# Figo: Mobility-Aware In-Flight Service Assignment and Reconfiguration with Deep Q-Learning

Amir Varasteh, Henrique Soares Frutuoso, Mu He, Wolfgang Kellerer, and Carmen Mas-Machuca
Chair of Communication Networks, Department of Electrical and Computer Engineering,
Technical University of Munich, Germany
Email:{amir.varasteh, henrique.frutuoso, mu.he, wolfgang.kellerer, cmas}@tum.de

***Abstract*—Today, on-board passengers desire to have in-flight services such as Voice-over-IP (VoIP) and video streaming. These services are usually hosted by geographically distributed Data Centers (DCs) that are built/rented by the airline companies. Flights can be connected to these DCs using two types of Air-to-Ground (A2G) communication alternatives: *i)* satellite (SC), and *ii)* Direct-Air-to-Ground connections (DA2G). These two options are different in terms of propagation delay, capacity, and availability. Focusing on reducing the delay of the in-flight services, each airplane should be assigned to a nearby DC. However, due to the mobility of flights, a permanent DC assignment might not lead to an acceptable service delay for the flight duration. Therefore, the flight needs to be reassigned to another DC (reconfiguration) along its route, which comes with a cost. The real challenge in this work is to find the best assignments of each airplane to DC(s) and determine the required reconfigurations such that the sum of routing and reconfiguration delay is minimized.**

**We model this problem as a Multi-Period Generalized Assignment Problem (MPGAP) and formulate it as an Integer Linear Programming (ILP) optimization model. To overcome the scalability issues of the ILP, we propose *Figo*, a flight control framework that solves the MPGAP problem using deep Q-learning. Considering a realistic European-based Space-Air-Ground-Integrated Network (SAGIN) and a real set of flights, we compare the performance of *Figo* against the optimal. The results indicate that *Figo* can achieve 7% optimality gap in the worst case, while reducing the runtime from hours to seconds.**

***Index Terms*—Assignment, Reconfiguration, Service Migration, Mobility-Aware, Flight, Deep Q-Learning**

## I. Introduction and Background

The era of long flights with a slow or no Internet connection is over. Today, with a significant increase of worldwide flight traffic volume, providing Internet-based in-flight services (such as Voice-over-IP (VoIP), video conferencing, and gaming) to on-board passengers, has become an important goal for airline companies. Due to the cost, connectivity, and weight limitations of planes, these services cannot be stored on the plane itself. Thus, airline companies usually rent/build Data Centers (DCs) on the ground to provide resources for hosting their in-flight service instances. To ensure these services are reachable during all the flight period with acceptable Quality-of-Service (QoS) and also to improve the reliability, these DCs are usually distributed geographically. Therefore, the problem is to decide which flight should be assigned to which DC at a given time with minimum delay. In any case, to be able to provide the services to the passengers, flights must be able to communicate to their assigned DCs using an Air-to-Ground (A2G) technology. The traditional A2G method is satellite communication (SC) where the airplane traffic (services) is relayed by the satellite towards its gateway located on the ground network, where the DCs are located (See Fig. 1). The advantage of SC is that it is available globally to the flights. However, this A2G type has some limitations, especially in terms of latency and capacity of the link.

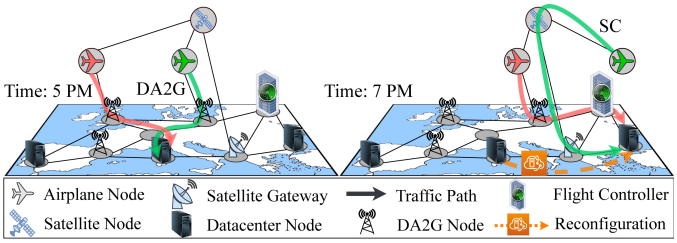



Fig. 1: An example of a Space-Air-Ground-Integrated Network (SAGIN) at two different timeslots: At 5 PM (left side), two flights are assigned to the same middle DC to receive their services, both using a DA2G connection. After 2 hours, at 7 PM (right side) based on their movement, they are reassigned to the right DC, incurring two reconfiguration operations. This example shows how the green plane at 5 PM connects to the middle DC through the DA2G, whereas, at 7 PM, the airplane uses the SC to connect to the right DC.

Recently, broadband Direct-Air-to-Ground (DA2G) has brought certain advances in the A2G communication [1]. With this method, the flights are directly connected to the base stations (BSs) located on the ground network (See Fig. 1). As DA2G BSs are distributed through the continents, the DA2G communication can be offered over the mainland, but not over the oceans. Nevertheless, the main advantage is the easy and cheap deployment of DA2G BSs with a high capacity and low link delay, compared to SC [1]. Thus, considering the different characteristics of A2G connections, routing the traffic from a flight to the selected DC becomes another important challenge.

