UAV mission optimization in 5G: On reducing MEC service relocation

Samir Si-Mohammed[1,2], Adlen Ksentini[1], Maha Bouaziz[1], Yacine Challal[2], and Amar Balla[2]

[1]EURECOM, Sophia Antipolis, France
[2]Laboratoire des Méthodes de Conception de Systèmes, Ecole nationale Supérieure d'Informatique, Algiers, Algeria
[1]{name.surname}@eurecom.fr
[2]{fs_si_mohammed, y_challal, a_balla}@esi.dz

***Abstract*—Unmanned Aerial Vehicle (UAV) applications and services have gained a huge deployment and adoption in different fields, such as the military domain (Defense or reconnaissance) and the civilian domain (Healthcare, surveillance, and transport). UAV operations are generally critical and require, during operations, a control link with the drones, which should be reliable with very low latency. To ensure low-latency, 5G architecture intends to deploy Mobile Edge Computing (MEC) servers, which provide cloud computing capabilities close to the end-users. Consequently, it is envisioned that the AutoPilot application will be deployed at the MEC in order to ensure a low latency connection to the drones. However, the high mobility of drones makes the migration of the AutoPilot applications among MEC servers unavoidable; in order to maintain a low latency connection with the flying drones. This may lead to frequent downtime of the service, which may impact the AutoPilot performances, and hence service migrations should be limited as much as possible. Accordingly, this paper aims to reduce the number of service migrations of drones by introducing novel algorithms that act at the mission planning phase, where the path of the drones is defined.**

***Index Terms*—UAV, Flight planning, Drones, 5G, MEC.**

I. Introduction

Among the industries that have lately seen a huge and fast development, we may quote the Unmanned Aerial Vehicle (UAV) industry. With several applications, from surveillance to healthcare, drones are knowing great popularity among different users, from companies for Business purposes, to governments for Security aims. In all cases, the necessary conditions of the safe progress of a drone's flight is the reliability of the wireless communication held with the drones in order to control their trajectory and avoid collisions with other flying drones. The communication link from the ground, or from the Drone Pilot to the Drones is known in UAV terminology, as the control and command link, or C2 Link [1]. It is foreseen that C2 link is extremely important as, in addition to carrying the command of the Drone Pilot to control the drones' trajectory, it gathers information from drones, such as the position or the speed. This information allows the Drone Pilot to detect any risk of collision with an obstacle or other drones. Consequently, this link should be secure, and benefiting from very low latency aiming at ensuring that the control requests will be done in real-time.

Meanwhile, the recent rise of 5G technologies, and their ability to support highly reliable and low latency services, via the Ultra Reliable Low Latency (uRLLC) [2] service, represents an interesting opportunity for the expansion of UAV applications and services. Indeed, 5G technology is assumed to guarantee wider bandwidth and very low-latency connectivity, which definitely places it as a key enabler of UAV-based services and applications. 5G brings into light several new concepts that can be beneficial for UAV. 5G New Radio (NR) provides larger bandwidth (up to 100 MHz in $<$ 6 GHz frequency band, and up to 400 MHz in $>$ 6 GHz frequency band) [3] to accommodate high data-rate demanding applications, such as VR/AR, 4K video streaming, which may be used by UAVs for high-quality video streaming or remote steering of the UAVs. Moreover, 5G NR uses new physical layer numerology that drastically reduces Radio Access Network (RAN) latency, which when combined with Mobile Edge Computing (MEC) [4] capabilities at the vicinity of the radio network, will allow achieving a very low latency for the C2 link. Besides using 5G for communicating with drones, the Drone Pilot will be hosted at the MEC; hence guaranteeing very low-latency communication for the C2 link. The challenges that arise in a MEC-enabled 5G architecture for the efficient placement of the virtual network functions (VNFs) are discussed in [5].

