HOMA: Online In-Flight Service Provisioning with Dynamic Bipartite Matching

Amir Varasteh[†], Saeed Akhoondian Amiri[⋆], and Carmen Mas-Machuca[†]

[†]Chair of Communication Networks, TU Munich
Email: {amir.varasteh, cmas}@tum.de

[⋆]Department of Computer Science, University of Cologne
Email: amiri@cs.uni-koeln.de

***Abstract*—Airline companies are currently investigating means to improve in-flight services for passengers. Given emerging Air-to-Ground (A2G) communication technologies and the high desire of passengers for in-flight services, the servers providing in-flight services can be moved from the airplane to Data Centers (DCs) on the ground. In this scenario, network nodes (airplanes) demanding network services move over a ground core network. Therefore, the selection of DCs to connect to, as well as the underlying routing decisions are challenging. In particular, to keep a low-delay in-flight connection during the flight, airplanes connections can be reconfigured from a DC to another one, which comes at a delay cost.**

This paper presents a formal model for the in-flight service provisioning problem, also as an Integer Linear Program (ILP). We show that the problem is NP-hard and hence propose an efficient online heuristic, HOMA, which addresses the above challenges in polynomial time. HOMA models the problem as a dynamic matching with special properties, and then efficiently solves it by a transformation into the shortest-path routing problem. Our simulation results indicate that HOMA can achieve near-optimal performance and outperform the baseline and state-of-the-art algorithms by up to 15% while reducing the runtime from hours to seconds.

***Index Terms*—Service Selection, Reconfiguration, Service Migration, Dynamic Service Allocation, Mobility-Aware, Mobile Users.**

I. Introduction and Background

Today, with a significant increase in worldwide flight traffic volume, providing in-flight services such as entertainment to onboard passengers has become an important goal for airline companies [1]. Recently, there have been advances in providing high-capacity Air-to-Ground (A2G) communications to airplanes [2], [3]. In particular, Low Earth Orbit (LEO) satellites can relay the airplane traffic towards a gateway on the ground with a high link capacity [4] Also, Direct-Air-to-Ground (DA2G) connections enable airplanes to communicate directly to the base stations located on the ground network [5], [6]. These developments have made new opportunities for airlines to deploy new *online* and interactive services, such as air-to-ground voice/video calls and gaming to improve passenger experience and grow the revenues [1]. To realize these services, airline companies can move the servers providing the in-flight services from the airplane to the distributed Data Centers (DCs) located on the ground. This can also reduce the weight of the airplane, hence the fuel consumption. Due to their real-time and interactive design, these types of services are usually demanding in terms of delay [7]. Thus, it is important to determine the DC to connect to as well as the underlying routing decisions to provide low-delay services to the airplane. Moreover, due to the mobility of airplanes over time, a DC may not remain the efficient one over time (i.e., the distance of the airplane to DC and hence the routing delay can increase during flight).

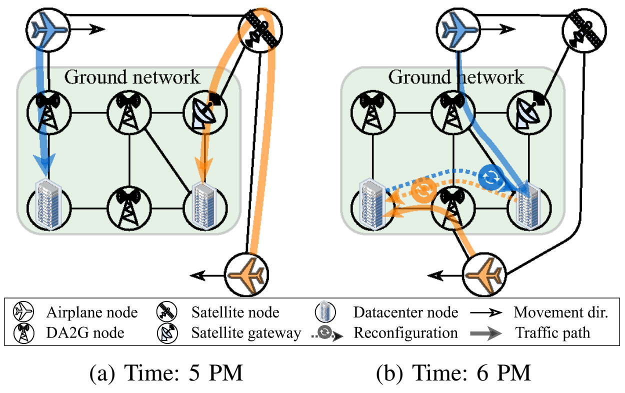


Fig. 1: An exemplary scenario with two flights in two snapshots. (a) shows two airplanes that are connected to the left and the right DC, respectively, at 5 PM. Also, the orange (bottom) airplane cannot use the DA2G connection, since it is out of the tower's range. (b) an hour later, the airplanes are located in different positions with different connectivity opportunities. To improve the delay, the connection of the blue (top) airplane is reconfigured from the left to right DC. With similar reasoning, the orange (bottom) airplane also starts using the left DC. Therefore, we see two reconfigurations at 6 PM.

Therefore, to maintain a continuous low-delay service delivery to the airplanes, the airplane-to-DC assignments can be adapted over time, henceforth called reconfigurations[1]. While these reconfigurations can improve the airplane to DC routing delay, they come at a cost, for example, in terms of the

[1]In this paper, reconfiguring/reconfiguration and reassigning/reassignment terms are used interchangeably

delay (e.g., airplane session's data synchronization between two DCs). For clarification, an exemplary scenario with two flights is shown in Fig. 1.

Given a set of flights, their A2G connections, and a ground core network with capacitated DCs and links, it is of critical importance to choose the best DCs for airplanes to connect to using an efficient routing, as well as the necessary reconfiguration decisions during their flight period to keep the delay of in-flight service delivery minimized in a best-effort approach.

We see our work partially embedded in the Mobile Edge Computing (MEC) research area [8]–[10], with use-cases such as Internet-of-Vehicles (IoV) [11], [12]. Some works [10], [11] assuming that there is a single access point in the network and it is always available for the users to connect to, they focused on service placement on the available MEC servers. Further, by overlooking the end-to-end routing decisions, some works [8], [9] have focused on selecting the access point for the users. While in our case, in addition to selecting the A2G, we determine the routing from an airplane node to the DC with minimum delay. Also, in our scenario, the A2G availability can change over time, e.g., due to DA2G congestion or when an airplane is located out of DA2G communication range. This makes the joint A2G selection, and routing, considering the reconfiguration decisions more challenging. Moreover, some works [10], [11] ignored the capacity of the hosting resources, while we consider the DC capacity in our scenario. Further, in contrast to the existing works [11], we know the future positions of the users (airplanes), since the flight path is predetermined and static, at least in normal circumstances. This feature gives us opportunities to investigate novel approaches to tackle the mobility-aware resource management problems. Further, in the area of in-flight service provisioning, some works [6], [13], [14] have focused on offline algorithms for service-delivery cost optimization and A2G throughput improvement. However, an online algorithm with dynamic and low-delay service provisioning is missing. We believe an online solution would improve the network and resource allocations as it will adapt to late and/or unexpected flight changes with a low computational time.

In summary, the questions that we need to answer can be stated as follows:

1) At each timeslot, which airplane should be assigned to which DC? (Airplane-to-DC Connection)
2) When to reconfigure an airplane connection from a DC to another one? (Reconfiguration Decisions)
3) How to route the airplane traffic towards the selected DC? (Airplane-to-DC Routing)
4) How to jointly answer the above questions with minimum routing and reconfiguration cost (end-to-end delay) over the whole time horizon?

There are several challenges in designing this solution. Firstly, the designed solution needs to be efficient in terms of achieving minimum delay for the overall active flights in the network. Secondly, this algorithm needs to be *fast*, considering the incremental and dynamicity of the input (i.e., the arrival of unknown flight requests, and even emergency changes in the flight paths). Finally, the algorithm must meet different resource constraints such as network and DC capacities.

Thus, the contributions of this paper can be summarized as follows:

- First, we formulate the offline problem as an Integer Linear Programming (ILP) optimization model. Thereafter, by reducing the Generalized Assignment Problem (GAP) [17], we prove its NP-hardness. This shows that the ILP formulation is not scalable for large problem instances. However, the ILP formulation is still useful to compare the optimality of the proposed heuristics.
- To cope with the high complexity of ILP, we propose HOMA, an efficient online algorithm that addresses the in-flight service provisioning problem in polynomial time. We consider a flight-by-flight approach to solve the problem in online settings, where the flight requests are coming in sequence and with an unknown order. From the algorithmic viewpoint, our problem can be mapped to dynamic bipartite matching in graph theory [15], [16]. Specifically, in our scenario, a minimum-weight (routing delay) matching between airplanes and DCs needs to be maintained during the time, while the graph is changing dynamically over time due to the mobility of flights and hence, the availability of different A2G connections to them. Therefore, given a sequence of graphs as input, we formulate the problem into a dynamic bipartite matching problem with special properties. Having a sequence of bipartite matchings to be solved, we transform the matching and reconfiguration decisions in an auxiliary graph and use a shortest-path algorithm (e.g., Dijkstra) to solve the problem in a flight-by-flight fashion. We also show that HOMA finds the optimal solution in specific scenarios.
- Finally, we evaluate the performance of HOMA using extensive simulations in a European-based network with realistic settings. The simulation results show that the HOMA while significantly reducing the runtime, performs very closely to the optimal solution obtained from an ILP solver and outperforms the baseline and state-of-the-art algorithms.

Outline. We model and formulate the offline optimization problem in Sections II and III. In section IV, we introduce HOMA heuristic framework in detail. Then, we evaluate the performance of the proposed approach in Section V. Finally, Section VI discusses the related work, followed by the conclusions in Section VII.

II. PROBLEM FORMULATION

In this section, we present the formulation of the problem, using the notations summarized in Table I. Let us start the mathematical formulation by defining dynamic graphs. Informally speaking, a dynamic graph is a graph that changes over time, either node and/or links. The formal definition is given by Definition 1.

Definition 1. *A discrete and finite number of timeslots is defined as T. We define a dynamic graph as a sequence of T static graphs $G_1 = (N_1, L_1), G_2 = (N_2, L_2), ..., G_T = (N_T, L_T)$.*

Parameter	Definition
T	Set of timeslots
$G_t = (N_t, L_t)$	Dynamic graph at timeslot $t \in T$, N and L are the set of nodes and links of the graph
$\mathcal{N}_{DC}$	Set of data center nodes
$\mathcal{N}_t^F$	Set of airplane nodes at time t
C_j^{DC}	Capacity of DC j
$\mathcal{N}_C$	Set of ground core network nodes
$\mathcal{L}_C$	Set of ground core network links
F	Set of flights
$\mathcal{T}_f$	The duration of flight $f \in F$
$p_{f,t}$	The position of flight f at timeslot t
$b_{f,t}$	The traffic volume of flight f at timeslot t
$D_{u,v}$	The delay of link $uv \in L$
$R_{i,j}$	Reconfiguration delay of assigning an airplane from DC node i to j
$\Psi^+(i), \Psi^-(i)$	Outgoing and incoming nodes from and to node i
$\mathcal{D}_{routing}^T$	Total routing delay
$\mathcal{D}_{reconf}^T$	Total reconfiguration delay
$\mathcal{S}$	Matching solution for a dynamic bipartite graph
$\mathcal{C}(\mathcal{S})$	Total routing delay of solution $\mathcal{S}$
$\mathcal{R}(\mathcal{S})$	Total reconfiguration delay of solution $\mathcal{S}$
$d(j)$	Degree of node j
Decision Variables	
$x_j^{p_{f,t}} \in \{0,1\}$	=1 if flight node $p_{f,t}$ is assigned to DC j at timeslot t
$l_{u,v,j}^{p_{f,t}} \in \{0,1\}$	=1 if link u, v is used to route flight node $p_{f,t}$ to DC j at timeslot t
$r_{i,j}^{p_{f,t}} \in \{0,1\}$	=1 if flight node $p_{f,t}$ is reassigned from DC i to j at timeslot t

TABLE I: Notation definition.