These advances in A2G communications have brought the opportunity of providing online and interactive services to the passengers, such as air-to-ground voice/video calls, gaming, etc. These kinds of services demand a low-delay connection between the client (passengers) and servers (DCs on the

ground) [2]. Accordingly, to be able to provide these services with an acceptable delay to the passengers, it is important to assign each flight to the closest DC (i.e., with the lowest routing delay). However, due to the mobility of airplanes overtime, the routing delay from the airplane to DC can increase, making a single DC inefficient for the whole flight duration. Therefore, to maintain the service delay, we have the option of reconfiguration, i.e., changing the DC assignments during the flight[1] (See Fig. 1). However, reconfiguration comes with a penalty, for example, in terms of delay in migrating the flight's session data. Hence, it is crucial to control the unnecessary reconfigurations.

Therefore, in this paper we tackle three challenges: *i)* airplane-to-DC assignment, *ii)* airplane-to-DC routing, and *iii)* reconfiguration decisions for the flight period. We first model the problem as a Multi-Period Generalized Assignment Problem (MPGAP) using Integer Line Programming (ILP) to minimize the total routing and reconfiguration costs (in terms of delay). Due to the high complexity of the MPGAP [3], finding an optimal solution is not straightforward for practical-size problem instances, e.g., several hours for 100 flights. To overcome these limitations, we propose *Figo*, a flight control framework based on deep Q-learning to solve the presented problem. *Figo* can be deployed on the central flight controller, owned/leased by the airline company (See Fig. 1). It can react in real-time to network changes like new flights and emergency flight route changes. In case of assigning an airplane to a DC, the flight controller node sends a command to the in-flight router (e.g., OpenFlow message) using an A2G connection to configure the routing for forwarding the traffic (using A2G options) towards the selected DC. Also, to reassign an airplane to another DC, the controller triggers the session and data migration/synchronization between the source and destination DCs.

**Outline:** The rest of this paper is organized as follows. Section II and III presents the related work and system model, respectively. In Section IV, the ILP optimization model is presented, followed by the *Figo* framework design in Section V. We evaluate the performance of *Figo* compared to the optimal solution in Section VI. Finally, Section VII concludes the paper.

## II. Related Work

As an emerging network architecture, Space-Air-Ground Integrated Network (SAGIN) [4] has been attracting research in various areas like resource allocation [5], [6], mobility management [7], and energy-efficiency [8]. Considering the in-flight services, there are not many works available in the literature. In our previous work [6], [9], considering the scenario of this paper, we have presented a mathematical formulation for joint placement, routing, and migration of Virtual Machines (VMs) hosting flight services and proposed two heuristics [10] to solve it. From the modeling point of view, some works in the literature have worked on the mobility-aware resource management schemes. For instance, in the area of Internet of Vehicles (IoV), the authors in [11] have presented a solution for placement and migration of vehicle services while guaranteeing the delay in a software-defined-network urban environment. They presented an algorithm that continuously monitors the delay level of the connection. Neglecting the cost of reconfiguration, they migrate the service if a certain minimal delay threshold is reached. Thus, it only reacts to pre-defined thresholds, thus providing an unstable QoS.

Moreover, in Mobile Edge Computing (MEC), authors in [12], [13] have proposed an approach to dynamically place network functions on MEC servers for mobile users, according to the handover probability of users for the next time-slot. They formulate two optimization models with the objective of minimizing network function migrations and communication costs (i.e., QoS/QoE) between users and network functions. Also, the paper [14] takes into account the high mobility and uncertainty of MEC users and proposes a task migration algorithm based on reinforcement learning that determines the migration policy, while predicting the users mobility patterns.

However, [15] sheds light on the SAGIN architecture which is more challenging compared to the problems other areas. In particular, there is a high complexity coupled with high dynamicity in the SAGIN. For example, different flights with various service requests can dynamically join and leave the network. Also, the calculation of an efficient routing from the airplanes to the DCs is challenging, since using different A2G connections can lead to very different solution qualities in the long-term. Finally, we consider the cost of reconfiguration, and try to keep the QoS at an acceptable level during the whole problem horizon. To the best of our knowledge, there is no work in the state-of-the-art that solves the mobility-aware in-flight service assignment and reconfiguration in a realistic SAGIN based on reinforcement learning.

## III. System Model and Problem Statement

### A. System Model

Let us define the finite time horizon $T$ and divide it into $T/\omega$ equal timeslots, where $\omega$ is the duration of each timeslot. We define a graph $G = (N, L)$, where $N$ and $L$ are the sets of nodes and links of the graph in the timeslot $t \in T$, respectively. In the modeled SAGIN, a graph $G$ consists of four parts:

*1) Ground Network*: We define $N_G \subset N$ and $L_G \subset L$ as the set of nodes and links of the ground network, respectively. Besides, we consider a set of DCs at the ground nodes $J \subset N_G$. These DCs host the services of the airline company.

*2) DA2G Nodes/Links*: As mentioned before, airplanes can communicate to the ground network through DA2G connections (via DA2G BSs). We denote the set of DA2G BSs in the ground network as $N_{DA2G} \subset N$. These DA2G BSs are connected to the closest network node $N_G$ with a ground link $L_G$. Each DA2G BS has a communication range $\Psi_{DA2G}$ km (i.e., the communication to a plane is possible if the distance between them is $\leq \Psi_{DA2G}$). We denote $L_{DA2G} \subset L$ as the set of links that connects airplanes to the closest BS.

