Approximating Airports and Railways

Anna Adamaszek
University of Copenhagen, Copenhagen, Denmark
anad@di.ku.dk

Antonios Antoniadis
Saarland University and MPII, Saarbrücken, Germany
aantonia@mpi-inf.mpg.de

Amit Kumar
IIT Delhi, Delhi
amitk@cse.iitd.ac.in

Tobias Mömke
Saarland University, Saarbrücken, Germany and
University of Bremen, Bremen, Germany
moemke@cs.uni-saarland.de

Abstract

We consider the airport and railway problem (AR), which combines capacitated facility location with network design, both in the general metric and the two-dimensional Euclidean space. An instance of the airport and railway problem consists of a set of points in the corresponding metric, together with a non-negative weight for each point, and a parameter k. The points represent cities, the weights denote costs of opening an airport in the corresponding city, and the parameter k is a maximum capacity of an airport. The goal is to construct a minimum cost network of airports and railways connecting all the cities, where railways correspond to edges connecting pairs of points, and the cost of a railway is equal to the distance between the corresponding points. The network is partitioned into components, where each component contains an open airport, and spans at most k cities. For the Euclidean case, any points in the plane can be used as Steiner vertices of the network.

We obtain the first bicriteria approximation algorithm for AR for the general metric case, which yields a 4-approximate solution with a resource augmentation of the airport capacity k by a factor of 2. More generally, for any parameter $0 < p \leq 1$ where $p \cdot k$ is an integer we develop a $(4/3)(2 + 1/p)$-approximation algorithm for metric AR with a resource augmentation by a factor of $1 + p$.

Furthermore, we obtain the first constant factor approximation algorithm that does not resort to resource augmentation for AR in the Euclidean plane. Additionally, for the Euclidean setting we provide a quasi-polynomial time approximation scheme for the same problem with a resource augmentation by a factor of $1 + \mu$ on the airport capacity, for any fixed $\mu > 0$.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms, Theory of computation → Routing and network design problems, Theory of computation → Facility location and clustering

Keywords and phrases Approximation Algorithms, Network Design, Facility Location, Airports and Railways

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.5

Funding Research supported in part by the Danish Council for Independent Research DFF-MOBILEX mobility grant and by Deutsche Forschungsgemeinschaft grants AN 1262/1-1 and MO 2889/1-1.

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 5; pp. 5:1–5:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

1 Introduction

This paper studies the airport and railway problem, which combines facility location and network design, and has been introduced Adamaszek et al. [1]. In the airport and railway problem the input consists of a complete n-vertex graph G with vertex costs $a\colon V(G) \to \mathbb{R}_{\geq 0}$ and edge costs $\mathrm{len}\colon E(G) \to \mathbb{R}_{\geq 0}$, and a parameter k. The vertices represent cities, vertex cost represents the cost of opening an airport in the corresponding city, and edge cost models the cost of building a railway connecting the corresponding pair of cities. Finally, the parameter k represents a maximum capacity of an airport. The goal is to compute a minimum cost network of airports $A \subseteq V(G)$ and railways $R \subseteq E(G)$ which satisfies the following properties: (i) each $v \in V(G)$ is connected with some vertex from A via a path of edges from R (i.e., all cities are connected by the network), and (ii) each connected component of the network contains at most k vertices (i.e., each airport serves at most k cities). The cost of the network equals $a(A) + \mathrm{len}(R) = \sum_{v \in A} a(v) + \sum_{e \in R} \mathrm{len}(e)$. As the cost functions a and len are non-negative, an optimal solution to AR is a forest, with the cheapest airport opened in each connected component.

We consider both the case where $(V(G), \mathrm{len})$ is a general metric space, and the case where it is the Euclidean plane, i.e., the set of vertices $V(G)$ is represented by a set of points in the Euclidean plane, and the edge cost len is the Euclidean distance between the corresponding points. The goal in both cases is to find a minimum cost network spanning all vertices $V(G)$ and consisting of *components*, such that each component spans at most k vertices and contains an open airport. Furthermore, for the Euclidean metric case, we assume that each point in the Euclidean plane can be used as a Steiner vertex within the components. Note that in the Euclidean plane we allow edges corresponding to different components to cross, without a Steiner vertex at the intersection.

We also consider AR with resource augmentation, denoted by AR_α for a constant $\alpha > 1$, where we are allowed to assign $\alpha \cdot k$ cities to an airport of capacity k. We then compare the cost of the obtained solution against an optimal solution without resource augmentation.

Related work. The airport and railway problem AR is the most general problem within the framework introduced by Adamaszek et al. [1]. Several interesting novel problems can be defined within this framework by starting with AR and imposing additional constraints to the underlying network. It was shown in [1] that two-dimensional Euclidean AR is NP-hard, even when all the vertex costs are uniform. This uniform-vertex-cost case admits a polynomial time approximation scheme. Furthermore, when the airport capacity k is unbounded, AR can be solved exactly in polynomial time, even with both arbitrary vertex costs and arbitrary edge costs. Additionally, [1] considered the related AR_P problem. In AR_P, each component of the network is required to be a path, with airports at both of its endpoints. This problem is of particular interest because it models the Capacitated Vehicle Routing Problem (CVRP). Two-dimensional Euclidean AR_P is shown to be NP-hard even with uniform airport costs and unbounded parameter k. For the setting where either the airport costs are uniform or the parameter k is unbounded, a PTAS for AR_P has been presented.

Since AR combines the classical capacitated facility location (CFL) problem and network design (ND), we shortly describe these problems.

