Airports and Railways with Unsplittable Demand

Hossein Jowhari, Shamisa Nematollahi

Airports and Railways with Unsplittable Demand

Hossein Jowhari[1*] Shamisa Nematollahi[2,3 †]

[1]K. N. Toosi University of Technology. Tehran, Iran.
[2]CNRS & IRIF, Université Paris Cité.
[3]Shahid Rajaee Teacher Training University, Tehran, Iran.

September 8, 2024

Abstract

In the problem of airports and railways with unsplittable demand (ARUD), we are given a complete graph $G = (V, E)$ with weights on the vertices $a : V \to \mathbb{R}^+$, and the length of the edges $\ell : V \times V \to \mathbb{R}^+$. Additionally, a positive integer k serves as the capacity parameter. We are also provided with a function $b : V \to \mathbb{N}$ that defines a non-zero demand for each city. The goal is to compute a spanning forest R of G and a subset $A \subseteq V$ of minimum cost such that each component in R has one open facility and the total demand in each component is at most k (the capacity constraint). The cost of the solution (A, R) is defined as $\sum_{v \in A} a(v) + \sum_{e \in E(R)} \ell(e)$. This problem is a generalization of the Airport and Railways (AR) problem introduced by Adamaszek *et al.* (STACS 2016). In Adamaszek *et al.* version, each vertex has a unit demand.

This paper presents a bi-criteria approximation algorithm for the metric ARUD problem in the sense that the algorithm is allowed to exceed the capacity constraints by $O(k)$ while the cost of the solution is compared with the cost of an optimal solution that does not violate the capacity constraint. Our approach builds upon an existing approximation algorithm for the metric AR problem, developed by Adamaszek *et al.* (STACS 2018), and further leverages the well-known rounding algorithm of Shmoys and Tardos for the Generalized Assignment Problem (GAP). Assuming the total demand is polynomially bounded in the number of vertices, our algorithm runs in polynomial time. We also show that it is NP-hard to find an approximate solution for ARUD within any factor without violating the capacity constraints. This is the case even when each demand is polynomially bounded in the number of vertices. Furthermore, we determine the complexity of ARUD for some fixed values of k.

*The author is also affiliated with the School of Mathematics at the Institute for Research in Fundamental Sciences (IPM) in Tehran, Iran. Email: jowhari@kntu.ac.ir

†The author has received funding from the French Agence Nationale de la Recherche (ANR), under grant ANR-21-CE48-0016 (project COMCOPT). Email: shamisa@irif.fr

1 Introduction

The work by Adamaszek et al. [AAM16] integrates the facility location problem with network design in the following manner. Suppose we have a set of cities that require airports. We can build an airport in a city, directly meeting the demand of that city, or we can connect the city to an airport constructed elsewhere via railways. Naturally, airports have capacity limits; each airport can serve at most k cities (or, more generally, k units of demand). Under these constraints, the objective is to minimize the total cost required to satisfy the cities' demands. The total cost comprises both the expense of constructing the airports and the cost of building the railway network.

The Airports and Railways problem (AR), as articulated in [AAM16], can be described in a more formal manner. We are presented with a complete graph $G = (V, E)$ where V corresponds to the cities and E is the potential railways between the cities. We are also given $a : V \to \mathbb{R}^+$ which defines the cost of opening facilities (airports) at the vertices. Additionally, $\ell : E \to \mathbb{R}^+$ defines the cost of the edges (the cost of constructing railways between the cities.) The parameter k in the input designates the capacity of each facility. The goal is to select a subset $A \subseteq V$ (the facilities) and a spanning forest $R = \{R_1, R_2, \ldots\}$ of G such that each component R_i of R has one vertex in A, i.e. has an opened facility. Moreover, the size of each component is at most k (the capacity constraint.) The cost of a feasible solution (A, R) is measured as

$$\mathsf{COST}(A, R) = \sum_{u \in A} a(u) + \sum_{e \in E(R)} \ell(e).$$

We define the unsplittable demand version of the problem, denoted by ARUD, by introducing the function $b : V \to \mathbb{N}^+$, which determines the demand of each vertex. Similarly, we have the capacity constraints

$$\forall R_i \in R, \quad \sum_{u \in R_i} b(u) \leq k\,.$$

Note that in the AR problem, we have $b(u) = 1$ for all $u \in V$, while in ARUD, we have $1 \leq b(u) \leq k$ for each $u \in V$. In the metric version of the AR problem (also in ARUD), the length function $\ell : V \times V \to \mathbb{R}^+$ defines a metric over V. In this paper, we establish the convention that any reference to the AR or ARUD problem will pertain to their metric version, unless explicitly indicated otherwise.