[1] In this paper, reconfiguring/reconfiguration is used interchangeably with reassigning/reassignment.

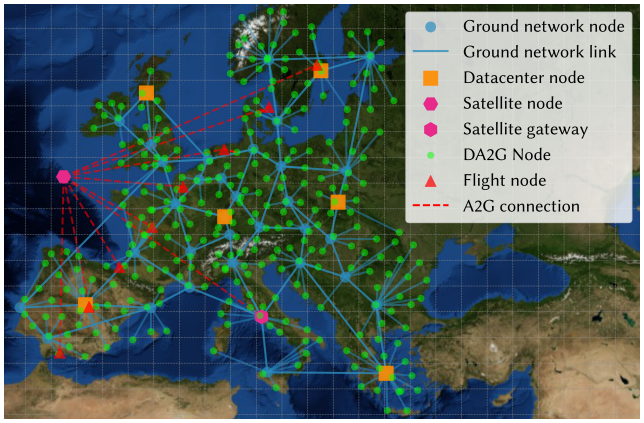



Fig. 2: An exemplary SAGIN (graph $G$) with one flight from Spain to Sweden in $|T| = 8$ timeslots.

*3) Satellite Nodes/Links:* The set of satellite nodes and links are denoted by $N_{SAT} \subset N$ and $L_{SAT} \subset L$, respectively. $N_{SAT}$ relays the airplane traffic towards the satellite gateways located in the ground network, defined as $N_{SG} \subset N_G$.

*4) Flight Nodes/Links:* We define $F = \{f_1, f_2, ..., f_{|F|}\}$ as the set of flights over $|T|$ equal timeslots (i.e., the number of flights during $T$ does not change). Each flight has a location for each timeslot (considered to be the location in terms of latitude and longitude at the starting time that timeslot). Each location corresponds to a different network node. Therefore, we define the set of airplane nodes denoted by $N_F \subset N$ where $|N_F| = |F| \times |T|$. For example, in Fig. 2, there is only one flight $F = \{f_1\}$ from Spain to Sweden, and $|N_F| = 8$ timeslots.

We define the set of requested services at each timeslot $t \in T$ as $A_t$. We aggregate the services of a plane and represent it as one service. These services are generated from the airplane nodes at a specific timeslot $t$. For example, in Fig. 2, at $t = 1$ the first airplane node generates the service request set $A^1$, and at $t = 2$ the second airplane node generate $A^2$, and so on. At each timeslot, each service request $a \in A^t$ should be assigned to a DC to receive its services. This assignment can be realized using either SC or DA2G connections (refer to the green plane in Fig. 1, which uses the DA2G connection at 5 PM and the satellite connection at 7 PM). These links have different characteristics in terms of latency, capacity, and availability which can be considered as the routing cost function.

Considering the mobility of airplanes reconfiguring an airplane connection from a DC to another one (as shown in Fig. 1 at 7 PM) can reduce the connection delay from an airplane to another DC to reduce the overall delay in long-term. This reconfiguration comes with a penalty, for instance, the delay of data migration and state synchronization between the source and destination DCs. The reconfiguration cost (i.e., delay) of reconfiguring a service request from DC $i$ to $j$ is denoted by $R_{i,j}$.

### B. Problem Statement

Given a set of finite timeslots and a SAGIN graph at each timeslot, the problem is to find: *i) Assignment:* Which airplane should be assigned to which DC? *ii) Routing:* How to reach from the airplane to the assigned DC? *iii) Reconfiguration:* Which airplanes should be reassigned to which DC?, and *iv) Delay consideration:* How to jointly answer the above questions by minimizing the total routing and reconfiguration delay over the whole time horizon? In this problem, we assume that the DCs and network links are uncapacitated. Further, the decisions are made per timeslot and assumed to be valid for the entire timeslot.

## IV. Offline Optimization Formulation

In this subsection, we formulate the above problem as an Integer Linear Programming (ILP) optimization model. To deliver low-delay services to flights, there are two delays that should be considered: First, we define $\mathcal{D}^T_{routing}$ as the sum of the propagation delay of each link of the path from all the airplane nodes to the assigned DC at each timeslot:

$$\mathcal{D}^T_{routing} = \sum_{t\in T}\sum_{a\in A^t}\sum_{j\in J}\sum_{(u,v)\in L} l^{u,v,t}_{a,j} D_{u,v} \quad (1)$$

where $l^{u,v,t}_{a,j} \in \{0,1\}$ is a decision variable that is equal to 1 if the service request from node $a \in A^t$ is routed towards DC $j \in J$ using the link $(u,v) \in L$ at timeslot $t \in T$. Also, $D_{u,v}$ denotes the propagation delay of link $(u,v) \in L$. Second, $\mathcal{D}^T_{reconf}$ is defined as the total reconfiguration penalties for all the flights for the whole period $T$:

$$\mathcal{D}^T_{reconf} = \sum_{t\in T}\sum_{a\in A^t}\sum_{(i,j)\in J\times J, i\neq j} r^t_{a,i,j} R_{i,j} \quad (2)$$