- **Capacitated Facility Location (CFL):** We are given a complete graph G with $V(G) := \{v_1, \ldots, v_n\}$, edge costs $d\colon E(G) \to \mathbb{R}_{\geq 0}$ and vertex costs $c\colon V(G) \to \mathbb{R}_{\geq 0}$, along with a capacity parameter k. Intuitively, $d(v_i, v_j)$ denotes the distance between vertices v_i and v_j, and $c(v_i)$ denotes the cost of opening a facility at v_i. A feasible solution to CFL consists of a set of open facilities $F \subseteq V(G)$, and an assignment of each vertex v_i to some open facility $f(v_i) \in F$ so that each facility has at most k vertices assigned. The cost of a solution is given by the sum of the cost for opening the facilities and the cost of connecting all vertices to the assigned facilities by *direct links*, i.e., $\sum_{v \in F} c(v) + \sum_{v \in V(G)} d(v, f(v))$. The goal is to find a minimum cost feasible solution. For CFL, the currently best results are a 1.488-approximation algorithm [8] and an LP based constant factor algorithm [4].

- **Network Design (ND):** In the framework of network design we are given a graph G with weights on the edges, and in some cases also on the vertices. Furthermore, we are given a set of constraints, e.g., connectivity constraints. The goal is to find a set of edges of minimum cost that satisfy the constraints.

Another problem closely related to AR is the *capacitated minimum spanning tree problem* (CMST). In CMST, the goal is to construct a minimum cost collection of trees covering all the input vertices, each tree spanning at most k vertices, connected to a single pre-specified root. Jothi and Raghavachari [7] give a 3.15-approximation algorithm for Euclidean CMST and a $2+\gamma$ approximation for metric CMST, where $\gamma \leq 2$ is the ratio between the cost of a Steiner tree and a minimum spanning tree. Both results allow demands on the vertices. We note that the AR problem can be modelled as CMST with an arbitrary (i.e., non-metric) cost function[1]. However, to the best of our knowledge, such version of CMST has not been studied before.

Ravi and Sinha [10] consider a related *capacitated-cable facility location problem* (CCFL) obtaining a constant factor approximation algorithm. The problem is based on the uncapacitated facility location (UFL) problem, i.e., there is a set of facilities with unbounded capacities and non-negative opening costs. But instead of connecting clients to facilities by direct links, they are connected by a network of capacitated cables (i.e., each edge of the constructed network can accommodate at most u units of flow from the clients to the facilities, where u is a parameter). When the cable capacity u is 1, the problem is equivalent to UFL. When the cable capacity $u = \infty$, CCFL is equivalent to AR with $k = \infty$. In general, the problem differs considerably from AR, as in CCFL, once a facility has been opened, it can receive an unbounded amount of flow. CCFL resembles AR, when instead of a bound on the airport capacity we have a bound on the railroad capacity.

Another problem related to AR and CCFL is *capacitated geometric network design* (CGND), where the goal is to create a network of capacitated links which allows sending flow from all the vertices to a single, pre-specified sink. In CGND the optimal network can be more complicated than a tree. Adamaszek et al. [2] provide a PTAS for the two-dimensional Euclidean CGND for link capacities $k \leq 2^{O(\sqrt{\log n})}$, where the network can use Steiner vertices anywhere in the plane.

Maßberg and Vygen [9] obtained a 4.1-approximation for another problem related to AR with uniform airport costs, called the *sink clustering problem.* They construct a network consisting of components, where instead of a capacity constraint for each component, they have a different constraint which incorporates both the capacity and the length of the edges of the component.

---

[1] To model an instance (G, a, r, k) of AR as a CMST problem, we proceed as follows. We extend the graph G to G' by adding a new vertex v and connecting it with all other vertices of G'. We extend the edge cost function r to a function r' as follows. Each edge $\{u, v\}$ for $u \in V(G)$ has cost equal to $a(u)$ in G'. Then (G', r', v, k) is an instance of CMST, with a pre-specified root v and parameter k. The corresponding instances of AR and CMST have corresponding solutions of the same costs, where adding an edge $\{u, v\}$ to a solution for CMST corresponding to opening an airport at u in AR.

Our results. We initiate the study of AR for general metric spaces from the perspective of bicriteria approximation, where we allow resource augmentation for the airport capacity parameter k. We prove the following theorem, which is the first result for the airport and railway problem on general metric spaces.

▶ **Theorem 1.** *There is a* 4*-approximation algorithm for metric* AR_2*. More general, for any* $0 < p \leq 1$ *such that* $p \cdot k$ *is integer, there is a* $\frac{4}{3}(2 + \frac{1}{p})$*-approximation algorithm for metric* AR_{1+p}*.*

The algorithm starts with computing an infeasible solution to the problem, returned by an algorithm for uncapacitated AR (i.e., assuming $k = \infty$). Then, this infeasible solution is transformed into a feasible one in a sequence of (technically involved) steps. First, the connected components of the uncapacitated solution are partitioned into paths, where each path contains $k \cdot p$ cities, plus one shorter path per component of the uncapacitated solution. These shorter paths get connected to the airports open by the uncapacitated solution, and they are the reason for the required resource augmentation. Then, the paths containing $k \cdot p$ cities each are assigned to airports using min-cost max-flow computation in a specially constructed graph, where each airport gets connected to at most $1/p$ paths. This requires the solution to open additional airports, and the solution has to be modified again so that each component contains one airport.

We then turn our attention to AR in the Euclidean plane, providing the first approximation algorithm for this setting. Note that this algorithm, in contrast to the algorithm from Theorem 1, does not use resource augmentation.

▶ **Theorem 2.** *For any fixed* $\epsilon > 0$ *there is a* $(2 + \frac{k}{k-1} + \epsilon)$*-approximation algorithm for* AR *with the airport capacity* $k \geq 2$ *in the Euclidean plane.*

Note that for $k = 1$ the AR problem becomes trivial (as the solution requires opening airports in all the cities). For $k \geq 2$, the approximation factor of the algorithm from Theorem 2 is at most $4 + \epsilon$.