Exact complexity of ARUD. As observed in [AAM16], AR and consequently ARUD are NP-hard even for Euclidean graphs with unit airport costs. In a subsequent work [AAKM18], the authors have shown a polynomial time algorithm for the case where $k = \infty$. Their algorithm works for non-metric length functions as well. Here, we show in the case of $k = 2$, the ARUD problem (with even non-metric edge lengths) admits a polynomial time algorithm. This fact is established through a reduction to the min-cost perfect matching problem. However, when $k = 3$, we show metric AR is NP-hard even for inputs with uniform airport costs. The proofs of these facts are presented in Section 3.

Approximation algorithms for ARUD. A simple reduction from the Bin-Packing problem illustrates that there is no polynomial time approximation algorithm for ARUD that satisfies the capacity constraint. More formally we have the following hardness result for the ARUD problem.

Theorem 1.1. *Unless $P = NP$, there is no polynomial time approximation algorithm for the ARUD problem without violating the capacity constraint, even when the demand of each vertex is polynomially bounded in the number of vertices.*

We give a proof of this theorem in Section 3. It must be noted a similar fact has been previously reported regarding related problems, including the capacitated facility location problem with unsplittable demand [BH12, BSS16].

To overcome the computational barrier of approximating ARUD, we resort to bi-criteria approximation algorithms. More specifically, we consider algorithms that are allowed to violate the capacity constraint in a limited manner [1] The following definition formalizes the concept of bi-criteria (or bi-factor) approximation algorithms for the ARUD problem.

Definition 1.2. *Let $\mathsf{OPT}(\mathcal{I})$ denote an optimal solution for a given instance $\mathcal{I}$ of the ARUD problem. Let $f \geq 1$ and $f' \geq 1$ be reals. Algorithm $\mathcal{A}$ is called a (f, f') approximation algorithm for ARUD if given an instance $\mathcal{I}$ the algorithm finds a solution (A, R) with the cost at most f times $\mathsf{COST}(\mathsf{OPT}(\mathcal{I}))$. The solution (A, R) satisfies every constraint except that the size of each component in R is allowed to exceed k but not $f'k$.*

Note that, unlike ARUD, the AR problem can be approximated within a factor of $O(\log n)$ without violating the capacity constraints. This fact has been shown in a recent work by Salavatipour and Tian [ST24]. On the other hand, there exists a bi-factor approximation algorithm for the AR problem (with better approximation factor) as shown by Adamaszek *et al.* [AAKM18].

Lemma 1.3. *[AAKM18] Let $\varepsilon \in (0, 1]$ such that $k\varepsilon$ is an integer. There is a polynomial time $(\frac{4}{3}(2 + \frac{1}{\varepsilon}), 1 + \varepsilon)$ bi-factor approximation algorithm for the AR problem.*

In this paper we show a similar result for the ARUD problem by demonstrating a reduction to the AR problem. The following theorem, elaborated in Section 2, is the main technical achievement of this paper.

Theorem 1.4. *Given a polynomial time (α, β) bi-factor approximation algorithm for the AR problem, there is a polynomial time $(2\alpha, \beta+2)$ bi-factor approximation algorithm for the ARUD problem assuming the demands are polynomially bounded in the number of vertices.*

In order to show this result, we transform an instance of ARUD into an instance of AR by considering each unit demand as a separate vertex. This approach permits the division of demands and their routing to various facilities. Subsequently, we address the resulting uniform demand scenario by employing an algorithm designed for the AR problem, such as the one derived from Lemma 1.3. The solution obtained is cost-effective when compared to the optimal solution; however, it results in split demands. To merge the distributed demands and prevent them from being split, we apply the rounding algorithm developed by Shmoys and Tardos for the Generalized Assignment Problem [ST93, STA97]. This method allows us to achieve a cost-efficient solution that adheres to capacity constraints with only minor infringements. Consequently we get the following corollary. Note that here we have used the result of Lemma 1.3 while setting $\varepsilon = 1$.

Corollary 1.5. *There is a polynomial time $(4, 4)$ bi-factor approximation algorithm for the ARUD problem assuming the input demands are polynomially bounded in $|V|$.*

Note that, in a similar manner, we get a $(O(\log n), 3)$ bi-factor approximation algorithm by applying the result of Salavatipour and Tian [ST24] as mentioned above.