where $r^t_{a,i,j} \in \{0,1\}$ is a decision variable that is equal to 1 if the service request from airplane node $a \in A^t$ is reassigned from DC $i$ to $j$ in timeslot $t$. Therefore, the ILP optimization model can be formulated as follows:

$$\text{min.} \quad \left(\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}\right), \quad (3)$$

$$\text{s.t.} \quad \sum_{j\in J} x^t_{aj} = 1, \forall t \in T, \forall a \in A^t, \quad (4)$$

$$\sum_{v\in\delta^+(a)}\sum_{j\in J} l^{a,v,t}_{a,j} = 1, \forall t \in T, \forall a \in A^t, \quad (5)$$

$$\sum_{v\in\delta^+(u)}\sum_{j\in J} l^{u,v,t}_{a,j} - \sum_{v\in\delta^-(u)}\sum_{j\in J} l^{u,v,t}_{a,j} = \begin{cases} 0, \forall t \in T, \forall a \in A^t, \forall u \in N\backslash J, u \neq a \\ x^t_{a,u}, \forall t \in T, \forall a \in A^t, \forall u \in J \end{cases}, \quad (6)$$

$$l^{u,v,t}_{a,j} \leq x^t_{a,j}, \forall (u,v) \in L, \forall t \in T, \forall a \in A^t, \forall j \in J, \quad (7)$$

$$x^t_{a,j} = \sum_{i\in J} r^t_{a,j,i}, \forall t \in T^+, \forall a \in A^t, \forall j \in J, \quad (8)$$

$$x^t_{a,j} = \sum_{i\in J} r^{t+1}_{a,i,j}, \forall t \in T\backslash\{t_{|T|}\}, \forall a \in A^t, \forall j \in J, \quad (9)$$

$$\textit{vars:}\ x^t_{a,j} \in \{0,1\}, \forall t \in T, \forall a \in A^t, \forall j \in J,$$
$$r^t_{a,i,j} \in \{0,1\}, \forall t \in T, \forall a \in A^t, \forall i,j \in J \times J,$$
$$l^{u,v,t}_{a,j} \in \{0,1\}, \forall t \in T, \forall a \in A^t, \forall j \in J, \forall (u,v) \in L.$$

The objective function aims at finding a trade-off between the routing and reconfiguration delays. Constraints (4) force all the airplanes to be assigned to one and only one DC at each timeslot, by using the decision variable $x^t_{a,j} \in \{0,1\}$ which is equal to 1, if the airplane node $a \in A^t$ is assigned to DC $j \in J$ at timeslot $t \in T$. Constraints (5) generate the service request from the flights at all the timeslots. These service

requests must be going out from the airplane node towards a neighbor node in the graph which is either the DA2G or the satellite node. We note that $\delta^+(i)/\delta^-(i)$ is a function that returns the outgoing/incoming links from/to node $i \in N$. The flow conservation constraints (6) route the traffic flow through the network towards the assigned DC; the traffic should keep flowing until a DC node that serves the request. In addition to flow conservation, the service request traffic of each airplane node $a \in A^t$ must be forwarded towards the selected DC. This is ensured by constraints (7). The final set of constraints belong to the reconfigurations that have to be made. Constraints (8) and (9) link $x$ and $r$ variables in a flow conservation way. When $x^t_{a,j} = 0$, the airplane node $a$ is not assigned to DC $j$ at timeslot $t$, meaning no reconfiguration. $x^t_{a,j} = 1$ imposes that a reconfiguration assigns airplane node $a$ 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: $R_{i,j} = 0, \forall i, j \in (J \times J), i = j$. We note that no reconfiguration is possible at the first timeslot.

This problem is considered to be NP-Hard since for $|T| = 1$, it can be reduced from the Generalized Assignment Problem (GAP), which is proven to be NP-hard [3]. This makes the ILP not applicable to practical-size problem instances with a large number of flights and timeslots. In the next section, we propose an efficient algorithm to overcome this issue.

## V. Figo Framework Design

In this section, we propose *Figo*, a framework to solve the problem presented in Section III-B in an online manner. In particular, *Figo* takes a flight with its path and then pre-plan the airplane-to-DC assignments and the necessary reconfigurations in real-time. These decisions can be sent to the DCs and airplanes to apply the necessary configurations at the right time, e.g., preparing the airplane session migration from a DC to another one. *Figo* can be deployed on the central flight controller (owned by the airline company) to do the delay-aware service provisioning for the flights of the company. *Figo* uses Q-learning to can learn the efficient policy, aiming at solving the aforementioned problem. Reinforcement learning methods can observe and learn the dynamics of complex environments, which makes it suitable to tackle our decision-making problem [16]. The agent in reinforcement learning algorithms collects the system state for each episode and calculates the reward during the last timeslot. Then, it selects an action according to a predefined strategy, and the system transfers to a new state in the next timeslot. Similarly, the agent calculates the reward and chooses new action. We define the state space, action set, and the reward function as follows.