In order to obtain Theorem 2, we define a relaxed version AR' of the AR problem, where each component can contain multiple airports and multiple copies of the same edge, each component allows routing flow from all its cities to the airports, each airport serves at most k cities, and each copy of an edge can be used by at most k cities. Note that in this version of the problem the cities belonging to different airports can share the edges of the network. Building upon Arora's PTAS for the Euclidean TSP [5] we develop $(1 + \epsilon)$-approximation algorithm for AR' for any fixed $\epsilon > 0$. By applying a carefully-designed sequence of transformations on the solution returned by the algorithm for AR', we transform it to a feasible solution for AR. These steps resemble the steps of the algorithm from Theorem 1. However, we have to be more careful to avoid resource augmentation.

Finally, we provide a QPTAS for $\mathrm{AR}_{1+\mu}$ for any fixed $\mu > 0$, matching the corresponding result for capacitated facility location [6].

▶ **Theorem 3.** *For arbitrary* $\epsilon, \mu > 0$ *there is a* $(1 + \epsilon)$*-approximation algorithm for two-dimensional Euclidean* $\mathrm{AR}_{1+\mu}$ *running in quasipolynomial time.*

In Section 2 we study metric AR and prove Theorem 1. In Section 3 we study AR in the Euclidean plane. We prove Theorem 3 in Section 3.1 and Theorem 2 in Section 3.3.

2 The Metric Case

In this section we will assume that the edge cost len is metric, i.e., it satisfies the triangle inequality $(\mathrm{len}(\{u, v\}) + \mathrm{len}(\{v, w\}) \geq \mathrm{len}(\{u, w\})$ for each triple of vertices $u, v, w \in V(G))$.

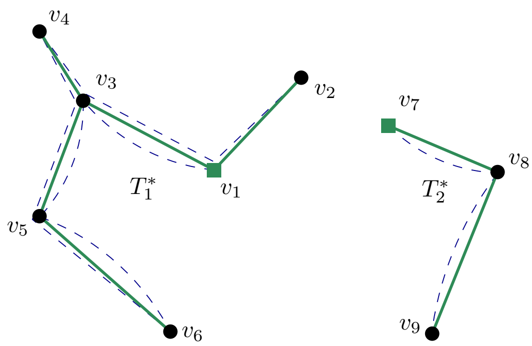

Figure 1 An infeasible solution S_0 for an instance I of AR for $k = 2$. The solution consists of the set of trees $\{T_1^\star, T_2^\star\}$, and airports open at vertices v_1 and v_7. The airports are denoted by squares. The dashed blue lines show an Eulerian tour needed in Step 1.

Fix a parameter $0 < p \leq 1$ to determine the amount of resource augmentation. We assume that p is a multiple of $1/k$, so that $p \cdot k$ is an integer.

Consider an instance I of AR, and let OPT be an optimal solution for I. Our algorithm consists of several steps, which we describe below.

Step 0: Preprocessing. Infeasible solution S_0. We create a new problem instance I_0 by taking I and setting the airport capacity $k = \infty$. I_0 is an instance of AR_F^∞, a relaxation of AR defined in [1], where we assume that the airport capacity is unbounded. By Theorem 4 in [1], there is a polynomial time algorithm for AR_F^∞. The algorithm is an extension of an algorithm finding a minimum spanning tree, and it always outputs a spanning forest.

Let S_0 be an optimal solution for I_0 output by the algorithm from [1]. Note that S_0 may contain components with more than k cities, and therefore it is not necessarily a feasible solution for AR. See an example in Figure 1. In the following lemma we prove various properties of S_0.

▶ **Lemma 4.** *The solution S_0 has the properties that (i) it is a forest, (ii) each connected component of S_0 contains exactly one airport, and (iii) $\texttt{cost}(S_0) \leq opt$.*

▶ **Observation 1.** *If for some instance I, the solution S_0 returned by the AR_F^∞ algorithm only contains components of size at most k, then by Lemma 4 S_0 is already optimal for AR on I. However, in general S_0 may contain components with more than k vertices – in which case it is not a feasible solution for AR.*

Step 1: Splitting each component of S_0 into paths. Infeasible solution S_1. We will split each connected component of S_0 into paths. Each path, except exactly one shorter path, will contain exactly $p \cdot k$ cities (vertices from $V(G)$). We proceed as follows.

By Lemma 4, the edges of S_0 form a forest $\{T_1^\star, \ldots, T_\ell^\star\}$ in G, where each tree $T_i^\star$ of S_0 contains one airport. For each tree $T_i^\star$ of S_0, denote by v_i the vertex of $T_i^\star$ with an airport, and consider $T_i^\star$ as a rooted tree with a root at v_i. Observe that $\texttt{cost}(S_0) = \sum_{i=1,\ldots,\ell} \texttt{cost}(T_i^\star)$, where $\texttt{cost}(T_i^\star) = a(v_i) + \sum_{e \in E(G) \cap T_i^\star} \text{len}(e)$ denotes the total cost of the tree $T_i^\star$.

First, in the solution S_1 we open the airport at v_i for each tree $T_i^\star$. Recall that S_0 also opens (and therefore pays for) these airports. The next step is transforming the forest $\{T_1^\star, \ldots, T_\ell^\star\}$ into a collection of paths.