[1] In some texts, this is referred to as approximation algorithms with resource augmentations.

1.1 Related Works

The ARUD problem extends traditional optimization challenges, including Bin-Packing (refer to Theorem 1.1) and Minimum Spanning Tree. Additionally, this problem is connected to well-researched optimization problems such as the Capacitated Facility Location with Unsplittable Demand [STA97, BH12, BSS16]. The latter bears a resemblance to ARUD, with the distinction that the components within the solution must adhere to a star topology. The works [AAM16, AAKM18] consider variants of the AR problem where the topology of the components is required to be a path. Another important related problem is the Capacitated Tree Covering Problem studied in [Trö19, TT20]. This problem is a special case of ARUD where the airport costs are uniform. The authors in [TT20] have shown a $1 + \frac{1}{7}$ approximation algorithm for this problem. Another well-studied related problem is the Capacitated Vehicle Routing Problem (CVRP) introduced by Dantzig and Ramser in 1959 [DR59]. In this problem, the goal is to cover all the vertices with a network of cycles with minimum cost such that each cycle contains a common vertex which is a network depot. Haimovich and Rinnooy Kan in [HRK85] gave the first constant factor approximation algorithm for the unsplittable demand version of CVRP. Blauth *et al.* [BTV21] have improved the approximation factor for this problem. Friggstad *et al.* [FMRS21] and Grandoni *et al.* [GMZ22] have given the best-to-date approximation ratios for CVRP in general metrics and in the Euclidean plane, respectively. Mathieu and Zhou gave a $(1.5+\epsilon)$-approximation algorithm for the unsplittable CVRP and a PTAS for the unit demand CVRP on trees [MZ23a, MZ23b]. Another related problem is the Capacitated Minimum Spanning Tree [JR05] where the goal is to find a network of trees with the minimum cost under the requirement that each tree should be connected to a specified root and satisfy a certain capacity constraint.

The Generalized Assignment Problem (GAP) and the rounding algorithm in [ST93] have been widely used in the context of approximation algorithms for various combinatorial optimization problems [ST93, STA97, KPR00, DKK$^+$00, RV18, BCFN19].

1.2 Preliminaries

The Generalized Assignment Problem (GAP). The Generalized Assignment Problem, as introduced by [ST93], involves a collection of jobs J and machines M. The objective is to assign each job $j \in J$ to a machine $i \in M$, where executing job j on machine i requires p_{ij} units of processing and has a cost of r_{ij}. In addition, each machine has a processing capacity of P_i, with a total budget constraint of B. The aim is to determine an assignment of jobs to the machines that meet the processing capacity of each machine while keeping the total cost below or equal to B.

The problem can be modeled (fractionally) with the following set of linear constraints. Here x_{ij} represents a fraction of job j that has been assigned to the machine j.

$$\sum_{i \in M} x_{ij}=1 \text{ for each job } j \in J \tag{1}$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq P_i \text{ for each machine } i \in M \tag{2}$$

$$\sum_{i \in M} \sum_{j \in J} r_{ij} x_{ij} \leq B \tag{3}$$

$$x_{ij} \geq 0 \text{ for each } i \in M \text{ and } j \in J \tag{4}$$

Shmoys and Tardos [ST93] have proven the following lemma (see also the discussion in [STA97] for a similar application of this result.)

Lemma 1.6. *Any feasible solution x can be rounded, in polynomial time, to an integer solution that is feasible if the right-hand side of (2) is relaxed to $P_i + \max_j p_{ij}$.*

2 A bi-factor approximation algorithm

This section presents our bi-factor approximation algorithm for the ARUD problem. Our algorithm consists of several steps, which are detailed in the following subsections. The main steps of the algorithm are summarized in Algorithm 3 (the Main Procedure).

2.1 Converting to uniform demand

Given the input $\mathcal{I} = (G, a, \ell, b, k)$, we begin by running the procedure **ConvertToUniform** to create a larger instance $\mathcal{I}^+$ with uniform demand. The description of **ConvertToUniform** procedure is stated in Algorithm 1. The following observation is crucial in the analysis of our algorithm.