*1) State-Space $\mathbb{S}$:* We define the state $S(t)$ for flight $f$ in form of three elements:

$$\mathbb{S} = \{S(t) = (p^t_f, p^{|T|}_f, x^t_{f,j}) \mid \forall f \in A^t, \forall j \in J\}, \quad (10)$$

where $p^t_f$ and $p^{|T|}_f$ corresponds to the current and final destination of the flight $f$, respectively. This conveys both a sense of mobility and enables the agent to plan for a long-term solution. In order to diminish our state-space we divided our map (graph $G$) into areas on a $100 \times 100$ grid (as shown in Fig. 2), from which the positions $p^t_f$ are taken. This enables an improved exploration during the training phase and also an increase in performance for unknown flights. The last component of the state $x^t_{f,j}$ is the current DC that the airplane is assigned to.

*2) Actions $\mathbb{X}$:* The actions correspond to each DC that we assign an airplane to. An action is performed in case of a reconfiguration operation. Therefore, we define the action set $\mathbb{X}$ as follows:

$$\mathbb{X} = \{\chi_j \mid j \in J\}, \quad (11)$$

where $\chi_j$ indicates if the given airplane is assigned to DC $j$.

*3) Reward Function $\mathbb{R}$:* At each timeslot, the agent gets a reward $\mathbb{R}(S, \chi)$ in a certain state $S$ after executing each possible action from $\mathbb{X}$. We define the reward function as the following equation:

$$\mathbb{R}(S, \chi) = \bar{\mathcal{D}}(S, j) - \hat{\mathcal{D}}^t_{reconf} - \hat{\mathcal{D}}^t_{routing}, \forall j \in J, \forall t \in T, \quad (12)$$

where $\bar{\mathcal{D}}(S, j)$ is the delay of Dijkstra's shortest-path from the current flight position to the closest DC $j$. Also, $\hat{\mathcal{D}}^t_{reconf}$ and $\hat{\mathcal{D}}^t_{routing}$ are the reconfiguration/routing delays at the timeslot $t$, respectively. The comparison with the minimum delay for each DC allows all timeslots to have a common baseline. This prevents large reward variance at each timeslot, thus avoiding an inconsistent training, which would lead to slow convergence.

*4) Q-Learning:* The objective is to find a policy that maximizes the cumulative reward of each timeslot. Hence, we need to define a policy that iterates through the state-space and updates each reward accordingly. In order to maximize the long-term reward, we need to learn the optimal policy by determining the optimal reward values for each action, at each state. That is, given the agent takes the action $\chi$ at state $S$, we define Q-values as an estimate of the expected long-term reward. These values can be estimated using the Bellman equation:

$$Q_{t+1}(S(t), \chi(t)) = (1 - \alpha) Q_t(S(t), \chi(t)) + \alpha[\mathbb{R}(t) + \gamma \cdot \max_{\mathbb{X}} Q_{t+1}(S(t + 1), \chi(t + 1))], \quad (13)$$

where $\alpha$ is the learning rate, and $\gamma$ is the discount factor, which allows us to balance the concentration on short or long-term rewards. This procedure is called Q-learning, which is not practical to be used in the case of a large state-space [17]. In particular, in a large state-space, it is much more difficult for an agent to visit each state and store all these Q-values. To tackle this issue, an alternative is Deep Q-learning (DQL), in which a Deep Neural Network (DNN) is deployed. In particular, it is used to approximate the optimal action-value function, enabling it to accurately predict the Q-values for each state. As shown in [16], the use of DQL allows learning of *good* policies within a large and complex state-space use-case.

*5) Training: Figo* learns by using the continuously received flight data. Long-term and continuous training enables *Figo* to improve the returned solutions for unknown scenarios.

The DNN is updated during training by using a batch of random picket data from past observations [18]. We define a

| Parameter | Value |
|---|---|
| Learning rate $\alpha$ | $5 \times 10^{-4}$ |
| Discount factor $\gamma$ | 0.9 |
| Final exploration factor $\epsilon$ | 0.02 |
| Exploration fraction | 30% |
| Batch size $n$ | 32 |
| NN Layers (inc. input and output) | 4 |
| Training length | $10^5$ timeslots |

TABLE I: Training hyperparameters

| Scenario ID | Setting $(\|J\|, \|T\|, \Psi_{DA2G})$ | Scenario ID | Setting $(\|J\|, \|T\|, \Psi_{DA2G})$ |
|---|---|---|---|
| 1 | (3, 4, 70 km) | 2 | (3, 4, 150 km) |
| 3 | (3, 8, 70 km) | 4 | (3, 8, 150 km) |
| 5 | (6, 4, 70 km) | 6 | (6, 4, 150 km) |
| 7 | (6, 8, 70 km) | 8 | (6, 8, 150 km) |

TABLE II: The considered scenarios with $|F| = 50$ with different number of DCs $|J|$, timeslots $|T|$, and DA2G coverage $\Psi_{DA2G}$.

decreasing $\epsilon$ value for the training algorithm to trade-off the exploration against the exploitation [19]. The last parameter to be considered is the discount factor $\gamma$ which balances the importance of long and short-term rewards. When the value of $\gamma$ is close to one, it indicates a higher importance for long-term rewards. After a grid-search, we decide to use the training hyperparameters listed in Table I.

## VI. Performance Evaluation

In this section, we introduce our simulation scenario and parameters for evaluation. Thereafter, we compare the proposed *Figo* framework against the optimal solution (i.e., ILP model in Section IV).