The deployment of MEC [6] is assumed to be very distributed, i.e. several MEC servers will be deployed close to end-users. One MEC server will cover a set of base stations, hence covering a limited geographical area. However, as drones are highly mobile, they can go out of the coverage area of a MEC server, which may increase the latency; hence perturbing the C2 link (lead to a threat in the safety of the drone's flight). One solution usually employed to keep the benefit of the MEC, in terms of low latency connection, is to migrate the Drone Pilot application among MEC servers (known as service migration) [7], by following drones mobility. This will ensure that the Drone Pilot application is always hosted by the MEC server covering the area (set of base stations) where the Drones are located. Nevertheless, service migration has a negative effect. Indeed, during the migration, the service is down (called downtime) for a few seconds. This may affect negatively the C2 link, and thus service migrations need to be limited to a minimum (only when deemed appropriate). In this paper, we propose to reduce the service migration downtime by minimizing the number of Drone Pilot migrations among MEC servers. To achieve this objective, we propose an algorithm to be used offline and during the mission planning phase, where the Drone Operator prepares the plan of the flight in accordance with the 5G Network Operator. The proposed algorithm aims at selecting the flight path, from the start point to the landing point (drone's flight plan), by considering not only the shortest path but also reducing the number of service migrations.

This paper is organized as follows: Section II details the steps for UAV Flight Planning. Section III formalizes the problem and introduces the proposed algorithms. Our results are presented in Section IV. Finally, the paper is concluded in Section V.

II. Flight Unfolding

In general, the deployment of drones should follow several steps [8], divided into three blocks: (i) Scope Definition Block; (ii) Drone Block consisting of Flight Planning, Flight Implementation, and Data Acquisition; (iii) Software Block where Data Analysis, Data Interpretation and Optimization are conducted. In the first block, the mission statement is clearly established and precise objectives are defined; ex. for network traffic analysis and behavioural studies. The second block consists of preparing the flight, considering Safety & Environment conditions, and route planning. The third and final block proceeds during the flight, where all the operations (Analysis, Interpretation and Optimization) on the Data acquired by the Drones are executed.

Flight Planning : This task is extremely important, for security reasons, since the conditions of the flight are negotiated at this level. The main concerned stakeholders [9] are, as showed in Fig. 1 :

- Customer: The entity or user wanting to benefit from a Drone Service. It can be an individual (For entertainment), a company (Delivery purposes, etc.) or even a government. The application on the UAV can vary depending on the customer.
- Drone Operator: The entity responsible of controlling the Drones, and offering UAV-based services, and proposing Flight plans depending on the needs of the Customer. The Drone Operator is the entity which deploys the Drone Pilot as an application at the edge.
- Network Operator (NOP): The entity holding the 5G infrastructure and offering 5G coverage.
- UAV Traffic Management (UTM): A centralized entity responsible of the management of drone's flights, since it holds information about all the drones flying in the the areas; all information such as presence of drones, their trajectories, locations [9], are centralized at the UTM.

The preparation of a flight plan for the mission consists of proposing a path (or route) followed by the drone during a given interval of time in accordance with Network Operator

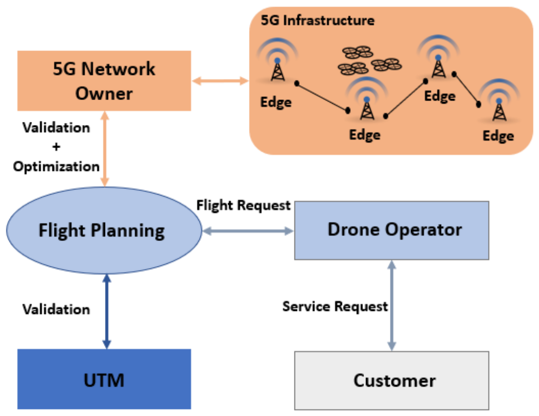


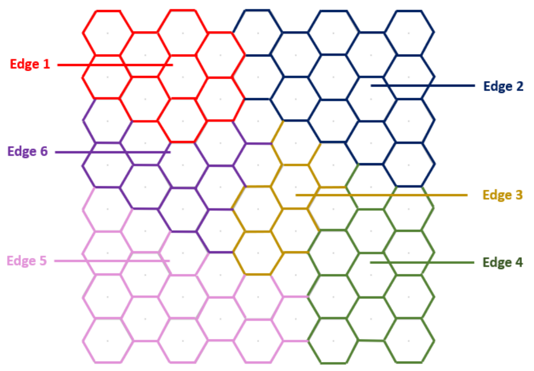
Figure 1: Flight Planning actors.