Algorithm 1 ConvertToUniform Procedure

Input: The instance $\mathcal{I} = (G, a, \ell, b, k)$ with non-uniform demand
Output: The instance $\mathcal{I}^+ = (G^+, a^+, \ell^+, k)$ with uniform demand

1. Let G^+ be an empty graph.
2. For each $u \in V(G)$ with demand $b(u)$, add the set of vertices $X(u) = \{u_1, \ldots, u_{b(u)}\}$ to G^+. Note that for each $u \in V(G)$ and $i \in [b(u)]$, u_i represents a vertex with unit demand in G^+.
3. For each $u \in V(G)$, set $a^+(u_1) = a(u)$ and $a^+(u_i) = \infty$ for $i > 1$. In other words, all the vertices in the cluster $X(u)$ have infinite costs except for one of them.
4. For each $u_i \in X(u)$ and $v_j \in X(v)$, let $\ell^+(u_i, v_j) = \ell(u, v)$.

Observation 2.1. $\mathit{COST}(\mathit{OPT}(\mathcal{I}^+)) \leq \mathit{COST}(\mathit{OPT}(\mathcal{I}))$.

Proof. Let (A^*, R^*) be an optimal solution for the non-uniform instance $\mathcal{I}$. The solution (A^*, R^*) can be converted to a solution for the uniform demand instance $\mathcal{I}^+ =$ **ConvertToUniform**$(\mathcal{I})$ without increasing the cost. To explain, a tree $R_i^* \in R^*$ that goes through a vertex $u \in G$, can be extended to a tree R_i^+ that spans all the vertices in the cluster $X(u)$. Since the lengths of the edges within the clusters are zero, the transformation does not add an extra cost. Moreover for $u \in A$, in the solution $\mathcal{I}^+$, we open a facility at $u_1 \in X(u)$, which has the same cost by definition. □

2.2 A solution with divided demands

Using the algorithm in [AAKM18], we compute a solution (A^+, R^+) for the uniform instance $\mathcal{I}^+$. The following observation about (A^+, R^+) is used in the subsequent arguments.

Observation 2.2. *Let $R^+ = \{R_1^+, \cdots, R_q^+\}$ be the trees in the forest R^+. Each component R_i^+ has exactly one opened facility and at most βk number of vertices. Moreover, each cluster $X(u)$ in G^+ has at most one opened facility.*

Proof. These facts follow from the description of **ConvertToUniform** procedure and the guarantees of the algorithm in Lemma 1.3. □

2.3 Building a non-split assignment

The pair (A^+, R^+) may not represent a viable solution for the non-uniform instance $\mathcal{I}$. Primarily, the forest R^+ spans a different graph, and more significantly, the demand within a cluster $X(u)$ could be directed towards various facilities. See Figure 1 for an example. Given (A^+, R^+), we build a solution (A, R) for the unsplittable instance $\mathcal{I}$ in several steps. First, we determine the set of facilities A which is the easiest part.

Definition 2.3. *For each $x \in V(G^+)$, let $Super(x)$ denote the vertex $u \in V(G)$ where $x \in X(u)$.*

We let $A = \{Super(x) \,|\, x \in A^+\}$. In other words, if $X(u)$ contains a facility in the graph G^+, we declare $u \in V(G)$ as a facility in graph G. The following observation follows from the definitions.

Observation 2.4. *Let $\mathsf{COST}(A^+) = \sum_{x \in A} a^+(x)$ and similarly let $\mathsf{COST}(A) = \sum_{x \in A} a(x)$. We have $\mathsf{COST}(A^+) = \mathsf{COST}(A)$.*

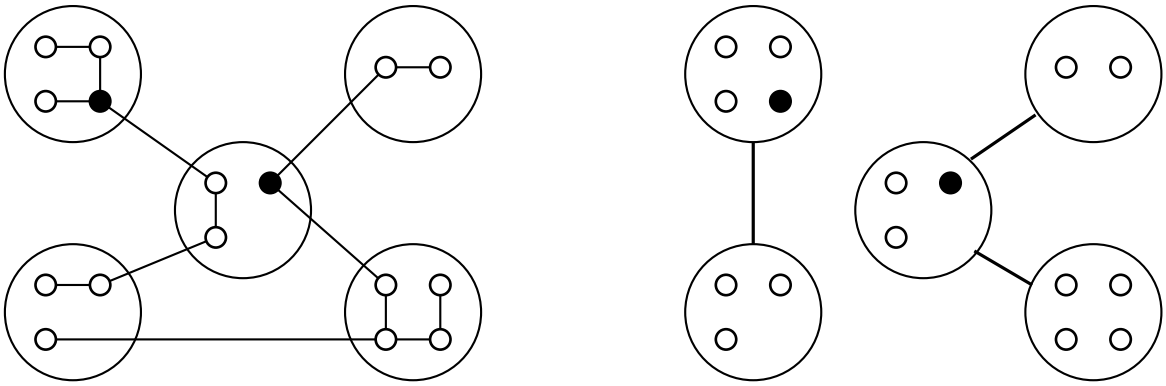
Concerning the facilities, we do not observe an escalation in the transformation cost. The primary challenge lies in identifying a suitable assignment $F : V(G) \to A$ that respects the capacity constraints in a reasonably approximate manner, while also ensuring that the connection cost does not rise significantly. For $v \in V(G^+)$, let $F^+(v)$ denote the assigned facility of v under (A^+, R^+). A crucial aspect to consider is that, to prevent a substantial rise in the connection costs, *when we set $F(u) = f$ where $f \in A$ we ensure there is a vertex $v \in X(u)$ which is served by a facility in $X(f)$ under the assignment F^+.* Note that the solution (A^+, R^+) has already paid the cost of connecting v to its designated facility in A^+. Therefore, intuitively, we expect a small increase in the cost of the railways. To achieve this, we follow the steps in the description of **NonSplitAssignment** procedure.

