Communication-Aware UAV Path Planning

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 21, 2019, accepted April 5, 2019, date of publication April 15, 2019, date of current version May 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911018

Communication-Aware UAV Path Planning

AFSHIN MARDANI, (Student Member, IEEE), MARCELLO CHIABERGE, (Member, IEEE), AND PAOLO GIACCONE, (Senior Member, IEEE)

Department of Electronics and Telecommunications, Politecnico di Torino, 10124 Turin, Italy

Corresponding author: Paolo Giaccone (paolo.giaccone@polito.it)

This work was supported by the PIC4SER (PoliTO Interdepartmental Centre for Service Robotics) laboratory at the Politecnico di Torino, Italy.

ABSTRACT Autonomous air drones, known as unmanned aerial vehicles (UAVs), often accomplish missions with real-time video streaming leveraging the available cellular network. Video streaming is a typical application with stringent quality of service (QoS) requirements, which are not always supported in the whole mission area. In this paper, we address the offline path planning problem of finding the optimal path from a source to a destination in 2-D space in order to maximize the communication quality, given a cellular coverage, and thus providing throughput guarantees to the video streaming. In addressing the problem, the restricted amount of available energy, the wind effect, and the path post-smoothing problem are considered. We propose two innovative path planning algorithms and we show that our algorithms outperform classical approaches that are oblivious of communication network coverage. Both algorithms are variants of classical A* algorithm and they optimize the path jointly in terms of distance and of the experienced throughput by the drone: in this way, the quality of the video streaming along the path is optimized while preserving the energy budget for the flight. We describe both of the algorithms and investigate their performance. Moreover, we introduce a novel path smoothing method that outperforms classing approaches in terms of distance and computation cost. Finally, in order to prove that our algorithms can be practically utilized in the real-world path planners, we integrate our proposed path planning algorithms into the popular QGroundControl (QGC) control station.

INDEX TERMS Unmanned aerial vehicles (UAVs), autonomous navigation, path planning algorithms.

I. INTRODUCTION

Autonomous air drones are becoming increasingly popular and typically are equipped with real-time video streaming for surveillance and safety purposes. In recent years, the UAVs' excessive variety of applications including delivery, search and rescue, and inspections, require a highly secure and reliable connection. Focusing on this objective leads to support the UAVs on cellular network coverage. Wireless technology can bring many privileges to UAVs such as ubiquitous coverage, high-speed mobile support, robust security, high reliability and quality of service (QoS). This aim has been attracted the attention of industry and companies [1], to provide cellular connectivity for UAVs. For instance, telecommunications leading companies like Qualcomm and AT&T have set up to develop the connectivity technologies, including optimization of LTE networks and advancement of 5G technology for drones [2]. The 3rd Generation Partnership Project (3GPP) is started the study on enhanced LTE support for UAVs in 2017 [3]. Accordingly, UAVs can leverage the cellular network infrastructure to support the video streaming communication, in case of long-distance flights. When planning an autonomous path between a set of distant way-points, we advocate the adoption of path-planning algorithms which are aware not only of the obstacles (as in classical approaches) but also of the cellular coverage, in order to guarantee the stringent requirements of Quality of Service (QoS) in the communication for video streaming.

In this work, an offline path planning problem between two generic way-points, taken into account the provided budget of energy and the required bandwidth in terms of either average or minimum throughput is addressed. In addition, the cellular coverage map and possible effects of the wind that could have an impact on the path trajectory is considered. Moreover, we address the smoothing problem of the resulting paths of our path-planning algorithms, for the purpose of compensating for the spatial sampling of the graphs needed for computation and obtaining straight flying trajectories.

Our main contributions are manyfold.

1) We propose two innovative path-planning algorithms, which are variants of the popular A* algorithm. The main focus of AT-PP (Average Throughput Path

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiyong Chen.

Planning) algorithm is to maximize the average throughput along the path, with the purpose of guaranteeing an average level of communication QoS. The main focus of MT-PP (Minimum Throughput Path Planning) algorithm is instead to maximize the minimum throughput along the path, with the purpose of guaranteeing a minimum level of communication QoS. Each of the two algorithms take into account the on-board accessible amount of energy for the flight and lengthen the path for the sake of optimizing the communication quality.

2) By simulation, we evaluate the performance of the MT-PP and AT-PP algorithms, in both presence and absence of the wind. It is numerically shown that our communication aware approach is capable of outperforming classical approaches (oblivious of cellular coverage) significantly, concerning throughput, without exceeding the constraint of energy. Furthermore, we investigate the effect of energy budget on the resulting paths of MT-PP and AT-PP, and we show how the energy budget can affect the resulting path.
3) A novel path post-smoothing approach (IPS) is proposed that leads to a reduction in the computation time comparing with classical approaches.
4) We integrate seamlessly our path planning algorithms into the open-source QGroundControl control station [46], in order to modify the path between way-points considering the cellular coverage map. We prove that our algorithms are practical, by employing them in a real-world path planner.

The paper is organized as follows. The related works are discussed in Sec. II. The path planning problem is described in Sec. III. Our innovative communication aware path planning algorithms are proposed in Sec. IV. The achievable performance resulting from our proposed approach is assessed in Sec. V. In Sec. VI, we describe the integration of our algorithms in QGroundControl control station. Ultimately, we conclude the paper in Sec. VII.

A preliminary version of this work appeared in [4].

II. RELATED WORK

Up to now, various path-planning methods and algorithms for UAVs have been developed [5], but they are entirely unaware of the cellular network coverage. The algorithm in [6] is a heuristic search based on A*, which iteratively optimize the path during the available search time. Lazy Theta*, proposed in [7], is an any-angle path finding algorithm in which the resulting paths are not constrained to the form of graph edges and is faster than Theta* algorithm [8]. In [9], a heuristic-based re-planning algorithm (AD*) is presented that continuously improves its solution while time allows, and corrects its solution when updated information is received. In [10], inspiring by A* algorithm and Dubins path, a path planning method to discover the shortest, flyable and safe path for fixed wing UAVs with obstacle escape is presented. In [11], the authors discussed any-angle path planning algorithms that are variants of classical A* algorithm. In their method, through propagating information along grid edges, short paths are found without constraining the resulting paths to grid edges. In [12], a path planning method to create trajectories for aerial vehicles in a two dimensional space is proposed in order to avoid risky areas, conflict, and no-fly zones. In sampling based algorithms like Rapidly-exploring Random Trees (RRT) that is introduced in [13], the algorithm is able to find a sub-optimal path in a high dimensional space while some pre-defined information about the robot operating space is required. The RRT has no optimization process and re-planning procedure. Therefore, an improved version RRT* [14] is proposed to bridge this gap.