infrastructure information. The flight plan needs to be later validated by the UTM. The information that the NOP has to indicate to prepare the flight plan is the network coverage of the flying area, to check whether the infrastructure can offer the needed 5G network coverage and states. For instance, if at a given instant, the network infrastructure is overloaded with huge traffic, ensuring very low latency to the C2 link for UAV may be difficult. In this case the flight plan should be modified, and another flight time should be proposed. Obviously, we assume that the NOP is aware of the state of network in its infrastructure during the day long. Another information that the NOP should provide is the number of MEC servers and their mapping with 5G base stations. At this step, our proposed algorithm proceeds, by helping the Drone operator to find a path that reduces the number of service relocation, and avoid overloaded base stations. Regarding the UTM, the validation consists of checking if the area is safe in terms of environmental conditions (Weather, accidental zones...), and UAVs collisions; for example, if the proposed flight plan is in an area where other Drones are flying in the same altitude, then there will be a risk of conflicts and collisions between drones, which means that the flight plan will be rejected and another flight plan should be proposed. The Flight Preparation step ends with a validated and agreed on flight plan, which contains the list of cells followed by the drone, and the corresponding time.

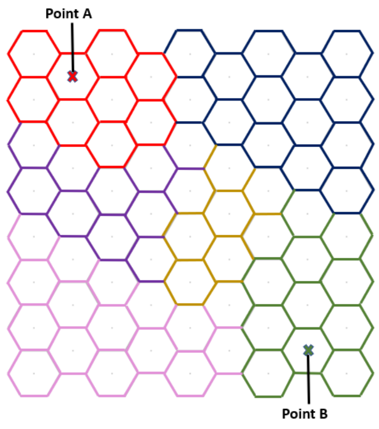
III. The Proposed Solution

A. System modeling

As stated earlier, our proposed algorithm intervenes at the route planning step of the mission preparation. As usually modeled, the mobile network is composed of a set of base stations, where each base station has an hexagonal coverage [10]. In Fig. 2a, we show an example of a mobile network topology, where each MEC Server (noted edge) covers an area composed of a group of cells. The Drone Pilot application can be deployed at the MEC server to ensure low latency. We used colors to show the relation between a MEC server



(a) Topology Edges



(b) Topology Points

Figure 2: Network Topology

and a group of cells it covers. Let us suppose that a mission consists of flying a drone from a point A to point B (see Fig. 2b), in different areas. We consider then that the Drone Pilot application is first instantiated in Edge1 server as it covers the initial position of the drone. Then, the Drone Pilot application is migrated among the servers according to the drone mobility.

The straight path between the two points consists in minimizing the distance traveled by the drone. But, in some cases, like the one depicted in Fig. 2b, the straight path between the two points will require not less than three service migrations, since the drone will pass through both Edge1, Edge6, Edge3 and finally Edge4; which requires to migrate the Drone Pilot application accordingly. This would impact the performances of the C2 link as the duration of downtime could be consequent. However, from Fig. 2b, we can see that another path is much more interesting, in terms of service migrations; if the drone goes from Edge1 to Edge2, then to Edge4, it will get to the final destination with only two service migrations, which will considerably reduce the downtime duration. Hence, there is a need of an algorithm that returns the best path between the two points, in terms of minimizing the number of service migrations.

B. Problem formulation

We propose to model the network topology of Fig. 2 by an oriented graph, where: the vertices represent the cells, the edges are weighted with either the distance between the two cells, or the cost of service migration in the case where the two cells are under the coverage of different edges. The objective is to find the optimal trajectory, i.e. a set of cells to cross through that reduces the service migrations from the starting to the landing point. We denote by E$_i$, the Edge server covering the area $\{C_{1i}, C_{2i}, ..., C_{ni}\}$, which consists of a set of cells, where C_{ji} represents the cell identified by j in the area covered by the Edge E_i. As indicated earlier, we model the topology as a non oriented graph (V, E) where V is the set of cells $\{C_{11}, C_{21}, ..., C_{n1}, ..., C_{1k}, C_{2k}, ..., C_{nk}\}$, k is the number of cells per Edge node, and n is the number of Edges.