The dynamic graph consists of four parts:

1) Flights: The set of flights is defined as $F = \{f_1, f_2, ..., f_{|F|}\}$. At each timeslot, each flight f has a certain position over the earth (in terms of latitude and longitude), which is denoted by $p_{f,t}$. Also, each flight f has a flying duration of $\mathcal{T}_f$ timeslots. Each airplane location $p_{f,t}$ is a network node in G_t. Therefore, we define the set of airplane nodes as $\mathcal{N}_t^F = \{p_{f,t} : f \in F, t \in \mathcal{T}_f\}$ and $\mathcal{N}_t^F \subset N_t$. Note that a feature of our problem is the prior knowledge of flight positions since the flight path is pre-determined and normally does not change. Moreover, the data rate of each flight (for all the passengers) is denoted by $b_{f,t}$ (e.g., in Mbps). These in-flight services are considered to be interactive/real-time, such as air-to-ground voice/video calls, gaming, etc., with delay requirements of around 100 ms [7].

2) Ground Network: We define $\mathcal{N}_C \subset N_t$ and $\mathcal{L}_C \subset L_t$ as the set of nodes and links of the ground core network, respectively. These services are hosted by DCs located on the ground, defined as $\mathcal{N}_{DC} \subset \mathcal{N}_C$. A limited amount of resources are rented at each DC. Thus, each DC j can process a maximum traffic volume of C_j^{DC}.

3) DA2G Base Stations: We denote the set of DA2G base station nodes as $\mathcal{N}_{DA2G} \subset N_t$, distributed/located on the ground. These nodes can directly communicate to the airplane flying over them and connect them to the ground network. Each DA2G base station has a communication range, usually between 150-250 km based on the technology that they use [17].

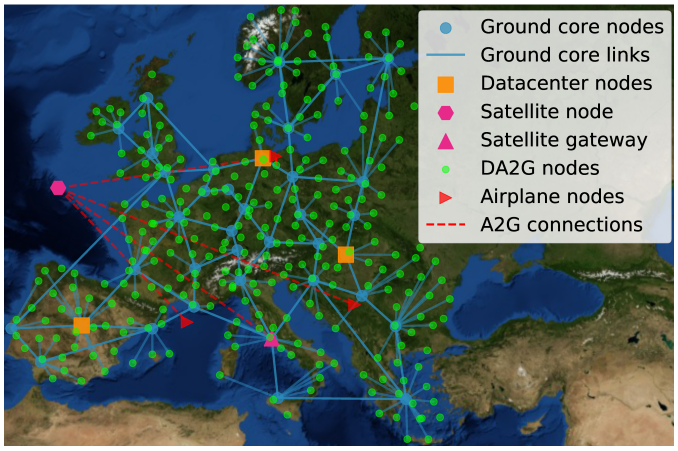


Fig. 2: An instance of graph G_t at a specific timeslot t. It is a realistic Europe-based core network with three DC nodes ($| \mathcal{N}_{DC} | = 3$), and real DA2G base station locations. It shows three flights over Europe ($| \mathcal{N}_t^F | = 3$) with the A2G connections: a representative satellite node and its gateway in Rome, and the connection to the distributed DA2G base stations in Europe.

4) Satellite Connections: We use a single representative satellite node that relays the airplane traffic towards the satellite gateway located on the ground core network. We assume the satellite connection to be available to all the airplanes during the flight. Considering the dynamicity of the satellite constellations are out of the topic of this paper. Therefore, in graph G_t, we connect all the airplane nodes to the satellite node at all its timeslots. Fig. 2 shows an exemplary graph G_t with three flights at a specific timeslot. Finally, we associate a constant propagation delay and a bandwidth capacity to each link $(u, v) \in L_t$, denoted by $D_{u,v}$ and $C_{u,v}^L$, respectively. We note that these values are different for each connection (link) type, i.e., core, satellite, and DA2G. The dynamicity of G_t comes from the mobility of flights and their DA2G connections. As shown in Fig. 3, the availability of DA2G base stations around the airplane position can change due to a congestion model [13], hence the available links of the graph G_t in a particular timeslot.

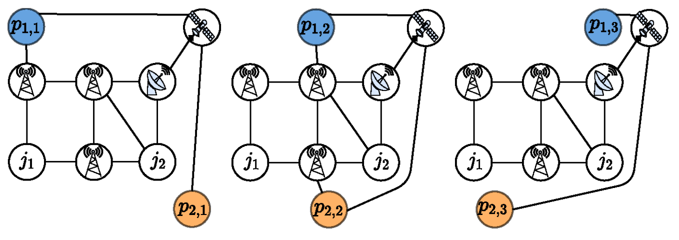
III. Integer Linear Program Formulation

In this section, we present the offline optimization formulation as an Integer Linear Programming (ILP) model. Let us start with two delay functions:

1) Routing Delay: The routing delay (i.e., here delay) is the defined sum of the propagation delay of the path from the airplane to the selected DC in all timeslots, and is denoted by $\mathcal{D}_{routing}^T$:

$$\mathcal{D}_{routing}^T = \sum_{t \in T} \sum_{f \in F} \sum_{j \in \mathcal{N}_{DC}} \sum_{(u,v) \in L_t} l_{u,v,j}^{p_{f,t}} D_{u,v} \quad (1)$$

where $l_{u,v,j}^{p_{f,t}} \in \{0,1\}$ is a decision variable that is equal to 1 if the traffic of airplane node $p_{f,t}$ is routed towards DC j



(a) G_1 $(t = 1)$ (b) G_2 $(t = 2)$ (c) G_3 $(t = 3)$

Fig. 3: An example of the problem, where input is a dynamic graph with two flights, two DC nodes, and three timeslots. With the change of airplane positions over time, the available connection links might change. For example, Flight 2 at $t = 1$, (i.e., node $p_{2,1}$) is connected only to the satellite, while at $t = 2$ (i.e., node $p_{2,2}$), it gets into a DA2G base station range, thus, both A2G options are accessible.

using the link (u, v) at timeslot t.

2) Reconfiguration Delay: Due to the mobility of airplanes, the reassignment of an airplane connection from a DC to another one requires state synchronization and data migration between two DCs. This reassignment has a penalty in terms of overhead traffic and effort, which is defined as the total reassignments delay for all the required reconfigurations $\mathcal{D}^T_{reconf}$ as:

$$\mathcal{D}^T_{reconf} = \sum_{t\in T}\sum_{f\in F}\sum_{(i,j)\in\mathcal{N}_{DC}\times\mathcal{N}_{DC}} r^{p_{f,t}}_{i,j} R_{i,j} \tag{2}$$

where $r^{p_{f,t}}_{i,j} \in \{0, 1\}$ is a decision variable that is equal to 1 if the connection of airplane node $p_{f,t}$ is reconfigured from DC node i to j at timeslot t. Also, $R_{i,j}$ represents the time it takes for an airplane to change its connection from DC i to j.

Let us now define the constraints that need to be met by the model.

Assignment: Assuming that flows are not splittable, so each airplane must be assigned to one and only one DC at each timeslot t:

$$\sum_{j\in\mathcal{N}_{DC}} x^{p_{f,t}}_j = 1, \forall t \in T, \forall f \in F, \tag{3}$$

where $x^{p_{f,t}}_j \in \{0, 1\}$ is a binary decision variable which is equal to 1, if the airplane node $p_{f,t}$ is assigned to DC j at timeslot t.

Traffic Generation: The traffic is generated from the airplane nodes towards a neighbor node in the graph, either the satellite node or a DA2G node (if in range):

$$\sum_{v\in\delta^+(p_{f,t})}\sum_{j\in\mathcal{N}_{DC}} l^{p_{f,t}}_{p_{f,t},v,j} = 1, \forall t \in T, \forall f \in F. \tag{4}$$

We note that $\delta^+(i)/\delta^-(i)$ is a function that returns the outgoing/incoming links from/to node i.

Flow Conservation: These constraints route the traffic flow through the network towards the assigned DC:

$$\sum_{v\in\delta^+(u)}\sum_{j\in\mathcal{N}_{DC}} l^{p_{f,t}}_{u,v,j} - \sum_{v\in\delta^-(u)}\sum_{j\in\mathcal{N}_{DC}} l^{p_{f,t}}_{u,v,j} \tag{5}$$
$$= \begin{cases} 0, \forall t \in T, \forall f \in F, \forall u \in N_t\setminus\mathcal{N}_{DC}, u \neq p_{f,t} \\ x^{p_{f,t}}_u, \forall t \in T, \forall f \in F, \forall u \in \mathcal{N}_{DC} \end{cases}.$$

Routing-Assignment Relation: We need to ensure that the traffic flow of each airplane node at a particular timeslot is forwarded towards the assigned DC:

$$l^{p_{f,t}}_{u,v,j} \leq x^{p_{f,t}}_j, \forall (u, v) \in L_t, \forall t \in T, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \tag{6}$$

In the above constraints, the traffic of the airplane node $p_{f,t}$ can be routed towards DC j, if and only if the airplane node is assigned to it at timeslot t (i.e., $x^{p_{f,t}}_j = 1$).