For each tree $T_i^\star$ we construct a path $P_i^\star$ starting in the vertex v_i, visiting all cities of $T_i^\star$, and such that the cost of edges of $P_i^\star$ is at most twice the cost of the edges of $T_i^\star$. We

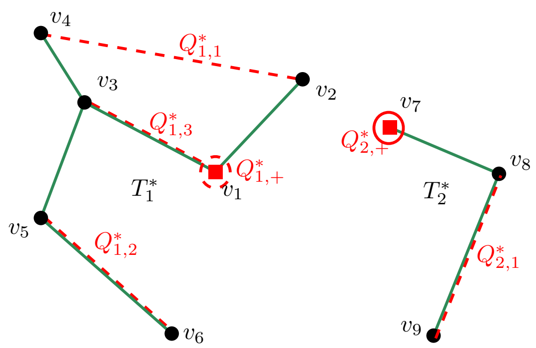

Figure 2 An infeasible solution S_1 for the instance I of AR from Figure 1 (green) is pictured in red (dashed). Here $k = 2$ and $p = 1$, i.e., each subpath $Q^\star_{i,j}$ contains 2 cities, and each subpath $Q^\star_{i,+}$ contains either 0 or 1 cities. $Q^\star_{1,+}$ is empty, incident with the vertex v_1, and the airport v_1 does not serve any city. $Q^\star_{2,+}$ consists of a city v_7, and the airport v_7 serves this city.

do that by doubling all edges of $T^\star_i$, constructing an Eulerian tour of $T^\star_i$ (note that after doubling the edges each vertex has even degree), shortcutting the tour so that each vertex is visited only once, and removing from the obtained tour one edge incident with v_i.

We break each of the paths $P^\star_i$ into subpaths $Q^\star_{i,j}$, each containing exactly $p \cdot k$ cities, and exactly one shorter subpath $Q^\star_{i,+}$ whose number of cities lies in the range $[0, p \cdot k - 1]$. We do this so that the subpath $Q^\star_{i,+}$ is the one closest to the root v_i. In particular, if $Q^\star_{i,+}$ contains at least one city, then it contains the city v_i. If $Q^\star_{i,+}$ is an empty path, we think of it as a path consisting of a single vertex v_i, but not containing the city at v_i. We add all the edges of all the paths $Q^\star_{i,j}$ and $Q^\star_{i,+}$ to S_1. See Figure 2 for an example of this construction.

For each airport v_i, we *assign* the subpath $Q^\star_{i,+}$ to v_i. This means that every city in the (possibly empty) subpath $Q^\star_{i,+}$ is *served* by the airport v_i. Note that this can be done, as by the construction of S_1 all vertices of $Q^\star_{i,+}$ are in the same connected component of S_1 as v_i. [2]

In the subsequent step, we will initially ignore the already assigned cities from the subpaths $Q^\star_{i,+}$, and concentrate on assigning the subpaths $Q^\star_{i,j}$ to airports. The already assigned cities from the subpaths $Q^\star_{i,+}$ will later be added, and they will induce a resource augmentation of the airport capacities.

▶ **Lemma 5.** *The solution S_1 satisfies that (i) each airport serves at most $k \cdot p - 1$ cities*[3]*, (ii) each city from a subpath $Q^\star_{i,+}$ is served by an airport, and (iii)* $\texttt{cost}(S_1) \le 2 \cdot \texttt{cost}(S_0) \le 2 \cdot opt$.

▶ **Observation 2.** *If for some instance I every component of S_1 has an airport that serves all cities of the component (including itself), then S_1 is a feasible, 2-approximate solution for I. Such a solution does not use resource augmentation, and each airport is only used up to a capacity of $pk - 1$. However, in general S_1 may have components of size exactly $k \cdot p$ whose cities are not served by any airport, and it is therefore in general not a feasible solution for* AR.

[2] Note that in case when $Q^\star_{i,+}$ is an empty path, the connected component of S_1 containing v_i consists of some path $Q^\star_{i,j}$. However, we do not assign the subpath $Q^\star_{i,j}$ to the vertex v_i, i.e., for now we treat the cities of $Q^\star_{i,j}$ as not served by any airport. Later the solution will be modified, and the cities from $Q^\star_{i,j}$ will be served by some airport (possibly different from v_i).

[3] Recall that in the case when $Q^\star_{i,+}$ is empty, the airport v_i does not serve any city, and the connected component of S_1 containing v_i contains $k \cdot p$ unserved cities.

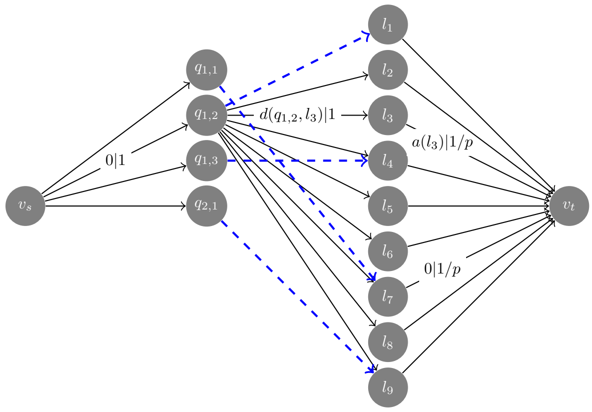

Figure 3 Example of an instance of the min-cost max-flow problem, corresponding to the AR instance from Figures 1 and 2. The goal is to send flow from v_s to v_t. Here $Q = \{q_{1,1}, q_{1,2}, q_{1,3}, q_{2,1}\}$ and $L = \{l_1, \ldots, l_9\}$. An label $x|y$ denotes an edge with cost x and capacity y. The airport l_7 has already been opened in S_1 (and therefore the edge $\{l_7, v_t\}$ has cost 0), while the airport l_3 has not been opened in S_1 (and the edge $\{l_3, v_t\}$ has non-zero cost). Note that for the clarity of presentation not all edges of E_{QL} have been drawn. The blue dashed edges denote a potential solution, i.e., a potential assignment of subpaths to airports.

Step 2: Assigning the subpaths $Q^\star_{i,j}$ to airports using network flows. Infeasible solution S_2. In this step we will make sure that every connected component is assigned to a neighboring airport, and therefore all the cities are served. For that, we will choose a set of additional airports to be opened. We will assign (and connect by choosing the appropriate edges of G) at most $1/p$ many subpaths $Q^\star_{i,j}$ to each airport, considering both the newly opened airports and the airports opened in Step 1. Note that if $1/p$ subpaths get assigned in this step to an airport that has been opened in Step 1 (and therefore might already be serving some cities), the capacity of the airport can become violated. Therefore, the solution S_2 constructed in this step requires resource augmentation. For now, we allow assigning subpaths $Q^\star_{i,j}$ to airports from different components. We will fix that in the subsequent step.