Algorithm 2 NonSplitAssignment Procedure

Input: $A, F^+ : V(G^+) \to A^+$
Output: $F : V(G) \to A$

1. Let C be the clusters in G^+ without an open facility.
2. Run the procedure $\mathbf{GAP}(C, A)$ as explained in Section 2.3.1. Let $F' : C \to A$ be the returned assignment.
3. For each cluster $S \in C$, choose an arbitrary vertex x from S and let $F(Super(x)) = F'(S)$.
4. For each cluster S that has an open facility $y \in A^+$, we let $F(Super(y)) = Super(y)$.

The steps $1 - 3$ of the procedure handle the clusters without an open facility. To find facilities for these clusters, we use a reduction to the Generalized Assignment Problem and the rounding algorithm of Lemma 1.6.



(a) (b)

Figure 1: (a) A pictorial representation of an intermediate solution with divided demands. Big circles represent the cluster of vertices. Vertices in black represent the open facilities. (b) The figure shows the associated non-split solution after running the steps 4-5 of the Main Procedure in Algorithm 3

2.3.1 Reduction to GAP

To model the problem as a GAP instance, the clusters without an open facility, namely $C = \{C_j\}_{j\in J}$, represent the jobs and the facilities $A = \{a_i\}_{i\in M}$ represent the machines. For all facilities $i \in M$, we set the capacity $P_i = \beta k$. For cluster C_j and the facility a_i, we set $p_{ij} = |C_j|$. We set $r_{ij} = 0$ iff the component in R^+ that owns the open facility in $X(a_i)$ intersects with C_j, otherwise it is set to infinity (a large enough number). Finally we set $B = 0$. Note that the last two assignments are done to prevent the allocation of a cluster C_j to a facility a_i where the open facility in $X(a_i)$ does not serve a vertex in C_j according to the assignment F^+.

Fact 2.5. *Using the setup as detailed above, the associated GAP instance has a feasible fractional solution x.*

Proof. If the cluster C_j has t members that are served by the facility a_i, we set $x_{ij} = \frac{t}{|C_j|}$. All other variables are set to zero. The reader can check that this is a feasible solution. □

Consequently, by Lemma 1.6, in polynomial time, we can round the solution x and compute an assignment $F' : C \to A$ such that meets the requirement mentioned above, and further no facility receives more than k new vertices. Note that the extra k vertices come from the GAP rounding algorithm.

The clusters with an open facility. Finally, in step 4 of the procedure, we handle the clusters with an open facility. We give all the demand within the cluster $X(u)$ with an open facility to the vertex u. This increases the load of a facility by at most k (since the total demand in a cluster is at most k.) Wrapping up, we can state the following observation.

Observation 2.6. *The assignment $F : V(G) \to A$ satisfies the following two properties:*

1. *If $F(z) = u$ then there is $v \in X(z)$ where $F^+(v) = u_1$.*
2. *The load of each facility $a_i \in A$ is at most $(\beta + 2)k$.*

2.4 The last step: building the trees

Having the assignment function $F : V(G) \to A$, it remains to construct the associated trees. For a facility $u \in A$, let $F^{-1}(u)$ be the set of vertices that have been assigned to u. For each $u \in A$, we build an MST over $F^{-1}(u)$. Let R_u be the resulting tree. The algorithm's output includes the collection $R = \{R_u\}_{u \in A}$ along with the set of facilities A. The following lemma relates the cost of R to the cost of the edges in R^+. This lemma and Observations 2.6 and 2.4 together prove our main theorem.