DC Capacity: As mentioned before, the amount of airplane traffic that is being processed by each DC is limited:

$$\sum_{f\in F} x^{p_{f,t}}_j b_{f,t} \leq C^{DC}_j, \forall t \in T, \forall j \in \mathcal{N}_{DC}, \tag{7}$$

Link Capacity: The amount of the airplane traffic volume that can be delivered on a network link is limited to the link's capacity. To express this, we define the following constraints:

$$\sum_{f\in F}\sum_{j\in\mathcal{N}_{DC}} l^{p_{f,t}}_{u,v,j} b_{f,t} \leq C^L_{u,v}, \forall (u, v) \in L_t, \forall t \in T \tag{8}$$

Reconfigurations: The final set of constraints belongs to the reconfigurations that have to be performed:

$$x^{p_{f,t}}_j = \sum_{i\in\mathcal{N}_{DC}} r^{p_{f,t}}_{i,j}, \forall t \in T\setminus\{0\}, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \tag{9}$$

$$x^{p_{f,t}}_j = \sum_{i\in\mathcal{N}_{DC}} r^{p_{f,t+1}}_{j,i}, \forall t \in T(t \neq T), \forall f \in F, \forall j \in \mathcal{N}_{DC}. \tag{10}$$

Constraints (9) and (10) link x and r variables in a flow conservation way. If $x^{p_{f,t}}_j = 0$, the airplane node $p_{f,t}$ is not assigned to DC j at timeslot t, meaning no reconfigurations. $x^{p_{f,t}}_j = 1$ implies that a reconfiguration assigns the airplane node $p_{f,t}$ to DC j at timeslot t and reassigns it at $t+1$. This reassignment can be from the DC j to j, which incurs zero reconfiguration delay: $\forall i, j \in (J \times J)$, if $i = j$, $R_{i,j} = 0$. We note that reconfiguration is not possible at the first timeslot. Finally, the ILP formulation can be presented as follows. The objective function aims at minimizing the routing and reconfiguration delays over the whole time horizon.

$$\begin{aligned} \text{minimize} \ & \left(\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}\right), \\ \text{s.t.} \ & \text{Constraints } (3) - (10), \\ \text{vars:} \ & x^{p_{f,t}}_j \in \{0, 1\}, \forall t \in T, \forall F \in F, \forall j \in \mathcal{N}_{DC}, \\ & l^{p_{f,t}}_{u,v,j} \in \{0, 1\}, \forall t \in T, \forall f \in F, \\ & \qquad \forall j \in \mathcal{N}_{DC}, \forall (u, v) \in L_t, \\ & r^{p_{f,t}}_{i,j} \in \{0, 1\}, \forall t \in T, \forall f \in F, \forall i, j \in \mathcal{N}_{DC}. \end{aligned} \tag{11}$$

Theorem 1. *The optimization model (11) is NP-hard.*

Proof. Let us begin by defining the Generalized Assignment Problem (GAP). According to [18], GAP determines the minimum-cost assignment of n jobs to m agents, such that considering the capacity restrictions on the agents, each job is assigned to exactly one agent. Therefore, we are able to reduce from GAP to an instance of model (11) with $T = 1$:

$$\begin{aligned} \text{Minimize} \ & \sum_{f \in F} \sum_{j \in \mathcal{N}_{DC}} c_j^{p_f} x_j^{p_f}, \\ \text{s.t.} \ & \sum_{j \in \mathcal{N}_{DC}} x_j^{p_f} = 1, \forall f \in F, \\ & \sum_{f \in F} x_j^{p_f} b_f \leq C_j^{DC}, \forall j \in \mathcal{N}_{DC}, \\ \text{vars:} \ & x_j^{p_f} \in \{0,1\}, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \end{aligned} \tag{12}$$

where $c_j^{p_f}$ is the assignment cost of airplane node (job) p_f to DC (agent) j (e.g., the shortest-path delay value). This transformation is doable in polynomial time. If there exists an algorithm that solves the model (11), it solves the corresponding GAP as well. Given the NP-hardness of GAP [18], the model (11) must be NP-hard too. □

IV. HOMA: THE HEURISTIC FRAMEWORK

In this section, we present HOMA, a deterministic approach that solves the problem mentioned in Section I in polynomial time. The proposed heuristic is formulated as a Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM) problem, as explained in Section IV.A. In this context, a bipartite matching should be maintained (considering the reconfiguration cost) for a sequence of graphs that may change over time dynamically. We solve the DC-MWBM in a flight-by-flight manner where a set of flight requests for the service provisioning during their flight time. This set of flight requests is assumed to arrive in an online manner. Also, the size of this set can be variable in different settings (later we assume it as 1 for the worst case). To solve the problem for a given flight, we transform the problem into a shortest-path routing problem, where using an auxiliary graph, the airplane-to-DC assignments, routing, and reconfiguration decisions are taken. These steps are explained in the following.

A. Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM)

Given the sequence of graphs G_t (see Fig. 3), we transform each G_t to a bipartite graph $\mathcal{G}_t$. We define bipartite graph $\mathcal{G}_t = (\mathcal{N}_t^F, \mathcal{N}_{DC}, \mathcal{L}_t)$. The set of links is defined as $\mathcal{L}_t = \{(f, j, n) : f \in \mathcal{N}_t^F, j \in \mathcal{N}_{DC}, n \in \delta^+(p_{f,t})\}$ which shows a matching from airplane node $p_{f,t}$ to DC j through A2G node n (i.e., n can be satellite or DA2G connection). Each $(f, j, n) \in \mathcal{L}_t$ is associated with a weight function $w(f, j, n) : \mathcal{L}_t \rightarrow R^+$ which is defined as the routing delay between $f \in \mathcal{N}_t^F$ and $j \in \mathcal{N}_{DC}$ through the A2G node n (routing is performed in G_t). Since the position of airplanes changes per timeslot, the link weights from w function also change, i.e., due to the airplanes' mobility, the routing delay between the airplane nodes and DCs changes. Considering that, the total number of links for each $\mathcal{G}_t$ can be at most $2 \cdot \mid \mathcal{N}_t^F \mid \cdot \mid \mathcal{N}_{DC} \mid$. For clarification, Fig. 4 shows the bipartite graphs created for the scenario in Fig. 3. In this example, it is assumed that $\forall f \in F, \mathcal{T}_f = T = 3$.

Having the sequence of bipartite graphs $\mathcal{G}_t$, we define the DC-MWBM problem:

Definition 2. *Given a sequence of bipartite graphs, find a sequence of per-graph matchings, such that the sum of total matching weights and their changes (i.e., reconfigurations) is minimized.*

In this problem, we need to select a set of edges $\mathcal{M}_t \subset L_t$ such that each DC node j has a node degree at most equal to its capacity C_j^{DC}. For simplification purposes, we consider the DC capacity as the maximum number of airplanes that can use the DC resources at the same time.

$$d(j) \leq C_j^{DC}, \forall j \in \mathcal{N}_{DC}, \tag{13}$$

where $d(j)$ indicates the node degree of node j. Also, since we focus on the best-effort delay minimization, we relax the bandwidth constraints for now. However, we later show that in our solution, the bandwidth constraint can be addressed by simply removing the links with not enough available capacity.

Remark 1. *Considering Hall's theorem [19], $\mathcal{G}_t$ can have a matching with cardinality $\mid \mathcal{N}_t^F \mid$. Although $\mid \mathcal{N}_t^F \mid > \mid \mathcal{N}_{DC} \mid$ in $\mathcal{G}_t$; however, by making C_j^{DC} copies of nodes $j \in \mathcal{N}_{DC}$ in $\mathcal{G}_t$, we have $\forall U \subseteq \mathcal{N}_t^F, \mid U \mid \leq \mid \delta^+(U) \mid$ (where $\delta^+(U)$ is the set of DC nodes, i.e., copied nodes, that airplanes can connect to them). This condition is true by having the lower bound value for the DC capacity as $\forall j \in \mathcal{N}_{DC}, C_j^{DC} = \left\lceil \frac{|F|}{|\mathcal{N}_{DC}|} \right\rceil$ (and thus a feasible solution).*

Hence, we ensure that all the airplane nodes in $\mathcal{G}_t$ are matched (connected) to one DC node, i.e.,:

$$d(p_{f,t}) = 1, \forall p_{f,t} \in \mathcal{N}_t^F, t \in T. \tag{14}$$

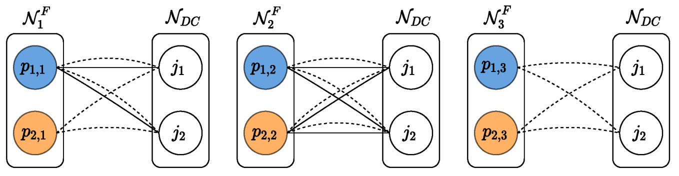
Having a sequence of dynamic bipartite graphs $\mathcal{G}_t$, the solution of the offline DC-MWBM problem is a sequence of matchings $\mathcal{S} = \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_T$, where $\mathcal{M}_t$ is the matching of bipartite graph $\mathcal{G}_t$. Similar to the objective function in ILP model (11), the cost of solution $\mathcal{S}$ can be defined as follows:

1) Matching Cost (routing delay): The total matching cost of a solution $\mathcal{S}$ is the sum of matchings (selected links in bipartite graphs) in all timeslots:

$$\mathcal{C}(\mathcal{S}) = \sum_{t \in T} \sum_{l \in \mathcal{M}_t} w(l), \tag{15}$$

where $l = (f, j, n)$ in $\mathcal{G}_t$. We remind that $\mathcal{C}(\mathcal{S})$ is the sum for the routing delay of all the airplanes to the matched DC nodes in all timeslots.

2) Reconfiguration Cost: We define a per-link reconfiguration cost to model the cost of matching changes (i.e.,



(a) $\mathcal{G}_1$ (b) $\mathcal{G}_2$ (c) $\mathcal{G}_3$

Fig. 4: The sequence of bipartite graphs $\mathcal{G}_t$ is based on the example in Fig. 3. At each timeslot t, airplane nodes $p_{f,t}$ can be matched to a DC node j through an A2G if available: satellite (dashed lines) or DA2G (solid lines). For example, at the $t = 1$, Flight 1 has both DA2G and satellite connections, whereas Flight 2 has only access to the satellite. At $t = 2$, both flights have access to both A2G connections, while at $t = 3$, only have the satellite connection available. Also, at each timeslot t, due to the mobility of airplanes, the link weight (airplane-to-DC routing delay) changes dynamically.

reconfiguration delay). We denote the total reconfiguration cost of solution $\mathcal{S}$ as $\mathcal{R}(\mathcal{S})$, defined as:

$$\mathcal{R}(\mathcal{S}) = \sum_{t \in T \setminus 0} \sum_{(f,j) \in \mathcal{M}_t} \sum_{(f',j') \in \mathcal{M}_{t-1}, (f=f', j \neq j')} R_{j',j}, \quad (16)$$

where $R_{j',j}$ is the cost (delay) of reconfiguring an airplane connection from DC node j' to j in two consecutive timeslots. We note that reconfigurations are not possible at the first timeslot.

Having the cost functions in hand, our goal is to calculate $\mathcal{S}$ for the sequence of $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_T$, minimizing the total matching and reassignment costs:

$$\text{Minimize } \big(\mathcal{C}(\mathcal{S}) + \mathcal{R}(\mathcal{S})\big). \quad (17)$$

Similar to Theorem 1, the DC-MWBM can be proved to be NP-hard, which is omitted due to the lack of space.