To decide which additional airports should be opened, and how we should assign (and connect) the subpaths $Q^\star_{i,j}$ to the airports, we use min-cost max-flow computation. In this computation we ignore the subpaths $Q^\star_{i,+}$ containing less than $k \cdot p$ cities each, as they have already been assigned (and connected) to the airports.

We construct a directed graph G', with capacities and a cost function d on the edges, as follows (see Figure 3). We introduce a vertex $q_{i,j}$ corresponding to each subpath $Q^\star_{i,j}$ (but not for the subpaths $Q^\star_{i,+}$), and we denote this set of vertices by Q. We also introduce a vertex l_v for each vertex $v \in V(G)$ of the original instance, and we denote this set of vertices by L. For each pair of vertices $(q_{i,j}, l_v) \in Q \times L$ we introduce an edge with capacity 1 and cost $d(q_{i,j}, l_v) := \min_{u \in V(Q^\star_{i,j})} \mathrm{len}(\{u, v\})$, directed from $q_{i,j}$ to l_v. Note that $d(q_{i,j}, l_v)$ equals the minimum distance between a vertex of the subpath $Q^\star_{i,j}$ represented by $q_{i,j}$, and the vertex v corresponding to l_v. We denote this set of edges by E_{QL}. Intuitively, sending flow 1 through an edge $\{q_{i,j}, l_v\} \in E_{QL}$ means connecting the subpath $Q^\star_{i,j}$ to an airport at the vertex v.

We then introduce a source vertex v_s and directed edges from v_s to all vertices in Q, each edge $\{v_s, q_{i,j}\}$ with capacity 1 and cost $d(v_s, q_{i,j}) = 0$. Finally, we introduce a sink vertex v_t and directed edges from each vertex $l_v \in L$ to v_t. We denote these sets of edges by E_Q and E_L, respectively. The cost $d(l_v, v_t)$ of an edge $\{l_v, v_t\}$ is zero if an airport at v has been opened in S_1 (i.e., in Step 1 of the algorithm), and $a(v)$ otherwise. Each edge of E_L has a capacity of $1/p$, which enforces that no more than $1/p$ subpaths (and therefore no more than $(k \cdot p) \cdot (1/p) = k$ vertices) are assigned to each airport at this step of the algorithm.

The graph G', together with the edge capacities and edge costs, yields an instance of the min-cost max-flow problem, where we want to send flow from the source vertex v_s to the sink vertex v_t. Clearly, the instance admits a solution where the amount of flow is $|Q|$. We can send one unit of flow from v_s to each of the vertices $q_{i,j}$, then also one unit of flow from each $q_{i,j} \in Q$ to some vertex $l_v \in L$ so that $v \in Q^\star_{i,j}$, and as each vertex from L gets at most one unit of flow, it can be sent to the sink vertex v_t. We note that this is the maximum amount of flow that can be sent, since the total capacity of the outgoing edges from v_s is $|Q|$.

It is well known that one can find an optimal, integral solution for the min-cost max-flow problem in time polynomial in the number of vertices and edges of the input instance (cf. [3], Chapters 9 and 10 or [11]). Let S' be this optimal, integral min-cost max-flow problem solution for G'. We denote the cost of S' by $\texttt{cost}(S')$.

▶ **Lemma 6.** *The following inequality holds:* $\texttt{cost}(S') \leq opt/p$.

Proof. We will show how we can translate OPT into a solution OPT$'$ to the min-cost max-flow problem, with cost of at most opt$/p$. By the optimality of S', the cost of S' is not greater than the cost of OPT$'$, and therefore it is at most opt$/p$.

Consider an optimal solution OPT for the instance of AR. We construct a (fractional) flow in G' of capacity $|Q|$ corresponding to OPT in the following way. We send a flow of 1 along each edge $\{v_s, q_{i,j}\}$ leaving the source. At each vertex $q_{i,j} \in Q$, this flow of 1 is split into $k \cdot p$ equal parts, one for each vertex; recall that each $Q^\star_{i,j}$ has exactly $k \cdot p$ vertices. For each vertex $u \in Q^\star_{i,j}$, we send the amount of $1/(k \cdot p)$ flow along the edge $\{q_{i,j}, l_v\}$, where v is the airport serving u in the solution OPT. Finally, for every vertex $l_v \in L$ we forward all the received flow (which by feasibility of OPT cannot be greater than $\frac{1}{k \cdot p} \cdot k = 1/p$) along the outgoing edge $\{l_v, v_t\}$ to the sink v_t.

The constructed flow OPT$'$ has capacity $|Q|$ and is feasible, as the only edges where the amount of flow might be greater than 1 are the edges of E_L, and in the reasoning above we have shown an upper bound of $1/p$ on the flow on these edges.

We will now upper bound the cost of OPT$'$ with respect to the cost of OPT. Let opt $=$ $\texttt{cost}_e(\text{OPT}) + \texttt{cost}_a(\text{OPT})$, where $\texttt{cost}_e(\text{OPT})$ is the edge cost of OPT and $\texttt{cost}_a(\text{OPT})$ is the airport cost of OPT. For any $v \in V(G)$, let $b(v)$ denote the airport serving v in OPT. As each airport serves at most k cities, we have $\texttt{cost}_e(\text{OPT}) \geq \frac{1}{k} \sum_{v \in V(G)} \text{len}(v, b(v))$.