Recently, a lot of research has been done to optimize the path for UAV communication systems for different setups. In [15], the authors used an aerial wireless coverage 3-D modeling in mission planning of aerial vehicles for monitoring and surveillance of vast areas like long linear utility infrastructures. In [16], the flying trajectory of UAV is optimized for up-link communications. In [17], the authors used a UAV-based mobile relay in order to send data to different group of users. They optimized the data volume and relay trajectory based on the visiting sequence to the different group of users. In [18] and [19], the UAVs' movement is optimized to improve the network connection of a UAV assisted ad-hoc network. In [20], a multi-antenna UAV flying over a collection of single-antenna mobile ground nodes for providing relay services for mobile *ad hoc* networks is considered. They optimized the heading of UAV in order to optimize the up-link communication performance. In [21], they investigated a communication system with multi-antenna UAVs as relays between ground-based terminals and a network base station. They developed a closed form expression to optimize the UAV heading based on knowledge of the user terminal's future position, in order to maximize the uplink data rate. They attempted to keep the rate of each individual link above a certain threshold. In [22], the authors employed the UAVs as aerial base stations as a promising solution to enhance the performance of existing cellular systems. They exploit the UAV's high mobility to serve a group of users on the ground in order to maximize the minimum average rate among all the users, by jointly optimizing the trajectory of the UAVs, controlling the transmit power, and scheduling the user association and communication. In [23], a UAV-enabled orthogonal frequency division multiple access (OFDMA) system is studied, where the UAV is dispatched as a mobile base station to serve a group of users on the ground. The minimum average throughput of all ground users is maximized while meeting a given set of constraints, by jointly optimizing the UAV trajectory and OFDMA resource allocation. In [24], a comprehensive framework for hyper-dense small-cell networks assisted by caching UAVs is proposed, its feasibility is analyzed, and the secure transmission is discussed. UAVs are utilized to provide data traffic to mobile users cooperatively with small-cell base stations, due to their lower cost and higher mobility. The work in [25] studies a hybrid cellular

network with UAV-aided offloading at the edges of multiple cells, by accounting for the interference between ground base stations and UAV. The UAV trajectory is selected to maximize the sum rate of edge users by avoiding the interference, while the rate requirements of all the users are guaranteed.

However, in none of the above works the energy consumption of the UAVs and the effect of wind on the energy consumption has been considered.

In general, UAV communication systems may encounter various new challenges [26]. In fact, the performance of UAV communication networks are highly confined by on-board energy. Thus, maximizing the information bits while minimizing the energy consumption in the same time is of paramount importance. Furthermore, minimizing the energy consumption of UAV systems regarding the energy expenditure in propulsion power consumption to support the movements and maintaining the UAV in the sky, is more important than reducing the energy consumption in communication circuits and signal transmission.

Moreover, in many researches trajectory optimization has been studied, but not particularly for communication purposes. Many algorithms have been developed on energy-aware UAV path planning. For instance in [27], path optimization problem is studied considering the energy consumption of UAV, but the communication performance is not covered. In [28], the task of seeking out the goal while considering the total energy consumption of UAV to be minimized is studied. This optimization depends on unknown disturbances, like wind. In this study, the communication is neglected as well.

In [29], the authors represented a multi-layer architecture for intelligent navigation of Remotely Piloted Aircraft Systems (RPAS) in urban environments. In the path planning layer, the planner computes an optimal way point-based path, then the on-line planner continuously updates the offline path. Their algorithm searches for an optimal path that minimizes the cost function. It minimizes the risk-cost, the estimated energy consumption, and the length of the path. Another example is [30] which proposes a shortest trajectory planning for UAVs on an embedded GPU. A fast, energy-efficient global planner for multi-rotor UAVs has been developed supporting human operator during rescue mission. The planner is suitable for real-time path re-computation in dynamically varying environments but of small area (no more than 200 m^2). The work in [31] proposes a multi-objective path finder that can discover Pareto-optimal solutions concerning energy consumption and length of the path. Their solution is on the basis of NAMOA* search algorithm that exploits a monotone heuristic cost function. Unfortunately, in all of these works the effect of network coverage is completely neglected.

Several recent works have targeted the path planning for UAVs considering the energy efficiency and network yield, simultaneously. But the absence of wind as a crucial factor on the UAV's energy consumption and trajectory planning is sensible. In [32], the energy-efficient designs for drone communication is studied, where a drone is exploited to communicate with the base station. In this work, the energy efficiency maximization via path optimization is performed. The work in [33] proposed a computationally efficient suboptimal algorithm that can saves energy by 50 percent, increases network throughput by 15 percent, and extend network lifetime by 33 percent compared to the-state-of-the-art. In [34], the offline path finding problem for UAVs is addressed. The authors in this work attempted to discover paths that meet mission objectives, are safe considering collision and grounding, are fuel efficient, and satisfy criteria for communication. In [35], they proposed a joint UAV trajectory and power control scheme that significantly enhances the achievable rate of UAV communication system comparing to benchmark schemes. Particularly, they maximized the average achievable rate from the UAV to the ground receiver over a finite communication period by jointly optimizing the UAV trajectory and transmit power allocation. These are subject to maximum speed of UAV, initial/final locations, and average transmit power constraints. Nevertheless, in all of the above cited works, the wind effects on the energy consumption is neglected.

A lot of research has been done on following certain trajectories by UAVs under some circumstances. They mostly investigate the effect of wind on following a generated path accurately in order to accomplish the assigned tasks. This effect can be considered in the control loop design of the UAVs to keep the position error of flight path under a determined error with respect to the desired path. Therefore, the wind disturbances are not considered in the path generation. The work in [36] considered the effect of wind that causes the aircraft to drift in a certain direction. A method is designed based on an accelerated A* algorithm that follows the trajectory planner to take into account the wind effects. In [37], UAV path following in cluttered environments under windy conditions in a two dimensional configuration space is investigated. They designed a novel guidance law with low computational complexity which allows the UAV to follow the path with minimum deviation. In [38], the authors presented an algorithm based on the idea of following a vector field that converges smoothly to the desired path. Their work includes the technique of dealing with wind disturbances when following a generic sufficiently smooth 2-D path.