B. Solving DC-MWBM

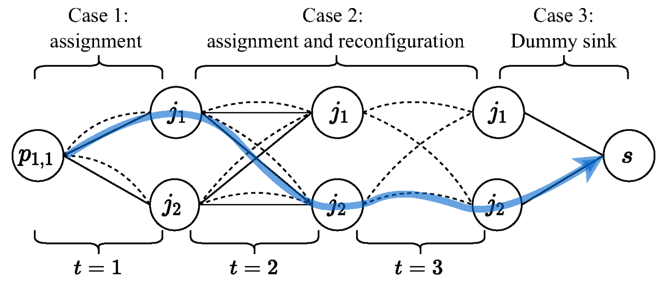
In this subsection, we propose an approach to solve the DC-MWBM problem. At this stage, a request for service provisioning for a set of flights is given as an instance of DC-MWBM. To solve this matching problem, HOMA is designed to work in a flight-by-flight manner. In practice, when a flight is parked in an airport, before takeoff, the control center determines its flight path during the flight. HOMA uses this information (i.e., the airplane locations during the flight) to schedule the DC connections and reconfigurations for the whole flight period. Therefore, given a flight f with duration $\mathcal{T}_f$, we first build an auxiliary graph $\mathcal{G}_A$. We then introduce and solve a shortest-path routing problem in $\mathcal{G}_A$, which gives the solution of service assignment and reconfiguration problem for the given flight.

The main algorithm is aligned in Algorithm 1. The input of the algorithm is a single flight f with duration $\mathcal{T}_f$, for which, we find the airplane-to-DC assignment and the necessary reconfigurations at each timeslot. Also, to keep track of DC capacities, a dictionary with key-value pairs $\big(j, (t, C_j^{DC})\big), j \in \mathcal{N}_{DC}, t \in \mathcal{T}_f$ is calculated and given using the network status from previously served flights. In line 1, we initialize the sequence of bipartite graphs $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_{\mathcal{T}_f}$ for the given flight f. In line 2, we create the auxiliary graph $\mathcal{G}_A$ as a multigraph. We create a dictionary in line 3 to keep the DCs with enough capacity at each timeslot (line 5). Thereafter, we start to build $\mathcal{G}_A$.

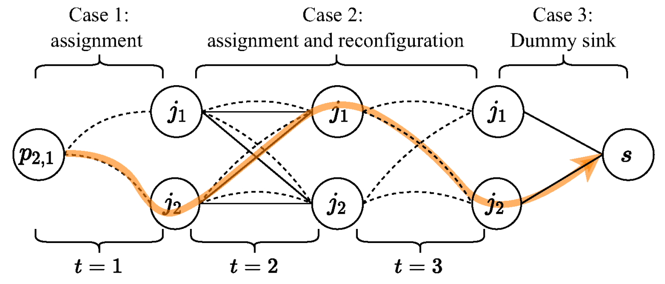
Algorithm 1: HOMA Online Algorithm

```
Input  : Flight $f$, $\mathcal{T}_f$, dc_capacity_dic
Output: Solution $\mathcal{S}$
1  initialize $\mathcal{G}_t, \forall t \in \mathcal{T}_f$ ;
2  $\mathcal{G}_A$ = new MultiGraph() ;
3  available_dcs = {} ;
4  foreach $t \in \mathcal{T}_f$ do
5      available_dcs[t] = get_available_dcs(dc_capacity_dic[t]);
6      airplane_a2g_nodes = $G_t$.get_node_neighbors($p_{f,t}$);
       /* Case 1 */
7      if $t == 1$ then
8          $\mathcal{G}_A$.add_node($p_{f,t}$) ;
9          foreach $j_t \in available\_dcs[t]$ do
10             $\mathcal{G}_A$.add_node($j_t$);
11             foreach $n \in airplane\_a2g\_nodes$ do
12                 delay1 = $G_t$.dijkstra_path_length($p_{f,t}$, $n$) ;
13                 delay2 = $G_t$.dijkstra_path_length($n$, $j_t$);
14                 sum_delay = delay1 + delay2 ;
15                 $\mathcal{G}_A$.add_link($p_{f,t}$, $j_t$, $sum\_delay$);
16     else
           /* Case 2 */
17         foreach $j'_{t-1} \in available\_dcs[t-1]$ do
18             foreach $j_t \in available\_dcs[t]$ do
19                 $\mathcal{G}_A$.add_node($j_t$);
20                 foreach $n \in airplane\_a2g\_nodes$ do
21                     delay1 = $G_t$.dijkstra_path_length($p_{f,t}$,
                        $n$);
22                     delay2 = $G_t$.dijkstra_path_length($n$, $j_t$);
23                     sum_delay = delay1 + delay2;
24                     if $j'_{t-1} \neq j_t$ then
25                         sum_delay += $R_{j'_{t-1},j_t}$;
26                     $\mathcal{G}_A$.add_link($j'_{t-1}$, $j_t$, $sum\_delay$);
   /* Case 3 */
27 $\mathcal{G}_A$.add_node($s$);
28 foreach $j_t \in available\_dcs[\mathcal{T}_f - 1]$ do
29     $\mathcal{G}_A$.add_link($j_t$, $s$, 0);
30 $\mathcal{S}$ = $\mathcal{G}_A$.dijkstra_path($p_{f,0}$, $s$);
31 update_capacity(dc_capacity_dic, $\mathcal{S}$));
32 return $\mathcal{S}$;
```

1) Case 1, $t = 1$: This is the case for the first timeslot of the flight f. In this case, we first add the airplane node $p_{f,t}$ to $\mathcal{G}_A$ as the first node (line 8). Then, we create a node for each available DC node (line 10), and connect the airplane node $p_{f,t}$ node at $t = 1$ to it (lines 11-15). These links represent an assignment decision, for instance link $(p_{f,1}, j_1)$ means airplane node f is assigned to DC j_1 at timeslot $t = 1$. As mentioned before, each airplane node can connect to a DC using one of the A2G connections (i.e., satellite or DA2G), which is determined in line 6. Therefore, for each available A2G (line 11), and add a link for each of them (line 15). For example, in Fig.3a, it can be seen that at $t = 1$, flight $f = 2$ has only satellite connection. Therefore, in Fig. 5b



(a) $\mathcal{G}_A$ for the exemplary flight $f = 1$.



(b) $\mathcal{G}_A$ for the exemplary flight $f = 2$.

Fig. 5: Building the auxiliary graph $\mathcal{G}_A$, and solving the DC-MWBM problem for the scenario in Figs. 3 and 4 in a flight-by-flight manner. (a) The auxiliary graph $\mathcal{G}_A$ and solution for $f = 1$ is built with the possible assignment and reconfiguration decisions through the flight time. It can be seen that at $t = 3$, only satellite connection is available (dashed line). The calculated shortest-path from $p_{1,1}$ to s is $p_{1,1} \dashrightarrow j_1 \rightarrow j_2 \dashrightarrow j_2 \rightarrow s$. This path means at $t = 1$, the airplane node $p_{1,1}$ is connected to DC j_1 through DA2G (the one in range), at $t = 2$ the airplane node $p_{1,2}$ is connected to j_2 using DA2G (one reconfiguration from j_1 to j_2), and at $t = 3$, the airplane node $p_{1,3}$ continues connecting to DC j_2 through satellite. (b) The calculated shortest-path here is $p_{2,1} \dashrightarrow j_2 \rightarrow j_1 \dashrightarrow j_2 \rightarrow s$, which uses satellite, DA2G, and satellite to connect to DCs j_2, j_1, and j_2 at timeslots $t = 1$, $t = 2$, and $t = 3$, respectively. For this flight, two reconfiguration at $t = 2$ and $t = 3$ between DC $j_2 - j_1$, and $j_1 - j_2$ are determined, respectively. We note that the reconfiguration between DA2G and satellite connection inside the airplane can be done with an almost instant routing table update using e.g., OpenFlow [20].

at $t = 1$, we connect the airplane node $p_{2,1}$ to the DCs only using the satellite connection (dashed line). For each of these links, we define a weight, which will be used later to calculate the shortest path. These weights are defined as the delay of the shortest path between the airplane and DC nodes (in the original graph G_t) through the corresponding A2G connection. Note that, before calculating the shortest path, the links with lower than $b_{f,t}$ available capacity are removed from the network. Therefore, in lines 12 and 13, we calculate the shortest-path delay from the airplane node $p_{f,t}$ to the neighbor node n (i.e., either satellite or DA2G node), and from n to the DC node, respectively. We then sum up these two delay values (line 14) and finally create a link between the airplane and DC node with the total delay value as its weight (line 15).

2) Case 2, $1 < t \leq \mathcal{T}_f$: In this case, we continue building $\mathcal{G}_A$ by connecting DC nodes in two consecutive timeslots. We create a link between the DC nodes in $t-1$ and $t, \forall 1 < t \leq \mathcal{T}_f$ (lines 17-26). The link (i_{t-1}, j_t) which connected DCs i and j in $\mathcal{G}_A$, indicates that in two consecutive timeslots $t-1$ and t, the airplane nodes $p_{f,t-1}$ and $p_{f,t}$ are assigned to DC nodes i and j, respectively. Compared to the first case, in addition to the assignment decisions, the possibility of reconfigurations is enabled here. When $i \neq j$, the link represents a reconfiguration from DC i to j at timeslot t. We loop through the previous DC nodes at $t-1$ (line 17) and also at t (line 18). We create a node in $\mathcal{G}_A$ for all the available DC nodes j_t (line 19). Similar to the first case, a maximum of two links can exist between nodes (per A2G connection type). We calculate the delay of the shortest-path between $p_{f,t}$ and DC t through the neighbor node (i.e., A2G connections) n in G_t. In case two DC nodes at $t-1$ and t are not the same node in G_t, it indicates a reconfiguration decision. Therefore, we add the reconfiguration cost between two DC nodes to the link weight (line 25). Finally, in line 26, we add a link to $\mathcal{G}_A$ from node j_{t-1} to j_t with the summed routing delay as the weight.

3) Case 3, dummy sink: This is the case where we add a dummy sink node s to $\mathcal{G}_A$ in line 27. Thereafter, we connect the DC nodes in timeslot $t = \mathcal{T}_f$ (last flight timeslot) to s with weight 0 (line 29).

Having the auxiliary graph $\mathcal{G}_A$ built, we can calculate the solution for the flight f. To do this, we simply need to calculate the shortest path from $p_{f,1}$ and the dummy sink node s nodes using conventional algorithms e.g., Dijkstra (line 30). The returned path gives us the assignment, routing, and reconfiguration decisions with minimum total delay. A comprehensive example is presented in Fig. 5. Finally, the DC capacities are updated in line 31 to be used for future flights.