For any $l_v \in L$, OPT$'$ sends some flow along the edge $\{l_v, v_t\}$ only when OPT opens an airport at v. As the capacity of each edge of E_L in G' is $1/p$, the cost of OPT$'$ on the edges of E_L is therefore at most $\texttt{cost}_a(\text{OPT})/p$. We will now upper bound the cost of OPT$'$ on the edges of E_{QL}. By the construction of OPT$'$, this cost equals

$$\sum_{q_{i,j} \in Q} \sum_{u \in Q^\star_{i,j}} \frac{d(q_{i,j}, b(u))}{k \cdot p} \leq \sum_{q_{i,j} \in Q} \sum_{u \in Q^\star_{i,j}} \frac{\text{len}(u, b(u))}{k \cdot p} \leq \sum_{u \in V(G)} \frac{\text{len}(u, b(u))}{k \cdot p} \leq \frac{\texttt{cost}_e(\text{OPT})}{p} .$$

The edges of G' which are in E_Q have cost 0, so they do not contribute towards the cost. Therefore $\texttt{cost}(\text{OPT}') \leq \text{opt}/p$. As S' is an optimal solution for the min-cost max-flow instance, we get $\texttt{cost}(S') \leq \texttt{cost}(\text{OPT}') \leq \text{opt}/p$. ◀

From the integral min-cost max-flow solution S' we construct S_2 as follows. We start by taking $S_2 = S_1$. Then, we open the airports $u \in V(G)$ which have not been opened by S_1, and for which S' has flow at least 1 on the corresponding edge $\{l_u, v_t\}$. Then, for each

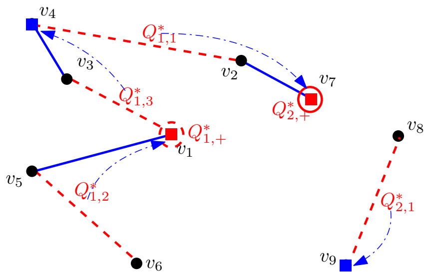

Figure 4 An infeasible solution S_2 for the instance I of AR from Figure 1 is pictured in red (dashed) and blue (solid). The new airports (blue squares) have been opened at vertices v_4 and v_9. The blue edges connect the subpaths $Q^\star_{i,j}$ with the assigned airports (the assignment is pictured by the dashed arrows). The solution is infeasible, as the airports serving $Q^\star_{1,2}$ and $Q^\star_{1,3}$ are in different components than the corresponding subpaths. (In the drawing, the components corresponding to the airports v_1, v_4 and v_7 got connected.) Note also, that the airport v_7 now serves three cities (v_2, v_4 and v_7), and therefore requires resource augmentation.

subpath $Q^\star_{i,j}$ we find the unique vertex $u \in V(G)$, such that S' uses the edge $\{q_{i,j}, l_u\}$. Note that in this case S_2 must have an airport at u. Let $v_{i,j}$ be the vertex of $Q^\star_{i,j}$ minimizing the distance $\text{len}(v_{i,j}, u)$. We add to S_2 the edge $\{v_{i,j}, u\}$, and we assign $Q^\star_{i,j}$ to the airport u (i.e., the cities from $Q^\star_{i,j}$ will be served by u). See Figure 4 for an example.

Note that in this construction each subpath $Q^\star_{i,j}$ is assigned to an airport of S_2, and therefore all cities from $Q^\star_{i,j}$ are served. As the capacity of the edges $\{l_u, v_t\}$ is $1/p$, at most $1/p$ subpaths get connected to one airport. We can show the following.

▶ **Lemma 7.** *The solution S_2 has the following properties: (i) $\texttt{cost}(S_2) \leq (2 + 1/p)opt$, and (ii) each city is served by some airport, and each airport serves strictly less than $(1 + p) \cdot k$ many cities.*

▶ **Observation 3.** *The solution S_2 is still not feasible. For any vertex $u \in V(G)$ it may happen that the city u is served by an airport at some vertex $v \in V(G)$ with $v \neq u$, and at the same time the airport at u has been opened and serves some other component. In particular, a single component might contain a large number of airports, each of them serving a different component. (Then, when considering the edges of S_2, such components create a single connected component of S_2.) This is not consistent with the definition of the* AR *problem, where the airport serving a component must belong to this component.*

Step 3: Making the solution feasible. Solution S_3. In this final step we show how we can transform the solution S_2 to a feasible solution S_3, with only a small increase in cost and while increasing the size of each component by at most one vertex.

We consider the components of S_2 one by one. For each component T_ℓ, we consider the cities which belong to T_ℓ, as well as the airport $u \in V(G)$ serving the cities from T_ℓ. If the city u belongs to T_ℓ (i.e., it is served by the airport at u), we do not make any changes. Otherwise, we re-assign the city u to T_ℓ. We do that by removing u from its current component, and by adding it to T_ℓ. We denote the resulting solution by S_3.

▶ **Lemma 8.** *The re-assignment can be performed so that the solution S_3 has the following properties. (i) Each airport in S_3 serves at most $(1+p) \cdot k$ cities, (ii) $\texttt{cost}(S_3) \leq \frac{4}{3}\texttt{cost}(S_2)$, and (iii) S_3 is a feasible solution for AR_{1+p}.*

Theorem 1 follows from Lemmas 7 and 8.

3 The Euclidean Case

In this section we focus on AR in the two-dimensional Euclidean space. We first show in Section 3.1 that if we allow a small resource augmentation of the airport capacities, we are able to obtain a quasi-polynomial-time approximation scheme (QPTAS). We then, in Section 3.2, present a polynomial time $(1+\epsilon)$-approximation algorithm for a relaxed version of the problem that allows components to have size larger than k, but where each component must have enough airports in order to serve all clients. This approximation algorithm is then used in Section 3.3 in order to give a constant factor approximation algorithm for the general AR in two-dimensional Euclidean space, which is our main result for the section.