Lemma 2.7. *We have* $\sum_{e \in E(R)} \ell(e) \leq 2 \sum_{e \in E(R^+)} \ell^+(e)$.

Proof. Let R_u be a component in R with an open facility $u \in A$. By the definition of A, the solution (A^+, R^+) has opened a facility at the vertex $u_1 \in X(u)$. Let T be the tree in R^+ that owns the open facility $u_1 \in A^+$. We claim $\sum_{e \in R_u} \ell(e) \leq 2 \sum_{e \in E(T)} \ell^+(e)$. This will prove our lemma because for each $R_u \in R$, we have found a distinct $T \in R^+$ with the cost at least half of $\sum_{e \in R_u} \ell(e)$.

The edges of T are of two types: (a) the internal edges (the edges between the vertices in the same cluster), and (b) the cross-cluster edges (the edges between the vertices in different clusters.) Disregarding the internal edges, each cross-cluster edge in $E(T)$ corresponds to an edge of G. Let H be a subgraph of G consisting of the vertices and the edges in G that T has touched. Namely, we let $V(H) = \{Super(v) \mid v \in V(T)\}$ and

$$E(H) = \{(x, y) \mid x \in V(G), y \in V(G), x \neq y, \text{ and } \exists u \in X(x), \exists v \in X(y) \text{ where } (u, v) \in E(T)\}$$

An important observation here is that $V(R_u) \subseteq V(H)$. To see this, let z be an arbitrary vertex in R_u. By definition, we must have $F(z) = u$. By Observation 2.6, there is $v \in X(z)$ where $F^+(v) = u_1$. Therefore, the tree T spans the vertex v, and hence $z = Super(v)$ belongs to $V(H)$.

Also note that the subgraph H is connected. Finally, the tree R_u is an MST over $V(R_u) \subseteq V(H)$. We show $\sum_{e \in R_u} \ell(u) \leq 2 \sum_{e \in H} \ell(e)$ which follows from the metric property of the length function ℓ. Let K be an MST over $V(H)$. We double the edges of K and construct an Eulerian tour L of the doubled tree. The tour $L = v_1, \cdots, v_p = v_1$ visits all the edges in H exactly once. From the sequence L, we pick the first visit of a vertex in $V(R_u)$ and jump over the repeating vertices or those in $V(H) \setminus V(R_u)$. The result $v'_1, \cdots, v'_q$ is a permutation of the vertices in $V(R_u)$. By the metric property of the function $\ell : V \times V \to \mathbb{R}^+$, the length of the edge $\ell(v'_i, v'_{i+1})$ is at most the sum of the length of edges that have been jumped over. Consequently, we have

$$\begin{aligned}
\sum_{e \in R_u} \ell(e) &\leq \sum_{i=1}^{q-1} \ell(v'_i, v'_{i+1}) && \text{(since } R(u) \text{ is an MST over } V(R_u)) \\
&\leq \sum_{i=1}^{p-1} \ell(v_i, v_{i+1}) && \text{(since } \ell \text{ is a metric)} \\
&\leq 2 \sum_{e \in E(K)} \ell(e) && \text{(by the doubling of the edges in } K) \\
&\leq 2 \sum_{e \in E(H)} \ell(e) && \text{(since } K \text{ is an MST over } V(H) \text{ and } H \text{ is connected)} \\
&\leq 2 \sum_{e \in E(T)} \ell^+(e) && \text{(by definition of } H).
\end{aligned}$$

□

The main steps of our algorithm for the ARUD problem is stated in Algorithm 3.

Algorithm 3 The Main Procedure

Input: Non-uniform instance $\mathcal{I} = (G, a, \ell, b, k)$
Output: Solution (A, R)

1. Run **ConvertToUniform** procedure. Let $\mathcal{I}^+ = (G^+, a^+, \ell^+, k)$ be the output of the procedure.
2. Run the AKMM18 algorithm over $\mathcal{I}^+$. Let (A^+, R^+) be the output of the algorithm.
3. Having (A^+, R^+), compute the assignment $F^+ : V(G^+) \to A^+$
4. Run the **NonSplitAssignment** procedure to obtain the assignment $F : V(G) \to A$.
5. For each $u \in A$, build an MST tree R_u over $F^{-1}(u)$.
6. Ouput the forest $R = \{R_u\}_{u \in A}$ and the set A as the solution.