Remark 2. *Algorithm 1 can solve the problem for flights of any duration. In particular, we only need to keep track of DC capacities per timeslot. That is, the $\mathcal{G}_A$ can be formed for a flight with any duration. The number of nodes and links for $\mathcal{G}_A$ would depend on the flight duration.*

Remark 3. *For a given flight f with duration $\mathcal{T}_f$ timeslots, graph $\mathcal{G}_A$ can have $\mathcal{O}(\mathcal{T}_f \cdot \mid \mathcal{N}_{DC} \mid)$ nodes and $\mathcal{O}(\mathcal{T}_f \cdot \mid \mathcal{N}_{DC} \mid^2)$ links.*

Lemma 2. *For any given flight f with duration $\mathcal{T}_f$,* HOMA *is complete*[2] *and optimal.*

Proof. For a given flight f, we build the auxiliary graph $\mathcal{G}_A$ with the available DC nodes. Also, we add all the possible routing options (DA2G and/or satellite) with the weight defined as the shortest-path delay between the airplane and DC. Also, we consider all possible assignments and reconfiguration decisions in $\mathcal{G}_A$. Therefore, since Dijkstra is complete and

[2]An algorithm is complete if it always finds a solution if one exists. The completeness does not imply optimality.

optimal [21], for the given flight f, HOMA is also complete and optimal. $\square$

Theorem 3. *For any given set of flights F, if $\forall j \in \mathcal{N}_{DC}, C_j^{DC} \geq \mid F \mid$, HOMA is complete and optimal.*

Proof. Flights in F depend on each other through the DC capacity. If $\forall j \in \mathcal{N}_{DC}, C_j^{DC} \geq \mid F \mid$, it simply means at each timeslot, a single DC node can host all the flights. Thus, the DC capacity constraints can be relaxed in this case. Therefore, according to the Lemma 2, HOMA is complete and optimal for all the set of flight F. $\square$

Theorem 4. *For any given flight f, the complexity of* HOMA *for finding the optimal solution is $O(\mathcal{T}_f \cdot |N_t|^2)$.*

Proof. We calculate the runtime based on the number of times that the Dijkstra algorithm needs to be performed to find the solution for flight f. The Dijkstra algorithm needs to be executed for each link in the auxiliary graph $\mathcal{G}_A$, except the links for the dummy destination node s. Therefore, considering Remark 3, we need $2 \cdot (\mathcal{T}_f - 1) \cdot \mid \mathcal{N}_{DC} \mid^2 + \mid \mathcal{N}_{DC} \mid$ links (i.e., Dijkstra runs) to build the $\mathcal{G}_A$. Each Dijkstra run is executed for $G_t = (N_t, L_t)$, with runtime $O(\mid N_t \mid^2)$. Since $\mid \mathcal{N}_{DC} \mid \ll \mid N_t \mid$, the overall algorithm runtime for any flight f can be expressed as $O(\mathcal{T}_f \cdot |N_t|^2)$. We note that in use-cases where the duration of the timeslots is very small (close to 0), the algorithm complexity is pseudo-polynomial (based on $\mathcal{T}_f$). $\square$

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of HOMA in various scenarios and settings. We first introduce the algorithms that we compare with the optimal offline (ILP) solution. Thereafter, we explain the simulation setup, input parameters, and scenarios. Finally, we present the results of the performance evaluation of HOMA against the offline optimal solution (section III) and the two baseline algorithms.

A. Algorithms to Compare

For comparison purposes, we re-implemented the algorithm presented in the state-of-the-art work [11]. Further, we develop two variations of HOMA (Minimize Delay - MD, and Maintain Connection - MC) as the baseline algorithms. Below, we explain these approaches in short.

1) Threshold-Based Reconfiguration (TBR): The first algorithm to compare with HOMA is taken from the state-of-the-art work [11]. In [11], the authors propose a resource management framework for service placement and migration for the automotive use case, within the Mobile Edge Computing (MEC) scenario. Similar to us, they have focused on providing low-delay services to the mobile users (cars in their case), while determining the appropriate MEC server to host the services, and the necessary reconfigurations according to the mobility of the cars. In their approach, they set a threshold for which, if the current routing delay value is lower than a threshold value, the service is migrated to another MEC server. This MEC server (destination of the migration) is chosen as the farthest one, to lower the number of future migrations. For comparison purposes, we have chosen two threshold values, 20 and 40 ms. We show these two algorithm versions as TBR-20 and TBR-40, respectively.

However, in contrast to us, the approach presented in [11] lacks considering a few constraints. Firstly, they do not consider the DC and link capacities into the account. Secondly, they ignore the service migration (reconfiguration) penalty in their algorithm. Nevertheless, for a fair comparison, we have added these considerations into the implementation of their algorithm.

2) Minimize Delay (MD): The next algorithm is a baseline that focuses on minimizing the routing delay between the airplane nodes and the assigned DC. MD reconfigures the airplane connections whenever it leads to a lower routing delay. At $t = 1$, it chooses the DC with the lowest routing delay to connect the airplane to it. For $t > 1$, it reconfigures the airplane connection to a DC which keeps the routing delay at that specific timeslot minimized. Regarding the implementation, it is enough to set $\forall i, j \in \mathcal{N}_{DC}, R_{i,j} = 0$, i.e., reconfiguration becomes a decision with no cost. Thus, in the procedure of building $\mathcal{G}_A$, in the case 2 where $t > 1$, we update the weight of link $(i, j), i, j \in \mathcal{N}_{DC}$ with zero reconfiguration cost, i.e., changing line 25 to $sum\_delay \mathrel{+}= 0$ in Algorithm 1.

3) Maintain Connection (MC): As it appears from its name, this algorithm always maintains its connection to the first selected DC. The goal of developing this algorithm is to highlight the importance of providing flexibility in the network (i.e., reconfigurations). At the first timeslot of the flight, MC connects the airplane to the DC with minimum routing delay and keeps using it for the rest of the flight duration. The implementation is done with a small tweak in Algorithm 1, in Case 2, where $t > 1$. In particular, while building the $\mathcal{G}_A$, in two consecutive timeslots $t-1$ and t, MC adds links only between DC nodes i_{t-1} and j_t, if $i = j$ where $i, j \in \mathcal{N}_{DC}$. In this way, the reconfiguration of airplane connections would not be possible for the Algorithm (since there are no links between different DCs in $\mathcal{G}_A$).

B. Simulation Setup

In this subsection, we explain the simulation setup and input parameters used in the simulations for evaluating the performance of the proposed algorithm.

The set of flight requests (i.e., F) was exported from FlightRadar24 live air traffic for 24 hours on 9.11.2017. We extracted the position of flights at each timeslot and use it as $p_{f,t}$ parameter. In our experiments, we select a set of flights F, such that $\mid F \mid = \{20, 50, 100, 300, 500\}$. For simplification, we assume that all the airplanes are transmitting the same amount of traffic $b_{f,t}$ during all the timeslots. As the ILP takes several hours to solve a problem instance with 100 flights, the number of timeslots has been limited to eight, so we can have problem instances with a high number of flights. Thus, we choose F with two different flight duration, long and short flights, with four and eight timeslots, respectively. For simplicity, the duration of each timeslot is considered 30

Description	Values
Number of flights ($\mid F \mid$)	20, **50**, 100, 300, 500
Flight duration ($\mathcal{T}_f, \forall f \in F$)	4, **8**
Number of DCs ($\mid \mathcal{N}_{DC} \mid$)	3, **6**
DC capacity ($C_j^{DC}, \forall j \in \mathcal{N}_{DC}$)	*low*, ***medium***, *high*

TABLE II: A summary of important input parameters and their values used for the performance evaluation. The default value is in bold text.

minutes (this point is discussed in section V-D.D). Indeed, the number of timeslots depends on the airplane speed and the range of the DA2G and satellite coverage, which can lead to an even different number of timeslots per flight. However, to simplify the evaluation scenario, we consider the same number of timeslots and duration for all the flights.

The simulation settings are based on a realistic European-based flight space (See Fig. 2). We use the European Cost266 topology [22] for the ground core network and set the values of $\mathcal{N}_C$ and $\mathcal{L}_C$ accordingly. The delay of links ($D_{u,v}, u, v \in \mathcal{N}_C$) in $\mathcal{L}_C$ is determined according to the length of the optical fiber transmissions. Regarding the DC nodes, we choose two sets of locations, distributed over Europe [23]: a small set $\mathcal{N}_{DC}$={Hamburg, Madrid, Budapest}, and a larger set $\mathcal{N}_{DC}$={Strasbourg, Stockholm, Madrid, Athens, Glasgow, Krakow}. We set the value of DC capacity C_j^{DC} by considering three cases:

1) *low:* $\forall j \in \mathcal{N}_{DC}, C_j^{DC} = \left\lceil \frac{|F|}{|\mathcal{N}_{DC}|} \right\rceil$ (note that lower DC capacity makes the problem infeasible, see Remark 1).
2) *high:* it sets the DC capacity equal to the total number of flights in the network $\forall j \in \mathcal{N}_{DC}, C_j^{DC} = \mid F \mid$.
3) *medium (med):* the average of *low* and *high* values.

Since the focus of this work is to provide (best-effort) low-delay in-flight services, we consider $\forall f \in F, t \in \mathcal{T}_f, b_{f,t} = 1$ with a relaxation on the bandwidth constraints for the A2G and ground network links. There are 295 DA2G base stations (already deployed in Europe), and their location (in terms of latitude and longitude) is taken from [24]. Also, the propagation delay from airplane to the DA2G base station is set to 10 ms [17] (i.e., $D_{u,v}, u \in \mathcal{N}_t^F, v \in \mathcal{N}_{DA2G}$). Also, we consider the DA2G connectivity range as 350 km [25], although in practice it can vary based on, e.g., the antenna employed technology and weather conditions. Without loss of generality, we connect the airplane nodes $p_{f,t}$ to the closest DA2G base station in range. Further, to make our scenario more realistic, we consider a DA2G congestion model from [13]. Using the flight routes in our data that pass through the DA2G base station coverage, this congestion model selects a subset of DA2G nodes to be congested during all the timeslots (i.e., cannot accept any airplane connections).