3.1 A QPTAS with a Small Resource Augmentation

In this section we give a sketch of the proof of Theorem 3, i.e., a QPTAS for two-dimensional Euclidean $\mathrm{AR}_{(1+\mu)}$ for any fixed $\mu > 0$. Our algorithm is based on Arora's scheme [5].

First, using standard techniques, we partition the problem instance into independent subinstances and perform perturbation. This step reduces the original problem instance into a collection of independent subinstances, where each instance has all input points at points with integer coordinates, allows Steiner vertices only at points with integer coordinates, and is bounded by a polynomially sized bounding box. That increases the cost of the obtained solution only by a negligible factor.

Next, as in Arora's scheme [5], we introduce a shifted quadtree, which recursively decomposes the input box into smaller and smaller subsquares (called dissection squares), ending with leaf squares which contain only one point with integer coordinates each. Then, at the boundary of each dissection square we introduce a logarithmic number of equidistant portals. We then show that, by increasing the cost of the obtained solution only by a negligible factor, we can consider solutions where edges cross the boundary of the dissection squares only at the portals. By losing another negligible factor, we further restrict the solutions so that every component is $O(1)$-light, i.e., it crosses the boundary of each dissection square at at most $O(1)$ portals.

For each dissection square C, with each component T of the solution, we associate the *type* of T, which specifies the $O(1)$ portals used by T, a partition of these portals into sets such that each set corresponds to a connected component of $T \cap C$, information whether there is an open airport in T within C, and the number of points from $V(G)$ in $T \cap C$ rounded down to the nearest *threshold*, where the number of thresholds is polylogarithmic. That gives a polylogarithmic number of types of components.

This allows us to use a dynamic program that finds a near-optimal solution for $\mathrm{AR}_{(1+\mu)}$ for any constant $\mu > 0$. In the dynamic program, we have a set of possible *configurations* for each dissection cell C, where each configuration specifies the number of components of each of the polylogarithmic number of types. Therefore, the number of configurations is quasi-polynomial. For leaf dissection squares, we can find an optimal solution satisfying each configuration. Then, the DP proceeds bottom-up, computing solutions for all the configurations for larger dissection squares, based on the solutions for the subsquares. We can show that, by choosing the number of thresholds appropriately, the resource augmentation required for the DP solution can be upper-bounded by $1 + \mu$.

3.2 Relaxed AR: Allowing components with multiple airports

In this section we define a relaxed version of AR, which we denote by AR$'$. The difference between AR and AR$'$ is that each component of AR$'$ can contain multiple airports and multiple copies of the same edges. Moreover, each component allows routing all customers to the airports, where each airport serves at most k customers, and each copy of an edge can be used by at most k customers. As in the case for AR, AR$'$ can also use Steiner vertices.

Intuitively, AR$'$ is a relaxation of AR where cities assigned to different airports can share the same edges. We now define the problem formally.

▶ **Definition 9.** In the problem AR$'$ we are given a set of points $V(G)$ on the Euclidean plane, together with a cost $a(v)$ for each point $v \in V(G)$, and an integral capacity parameter k. A feasible solution is a subset of vertices $A \subseteq V(G)$ and a network consisting of edges $E(G)$ that allows routing the flow of one unit from each point in $V(G)$ to the points in A, such that (i) each edge in the network has capacity k, and (ii) each point from A can receive at most k units of flow. The network can use each point on the Euclidean plane as Steiner vertices, and parallel edges are allowed.

The goal is to find a feasible solution minimizing the total cost of the network, i.e., the value of $\sum_{v\in A} a(v) + \sum_{e\in E(G)} \text{len}(E(G))$.

We obtain a polynomial time algorithm that for every input instance I of AR finds a solution to instance I of AR$'$ with cost of at most $(1+\epsilon)\text{opt}_{\text{AR}}$, where opt_{AR} is the cost of an optimal solution to I for AR. The algorithm is based on a dynamic programming formulation, and builds on Arora's PTAS for the Euclidean TSP [5].

With each solution for an instance I of AR$'$ we can associate a network flow f that defines an assignment of cities to the airports within each component of the solution. Such flow can be computed efficiently.

3.3 A Constant-Factor Approximation Algorithm

In this section we use the algorithm from Section 3.2 as a building-block of a constant-factor approximation algorithm for the original AR problem in the two-dimensional Euclidean space. Note that this algorithm does not require resource augmentation.

We proceed similarly as in Steps 1 and 2 from Section 2, cutting the initial solution into pieces, and matching the pieces to the airports.

Step 0: Obtaining solution S_0. Given an instance I of AR, we run the algorithm from Section 3.2 on I, obtaining a solution S_0. Although S_0 is not feasible in general, as it may contain components with more than k cities and more than one airport, it is a good starting point, as we can upper bound its cost by opt.

▶ **Lemma 10.** *Consider the solution S_0 for an instance I of two-dimensional Euclidean* AR*. Let $\{C_1, C_2, \dots C_z\}$ be the set of connected components of S_0, where each component C_i contains h_i airports. The following holds: (i) $\texttt{cost}(S_0) \le (1+\epsilon)$opt, and (ii) for each component C_i, the number of points of C_i which are in $V(G)$ satisfies: $|C_i| \le k \cdot h_i$.*

We now show how to transform S_0 into a feasible solution for AR. For each connected component C_i of S_0 with $h_i > 1$, we proceed in two further steps that slightly resemble steps from Section 2. However, we have to be more careful in order to avoid resource augmentation. In the first step we will "cut" each such component C_i into singleton components containing the airports of S_0, and paths containing at most $k-1$ cities (with no airports) each. In the second step, then we will develop an algorithm that matches these paths to airports without increasing the cost by more than a constant factor.