We denote by $w_{(i,j)_(k,m)}$ as the weight between two neighboring, i.e. C_{ij} and C_{km}, which represents the cost of the service migration if the two cells are not under the same edge coverage, or the distance between them if they are in the same area. In our case, since the topology has a hexagonal form, all the distances are similar, and equal to 1 for simplicity. Such a graph is depicted in Fig. 3, where C is the fixed service migration cost between two edges.

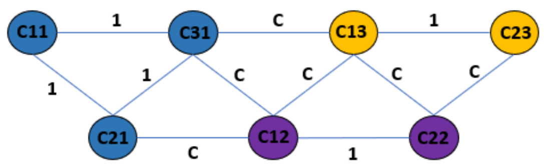
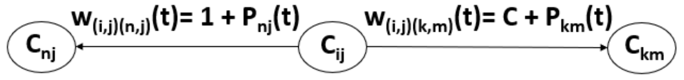


Figure 3: Graph Topology Weights.

As indicated earlier, another parameter that may impact the C2 link performances (i.e. latency) is the fact that a selected cell may be overloaded during the flight period, by other types of network traffic. Therefore, we add another parameter to the model, which is the cell overload probability (noted $P(t)$) that indicates the overload of a cell at an instant t. For example, at a given time, every cell of the topology will have a probability of being overloaded; indeed the probability of finding a cell busy at the rush time is different from other periods of the day. To introduce this parameter to our model, we include it in the weight of the edge between the cells. This way the selected path will consider the distance, the service migration cost, and the overloading of the destination cell. P(t) can be computed with the use of a forecasting model, trained using collected data on the mobile network traffic dynamic [11]. We derive the weight $w_{(i,j)_(k,m)}$ of an edge (Fig. 4) as follows:

$$w_{(i,j)_(k,m)}(t) = \begin{cases} 1 + P_{km}(t) & \text{if j = m} \\ C + P_{km}(t) & \text{else} \end{cases} \quad (1)$$

where C_{ij} is the source cell, C_{km} the destination cell, t the requested instant and P_{km}(t) the probability of the cell C_{km} being overloaded at the instant t.



$w_{(i,j)(n,j)}(t) = 1 + P_{nj}(t)$
$w_{(i,j)(k,m)}(t) = C + P_{km}(t)$
C_{nj}
C_{ij}
C_{km}

Figure 4: Graph Topology Weights.

To tune the impact of C and P_{km} on the edge's weight, which will drive the solution, we introduce a coefficient, noted α. Now the edge weight is expressed as follows:

$$w_{(i,j)_{(k,m)}}(t) = \begin{cases} 1+(1-\alpha)P_{km}(t) & \text{if j = m} \\ \alpha C+(1-\alpha)Pkm(t) & \text{else} \end{cases} \quad (2)$$

where $(0 \leq \alpha \leq 1)$.

Thus, one can use the value of α to steer the solution by giving more priority for reducing service migration, or more priority for visiting less loaded cells. Since P_{km}(t) expresses a probability, its value is between 0 and 1, which is not the case for the service migration cost (C). Hence, the two values are normalized to give them the same scale, to make them influence the model in the same way.

Having defined the weight of the link connecting two adjacent nodes (neighbor cells), we denote by $P = (c_1, c_2, c_3, ..., c_n\}$ a path in V; where $c_i \in V$, and c_i, c_{i+1} are two adjacent nodes. We note by $f(c_i, c_{i+1})$ the function that returns the weight of the link between c_i and c_{i+1} as defined in equation (2). Now the problem consists in finding a path P that minimizes $\sum_{n-1}^{i=1} f(c_i, c_{i+1})$.

C. Resolution

To find the optimal path P, considering both the service migration and the cell overload, from the initial point to the landing point, we propose two algorithms. The first one is based on the well-known Djikstra algorithm [12]; while the second one is based on a greedy algorithm (Prim [13]). The two proposed algorithms are not sensitive to any use-case, since they just compute the best path regarding the chosen metrics.

1) Djikstra-based: The Djikstra-based algorithm calculates for each node, the shortest distance from the source node to it. To do that, it first initializes the initial node with a current distance of 0 and the distances of the remaining nodes with infinity. Then, it sets the non-visited node with the smallest current distance as the current node (S). For each neighbor (N), it adds the weight of the connection between S and N to the distance from the source to S. If the new value is smaller than the previous distance from the source to N, it updates the latter with the calculated value. It repeats this process until all nodes are visited. Our contribution to the Djikstra algorithm is the usage of the overload probability of the target node when we compare and update the distances from the source node. Indeed, instead of taking only the weight of the connection between the nodes, we add the probability value to that connection.