3 Hardness of ARUD

This section states our results on the hardness of ARUD. We begin with the proof of Theorem 1.1. Then, we state two lemmas on the complexity of ARUD for fixed values of k.

Proof of Theorem 1.1. Let $X = (x_1, \cdots, x_n, t, B)$ be an instance of the Bin-Packing problem. The Bin-Packing problem asks if n items with non-zero integer sizes $x_1, \cdots, x_n$ can be packed into t bins where each bin has capacity B. It is known that Bin-Packing is NP-complete even when the item sizes $\{x_i\}$ are polynomially bounded in n [GJ79]. Given the instance X, we construct an instance $Y = (G, a, \ell, b, k = B + 1)$ of the ARUD problem as follows. The graph G has $n + t$ vertices. Corresponding to each item i, we put a vertex v_i in G. Additionally we have t extra vertices $u_1, \cdots, u_t$. For each $i \in \{1, \cdots, n\}$, we set $a(v_i) = 1$ and $b(v_i) = x_i$. Also for each $i \in \{1, \cdots, t\}$, we set $a(u_i) = 0$ and $b(u_i) = 1$. All the edge lengths $\ell(e)$ are set to zero. It is clear that the instance Y has zero cost if and only if X is a yes instance of the Bin-Packing problem. Consequently, an approximation algorithm for ARUD can distinguish between yes and no instances. This completes the proof. □

Lemma 3.1. *Assuming $k = 2$, one can find an optimal solution for a given ARUD instance $\mathcal{I} = (G, a, \ell, b, k)$ in polynomial time.*

Proof. To prove the statement, we establish a reduction to the minimum weight perfect matching problem [Edm65a, Edm65b]. Since $k = 2$, we have $b(v) \in \{1, 2\}$ for each $v \in V(G)$. Let $n = |V(G)|$. We construct the weighted undirected graph $G' = (V', E')$ on $2n$ vertices described as follows. For each $v \in V(G)$, we add a new vertex v^*. We let $V' = V(G) \cup V^*$ where $V^* = \{v^* \mid v \in V(G)\}$. The edge set of G' is defined as follows. For each $v \in V(G)$ we add the edge (v, v^*) to E' with length $a(v)$. Between all pairs vertices in V^*, we add an edge with zero length. Finally, for each pair $u, v \in V(G)$, we add the edge (u, v) if $b(u) + b(v) \leq 2$. In this case, we define the length of the edge (u, v) as $\min\{a(u), a(v)\} + \ell(u, v)$. This finishes the definition of G'.

Clearly, G' has a perfect matching. To prove the lemma, we show a feasible solution (A, R) with cost x for the instance $\mathcal{I}$ corresponds to a perfect matching M with cost x in G' and vice versa.

We show only one direction. The other direction follows in a similar manner. Let $M = \emptyset$. Note that a component in $R_i \in R$ is either an isolated vertex $u \in V(G)$ (an open facility) or a single edge $e = (u, v)$. In case R_i is an edge e, we include e in M, otherwise if $R_i = u$ we add the edge (u, u^*) to M. We do this for each component in R. To complete the perfect matching, we select an arbitrary perfect matching between the un-matched vertices in V^* and add it to M. It can be seen that the cost of M (the total weight of the edges in M) equals $\mathsf{COST}(A, R)$. This finishes the proof. □

Lemma 3.2. *The AR problem is NP-hard even when $k = 3$ and the airport costs are uniform.*

Proof. We establish a reduction from the P_2 Partition problem. In the P_2 Partition problem, we are given an undirected graph H on $3n$ vertices and we are asked if there are n vertex-disjoint paths of length 2 in graph H (in other words, we ask if H has a perfect 2-path matching [2].) It is known that P_2 Partition is NP-Complete [GJ79]. Let H be an instance of the P_2 Partition problem. We construct an instance $\mathcal{I} = (G, a, \ell, k)$ of the metric AR problem as follows. Let $V(G) = V(H)$. We define $\ell(u, v) = 1$ iff the edge (u, v) exists in H otherwise we set $\ell(u, v) = 2$. This defines a metric since the edge weights are either 1 or 2. Moreover, we set $a(u) = 2$ for each $u \in V$.