Before we start, we perform the following operations. First, we compute a flow f in S_0, and we modify f into a flow f', so that each airport v_i serves the city at v_i. We will modify the instance I_0 into an instance I_0', and the solution S_0 into S_0', so that each airport serves exactly k cities. We do that by introducing for each airport v_i *additional cities*, coincident with the airport v_i and being served by v_i, so that v_i serves exactly k cities. The cost of S_0' is then the same as the cost of S_0 (as the additional cities are served for free). We will transform S_0' into a feasible solution for the instance I' of AR, while increasing its cost only by a constant factor, and then by dropping the additional cities we will obtain a solution for the instance I.

Step 1: Cutting each component C_i. Solution S_1. Consider a connected component C_i of S_0' (and therefore also of S_0), which contains $|C_i|$ cities. We first transform C_i into a path P_i that visits all vertices of C_i that do not contain an open airport. We do this by first doubling all edges of C_i, obtaining an Eulerian tour on it, shortcutting the resulting tour so that it only visits cities that do not have an airport[4], and then removing a single edge from the tour. We have

$$\sum_{i=1}^{z} \mathtt{cost}(P_i) \leq 2 \cdot \mathtt{cost}_e(S_0), \tag{1}$$

where $\mathtt{cost}_e(S_0)$ refers to the edge cost of the solution S_0.

We now break each path P_i into a collection of h_i subpaths $\{Q_{i,1}, Q_{i,2}, \dots Q_{i,h_i}\}$, such that each subpath $Q_{i,j}$ contains exactly $k-1$ cities of I'. Note that we can do this, as in S_0' the component C_i contains exactly $k \cdot h_i$ cities (including the additional cities), and after removing the airports the path P_i contains $(k-1) \cdot h_i$ cities.

Let S_1 be a solution consisting of the airports open by S_0 (and therefore also by S_0') that form a singleton components $\{v_i\}$ and serve the cities at v_i, and of the paths $Q_{i,j}$ that do not contain open airports and contain exactly $k-1$ cities of I' each.[5] We now upper bound the cost of S_1.

▶ **Lemma 11.** $\mathtt{cost}(S_1) \leq 2 \cdot \mathtt{cost}(S_0)$.

Step 2: Matching the subpaths $Q_{i,j}$ to the airports within each component C_i. Solution S_2. In order to assign the subpaths $Q_{i,j}$ to the airports of component C_i, we develop an instance of of minimum cost perfect matching in a bipartite graph. This can be seen as a simplified version of our network flow construction in Section 2. We do this for each component C_i separately.

Consider a component C_i of S_0' (and therefore also of S_0). We construct a bipartite graph G_i' as follows. For each subpath $Q_{i,j}$ we introduce a vertex q_j, and denote the set of such vertices by Q. For each vertex $u \in C_i$ with an airport in S_0, introduce a vertex l_u and denote this set of such vertices by L. Now we form a complete bipartite graph G_i', where the vertices of Q are at one side of the bipartition, and the vertices of L are at the other side. An edge $\{q_j, l_u\}$ has cost equal to the minimum distance between the subpath $q_{i,j}$ and the vertex u. More formally, the cost of the edge $\{q_j, l_u\}$ equals $\min_{v \in Q_{i,j}} \mathrm{len}(\{v, u\})$.

[4] Note that if there are additional cities coincident with a city v_i with an airport, the path should not visit the city v_i, but it should still visit the coincident additional cities.

[5] Note that the paths $Q_{i,j}$ can have coincident endpoints, if they visit the coincident additional cities.

We construct the solution S_2 as follows. We start by setting $S_2 = S_1$. For each component C_i of S_0 we compute (in polynomial time) a min-cost perfect matching for the graph G'_i described above. Such a matching exists, as C_i has h_i airports and h_i subpaths, therefore both parts of the bipartition have equal size. For each matching edge $\{q_j, l_u\} \in Q \times L$ we add to the solution S_2 an edge $\{v, u\}$, where $v \in Q_{i,j}$ has minimum distance to u out of all vertices of $Q_{i,j}$. We then remove the additional cities from each component of S_2.

▶ **Lemma 12.** *Solution S_2 has the following properties:* (i) *it is a feasible solution for the* AR *instance (without resource augmentation), and* (ii) $\texttt{cost}(S_2) \leq \left(2 + \frac{k}{k-1}\right)(1+\epsilon)opt$.

The proof of Theorem 2 now directly follows from Lemma 12 and by substituting ϵ with $\epsilon/4$.

References

1. Anna Adamaszek, Antonios Antoniadis, and Tobias Mömke. Airports and railways: Facility location meets network design. In *STACS*, volume 47 of *LIPIcs*, pages 6:1–6:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
2. Anna Adamaszek, Artur Czumaj, Andrzej Lingas, and Jakub Onufry Wojtaszczyk. Approximation schemes for capacitated geometric network design. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2011. doi:10.1007/978-3-642-22006-7_3.
3. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications.* Prentice Hall, 1993.
4. Hyung-Chan An, Mohit Singh, and Ola Svensson. LP-based algorithms for capacitated facility location. In *FOCS*, pages 256–265. IEEE Computer Society, 2014.
5. Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.
6. Babak Behsaz, Mohammad R. Salavatipour, and Zoya Svitkina. New approximation algorithms for the unsplittable capacitated facility location problem. *Algorithmica*, 75(1):53–83, 2016.
7. Raja Jothi and Balaji Raghavachari. Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design. *ACM Transactions on Algorithms (TALG)*, 1(2):265–282, 2005.
8. Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013.
9. Jens Maßberg and Jens Vygen. Approximation algorithms for a facility location problem with service capacities. *ACM Trans. Algorithms*, 4(4):50:1–50:15, 2008.
10. R. Ravi and Amitabh Sinha. Approximation algorithms for problems combining facility location and network design. *Operations Research*, 54(1):73–81, 2006. doi:10.1287/opre.1050.0228.
11. Alexander Schrijver. *Combinatorial Optimization.* Springer, 2003.