Algorithm 1: Djikstra-based Algorithm

```
P := {};
d[S] := ∞ for all S in the graph;
d[start] := 0;
while There is a node out of P do
    Choose a node S out of P with min distance d[S];
    Put S in P;
    for Each node B out of P and neighbor of a do
        if d[B] > d[S] + weight(S,B) +
          overload_probability(B) then
            d[b] := d[a] + weight(S,B) +
              overload_probability(B);
            previous[B] := S;
        end
    end
end
```

2) Prim-based: In addition, to the Dijkstra-based algorithm, we also introduce a greedy one, namely Prim algorithm [13]. It creates from a given graph, the Minimum Spanning Tree (MST), which is another graph extracted from the initial one, where all the vertices are connected via a path, and where the sum of all the weights is the minimum, taking into consideration the service migration as well as the overload probability. This algorithm first initializes the MST as an empty set, and then takes at every step the minimum weight edge from the initial graph, and add it to the MST in the case that an edge is valid. A valid edge between two nodes is when one end of it is already included in the MST and the other one is not. These steps are repeated, and the number of edges in MST (nbEdges) is incremented at each step until the MST holds a number of edges equal to the number of nodes in the initial graph (graphSize) minus 2 (nbEdges = graphSize - 2). Once the MST is formed, we reconstruct the path from a source to a target node using Breadth-first search algorithm [14], which is an algorithm for exploring a graph by going through all of the neighbor nodes at the present depth, then moving on to the nodes at the next depth level and so on, until we find the target node. This way we give the expected path using Prim algorithm.

Algorithm 2: Prim-based Algorithm

```
nbEdges := 1;
MST := {};
while nbEdges < graphSize - 1 do
    Find minimum weight edge E;
    If E is valide edge then add it to MST;
    nbEdges := nbEdges +1;
end
```

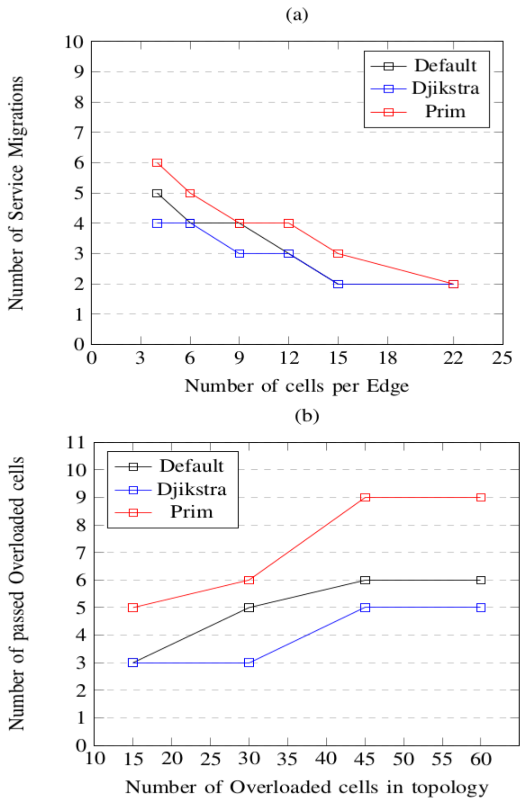


Figure 5: Metrics evolution for Scenario 1.