There is not a unique approach to compensate for wind effects in trajectory planning for UAVs. For instance, in [39], the authors dealt with the wind disturbances on trajectory planning by iteratively solving a no wind case problem with a moving virtual target. In most of the work like in [40]–[42], the authors usually compensate the wind in the control loop design of the UAVs to keep the position error of flight path as minimum as possible with respect to the desired path. However, we are compensating for the wind disturbances during the path computation, before the flight. This method helps us to be able to explore the effect of the wind in energy consumption of the drone. Accordingly, we are able to precisely calculate the energy consumption of the drone

and plan the most appropriate path for it. We calculate the near-optimal path offline, considering the *expected* wind. The drone internal flight controller compensates in real time for the actual wind conditions experienced while following the path computed according to the expected wind.

III. THE PATH PLANNING PROBLEM

In this work, the problem of generating the near-optimal path from an initial way-point to another one in a 2-D space for an autonomous UAV is addressed. We maximize the quality of the video streaming application during the flight in our innovative communication-aware approach by considering the accessible energy on-board and the expected wind conditions. Noteworthily, we compute the trajectory offline and upload it to the drone flight controller. Then, it follows the offline path and compensates for the actual disturbances during the flight.

A. FLIGHT AREA

We modelled the flight area, as shown in Fig. 1, based on a grid graph. Indeed, we subdivided the cellular coverage area into parts with a regular tessellation and placed each node at the center of each square. We associated each node with a throughput value which is an average throughput value experienced within the corresponding square. An edge connects the nodes corresponding to adjacent squares. In our study, 8-degree grid graph is considered, i.e. eight neighbors for every node. It is notable that our approach is extendable to any other type of grid graph.

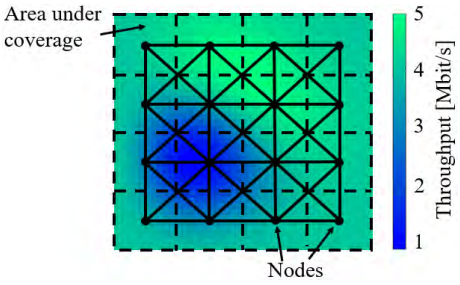

FIGURE 1. 8-Degree grid graph and the corresponding coverage map.

Formally, we define the flight area with an undirected grid graph $G = (V, E)$. We associate node $i \in V$ with a physical position (x_i, y_i) and the throughput b_i. b_i is the drone experienced throughput in the area in the proximity of node i when uploading the streaming data to the cellular network. The physical distance d_{ij} is associated with the edge $(i, j) \in E$ connecting nodes i to j.

This work is focusing on the path planning for the UAVs. We are concentrating on how to design a path for a mission that the drone experience the maximum throughput while the throughput map is given. Therefore, in this work, the average throughput experienced in each square is assumed to be known in advance. This information can be obtained through some coverage maps obtained by means of on-field measurements or on some channel model applied into the square area.

For the sake of simplicity, obstacles are not considered in the flight area, even though we can easily extend the methodology to this scenario by removing the corresponding nodes and edges.

B. DRONE MOBILITY MODEL

The drone is assumed to fly at constant *air speed* v_d, and its power consumption is equal to P. The drone's total available on-board energy is E_0. The required energy due to traveling from node i to node j is E_{ij}. The *source node* i_s is the node that the drone departs from as the starting way-point, and the *destination node* i_d is the node that the drone terminates the mission as the destination way-point. A path comprised of multiple way-points decomposes into several segments, and the algorithm runs for each segment.

Let $\mathcal{P}$ be the set of all possible loop-less paths connecting i_s to i_d and let $p \in \mathcal{P}$ be a generic path. The path planning aim is finding a path $p \in \mathcal{P}$ that connects i_s to i_d such that maximizing the communication performance, contingent on the following amount of energy:

$$\sum_{(i,j)\in p} E_{ij} \leq E_0 \tag{1}$$

C. MOBILITY PLANNING BETWEEN TWO NODES

Considering the problem in which the drone needs to move from way-point i to j. Let v_{dx} be the drone *air speed*, along the x-axis and v_{dy} be the drone *air speed*, along the y-axis. Let v_{gx} and v_{gy} be the drone *ground speed*, along the x-axis and y-axis, respectively. Let t_{ij} be the flight time from i to j. The *wind speed* is assumed to be constant and its speed along the two axes are v_{wx} and v_{wy}.

Now the following system of equations can be written:

$$v_{gx} = v_{dx} + v_{wx} \tag{2}$$
$$v_{gy} = v_{dy} + v_{wy} \tag{3}$$
$$x_j - x_i = v_{gx} \cdot t_{ij} \tag{4}$$
$$y_j - y_i = v_{gy} \cdot t_{ij} \tag{5}$$
$$v_{dx}^2 + v_{dy}^2 = v_d^2 \tag{6}$$

where the effect of wind and the actual ground speed are related by (2) and (3); the traveled physical distance is related to the ground speed by (4) and (5); eventually, the air speed of drone is related to its two components by (6). The drone speed and its flight time are calculable by solving the above equations. Therefore, the flight distance travelled by the drone is $v_d \cdot t_{ij}$, and the energy consumption between node i and j is:

$$E_{ij} = P \cdot t_{ij} \tag{7}$$

The ground distance from node i to j is equal to the Euclidean distance between them. The total transferred data along the path from node i to j is $b_i \cdot t_{ij}$. Table 1 contains the introduced parameters and their descriptions altogether.

Considering the energy model employed in [27], the energy consumption of the drone depends on different operating conditions, such as speed, horizontal and vertical acceleration,

TABLE 1. Variable definitions.