For the satellite connection, we consider the LEO satellites. According to [26], the number of LEO satellites existing over Europe is rather low (5 out of the 72 LEO satellites in the Iridium constellation). Compared to the Geostationary Orbit (GEO) satellites, LEO causes lower delay, which is more suitable for our use case. Therefore, in this paper, we only consider the LEO constellation for the evaluation scenario. For simplification purposes, we apply an abstraction model, where the group of LEO satellites occupying Europe is represented as a single satellite node. However, to avoid unrealistic assumptions, we set the latency of the satellite link to the worst-case achievable latency 50 ms (i.e., $D_{u,v}, u \in \mathcal{N}_t^F, v \in \mathcal{N}_C$ through the satellite node). This value is calculated according to the LEO delay calculations provided in [27]. Also, the satellite gateway node is considered to be located in Rome, Italy [28]. An instance of the built European-based scenario can be seen in Fig. 2 (for a single timeslot).

Regarding the reconfiguration overhead, we use the distance between source and destination DCs:

$$R_{i,j} = \begin{cases} M * dijkstra\_path\_length(i,j), i,j \in \mathcal{N}_{DC} \ (i \neq j), \\ 0, i,j \in \mathcal{N}_{DC} \ (i = j), \end{cases}$$

M is a random number between 0 and 2. We note that the modeling of the service migration is out of the scope of this paper. However, different $R_{i,j}$ models can be plugged into the HOMA, depending on the specific application.

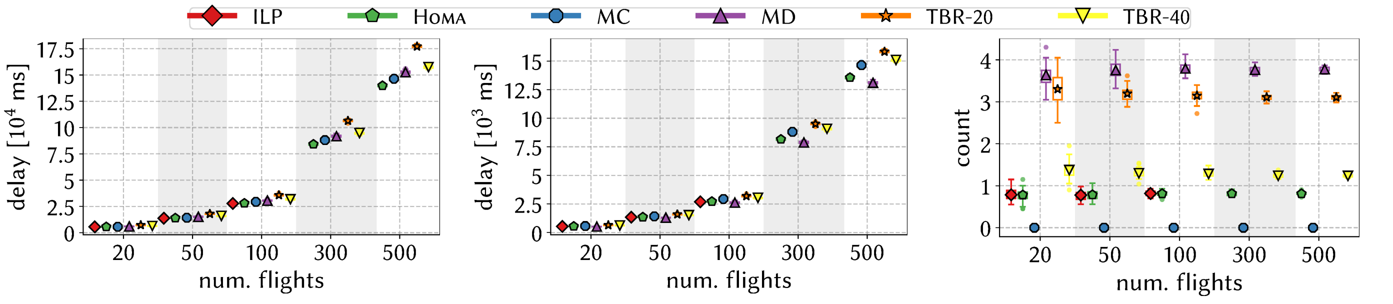
To design different experiments, we use the input parameters and their values in Table II where in each experiment, three parameters are fixed and the fourth one is varied. We implement and solve the ILP using Gurobi [29]. Also, HOMA, ***TBR***, ***MC***, and ***MD*** approaches are implemented in Python.

We perform the evaluations for 30 random sets of scenarios on a machine equipped with Intel Core i7-6700 CPU3.40 GHz, 16 GB RAM, and running Arch Linux with kernel 5.5.11-arch1-1.

C. Simulation Results

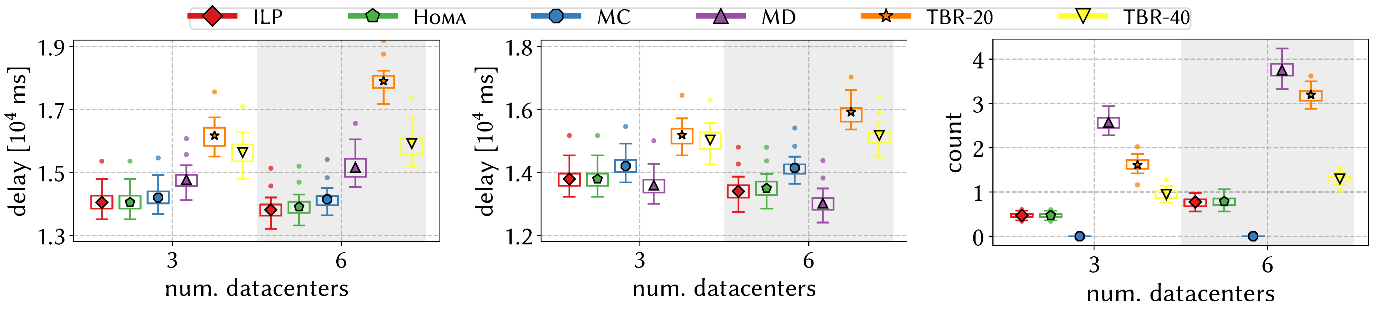
In this subsection, we present the simulation results for different parameters and experiments.

1) Number of Flights: We first consider the default values for flight duration ($\mathcal{T}_f$), number of DCs ($\mid \mathcal{N}_{DC} \mid$), and DC capacity ($C_j^{DC}$) according to Table II. The effect of number of flights on total delay (i.e., the objective function value), routing delay, and number of reconfigurations can be seen in Fig. 6a-6c. In Fig. 6a, it can be seen that HOMA achieves a near-optimal objective function value with around 1% of distance from the optimal. Also, HOMA outperforms the baseline and state-of-the-art algorithms for up to 15% on average for a high number of flights. The MD algorithm achieves a lower routing delay compared to HOMA (see Fig. 6b) since it greedily reconfigures the airplane connection to the closest DC to minimize the routing delay. However, it requires a significant number of reconfigurations to be able to minimize the airplane to DC delay (See Fig. 6c). Further, MC leads to a larger routing delay compared to HOMA, since it maintains the connection of the airplanes to the first selected DC for the whole flight period and does not reconfigure the connection. Therefore, during the flight, when the airplane gets far from the DC, the routing delay also increases. Also, it can be seen that the TBR-20 and TBR-40 lead to a higher objective value function. The reason is in TBR, determining the threshold value for triggering the reconfiguration is challenging, especially when



(a) Total delay ($\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}$). (b) Total routing delay ($\mathcal{D}^T_{routing}$). (c) Number of reconfigurations.

Fig. 6: Comparison of the simulation results for different number of flights.



(a) Total delay ($\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}$). (b) Total routing delay ($\mathcal{D}^T_{routing}$). (c) Number of reconfigurations.

Fig. 7: Comparison of the simulation results for different number of DCs.

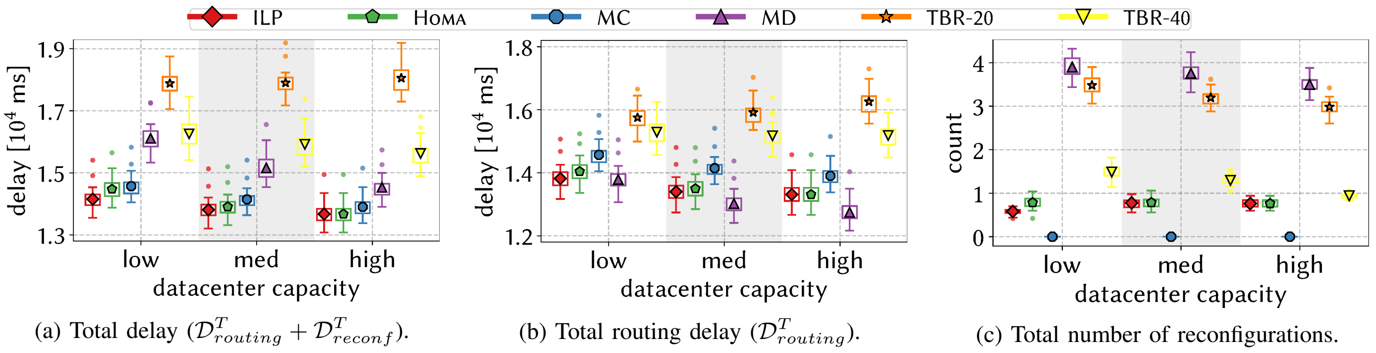
the goal is to minimize the overall delay. Lower threshold values (i.e., TBR-20) can lead to higher total delays and more reconfigurations are trigerred for TBR (see Fig. 6c). According to its design, TBR reconfigures the service to the farthest DC, which increases the routing delay. In addition, since TBR does not consider the reconfiguration cost into the account, the balance between the routing delay and reconfiguration delay is missed, hence, leading to sub-optimal decisions. Fig. 6c shows the number of per-flight reconfigurations for the flight duration, in which, the proposed HOMA approach can make near-optimal reconfiguration decisions for the flights. Also, as expected, MC does not perform any reconfigurations.

All in all, HOMA can accurately balance the airplane to DC routing delay and reconfigurations, demonstrating a near-optimal behavior, while outperforming the baseline and state-of-the-art algorithms. The algorithms exhibit similar behavior with short flights. However, we omit these results from the paper due to space limitations. We note that we can derive the average round-trip delay per flight per timeslot from Fig. 6a, which is around 68 ms in the worst case for 500 flights for the proposed HOMA approach (requirements being around 100 ms [7]), which can support a wide range of interactive services such as air-to-ground voice calls, video conferencing, and gaming [7].

2) Number of DCs: Fig. 7a-7c show the evaluation results of the algorithms for a varied number of DCs, while keeping the default values for the other parameters. The goal here is to compare the behavior of different approaches when the number of available DCs changes. This can assist operators in more accurate capacity planning. Regarding the total delay, Fig. 7a indicates that generally, a lower number of DCs leads to a higher delay value. That is because, with a higher number of DCs, airplane nodes get in a closer distance to a DC node on average during their flight. This can be seen also in Fig. 7b, where the routing delay is generally lower in the case with six DC nodes. However, the TBR algorithm does not follow this trend, with each reconfiguration, the routing delay increases (since the reconfiguration is done towards the farthest DC).

Also, it can be seen that the difference in total delay achieved by the algorithms, is generally lower for three DCs compared to six. This is because the role of DC capacity (which is the main cause of sub-optimality) is more critical with a higher number of DCs (i.e., there is more room to be sub-optimal).

However, regarding the number of reconfigurations, Fig. 7c shows that the case with six DCs has around 20% more reconfigurations compared to the case with three, since it brings more flexibility and more opportunities for reconfiguration, especially for the TBR algorithm). Of course, this value gets lower with services with higher reconfiguration delay. However, still, the case with three DC makes more sense, since the proposed approach, HOMA, leads to low-delay solutions, while the number of reconfigurations is kept low. The results show that using HOMA can reduce the operators' costs, by renting/building fewer DCs, while providing an *good* solution quality.



(a) Total delay ($\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}$).

(b) Total routing delay ($\mathcal{D}^T_{routing}$).

(c) Total number of reconfigurations.

Fig. 8: Comparison of the simulation results for different DC capacities.