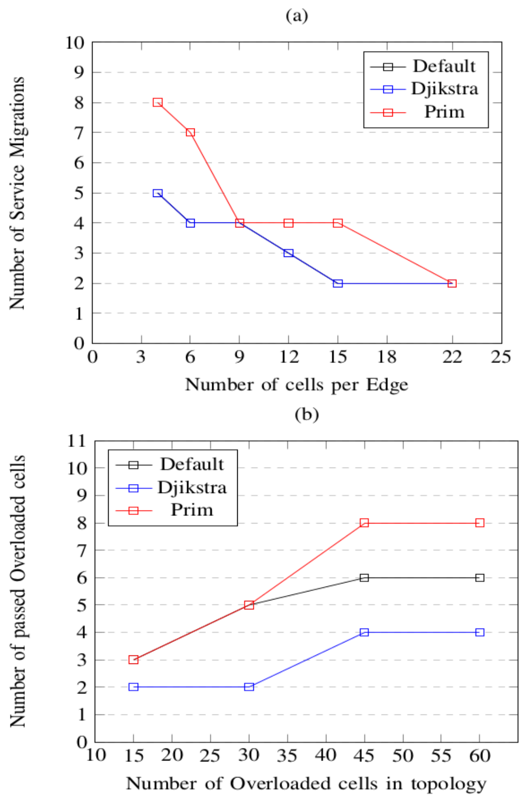


Figure 6: Metrics evolution for Scenario 2.

IV. Performance Evaluation

A. Topology and considered scenarios

To evaluate the performances of both algorithms, we executed them on the topology of Fig. 2. Note that this topology is just an example, we assume that each NOP has such a model for the geographical locations covered by its mobile network.

We considered two scenarios for tuning the expected solution; i.e. giving more priority for minimizing the service migrations, or for avoiding overloaded cells. To achieve this, we selected different values of α. It is worth noting that for all the scenarios, the migration cost C and the overload probability $\mathrm{P}(t)$ are normalized to give the same scale to the two parameters, by simply multiplying $\mathrm{P}(t)$ by 10, since the chosen default value of the service migration cost is 10, and the overload probability is ranged between 0 and 1. Any other way of normalization between the two variables can be easily done. From this point, all the values of C and $\mathrm{P}(t)$ are normalized.

- Scenario 1: In this scenario, more weight is given to the service migration, i.e. the expected solution tends to pass through an overloaded cell than to migrate a MEC service. The value that we used for alpha is $\alpha = 0.8$, so that the migration cost is greater than the overload probability.
- Scenario 2: Unlike Scenario 1, in this scenario, more weight is assigned to the overload probability, i.e. the expected solution avoids overloaded cells, and accepts more service migrations. We used for this scenario $\alpha = 0.2$, which will make the overload probability greater than the migration cost.

As stated earlier, we use the example of Fig. 2, which consists of a topology formed by 68 cells. For each scenario, we measure the performances of the three solutions in terms of the number of service migration, and the number of used cells that are overloaded. To measure the number of service migration, we first generate in a random way the overload probabilities of the cell and we vary the number of Cells per MEC Edge from 4 to 22; while for the number of overloaded cells we fixed the number of cells (9 per Edge server) covered by an Edge server and vary the number of overloaded cells in the topology. We assume that a cell is overloaded when its overload probability exceeds a certain threshold fixed by the Network owner (0.5 for example). For a sake of comparison, we also execute the Djikstra algorithm (As default in the figures) to find the shortest path, i.e. the weight of the graph edges are all equal to 1. This way, we obtain the shortest path, in terms of distance, between the initial and landing point. We then compute the number of service migrations and overloaded cells used by the path found by the three algorithms.

B. Results

Figures 5 and 6 show the performances of the three solutions, in terms of the two metrics, for scenario 1 and 2, respectively. The algorithms were implemented in Python. For both scenarios the number of service migration decreases as the number of cells per edge increases, which is logical as the higher the number of cells inside an edge is, the lesser the number of edges is, and the lesser the number of service migration is. Similarly, the number of overloaded cells selected in the proposed path increases as the number of overloaded cells in the topology increases. This is obvious as the higher the number of overloaded cells in the whole topology is, the higher the number of overloaded cells selected in the proposed path is. In addition, we remark that in Scenario 1 both algorithms (i.e. Djikstra-based and the default) achieve similar results and the best performances in terms of the number of service migrations. However Djikstra-based algorithm gives better results in terms of the number of overloaded cells held in the path. We argue this by the fact that the default algorithm finds always the same solution as the objective is to select the shortest path; while the Djikstra-based algorithm takes into consideration both metrics, by giving more weight for reducing the number of service migration. We also note that the greedy algorithm gives the worse solution than the two other solutions for both metrics.