Parameter	Description	Unit
b_i	Cellular network throughput at node i	bit/s
d_{ij}	Physical distance	m
E_{ij}	Energy consumed between two nodes	J
E_0	Total energy of UAV	J
t_{ij}	Time of flight	s
P	Power consumption	W
v_d	Air speed of UAV	m/s
v_g	Ground speed of UAV	m/s
v_w	Wind speed	m/s

and turning angle. We do not consider the consumed energy in climbing and descending, since it is common in all path plannings and does not affect the path. Moreover, we do not have any hovering period in our applications. Therefore, we consider the energy consumption during the rotations based on the work in [27]. The energy required to cover all the n turns (with angles $\theta_k, k \in \{1, 2, \ldots, n\}$) included in the path is computed as:

$$E_{\text{turns}} = \sum_{k=1}^{n} P_{\text{turn}} \frac{\theta_k}{\omega_{\text{turn}}} \tag{8}$$

where ω_{turn} represents the angular rotation speed, P_{turn} represents the power consumed during the rotations. Therefore the total consumed energy is computed considering also all the energy spent for all the turns along the path:

$$E_{tot} = E_{i_s i_d} + E_{\text{turns}} \tag{9}$$

IV. NEAR-OPTIMAL COMMUNICATION-AWARE PATH PLANNING

Our propounded approaches approximate the optimal path founded on the classical A* search algorithm for path planning, adopted in many contexts as robotics and video games. A* was designed for finding the shortest (or approximately shortest) path in a much shorter computation time than canonical Dijkstra's algorithm, by considering a smaller search subspace. Actually, A* employs a cost function which is the summation of two functions. Function g (in common with Dijkstra's algorithm) represents the cost of the path from the source to the current node, and function h is a user provided heuristic function that estimates the cost of the path from the current node to the goal. Computation time and optimality is conditional on the choice of heuristic function. Particularly, if the estimated cost by the heuristic function is equal to the real cost, the algorithm only expands the nodes on the least cost path from the start to the goal.

A. PATH PLANNING ALGORITHMS

In this section, we present our path planning algorithms as variants of the A* algorithm.

1) AT-PP ALGORITHM

This algorithm is a modified version of A* on grids. We provided the pseudo-code of the algorithm in the Algorithm 1.

Algorithm 1 Average-Throughput Path Planning (AT-PP)

1: **function** AT-PP($\mathcal{N}$, $\{b_v\}_{v\in\mathcal{N}}$, S, D, P, v_d, E_0, β)
2: **for each** vertex $v \in \mathcal{N}$ **do** ▷ Initialization, for each vertex
3: path_bw[v] $= -1$ ▷ Throughput from S to v
4: acc_path_bw[v] $= -1$ ▷ Cumulative throughput from S to v
5: parent[v] $= -1$ ▷ Parent of node v
6: ground_path_distance[v] $= \infty$ ▷ Ground distance from S to v
7: flight_path_distance[v] $= \infty$ ▷ Flight distance from S to v
8: path_hops[v] $= -1$ ▷ Number of nodes from S to v
9: path_time[v] $= -1$ ▷ Flight time from S to v
10: path_energy[v] $= \infty$ ▷ Energy from S to v
11: path_cost[v] $= \infty$ ▷ Cost based on A*
12: path_bw[S] $=$ acc_path_bw[S] $= b_S$ ▷ Setting the values for S
13: parent[S] $= S$
14: ground_path_distance[S] $=$ flight_path_distance[S] $=$ path_hops[S] $= 0$
15: path_time[S] $=$ path_energy[S] $= 0$
16: path_cost[S] $=$ GroundDistance(S, D)
17: $\mathcal{U} = \mathcal{N} \setminus \{S\}$ ▷ Unvisited nodes
18: $\mathcal{F} = \{S\}$ ▷ Frontier nodes
19: $\mathcal{V} = \emptyset$ ▷ Visited nodes
20: **while** $\mathcal{F}$ is not empty **do** ▷ Visit all the frontier nodes
21: $u = \arg\min_{v\in\mathcal{F}} \{\text{path\_cost}[v]\}$ ▷ Find the min cost node in $\mathcal{F}$
22: move u from $\mathcal{F}$ to $\mathcal{V}$
23: **if** path_energy[u]$> E_0$ **then** ▷ Check the required energy to reach u
24: **continue** ▷ If greater than E_0, consider a new node in $\mathcal{F}$
25: **if** $u = D$ **then** ▷ Check if arrived to destination
26: **return** ▷ End. Return the whole state, from line 4 to 11
27: **for each** neighbor $v \notin \mathcal{V}$ of u **do** ▷ Check all the neighbors of u that are in $\mathcal{U}$ or $\mathcal{F}$
28: **if** $v \in \mathcal{U}$ **then**
29: move v from $\mathcal{U}$ to $\mathcal{F}$ ▷ Move v to the frontier, if is not there
30: bw $=$ (acc_path_bw[u]$+b_v$) / (path_hops[u]$+1$) ▷ Average throughput along the path
31: path_bw[v] $=$ bw
32: acc_path_bw[v] $=$ acc_path_bw[u] $+b_v$
33: parent[v] $= u$
34: ground_path_distance[v] $=$ ground_path_distance[u] $+$ GroundDistance-TANCE(u, v)
35: flight_path_distance[v] $=$ flight_path_distance[u] $+$ FlightDistance(u, v, v_w, w_{dir})
36: path_hops[v] $=$ path_hops[u] $+ 1$
37: path_time[v] $=$ path_time[u]$+$FlightDistance(u, v, v_w, w_{dir})$/v_d$
38: path_energy[v] $=$ path_energy[u] $+P \times t_{uv}$
39: path_cost[v] $=$ (flight_path_distance[u]$+$FlightDistance (u, v, v_w, w_{dir})) $+$ FlightDistance(v, D, v_w, w_{dir}) $+\beta/$ path_bw[v] ▷ Main cost function
40: **return** error - unreachable destination

For the sake of readability, degenerate cases are neglected. In this algorithm the average throughput along the path is maximized, as:

$$\max_{p\in\mathcal{P}} \sum_{(i,j)\in p} \frac{b_i}{|p|} \tag{10}$$

contingent on the budget of energy. Accordingly, we modified the cost function of A* as is shown in ln. 39. Our cost function combines the estimated distance to the final destination D and the inverse of the average throughput (''path_bw'') up to some node. Therefore, by increasing the average throughput, an average level of communication QoS is guaranteed.

AT-PP algorithm maintains several values for every vertex v (ln. 3-11):

- *path_bw*(v) is the throughput from the start node to node v found so far. It is used in the cost estimation of node v in the cost function.
- *acc_path_bw*(v) is the cumulative throughput from the start node to node v, and it is utilized for average throughput calculation.
- *parent*(v) allows to retrieve the resulting path when the algorithm terminates.
- *ground_path_distance*(v) contains the ground distance from the start node to node v.
- *flight_path_distance*(v) contains the flight distance from the start node to node v considering the effect of wind. This value is used in the estimate of the cost of node v in the cost function.
- *path_hops*(v) contains the number of nodes from start node to node v.
- *path_time*(v) is the flight time from the start node to node v, considering the wind.
- *path_energy*(v) is the flight energy consumption from the start node to node v, considering the wind.
- *path_cost*(v) is the cost of the path from start node to node v.