***Simulation Setup:*** Our simulation settings are based on a realistic European-based SAGIN, similar to our previous work [6] (See Fig. 2). We use the Cost266 topology [20] for the ground network. The delay of links in $L_G$ is determined according to the link's length. We assume two DC sets: a reduced set $J$={Hamburg, Madrid, Budapest}, and a larger set $J$={Strasbourg, Stockholm, Madrid, Athens, Glasgow, Krakow}. The location of the DA2G BSs is taken from [21] ($|N_{DA2G}| = 295$) and its propagation delay is set to 10 ms [22]. Also, we consider two different DA2G connectivity ranges $\Psi_{DA2G}$ of 70 and 150 km, although in reality it can vary based on, e.g., the antenna type and weather conditions.

For the satellite A2G connection, we consider the LEO satellites. In contrast to the Geostationary Orbit (GEO) satellites, LEO has a lower delay, which is more suitable for our use case. According to [5], the number of LEO satellites over Europe space is rather low (5 LEO satellites in Iridium constellation). For simplification purposes, we abstract the group of LEO satellites covering Europe as a single satellite node, i.e., $|\bar{N}_{SAT}| = 1$. Further, to avoid unrealistic assumptions, we set the latency of satellite link to the worst-case achievable latency (i.e., 50 ms). This value is calculated according to the LEO delay calculations provided in [23]. Also, the satellite gateway node $N_{SG}$ is assumed to be located in Rome.

The set of flights has been exported from FlightRadar24 live air traffic for 24 hours on 9.11.2017. We choose a set of 50 random flights from the data, because of the ILP computation limitation; however, *Figo* can solve a larger set of flights. Let us consider a timeslot of $\omega = 30$ min and we consider flights with two different durations: *i) short* flights with the duration of 2 hours, i.e., $|T| = 4$, and *ii) long* 4-hour flights i.e., $|T| = 8$.

For the reconfiguration delay, we assume it to be independent of the source and destination DCs, thus

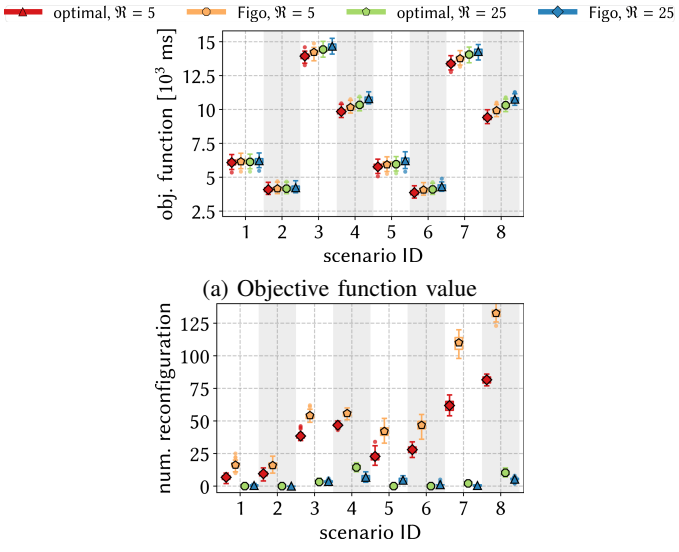
$$R_{i,j} = \begin{cases} \Re, \forall (i,j) \in |J| \times |J|, \text{ if } i \neq j, \\ 0, \forall (i,j) \in |J| \times |J|, \text{ if } i = j. \end{cases}$$

In our scenario, we consider two services with low and high reconfiguration delays, $\Re = 5$, and $\Re = 25$ ms, respectively.

The ILP model is implemented and solved with Gurobi [24] in Python. Moreover, *Figo* is implemented with Tensorflow in Python. The training and simulations are performed on a machine equipped with Intel Core i7-6560U CPU and 8 GB of RAM.

***Simulation Results:*** We present the simulation results in different categories and experiments. We note that each scenario is ran for 30 times randomly.

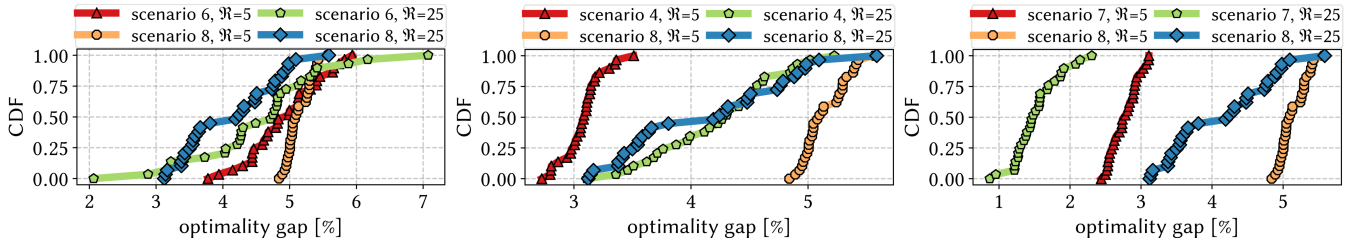
*1) Total Delay and Number of Reconfigurations:* We start by comparing the performance of *Figo* with the optimal solution in eight different scenarios summarized in Table II. As depicted in Fig. 3a a higher total delay can be observed for larger reconfiguration delays and an increase of reconfigurations for the smaller $\Re$ value. It can be seen that *Figo* can produce near-optimal results in all the scenarios. Also, the instances with




(a) Objective function value

(b) Number of reconfigurations

Fig. 3: Comparison of the objective function value and the number of reconfigurations for *Figo* and the optimal solution in the eight different scenarios of Table II.