For scenario 2, we see in Fig. 6 (a) that the Djikstra-based algorithm behaves as the default solution in terms of the number of service migration, while it achieves the best results in terms of the number of overloaded cells selected in a path (Fig. 6 (b)). We justify this by the fact that using the shortest path allows having interesting solutions for service migrations, while the Djikstra-based algorithm is seeking solutions that rather find a trade-off between the two metrics, with more weight given to reduce the number of overloaded cells. Indeed, we observe clearly in Fig. 6 (b) that the Djikstra-based algorithm achieves the best results in terms of the number of overloaded cells in the topology.

It is worth mentioning that, in the case of the Djikstra-based algorithm, the number of service migration in Scenario 2 is higher than in Scenario 1 (they are mostly between 3 and 8 in Scenario 2, while in Scenario 1 they are between 2 and 4). We argue this by the fact that the Djikstra-based algorithm gives more importance for avoiding the overloaded cells, which means that the model favorites a path with less overloaded cells. For the same reason, the number of overloaded cells runs through is less in Scenario 2 than in Scenario 1 (they are between 2 and 4 in this scenario while in Scenario 1 they are between 3 and 5).

These results clearly prove that our model is sensitive to the parameters' weights, which allows the Drones Operator to tune the model depending on its wishes, whether it wants to reduce the service migrations or the overloaded cells, or even equitably considering both parameters. Note that the difference between those numbers in the two scenarios is not consequent due to the used topology, where the edges are split in a way that the service migration cannot exceed a certain number. The difference should be more consequent and visible on a bigger topology, where the edges are numerous, which is expected in 5G.

V. Conclusion

In this work, we proposed a novel algorithm that intervenes at the flight planning step of a UAV mission, with the objective to reduce the service migrations as well as the overloaded cells that the drones will pass through. We have introduced a novel metric to measure the cost of UAV Pilot migration through Edge Servers. Based on this metric, we proposed two algorithms to solve the problem: a Prim-based algorithm and a Djikstra-based algorithm. Simulation results indicated that the Djikstra-based algorithm is more efficient regarding the two targeted metrics; hence improving the overall reliability of the C2link, which is a cornerstone requirement to enable a large scale adoption of UAV services and this allows the NOP to propose the best flight in terms of connectivity. In the future, since the proposed flight plan must be validated in terms of flight security too by UTM, our algorithm can be improved with a list of critical cells not to pass through, in order to ensure the avoidance of collisions with other drones in the same cells.

VI. Acknowledgment

This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program under the5G!Drones project (Grant No. 857031).

References

[1] International Civil Aviation Organization. (2019) Remotely Piloted Aircraft System (RPAS) Concept of Operations (ConOps) for International IFR Operations.

[2] A. Ksentini, P. Frangoudis, N. Nikaein, A. PC, "Providing low latency guarantees for slicing-ready 5g systems via two-level mac scheduling," *IEEE Network Magazine*, 2018.

[3] GSMA Public Policy Position. (2020, March) 5G Spectrum.

[4] A. Hueng, N.Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency mec framework for sdn-based lte/lte-a networks," *Proc. IEEE ICC*, 2017.

[5] Sarrigiannis, I., et al., "Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization," *IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2018.

[6] Rodrigo Romana, Javier Lopeza, Masahiro Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, January 2018.

[7] A. Aissioui, A. Ksentini, A. M. Gueroui, T. Taleb , "On Enabling 5G Automotive Systems Using Follow Me Edge-Cloud Concept ," *IEEE Trans. Vehicular Technology 67(6): 5302-5316*, 2018.

[8] Muhammad Arsalan Khana, Wim Ectorsa, Tom Bellemansa, Davy Janssensa and Geert Wetsa, "UAV-Based Traffic Analysis: A Universal Guiding Framework Based on Literature Survey," *Transportation Research Procedia*, 2017.

[9] Federal Aviation Administration. (2018) Unmanned Aircraft System (UAS) Traffic Management (UTM), Concept of Operations.

[10] David A. Levine, Ian F. Akyildiz and Mahmoud Naghshineh , "A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept ," *IEEE/ACM Transactions on Networking*, 1997.

[11] Imad Alawe, Adlen Ksentini, Yassine Hadjadj-Aoul, Philippe Bertin, "Improving traffic forecasting for 5g core network scalability: A machine learning approach," *IEEE Network Magazine*, November 2018.

[12] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, 1959.

[13] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, 1957.

[14] K. Zuse. Der plankalku. (1972) Der Plankalkül (BFS).