As in usual Dijkstra and A*, AT-PP algorithm creates three sets of nodes. The frontier nodes ($\mathcal{F}$) are stored in a priority queue and the visit procedure expands from them. The visited nodes ($\mathcal{V}$) are the nodes that have been already visited and will not be visited anymore. The unvisited nodes ($\mathcal{U}$) are the remaining nodes, i.e. not frontier nodes and not yet visited. As long as the frontier nodes list is not empty (ln. 20) (if it is empty, it announces that the destination is unreachable (ln. 40)), the algorithm selects the node with minimum path cost (ln. 21) from the frontier nodes as the current visiting node. First, it moves the current visiting node from frontier nodes to visited nodes (ln. 22), then it checks the required energy to reach the current visiting node (ln. 23). In case of exceeding the energy budget, the algorithm fetches a new node from the frontier nodes (ln. 22). Otherwise, in the next step, it checks if the current visiting node is the destination or not (ln. 25). If yes, the algorithm extracts the obtained path and terminates. Otherwise, it updates all the neighbors of the current visiting node which are not already visited (ln. 27). It checks each neighbor if it is in frontier nodes or not (ln. 28). If that neighbor is not included in frontier nodes, it will be moved there (ln. 29), and all the values will be updated (ln. 31-38). In this algorithm, throughput is the average along the path (ln. 30). Finally, the path cost will be updated in (ln. 39). This procedure continues until the algorithm reaches to the destination.

The fundamental difference in our algorithm from A* is that our algorithm does not examine whether the value of the g function of visiting node plus the length of the straight line from the visiting node to the next neighbor node is smaller than the value of the g function of the next neighbor node. In our algorithm, distance is considered in the cost function plus the inverse of the bandwidth of the next neighbor. Afterwards, the neighbor of the visiting node with minimum cost will be sent to the frontier nodes list of the algorithm. Therefore the node with minimum cost from the frontier nodes list will be the next visiting node. Then, the algorithm searches for the path with minimum cost. Consequently, both the throughput and the distance are optimized in the resulting path.

In the cost function, the bandwidth is weighted by a factor β. This coefficient is tuned to be a comparable value with the other part of the cost function. As shown in Sec. V-C, β is numerically tuned to maximize the throughput in different scenarios for a specific coverage map.

The available on-board energy E_0 is taken into account by pruning the visit as long as the energy consumption overshoots E_0 (see ln. 23). Notably, it is probable not to find any path due to the incompatibility of the energy consumption and E_0 (see ln. 40).

2) MT-PP ALGORITHM

In this algorithm the minimum throughput along the path is maximized, contingent on the budget of energy. We modified the cost function of A* and defined the worst-case throughput cost function for the algorithm as following:

$$\max_{p\in\mathcal{P}} \min_{(i,j)\in p} b_i \tag{11}$$

Therefore, by keeping the throughput above a minimum level, a minimum level of communication QoS is guaranteed.

We provided the pseudo-code of the algorithm in the Algorithm 2. This is an iterative algorithm and functions as follows, mimicking a dichotomic search. At the beginning, the algorithm finds the shortest path while the minimum throughput is maximized. In this situation, if the calculated energy consumption is less than the available on-board energy, the resulting path is the best and final path. On the contrary, if the calculated energy consumption for the resulting path is greater than the energy budget of UAV, an initial threshold value t_h is assumed by the algorithm. t_h could be set equal to the average throughput achievable in the area. In this part, the algorithm maintains three variables: *throughput level* (b_{level}), which contains the highest throughput value that the algorithm could obtain in the first stage for the resulting path, *throughput step* (b_{step}), which contains the decrement value

Algorithm 2 Minimum-Throughput Path Planning (MT-PP)

1: **function** DronePathPlanMT-PP()
2: $(Graph, S, D, P, b_n, v_d, (x_n, y_n), E_0, \beta, \mathcal{N}, b_{step}, b_{res}, b_{level})$:
3: PathMT-PP() ▷ Compute the path
4: **while** $b_{step} >= b_{res}$ **do** ▷ As steps are greater than resolution
5: **if** path_energy $> E_0$ **then**
6: $b_{step} = b_{step}/2$ ▷ Each step will be half of previous step
7: $b_{level} = b_{level} - b_{step}$ ▷ Throughput decreases by one step
8: **if** $b_{level} < b_{res}$ **then**
9: $b_{level} = \min\{b_n\}$ ▷ Throughput is minimum
10: PathMT-PP()
11: **else if** path_energy $< E_0$ and $b_{level} < \min\{b_S, b_D\}$ **then**
12: $b_{step} = b_{step}/2$
13: $b_{level} = b_{level} + b_{step}$ ▷ Throughput increases by one step
14: PathMT-PP()
15: **else**
16: **break**
17: **function** PathMT-PP()
18: **for each** vertex $v \in \mathcal{N}$ **do** ▷ Initialization, for each vertex
19: path_bw$[v] = -1$ ▷ Throughput from S to v
20: acc_path_bw$[v] = -1$ ▷ Cumulative throughput from S to v
21: parent$[v] = -1$ ▷ Parent of node v
22: ground_path_distance$[v] = \infty$ ▷ Ground distance from S to v
23: flight_path_distance$[v] = \infty$ ▷ Flight distance from S to v
24: path_hops$[v] = -1$ ▷ Number of nodes from S to v
25: path_time$[v] = -1$ ▷ Flight time from S to v
26: path_energy$[v] = \infty$ ▷ Energy from S to v
27: path_cost$[v] = \infty$ ▷ Cost based on A*
28: path_bw$[S]$ = acc_path_bw$[S] = b_S$ ▷ Setting the values for S
29: parent$[S] = S$
30: ground_path_distance$[S]$ = flight_path_distance$[S]$ = path_hops$[S] = 0$
31: path_time$[S]$ = path_energy$[S] = 0$
32: path_cost$[S]$ = GroundDistance(S, D)
33: $\mathcal{U} = \mathcal{N} \setminus \{S\}$ ▷ Unvisited nodes
34: $\mathcal{F} = \{S\}$ ▷ Frontier nodes
35: $\mathcal{V} = \emptyset$ ▷ Visited nodes
36: // Visit all the frontier nodes
37: **while** $\mathcal{F}$ is not empty **do**
38: $u = \arg\min_{v \in \mathcal{F}} \{\text{path\_cost}[v]\}$ ▷ find the min cost node in $\mathcal{F}$
39: move u from $\mathcal{F}$ to $\mathcal{V}$
40: **if** $u = D$ **then** ▷ Check if arrived to D
41: **return** All parameters from line 4 to 11
42: // Check all neighbors
43: **for each** neighbor $v \notin \mathcal{V}$ of u **do** ▷ Check all the neighbors of u that belongs to $\mathcal{U}$ or $\mathcal{F}$
44: **if** $v \in \mathcal{U}$ **then**
45: move v from $\mathcal{U}$ to $\mathcal{F}$ ▷ Enter v to the frontier, if its not
46: **if** (flight_path_distance$[u]$ + FlightDistance(u, v) < flight_path_distance$[v]$) **then** ▷ Update, in case of better flight distance
47: bw = min{path_bw$[u]$,b_v} ▷ Throughput is min along the path
48: path_bw$[v]$ = bw
49: parent$[v] = u$
50: ground_path_distance$[v]$ = ground_path_distance$[u]$ + GroundDistance(u, v)
51: flight_path_distance$[v]$ = flight_path_distance$[u]$ + FlightDistance(u, v)
52: path_time$[v]$ = path_time$[u]$ + FlightDistance$(u, v)/v_d$
53: path_energy$[v]$ = path_energy$[u] + P \times t_{uv}$
54: path_cost$[v]$ = (flight_path_distance$[u]$ + FlightDistance (u, v)) + FlightDistance(v, D) + β/ path_bw$[v]$ ▷ Cost function
55: **return** error - unreachable destination

