CYCLE-AR on graph G with the same cost, i.e. $2c \cdot \textsc{opt}$, as $\textsc{opt}' \leq 2\textsc{opt}$. Since a solution to CYCLE-AR on graph G can be easily transform 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 $2c \cdot \textsc{opt}$. ■

Now we show the transformation from G to G' preserves the triangle inequality.

Lemma 13 *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 α_A, α_B and α_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.

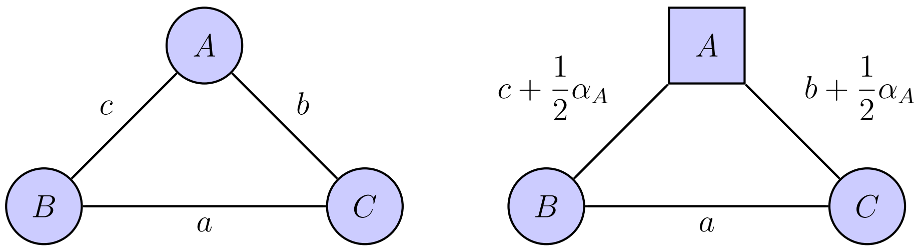
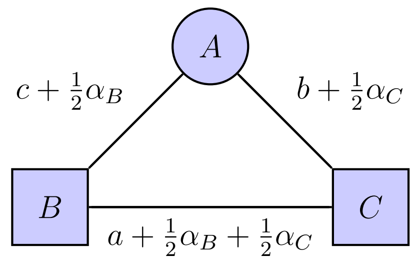


Figure 4.2: On the left is the original triangle in G. On the right is the scenario in G' where one of the vertex is a facility.

Say only one of the vertices is a facility. WLOG, assume A is the facility. It is easy to see the following inequalities hold.

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

Say two of the vertices are facilities. WLOG, assume B and C are the facilities.



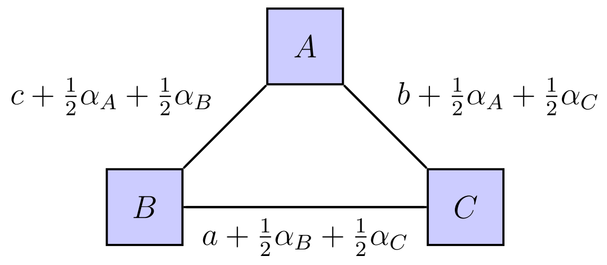

A
B
C
$c+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_B$
$b+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_C$
$a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C$

Figure 4.3: 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.

It is easy to see the following inequalities hold.

$$\left(a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C\right)+\left(b+\frac{1}{2}\alpha_C\right)\geq c+\frac{1}{2}\alpha_B$$
$$\left(a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C\right)+\left(c+\frac{1}{2}\alpha_B\right)\geq b+\frac{1}{2}\alpha_C$$
$$\left(b+\frac{1}{2}\alpha_C\right)+\left(c+\frac{1}{2}\alpha_B\right)\geq a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C$$

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

$$\left(a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C\right)+\left(b+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_C\right)\geq c+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_B$$
$$\left(a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C\right)+\left(c+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_B\right)\geq b+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_C$$
$$\left(b+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_C\right)+\left(c+\frac{1}{2}\alpha_A+\frac{1}{2}\alpha_B\right)\geq a+\frac{1}{2}\alpha_B+\frac{1}{2}\alpha_C$$

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

References

[1] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck, “VC-dimension and shortest path algorithms,” in *International Colloquium on Automata, Languages, and Programming*, Springer, 2011, pp. 690–699.

[2] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck, “Highway dimension and provably efficient shortest path algorithms,” *Journal of the ACM (JACM)*, vol. 63, no. 5, pp. 1–26, 2016.

[3] A. Adamaszek, A. Antoniadis, A. Kumar, and T. Mömke, “Approximating Airports and Railways,” in *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, R. Niedermeier and B. Vallée, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 96, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:13, ISBN: 978-3-95977-062-0. DOI: 10.4230/LIPIcs.STACS.2018.5. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2018/8518.

[4] A. Adamaszek, A. Antoniadis, and T. Mömke, “Airports and Railways: Facility Location Meets Network Design,” in *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, N. Ollinger and H. Vollmer, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 47, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 6:1–6:14, ISBN: 978-3-95977-001-9. DOI: 10.4230/LIPIcs.STACS.2016.6. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/5707.

[5] E. T. Bell, “Exponential polynomials,” *Annals of Mathematics*, pp. 258–277, 1934.

[6] E. T. Bell, “The iterated exponential integers,” *Annals of Mathematics*, pp. 539–557, 1938.

[7] D. Berend and T. Tassa, “Improved Bounds on Bell Numbers and on Moments of Sums of Random Variables,” *Probability and Mathematical Statistics*, vol. 30, no. 2, pp. 185–205, 2010.

[8] H. L. Bodlaender and T. Hagerup, "Parallel Algorithms with Optimal Speedup for Bounded Treewidth," *SIAM Journal on Computing*, vol. 27, no. 6, pp. 1725–1746, 1998. DOI: 10.1137/S0097539795289859. eprint: https://doi.org/10.1137/S0097539795289859. [Online]. Available: https://doi.org/10.1137/S0097539795289859.