	$\lvert F\rvert = 20$	$\lvert F\rvert = 50$	$\lvert F\rvert = 100$	$\lvert F\rvert = 300$	$\lvert F\rvert = 500$
ILP	187.1	569.2	12412.4	N/A	N/A
Homa	15.3	52.2	166.8	1175.4	3216.5
MD	16.8	44.5	151.9	1014.7	3140.1
MC	4.4	11.2	57.5	123.5	163.6
TBR-20	0.8	3.3	11.2	94.63	278.54
TBR-40	0.5	2.4	8.7	80.8	228.1

TABLE III: The comparison of the mean runtime of the different approaches (in seconds) for the default input parameters.

3) DC Capacity: In this study, we use the default values for the number of flights, flight duration, and the number of DC nodes, and vary the DC capacity as per Table II. The goal of this study is to find out what is the impact of different DC capacities on the performance of different algorithms. Fig. 8a shows the comparison of the achieved total delay by the algorithms for *low*, *medium*, and *high* DC capacities. Since our approach is online (flight-by-flight), the DC capacity can play a big role in having the available resources in a *good* DC location when needed. It can be seen that generally, the behavior of the algorithms is similar for the cases with different DC capacities. However, the total delay decreases with higher DC capacity. Also, the same trend is there for the routing delay as in Fig. 8b. The reason behind this is there are more DCs available to serve the airplanes (i.e., the average distance between airplanes and DCs is decreased). Moreover, as proved in Theorem 3, it is evident that HOMA is able to take optimal decisions for DC selection and reconfiguration in case of *high* DC capacity. Finally, Fig. 8c shows that the DC capacity does not change the number of reconfigurations much for the optimal and HOMA, i.e., they stay around one reconfiguration per flight for all the DC capacities.

These evaluations show that it is not necessary to rent plenty of resources at each DC. Medium or even low amount of resources could provide a *good* solution. This indicates that in contrast to other algorithms, HOMA can lead to cost savings for the operators.

4) Runtime: The runtime comparison of the optimal offline solution (i.e., ILP), HOMA, MD, MC, and TBR algorithms is presented in Table III. Firstly, this table shows that the ILP formulation is significantly slower than the other heuristics. Also, it can be seen that HOMA can solve the problem for 500 flights in around six seconds per flight. Also, MC has the lowest runtime, since it does not consider the reconfiguration decisions, thus the number of Dijkstra runs is considerably lower, compared to the HOMA, and MD approaches. Finally, the TBR algorithms are the fastest ones, since they are less sophisticated in nature.

D. Remarks

Let us summarize some remarks related to the design of HOMA. Firstly, we choose to adopt the discrete-time model mainly due to its simplicity. Furthermore, this model can be arbitrarily close to the continuous-time model by making the timeslot duration very small. However, it is not the case for our scenario. Considering the airplane speed and the A2G coverage range, the decision-making can be done in longer periods (on a scale of a few minutes).

Secondly, we consider the flight paths to be a priori knowledge, once planes are ready to take off. In practice, the paths can be changed in some emergencies (e.g., bad weather conditions). In this case, HOMA can be rerun to adapt the previously-made decisions using the updated network status (e.g., network link availability/congestion status) according to the emergency.

Third, in this work, the LEO satellite constellation has been modeled with a single representative node with the worst-case delay value. The integration of the dynamic satellite network considering the cost/delay of the satellite handover could be added to our approach as follows: the handover cost between two consecutive timeslots could be added to the cost of satellite connection links (i.e., dashed lines in Fig. 4).

Finally, we point out that HOMA does not guarantee the service-specific delay requirements but follows a best-effort approach toward delivering low-delay in-flight services to the airplane. However, the routing decision-making part of HOMA can be enhanced with deterministic networking theories such as Network Calculus [30], to provide a delay guarantee for the critical traffic flows (e.g., flight status).

VI. RELATED WORK

The related work can be discussed in different categories. A closely-related domain to our work is space-air-ground integrated (terrestrial) networks. [31], [32]. In this area, there are studies that have focused on resource allocation [13], [14], [25], [33]–[35], satellite-based vehicular networks [36], reliability [37], and energy-efficiency [38]. For example, Varasteh et. al. [13], [25] has presented an offline mobility-aware optimization formulation for the joint service placement, routing, and migration for flying airplanes. Further, in [35], they have extended their work to use reinforcement learning for decision making. Moreover, authors in [14], [34] have focused on the onboard A2G communication analysis. In particular, in [14], the authors have performed a throughput analysis of DA2G communication during its flight. Also, [34] has presented a QoS-aware approach to satisfy the passengers' traffic flows using different A2G alternatives during a flight.

On the other hand, some works have focused on the resource allocation and management of MEC environments [39]–[43]. In [39], authors have proposed a graph-based algorithm to determine the number of servers, their size, and the operation area. Considering a maximum resource capacity for the MEC servers, their goal is to serve the maximum number of users at the edge area. Similar to them, the authors in [40] tackled the problem of edge server placement. They have proposed a multi-objective formulation for balancing the workloads on the MEC servers and reducing the delay between the clients (industrial control centers in their case) and MEC servers.

A closer area to our work is where the works have considered the resource allocation reconfiguration problem with mobile users [8]–[10], [44]–[52]. Authors in [46], have proposed a mobility-aware service placement approach, considering the mobility of users and location, to minimize the service access delay and deployment costs. Further, some works [49], [51] have studied the dynamic placement of network functions on MEC servers according to the future handover probability of users. Specifically, [49] proposes two service migration solution for MEC-based applications: *reactive*, and *proactive*. There are other works with a focus on proactive service provisioning using a look-ahead window and pre-allocation techniques [43], [47]. Also, authors in [10] have proposed a Virtual Machine (VM) migration method, called Follow-Me Cloud, to deal with user mobility. In more detail, the aim of them is to always connect the mobile users to the optimal gateway, while the service instance (i.e., VM) *follows* the users, using the migration technologies. They have proposed an approach based on the Markov decision process to determine the appropriate migration decisions. However, they do not consider the network and servers capacity limits.

In the IoV area, there are similar works [11], [12], [53], [54] that provide service continuity to users (vehicles), considering their mobility. For instance, Zhang et. al in [53] have focused on the placement of a constant number of edge servers to improve the QoS (i.e., reducing the average waiting time of services in the IoV), and load balancing. Also, the authors in [11] present a 5G-enabled framework that migrates the services based on the mobility of the vehicles along the road. They keep monitoring the delay between the vehicle and the MEC server. In case the delay goes higher than a threshold, they migrate the service to another MEC in the direction of the vehicle movement. However, these works overlook the end-to-end routing, reconfiguration overheads, and resource capacities. Also, some works [10], [51] assume that the access points are always available to the users to connect to. Also, some of them overlooked the end-to-end routing decisions [8], [9], [11], [50], DC capacity [8], [9], [11].

Finally, from an algorithmic viewpoint, this work is related to the matching problem in graph theory. The matching problem has a long history of research, going back to decades ago [55], [56]. Due to the enormous applications of matching in practice [57], there is a huge body of research for online (bipartite) *static* matching and its variants [58]–[65], [65], [66]. However, due to the dynamic nature of many matching problems in practice (e.g., considering the temporal/dynamic graphs), recently, dynamic matching has been receiving attentions [15], [16], [67]–[69]. Despite the similarity in names, our problem in this paper is different from others. In our problem, there is a sequence of graphs that change dynamically due to airplanes' mobility. Therefore, the goal is to maintain a minimum-weight bipartite matching, considering the mobility of airplanes.

VII. Conclusions

This paper initiated the study of an online service provisioning problem in the context of emerging air-to-ground communication technologies. We presented a formal model together with an efficient algorithm, HOMA to decide about airplane-to-DC connections, routing, and possible reconfiguration decisions over the flight period. HOMA showed a *good* performance in realistic scenarios, achieving a near-optimal objective function value, while significantly reducing the runtime. We see our work as a first step and believe that it opens several interesting avenues for future research. In particular, it would be interesting to explore the achievable competitive ratio in deterministic and randomized online settings and explore how to optimize and re-optimize schedules in scenarios that feature some uncertainty and in which a bounded number of unexpected events may occur.

We note that a similar approach can be applied to other areas such as Autonomous Driving where a vehicle needs to travel from point A to B in a city. Considering the positions of the vehicle at different times, the required services can be scheduled on the MEC servers available in the city.

Acknowledgment

The authors would like to thank Nemanja Deric, Stefan Schmid, and Mojgan Madadikhaljan for their support and constructive feedback. This work is funded by Germany Federal Ministry of Education and Research under project AI-NET ANTILLAS (project ID #16KIS1318).

References

[1] Özlem Atalık, Mahmut Bakır, and Şahap Akan. The role of in-flight service quality on value for money in business class: a logit model on the airline industry. *Administrative Sciences*, 9(1):26, 2019.

[2] GoGo, The Inflight Internet Company. https://gogoair.com/, 2020.

[3] European Aviation Network (EAN). http://www.europeanaviasionnetwork.com/, 2020.

[4] Zhicheng Qu, Gengxin Zhang, Haotong Cao, and Jidong Xie. Leo satellite constellation for internet of things. *IEEE access*, 5:18391–18401, 2017.

[5] Ergin Dinc, Michal Vondra, Sandra Hofmann, Dominic Schupke, Mikael Prytz, Sergio Bovelli, Magnus Frodigh, Jens Zander, and Cicek Cavdar. In-flight broadband connectivity: Architectures and business models for high capacity air-to-ground communications. *IEEE Communications Magazine*, 55(9):142–149, 2017.

[6] Michal Vondra, Ergin Dinc, Mikael Prytz, Magnus Frodigh, Dominic Schupke, Mats Nilson, Sandra Hofmann, and Cicek Cavdar. Performance study on seamless da2gc for aircraft passengers toward 5g. *IEEE Communications Magazine*, 55(11):194–201, 2017.

[7] ETSI TS 123.203 V13.6.0. https://goo.gl/KvekZM, 2016.

[8] Yu Ma, Weifa Liang, and Song Guo. Mobility-aware delay-sensitive service provisioning for mobile edge computing. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 270–276. IEEE, 2019.

[9] Bin Gao, Zhi Zhou, Fangming Liu, and Fei Xu. Winning at the starting line: Joint network selection and service placement for mobile edge computing. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1459–1467. IEEE, 2019.

[10] Tarik Taleb, Adlen Ksentini, and Pantelis A Frangoudis. Follow-me cloud: When cloud services follow mobile users. *IEEE Transactions on Cloud Computing*, 7(2):369–382, 2016.