in each iteration, and *throughput resolution* (b_{res}), which determines the final resolution for the solution. At the beginning, b_{step} is set equal to half of the maximum achievable throughput in the coverage map. On the basis of a variant of AT-PP, the algorithm selects the path such that the throughput at some visited node is computed as the minimum along the path (unalike to ln. 30 of AT-PP).

On the termination of each iteration, t_h increases in order to maximize the minimum throughput. The algorithm may not find a path with a minimum throughput $\geq t_h$ (possibly due to the energy restriction), then it decreases t_h in the following iteration. A dichotomic search is adopted in order to optimize the selection procedure of t_h values. The process terminates as long as the algorithm achieves a given precision on the minimum throughput along the path.

The obtained path approximates the one with the maximum throughput possible compatible with the available energy. However, when the resulting path corresponds to the minimum distance path, and the calculated energy consumption is still not compatible with the available energy, then there does not exist any path for this mission that meets the energy budget.

B. PATH SMOOTHING

The MT-PP and AT-PP resulting paths are constrained to grid edges since their headings are actually restricted to be along the edges of the grid graph. Consequently, the resulting paths are longer than the shortest path on the free 2-D space. This shortcoming led to smooth the resulting paths, by employing a post-smoothing process leading to an increase in the run-time.

1) POST-SMOOTHING METHOD

We take into account the A* post-smoothing algorithm (A*PS) presented in [43]. The smoothing procedure starts from the first node of the obtained path by A*. It checks if the current node has line of sight to the successor of its successor in the path or not. We say node u and u' have line-of-sight (LOS) if and only if the straight line from u to u' neither passes through the low throughput (less than a specified minimum network throughput) nodes nor passes between low throughput nodes that share an edge. If so, the algorithm removes the intermediate node and connects the current node to the successor of its successor. A*PS continues this procedure until the current node does not have a LOS to the successor of its successor. The resulting paths of A*PS are usually shorter than A* on grids. In this method, implemented in our approach, the well-know line-drawing algorithm of Bresenham in computer graphics [44] is used to check LOS between two nodes [45]. Notably, we tailored specifically the *Grid* function to consider only nodes with a throughput larger than a minimum specified value.

2) IMPROVED POST-SMOOTHING METHOD

In this section, we introduce an innovative improved post-smoothing process (A*IPS). The resulting paths of our

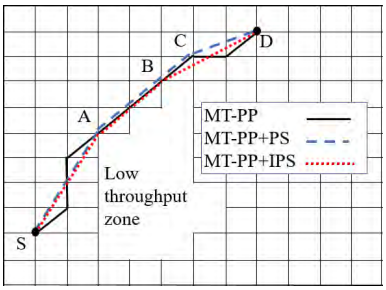

FIGURE 2. The difference between A*IPS vs A*PS shown in an example.

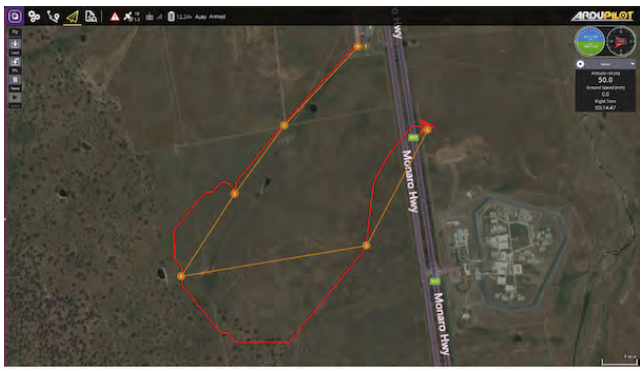

FIGURE 3. The GUI of QGroundControl integrated with our approach. The orange path is the original path, whereas the red one is the one computed by MT-PP and uploaded to the drone.

method are shorter than A*PS in some cases. A*IPS functions similar to A*PS, but the difference arises when the A*PS, during the graph visit, is checking the LOS from the visiting node to the successor of its successor. At this stage, A*IPS in parallel, is additionally checking the LOS from the successor of its successor to the goal node. A*IPS terminates the whole process as long as the LOS is discovered to the goal and the shortest path is achieved. In Fig. 2, we show an example to be more clear about the process. If we consider the point A as the starting point, the A*PS checks the LOS from point A, node by node, towards the goal. Since A*PS has no LOS (observing from point A) to any node located between point C and D, the process ends in point C and then starts over from point C to the goal. However, A*IPS is checking the LOS from both the starting point A and its following node to the goal simultaneously. As a result, A*IPS detects the LOS from point B to the goal and ends the process immediately. Notice that in this work, LOS is defined in terms of throughput values; we consider two points in LOS if all the closest nodes in the direct path among them possess throughput values of greater or equal to the current value.

In our algorithms, the post-smoothing process executes after finding the path in each iteration, enabling us to calculate the real energy consumption after smoothing (making the path shorter) and also considering the final turning angles of the path in calculating the energy consumption.

It will be shown inn Sec. V-C that the A*IPS resulting paths are shorter or equal to the A*PS, with a possible reduction in computation time.