[9] J. Bourgain, "On Lipschitz embedding of finite metric spaces in Hilbert space," *Israel Journal of Mathematics*, vol. 52, pp. 46–52, 1985.

[10] V. Cohen-Addad, H. Le, M. Pilipczuk, and M. Pilipczuk, *Planar and Minor-Free Metrics Embed into Metrics of Polylogarithmic Treewidth with Expected Multiplicative Distortion Arbitrarily Close to* 1, 2023. arXiv: 2304.07268 [cs.DS].

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[12] M. Cygan, F. V. Fomin, Ł. Kowalik, *et al.*, *Parameterized algorithms*. Springer, 2015, vol. 4.

[13] A. E. Feldmann, W. S. Fung, J. Könemann, and I. Post, "A $(1 + \epsilon)$-Embedding of Low Highway Dimension Graphs into Bounded Treewidth Graphs," *SIAM Journal on Computing*, vol. 47, no. 4, pp. 1667–1704, 2018. DOI: 10.1137/16m1067196. [Online]. Available: https://doi.org/10.1137%2F16m1067196.

[14] M. Gromov, *Metric Structures for Riemannian and Non-Riemannian Spaces*. Jan. 2007, ISBN: 978-0-8176-4582-3. DOI: 10.1007/978-0-8176-4583-0.

[15] P. Hall, "On Representatives of Subsets," *Journal of The London Mathematical Society-second Series*, pp. 26–30, 1935. [Online]. Available: https://api.semanticscholar.org/CorpusID:23252557.

[16] M. Hatzel and H. Seidler, *Advanced Algorithmics*, Acknowledgements: This unofficial document contains notes for the lecture "Advanced Algorithmics" given by Prof. Rolf Niedermeier (TU Berlin). We assume/accept no liability for being complete, correct and/or up-to-date. This script was initiated by Meike Hatzel and Henning Seidler., 2015. [Online]. Available: https://fpt.akt.tu-berlin.de/notes/aa-overview.pdf.

[17] J. Heinonen, *Lectures on Analysis on Metric Spaces*. Springer Science & Business Media, 2001.

[18] A. Jayaprakash and M. R. Salavatipour, "Approximation Schemes for Capacitated Vehicle Routing on Graphs of Bounded Treewidth, Bounded Doubling, or Highway Dimension," *ACM Trans. Algorithms*, vol. 19, no. 2, 2023, ISSN: 1549-6325. DOI: 10.1145/3582500. [Online]. Available: https://doi.org/10.1145/3582500.

[19] J. Kleinberg and E. Tardos, *Algorithm Design*, 2003.

[20] B. H. Korte, J. Vygen, B Korte, and J Vygen, *Combinatorial Optimization*. Springer, 2011, vol. 1.

[21] A. Lee and B. Xu, "Classifying approximation algorithms: Understanding the APX complexity class," *CoRR*, vol. abs/2111.01551, 2021. arXiv: 2111.01551. [Online]. Available: https://arxiv.org/abs/2111.01551.

[22] J. Matoušek, "Embedding finite metric spaces into normed spaces," in *Lectures on Discrete Geometry*, Springer, 2002, pp. 355–400.

[23] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge University Press, 2017.

[24] C. Moore and S. Mertens, *The Nature of Computation*. OUP Oxford, 2011.

[25] V. Nagarajan. "Approximation algorithms - lecture notes." Chernoff Bounds, Multicommodity Routing. (Feb. 2017), [Online]. Available: https://viswa.engin.umich.edu/wp-content/uploads/sites/169/2016/12/12.pdf.

[26] S. Norin, *Graph Theory & Combinatorics*, Instructor: Prof. Sergey Norin, Notes by: Tommy Reddad, Last updated: September 12, 2015. [Online]. Available: https://www.math.mcgill.ca/snorin/math350f2015/notes.pdf.

[27] M. Sipser, "Introduction to the Theory of Computation," *ACM Sigact News*, vol. 27, no. 1, pp. 27–29, 1996.

[28] K. Talwar, "Bypassing the Embedding: Algorithms for Low Dimensional Metrics," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04, Chicago, IL, USA: Association for Computing Machinery, 2004, 281–290, ISBN: 1581138520. DOI: 10.1145/1007352.1007399. [Online]. Available: https://doi.org/10.1145/1007352.1007399.

[29] V. V. Vazirani, *Approximation Algorithms*. Springer Science & Business Media, 2013.

[30] J. Verstraete, *Introduction to Graph Theory: A Short Course*, Department of Mathematics, University of California at San Diego, California, U.S.A, 2020. [Online]. Available: https://cseweb.ucsd.edu/~dakane/Math154/154-textbook.pdf.

[31] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[32] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge University Press, 2011.

[33] R. Wilson, *Introduction to Graph Theory*. Longman, 2010, ISBN: 9780273728894. [Online]. Available: https://books.google.ca/books?id=wwxTRAAACAAJ.