[11] Abdelkader Aissioui, Adlen Ksentini, Abdelhak Mourad Gueroui, and Tarik Taleb. On enabling 5g automotive systems using follow me edge-cloud concept. *IEEE Transactions on Vehicular Technology*, 67(6):5302–5316, 2018.

[12] Hong Yao, Changmin Bai, Deze Zeng, Qingzhong Liang, and Yuanyuan Fan. Migrate or not? exploring virtual machine migration in roadside cloudlet-based vehicular cloud. *Concurrency and Computation: Practice and Experience*, 27(18):5780–5792, 2015.

[13] Amir Varasteh, Sandra Hofmann, Nemanja Deric, Mu He, Dominic Schupke, Wolfgang Kellerer, and Carmen Mas Machuca. Mobility-aware joint service placement and routing in space-air-ground integrated networks. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[14] Sandra Hofmann, Adrian Exposito Garcia, Dominic Schupke, Hector Esteban Gonzalez, and Frank HP Fitzek. Connectivity in the air: Throughput analysis of air-to-ground systems. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[15] Markus Chimani, Niklas Troost, and Tilo Wiedera. Approximating multistage matching problems. pages 558–570, 2021.

[16] Monika Henzinger, Shahbaz Khan, Richard Paul, and Christian Schulz. Dynamic matching algorithms in practice. *arXiv preprint arXiv:2004.09099*, 2020.

[17] Ergin Dinc, Michal Vondra, and Cicek Cavdar. Multi-user beamforming and ground station deployment for 5g direct air-to-ground communication. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE, 2017.

[18] Dirk G Cattrysse and Luk N Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European journal of operational research*, 60(3):260–272, 1992.

[19] P. Hall. On representatives of subsets. In *Classic Papers in Combinatorics*, pages 58–62. Springer, 2009.

[20] Amaury Van Bemten, Nemanja Deric, Amir Varasteh, Andreas Blenk, Stefan Schmid, and Wolfgang Kellerer. Empirical predictability study of sdn switches. In *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 1–13. IEEE, 2019.

[21] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[22] Sophie De Maesschalck, Didier Colle, Ilse Lievens, Mario Pickavet, Piet Demeester, Christian Mauz, Monika Jaeger, Robert Inkret, Branko Mikac, and Jan Derkacz. Pan-european optical transport networks: An availability-based comparison. *Photonic Network Communications*, 5(3):203–225, 2003.

[23] Datacenter Map. https://www.datacentermap.com/cloud.html, 2020.

[24] Fast Internet for aircraft in Europe. https://goo.gl/KUFGGL, 2018. [Online; accessed 06-May-2020].

[25] Amir Varasteh, Wolfgang Kellerer, and Carmen Mas Machuca. Network in the air. In *Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies*, pages 20–22, 2019.

[26] Arled Papa, Tomaso De Cola, Petra Vizarreta, Mu He, Carmen Mas Machuca, and Wolfgang Kellerer. Dynamic sdn controller placement in a leo constellation satellite network. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 206–212. IEEE, 2018.

[27] Margaret M McMahon and Robert Rathburn. Measuring latency in iridium satellite constellation data services. Technical report, NAVAL ACADEMY ANNAPOLIS MD DEPT OF COMPUTER SCIENCE, 2005.

[28] The Dream of Affordable Internet Access for Everyone is Getting Closer. https://goo.gl/eTkRnL, 2017. [Online; accessed 06-May-2020].

[29] Gurobi Optimization Solver. https://www.gurobi.com/, 2020.

[30] Amaury Van Bemten, Nemanja Deric, Amir Varasteh, Stefan Schmid, Carmen Mas-Machuca, Andreas Blenk, and Wolfgang Kellerer. Chameleon: predictable latency and high utilization with queue-aware and adaptive source routing. In *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pages 451–465, 2020.

[31] Barry Evans, Markus Werner, Erich Lutz, Michel Bousquet, Giovanni Emanuele Corazza, Gerard Maral, and Robert Rumeau. Integration of satellite and terrestrial systems in future multimedia communications. *IEEE Wireless Communications*, 12(5):72–80, 2005.

[32] Jiajia Liu, Yongpeng Shi, Zubair Md Fadlullah, and Nei Kato. Space-air-ground integrated network: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2714–2741, 2018.

[33] Yurui Cao, Hongzhi Guo, Jiajia Liu, and Nei Kato. Optimal satellite gateway placement in space-ground integrated networks. *IEEE Network*, 32(5):32–37, 2018.

[34] David Tomić, Sandra Hofmann, Mustafa Ozger, Dominic Schupke, and Cicek Cavdar. Quality of service aware traffic management for aircraft communications. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–6. IEEE, 2020.

[35] Amir Varasteh, Henrique Soares Frutuoso, Mu He, Wolfgang Kellerer, and Carmen Mas-Machuca. Figo: Mobility-aware in-flight service assignment and reconfiguration with deep q-learning. In *IEEE Global Communications Conference (Globecom)*, 2020.

[36] Ning Zhang, Shan Zhang, Peng Yang, Omar Alhussein, Weihua Zhuang, and Xuemin Sherman Shen. Software defined space-air-ground integrated vehicular networks: Challenges and solutions. *IEEE Communications Magazine*, 55(7):101–109, 2017.

[37] Yurui Cao, Yongpeng Shi, Jiajia Liu, and Nei Kato. Optimal satellite gateway placement in space-ground integrated network for latency minimization with reliability guarantee. *IEEE Wireless Communications Letters*, 7(2):174–177, 2017.

[38] Yongpeng Shi and Jiajia Liu. Inter-segment gateway selection for transmission energy optimization in space-air-ground converged network. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[39] Mathieu Bouet and Vania Conan. Mobile edge computing resources optimization: A geo-clustering approach. *IEEE Transactions on Network and Service Management*, 15(2):787–796, 2018.

[40] Shahrukh Khan Kasi, Mumraiz K Kasi, Kamran Ali, Mohsin Raza, Hifza Afzal, Aboubaker Lasebae, Bushra Naeem, Saif ul Islam, and Joel JPC Rodrigues. Heuristic edge server placement in industrial internet of things and cellular networks. *IEEE Internet of Things Journal*, 2020.

[41] Nouha Kherraf, Hyame Assem Alameddine, Sanaa Sharafeddine, Chadi M Assi, and Ali Ghrayeb. Optimized provisioning of edge computing resources with heterogeneous workload in iot networks. *IEEE Transactions on Network and Service Management*, 16(2):459–474, 2019.

[42] Ying-Dar Lin, Yuan-Cheng Lai, Jian-Xun Huang, and Hsu-Tung Chien. Three-tier capacity and traffic allocation for core, edges, and devices for mobile edge computing. *IEEE Transactions on Network and Service Management*, 15(3):923–933, 2018.

[43] Jan Plachy, Zdenek Becvar, Emilio Calvanese Strinati, and Nicola di Pietro. Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users. *IEEE Transactions on Network and Service Management*, 2021.

[44] Muhammad Waqas, Yong Niu, Manzoor Ahmed, Yong Li, Depeng Jin, and Zhu Han. Mobility-aware fog computing in dynamic environments: Understandings and implementation. *IEEE Access*, 7:38867–38879, 2018.

[45] Jia Xu, Xuejun Li, Xiao Liu, Chong Zhang, Lingmin Fan, Lina Gong, and Juan Li. Mobility-aware workflow offloading and scheduling strategy for mobile edge computing. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 184–199. Springer, 2019.

[46] Xuhui Zhao, Yan Shi, and Shanzhi Chen. Maesp: Mobility aware edge service placement in mobile edge networks. *Computer Networks*, 182:107435, 2020.

[47] Shiqiang Wang, Rahul Urgaonkar, Ting He, Kevin Chan, Murtaza Zafer, and Kin K Leung. Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1002–1016, 2016.

[48] Lin Wang, Lei Jiao, Jun Li, and Max Mühlhäuser. Online resource allocation for arbitrary user mobility in distributed edge clouds. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1281–1290. IEEE, 2017.

[49] Ivan Farris, Tarik Taleb, Miloud Bagaa, and Hannu Flick. Optimizing service replication for mobile delay-sensitive applications in 5g edge network. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[50] Vajiheh Farhadi, Fidan Mehmeti, Ting He, Thomas F La Porta, Hana Khamfroush, Shiqiang Wang, Kevin S Chan, and Konstantinos Poularakis. Service placement and request scheduling for data-intensive applications in edge clouds. *IEEE/ACM Transactions on Networking*, 29(2):779–792, 2021.

[51] Tarik Taleb, Miloud Bagaa, and Adlen Ksentini. User mobility-aware virtual network function placement for virtual 5g network infrastructure. In *2015 IEEE International Conference on Communications (ICC)*, pages 3879–3884. IEEE, 2015.

[52] Tayebeh Bahreini and Daniel Grosu. Efficient placement of multi-component applications in edge computing systems. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–11, 2017.

[53] Jie Zhang, Jiawei Lu, Xuan Yan, Xiaolong Xu, Lianyong Qi, and Wanchun Dou. Quantified edge server placement with quantum encoding in internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[54] Giuseppe Avino, Paolo Bande, Pantelis A Frangoudis, Christian Vitale, Claudio Casetti, Carla Fabiana Chiasserini, Kalkidan Gebru, Adlen Ksentini, and Giuliana Zennaro. A mec-based extended virtual sensing for automotive services. *IEEE Transactions on Network and Service Management*, 16(4):1450–1463, 2019.

[55] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9):842, 1957.

[56] R. Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24(3):153–157, 1987.

[57] Atila Abdulkadiroglu and Tayfun Sönmez. Matching markets: Theory and practice. *Advances in Economics and Econometrics*, 1:3–47, 2013.

[58] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.

[59] Nikhil R Devanur and Kamal Jain. Online matching with concave returns. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 137–144, 2012.

[60] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *Acm Sigact News*, 39(1):80–87, 2008.

[61] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.

[62] A. Mehta. Online matching and ad allocation. 2013.

[63] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.

[64] Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 384–393. IEEE, 2014.

[65] Bala Kalyanasundaram and Kirk R Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325, 2000.

[66] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

[67] Julien Baste, Binh-Minh Bui-Xuan, and Antoine Roux. Temporal matching. *Theoretical Computer Science*, 806:184–196, 2020.

[68] Yongho Shin, Kangsan Kim, Seungmin Lee, and Hyung-Chan An. Online graph matching problems with a worst-case reassignment budget. *arXiv preprint arXiv:2003.05175*, 2020.

[69] Marthe Bonamy, Nicolas Bousquet, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Arnaud Mary, Moritz Mühlenthaler, and Kunihiro Wasa. The perfect matching reconfiguration problem. *arXiv preprint arXiv:1904.06184*, 2019.