(a) Impact of $|T|$ (b) Impact of $|J|$ (c) Impact of $\Psi_{DA2G}$

Fig. 4: The impact of $|T|$, $|J|$, and $\Psi_{DA2G}$ on the achieved optimality gap, showing that *Figo* achieves around 7% optimality gap in the worst-case.

higher $\Re$ have a higher objective function on average. The reason is that for small $\Re$, the algorithm has more flexibility in improving the objective function. This fact can be observed in Fig. 3b wherein all scenarios, the number of reconfigurations is lower in the case of $\Re = 25$.

An interesting observation in Fig. 3a is when we compare scenarios with different $|J|$ (i.e., scenarios (1,5), (2,6), (3,7), and (4,8)). In these scenarios, the objective function value is generally lower on average for $|J| = 6$ compared to $|J| = 3$. This can give the algorithm more reconfiguration possibility, as Fig. 3b indicates. In particular, in scenarios with $|J| = 6$, more reconfigurations are performed, especially when $\Re = 5$ due to even higher flexibility. That is why the improvement is generally lower for $\Re = 25$ compared to $\Re = 5$. Hence, it can be concluded that if the reconfiguration delay of the services is low/high, airline companies should go for a higher/lower number of DCs to achieve the same delay level for their in-flight services. In addition, Fig. 3a shows that the scenarios with higher $|T|$ (e.g., Scenarios 3, 7 compared to 1, 5) lead to obviously higher objective function. Also, higher values of $\Psi_{DA2G}$ (e.g., Scenarios 6, 8 compared to 5, 7) can lead to a lower objective function value, since higher $\Psi_{DA2G}$ makes the airplane nodes more reachable to the DC locations.

In the following, we investigate the impact of changes in $|T|$, $|J|$, $\Psi_{DA2G}$, and $|F|$ on the optimality gap of *Figo* with different reconfiguration $\Re$ penalties.

*2) Impact of $|T|$:* Let us choose some of the challenging scenarios for a closer analysis. Fig. 4a compares the effect of $|T|$ on the optimality gap by considering scenarios 6 and 8 where $|J|$ and $\Psi_{DA2G}$ are fixed. It shows that the optimality gap is the lowest for Scenario 8 ($|T| = 8$) and $\Re = 5$, and the same scenario with $\Re = 25$ has the highest gap. The reason is $\Re = 5$ brings more reconfiguration options than the $\Re = 25$ case. In this case, the reconfiguration delay is the deal-breaker in the optimality gap instead of the flight duration. This is an advantage of *Figo*, which performs with a close distance to optimal, even in more realistic scenarios where flights have a longer duration.

*3) Impact of $|J|$:* Fig. 4b compares the optimality gap for scenarios with fixed $|T|$ and $\Psi_{DA2G}$ and varying $|J|$ (i.e., Scenarios 4 and 8 with $|J| = 3$ and $|J| = 6$, respectively). We observe that Scenario 4 performs better than 8 when $\Re = 5$; however, for $\Re = 25$, they almost perform similarly. We know that Scenario 8 has a larger state-space, since it scales exponentially with the number of DCs, making it harder to achieve a close optimality gap. However, it can achieve a better optimality gap in both $\Re = 5$ and $\Re = 25$. We can conclude that the number of DCs has more impact on the performance than the reconfiguration delay.

*4) Impact of $\Psi_{DA2G}$:* In this part, we compare the performance of *Figo* against the optimal solution for different $\Psi_{DA2G}$ in Fig. 4c. The results indicate that the smaller DA2G range (i.e., Scenario 7) diminishes the number of DA2G connections. Considering $\Re = 25$, this fact leads to even less reconfiguration potentials since the ideal DC assignment would be located near the satellite gateway node.

*5) Impact of $|F|$:* A very important characteristic of *Figo* is its ability to solve the problem for a large number of flights with an acceptable optimality gap. To show this we ran scenario 8 for an additional 30 runs, taking 20, 50, and 100 flights into account. According to the results, it is evident that the number of flights does not influence the performance of *Figo*. In particular, Fig. 5 indicates that the optimality gap lies between 3% and 6% on average for different numbers of flights.