V. PERFORMANCE EVALUATION

A. SCENARIOS

Fig. 4 reports the two coverage maps on a 4 km×4 km area that we tested our algorithms on them. In the first map denoted as "cone" (left side of Fig. 4), maximum throughput shows in the center, then linearly it decreases to a minimum value, and finally it remains constant in the whole border with a higher value than the minimum. The performance of greedy approaches may be impaired by such discontinuity in the coverage (as the one considered in our work). Therefore, we can consider the cone map as a simple worst-case coverage scenario. In the second map denoted as "valleys" (right side of Fig. 4), the coverage is continuous with four low throughput valleys.

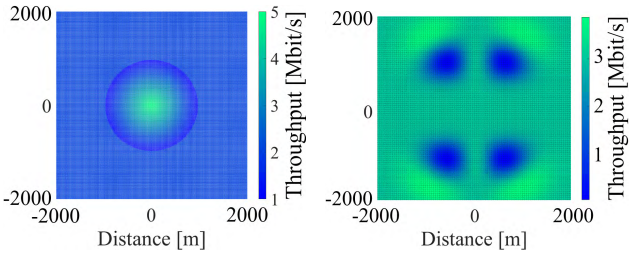

FIGURE 4. The valleys map (right), and the cone map (left).

The UAV mission starts from a starting point to an ending point. We tried to combine the selection of various source-destination pairs with the coverage map, defining the following scenarios:

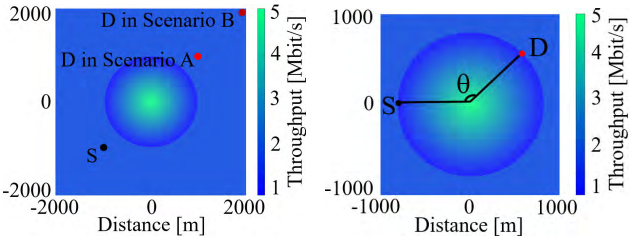

FIGURE 5. Scenario A, B (left) and scenario C (right). "S" stands for source position. "D" stands for destination position.

- Scenario A (Fig. 5): A symmetric case on the cone map. The source is positioned in (−1000,−1000) m and the destination in (1000,1000) m.
- Scenario B (Fig. 5): An asymmetric case on the cone map. The source is positioned in (−1000,−1000) m and the destination in (2000,2000) m.
- Scenario C (Fig. 5): We positioned the source and the destination at θ degrees from the peak of throughput on the cone map. It allows us to investigate how a path is deviated from its direction through the high throughput region.
- Scenario D (Fig. 7): The source is positioned in (−1000,−1500) m and the destination in (1000,1300) m, on the valleys map. Regarding the two low throughput regions between the source and destination on the direct

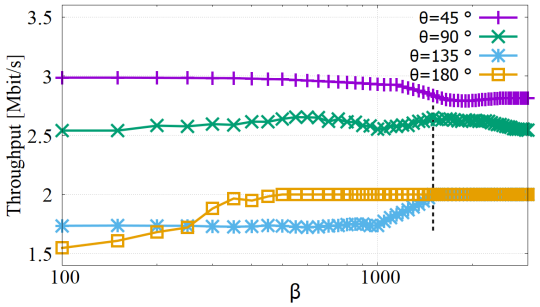

FIGURE 6. β optimization under scenario C.

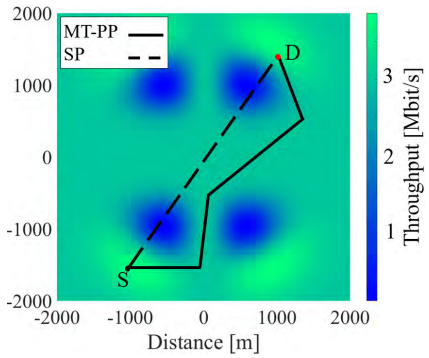

FIGURE 7. Resulting paths obtained by MT-PP and SP under scenario D.

path, this case enables us to demonstrate the deviation of the resulting path from the low throughput regions.

In this work, wind is called "head wind" when it is in the opposite direction of the direct path from source to destination, and is called "tail wind" when it is in the same direction.

B. METHODOLOGY

In this work, we compare the behavior of our algorithms with the Shortest Path (SP) algorithm, both in dealing with the wind presence and the on-board energy limit. SP is selected since it represents all the cited state-of-the-art algorithms in Sec. II, that compute the direct path from source to destination and are oblivious of the coverage map.

In the simulations of this work, as depicted in Fig. 1, 8-degree graph on a 101 $\times$ 101 grid with a distance of minimum 40 meters between two nodes is considered. Notably, the graph can simply be modified to represent the presence of obstacles by removing the corresponding nodes and edges. Thus, our algorithms can consider the presence of obstacles. However, we did not consider any obstacle in this work for the sake of simplicity.

We implemented our algorithms in MATLAB and executed them on a 2.67 GHz Core i7 PC with 8 GByte of RAM running Windows 10. We used `tic` and `tac` MATLAB commands to evaluate the running times.

C. NUMERICAL EVALUATION

Firstly the choice of coefficient β which is employed in the cost function of our algorithms is optimized. Towards this end, under scenario C, we selected four destinations in θ = 45, 90, 135, and 180 degrees. In Fig. 6, throughput in function of β is plotted. In order to obtain a high throughput value in most of the cases, it is set to β = 1500 (shown with dashed line in Fig. 6).

TABLE 2. Average throughput comparison.

Scenario	Algorithm	Average Throughput [Mbit/s]		
		no wind	head wind 2 m/s	tail wind 2 m/s
D	MT-PP + IPS	3.3	2.6	3.5
	AT-PP + IPS	2.7	2.3	2.8
	SP	2.2	2.2	2.2
C $\theta = 135°$	AT-PP + IPS	2.8	1.5	2.8
	MT-PP + IPS	1.5	1.5	1.5
	SP	1.5	1.5	1.5
A	MT-PP + IPS	2.0	2.0	2.0
	AT-PP + IPS	2.5	2.5	2.5
	SP	2.5	2.5	2.5
B	MT-PP + IPS	2.0	2.0	2.0
	AT-PP + IPS	2.3	2.3	2.3
	SP	2.3	2.3	2.3

TABLE 3. Minimum throughput comparison.