Suppose H has a perfect 2-path matching. In this case, there is a solution (A, R) for $\mathcal{I}$ with cost $4n$. Note that each 2-path costs 4, and we have n of these paths. Dividing the cost over the vertices, the average cost of satisfying the demand of each vertex is $\frac{4}{3}$. Now, suppose H has no perfect 2-path matching. In any solution (A, R), for instance $\mathcal{I}$, there will be a component $R_i \in R$ with a vertex $u \in R_i$ with a cost greater than $\frac{4}{3}$. There are three possibilities:

1. R_i is a 2-path that has an edge of length 2. In this case, the cost of each vertex in R_i is at least $\frac{5}{3}$.
2. R_i is a single edge. In this case, the cost of each vertex in R_i is at least $\frac{3}{2}$.
3. R_i is an isolated vertex. In this case, the cost of the isolated vertex is 2.

Consequently, when H has no perfect 2-path matching, there will be a vertex with a cost greater than $\frac{4}{3}$ in $\mathcal{I}$ (note that when we have $k = 3$ and $a(u) = 2$ for all u, the cost of each vertex will be at least $\frac{4}{3}$.) Hence, in this case, the cost of any solution for $\mathcal{I}$ is greater than $4n$. This finishes the proof. □

References

[AAKM18] Anna Adamaszek, Antonios Antoniadis, Amit Kumar, and Tobias Mömke. Approximating airports and railways. In *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[AAM16] Anna Adamaszek, Antonios Antoniadis, and Tobias Mömke. Airports and railways: Facility location meets network design. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[2] Here a 2-path refers to a simple path consisting of two edges.

[BCFN19] Suman Kalyan Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4955–4966, 2019.

[BH12] MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Assignment problem in content distribution networks: Unsplittable hard-capacitated facility location. *ACM Trans. Algorithms*, 8(3):20:1–20:19, 2012.

[BSS16] Babak Behsaz, Mohammad R. Salavatipour, and Zoya Svitkina. New approximation algorithms for the unsplittable capacitated facility location problem. *Algorithmica*, 75(1):53–83, 2016.

[BTV21] Jannis Blauth, Vera Traub, and Jens Vygen. Improving the approximation ratio for capacitated vehicle routing. In *Integer Programming and Combinatorial Optimization: 22nd International Conference, IPCO 2021, Atlanta, GA, USA, May 19–21, 2021, Proceedings 22*, pages 1–14. Springer, 2021.

[DKK+00] Milind Dawande, Jayant Kalagnanam, Pinar Keskinocak, F. Sibel Salman, and R. Ravi. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *J. Comb. Optim.*, 4(2):171–186, 2000.

[DR59] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[Edm65a] Jack Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *J. of Research at the National Bureau of Standards*, 69B:125—-130, 1965.

[Edm65b] Jack Edmonds. Path, trees, and flowers. *Canadian J. Math.*, 17:449—-467, 1965.

[FMRS21] Zachary Friggstad, Ramin Mousavi, Mirmahdi Rahgoshay, and Mohammad R Salavatipour. Improved approximations for cvrp with unsplittable demands. *arXiv preprint arXiv:2111.08138*, 2021.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GMZ22] Fabrizio Grandoni, Claire Mathieu, and Hang Zhou. Unsplittable euclidean capacitated vehicle routing: A $(2+\epsilon)$-approximation algorithm. *arXiv preprint arXiv:2209.05520*, 2022.

[HRK85] Mordecai Haimovich and Alexander HG Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of operations Research*, 10(4):527–542, 1985.

[JR05] Raja Jothi and Balaji Raghavachari. Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design. *ACM Trans. Algorithms*, 1(2):265–282, 2005.

[KPR00] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *J. Algorithms*, 37(1):146–188, 2000.

[MZ23a] Claire Mathieu and Hang Zhou. A PTAS for capacitated vehicle routing on trees. *ACM Transactions on Algorithms*, 19(2):1–28, 2023.

[MZ23b] Claire Mathieu and Hang Zhou. A tight (1.5+ ε)-approximation for unsplittable capacitated vehicle routing on trees. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).*, volume 261, pages pp–91, 2023.

[RV18] Dror Rawitz and Ariella Voloshin. Flexible allocation on related machines with assignment restrictions. *Discret. Appl. Math.*, 250:309–321, 2018.

[ST93] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62:461–474, 1993.

[ST24] Mohammad R Salavatipour and Lijiangnan Tian. Approximation algorithms for the airport and railway problem. In *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[STA97] David B Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, 1997.

[Trö19] Thorben Tröbst. Capacitated vehicle routing and cycle covering problems. *Master's thesis*, 2019.

[TT20] Vera Traub and Thorben Tröbst. A fast $(2 + 2/7)$-approximation algorithm for capacitated cycle covering. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 391–404. Springer, 2020.