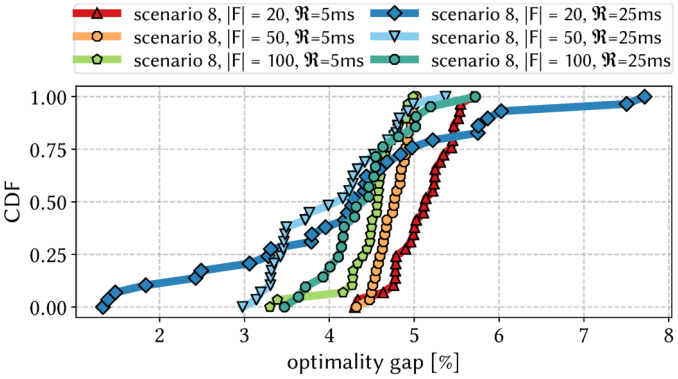



Fig. 5: Optimality gap for different number of flights.

*6) Training and Runtime:* Training duration is mainly influenced by two parameters, the sample size and convergence time. To reach an acceptable level of performance, the agent needs to go through a very large number of simulation runs. The sample size affects our algorithm in two ways. Firstly, a larger sample size means a larger graph to analyze. Since we

use Dijkstra to calculate the shortest path, a higher number of nodes leads to a longer calculation time for each Dijkstra run. Secondly, reinforcement learning needs a lot of different samples to learn a generalized behavior. This means that while we would be able to achieve convergence within seconds and a few hundred iterations with a single flight, we need to perform thousands of iterations to be able to teach the agent something meaningful. In our case, training phase takes around 90 minutes for $|F| = 100$, $|T| = 8$, and $|J| = 6$. After training, *Figo* is able to solve the problem for scenarios 7 and 8 with $|F| = 100$ in 26 seconds on average. Also, it takes around 18 seconds to solve scenarios 3 and 4 for the same number of flights (lower DCs leads to having lower runtime). Finally, we report that in an online manner, given a single flight, *Figo* takes around 0.5 seconds to find a solution on average for all the scenarios.

## VII. Conclusion

Aiming at minimizing the total delay, we determine the necessary airplane-to-DC assignment/reassignment operations along the flight duration. Since calculating the optimal solution is not scalable, we proposed *Figo*, a flight controller framework that efficiently solves the aforementioned problem by employing the Deep Q-Learning method. Moreover, considering the future airplane positions, it overlooks the short-term delay improvements for the long-term ones. Our simulations in a realistic scenario showed that *Figo* exhibits acceptable results with a maximum 7% optimality gap, while reducing the computation time from hours to seconds. We showed that *Figo* can solve the problem for a set of flights with different duration, while keeping an acceptable delay level compared to optimal. Moreover, it was shown that the services with lower reconfiguration costs can support more flexibility, enabling better delay improvements/adaptations with the mobility of the airplanes. *Figo* can be used by the airline companies to improve their in-flight service quality and employing an efficient and real-time resource management strategy. By using further data and network metrics collected by airlines during the flight operation, this work can be extended to consider parameters such as link capacity, dynamic service requests, and A2G availability patterns.

## Acknowledgment

The authors would like to thank the reviewers and Patrick Kalmbach for their fruitful feedback. This work is funded by Germany Federal Ministry of Education and Research under project OptiCON (grant IDs #16KIS0989K and #16KIS0991).

## References

[1] 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.

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

[3] Marshall L Fisher, Ramchandran Jaikumar, and Luk N Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management science*, 32(9):1095–1103, 1986.

[4] 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.

[5] 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.

[6] 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 *2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[7] Xiaoshen Xu, Baohua Kou, Ligang Fei, Dandan Fan, Lei Zhou, and Zhenhui Guo. Study on mobility technologies of space-ground integrated IP network toward GEO satellites. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 1832–1836. IEEE, 2016.

[8] 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.

[9] Amir Varasteh, Sandra Hofmann, Nemanja Deric, Andreas Blenk, Dominic Schupke, Wolfgang Kellerer, and Carmen Mas Machuca. Toward optimal mobility-aware vm placement and routing in space-air-ground integrated networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE, 2019.

[10] 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 (CONEXT)*, pages 20–22, 2019.

[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] 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.

[13] 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.

[14] Cheng Zhang and Zixuan Zheng. Task migration for mobile edge computing using deep reinforcement learning. *Future Generation Computer Systems*, 96:111–118, 2019.

[15] Nei Kato, Zubair Md Fadlullah, Fengxiao Tang, Bomin Mao, Shigenori Tani, Atsushi Okamura, and Jiajia Liu. Optimizing space-air-ground integrated networks by artificial intelligence. *IEEE Wireless Communications*, 26(4):140–147, 2019.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[17] Pierre Yves Glorennec and Lionel Jouffe. Fuzzy q-learning. In *Proceedings of 6th international fuzzy systems conference*, volume 2, pages 659–662. IEEE, 1997.

[18] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[19] Cesa-Bianchi N. Fischer P. Auer, P. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning 47, 235256*, 2002.

[20] S. De Maesschalck et al. Pan-european optical transport networks: An availability-based comparison. *Photonic Network Communications*, 5(3):203–225, 2003.

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

[22] Ergin Dinc, Michal Vondra, and Cicek Cavdar. Multi-user Beamforming and Ground Station Deployment for 5G Direct Air-to-Ground Communication. In *2017 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2017.

[23] M. McMahon et al. Measuring latency in iridium satellite constellation data services. Technical report, Naval academy Annapolis department of computer science, 2005.

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