Scenario	Algorithm	Minimum Throughput [Mbit/s]		
		no wind	head wind 2 m/s	tail wind 2 m/s
D	MT-PP + IPS	3.0	1.0	3.1
	AT-PP + IPS	1.8	1.0	2.2
	SP	0.3	0.3	0.3
C $\theta = 135°$	AT-PP + IPS	0.1	0.1	0.1
	MT-PP + IPS	0.1	0.1	0.1
	SP	0.1	0.1	0.1
A,B	MT-PP + IPS	2.0	2.0	2.0
	AT-PP + IPS	0.1	0.1	0.1
	SP	0.1	0.1	0.1

Table 2 compares the obtained average throughput and Table 3 compares the obtained minimum throughput along the resulting paths of our algorithms with SP, for all the defined scenarios, in presence and absence of wind. We employed the IPS method to smooth the resulting paths of both the MT-PP and AT-PP algorithms.

Consider for now the performance without wind. Fig. 7 is verifying the expected performance of MT-PP under scenario D. In this figure, the obtained paths by MT-PP and SP are depicted. Considering the Table 2 and 3, MT-PP and AT-PP both achieved a greater average throughput comparing with SP (50% greater for MT-PP) whereas the achieved minimum throughput is much greater than SP (9 times greater for MT-PP.)

Fig. 8 shows the path found by AT-PP under scenario C (θ = 135°), verifying the algorithm behavior. Specifically the resulting path tends to pass the close proximity of the cone vertex with maximum throughput. This path is optimal, considering that the average throughput is maximized. Thus, the average throughput of the path is ameliorated up to 1.3 Mbit/s comparing with SP.

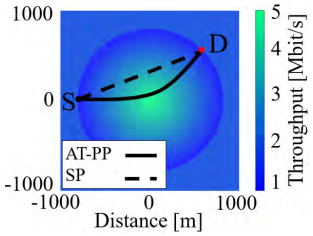

FIGURE 8. Resulting paths obtained by AT-PP and SP under scenario C ($\theta = 135°$).

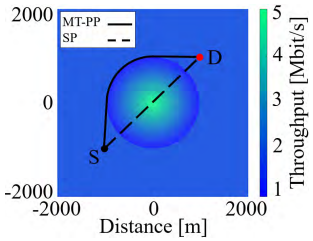

FIGURE 9. Resulting paths obtained by MT-PP and SP under scenario A.

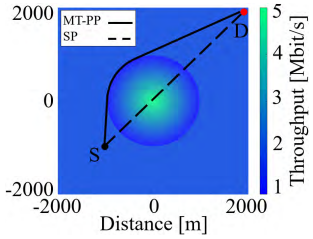

FIGURE 10. Resulting paths obtained by MT-PP and SP under scenario B.

In addition, in scenarios A in Fig. 9, and in scenario B in Fig. 10, MT-PP achieves a minimum throughput that outperforms the SP by about 20 times.

Now, we consider the wind effect. MT-PP and AT-PP gain less in the presence of head wind, comparing with either the tail wind or the wind absence. The reason behind is the restricted amount of on-board energy that prevents the drone from flying into the high throughput regions and compensating for the wind concurrently. The throughput of the path in scenario A and B is not influenced by the wind in either case, since MT-PP has a tendency to keep away from the low throughput regions so far as energy is provided. The resulting paths of AT-PP and MT-PP in scenario C and D, in presence of head wind, are approaching to SP leaving more energy behind for the wind compensation. As a consequence, the throughput of resulting paths are lower than the tail wind or no wind cases.

In the considered offline path-planning scenario, the wind can be taken into account just based on some forecast weather model. For an area of few km squares (or at most tens of them) where the UAV will fly, we expect to know in advance only an average wind value for each tile.

Nevertheless, the algorithm is compatible, without any modification, with variable winds, i.e. wind intensity

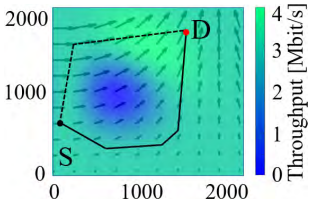

FIGURE 11. Comparing the performance of MT-PP in presence of variable wind (solid line), and absence of wind (dashed line).

different for each point of the grid map. In Fig 11, we considered variable wind in the field. In this scenario, the MT-PP algorithm planned a longer path comparing with the dashed line path which is the shorter path in the absence of wind. The resulting path of MT-PP in presence of variable wind is longer, but the energy consumption of the drone is lower, compared to the no wind case, because of the presence of tail wind during the path. Moreover, the energy consumption of the dashed line path in presence of variable wind is higher because of the presence of cross wind in half of the path.

TABLE 4. Path length and computation time comparison.

Algorithm	Scenario	Run time [s]	Path length [m]
MT-PP	A	1.18	3507
	B	2.91	4922
	C ($\theta = 135°$)	0.05	1535
	D	1.52	4715
MT-PP + PS	A	1.21	3325
	B	3.01	4655
	C ($\theta = 135°$)	0.07	1418
	D	1.55	4450
MT-PP + IPS	A	1.21	3325
	B	2.94	4610
	C ($\theta = 135°$)	0.05	1418
	D	1.57	4435

In Table 4, the effect of path smoothing methods is investigated. In this table, the path length and the corresponding computation time for the resulting paths of MT-PP algorithm alone (i.e., without any post-smoothing), MT-PP with standard post-smoothing, and MT-PP with our improved post-smoothing method under all the defined scenarios are compared. By construction, the least computation time belongs to MT-PP with no post-smoothing applied, while applying standard PS increases the computation time by 3.4%, in scenario B (as the worst case). Alternatively, applying IPS to MT-PP in scenario B adds up to 1.0% additional time. Regarding the path length, PS and IPS equally shorten the path lengths by 5.1% in scenario A, whereas they shorten the length of paths by 5.4% and 6.3% respectively, in scenario B. Therefore, IPS has always priority over PS in terms of both path length and computation time.

The effect of energy budget is investigated in Table 5. We compared the energy consumption of the MT-PP resulting paths of each iteration in scenario D with the SP, in one part of table, and the output paths of AT-PP in case of setting an energy bound for scenario C ($\theta = 135°$). In scenario D,

TABLE 5. Comparison of energy consumption.

Algorithm & Scenario	Path	Energy Consumption [kJ]
MT-PP + IPS Scenario D	1	74.5
	2	85.8
	3	88.5
	4	93.6
	MT-PP	90.5
	SP	64.6
AT-PP + IPS Scenario C ($\theta = 135°$)	1	31.0
	2	34.2
	AT-PP	36.7
	SP	29.4

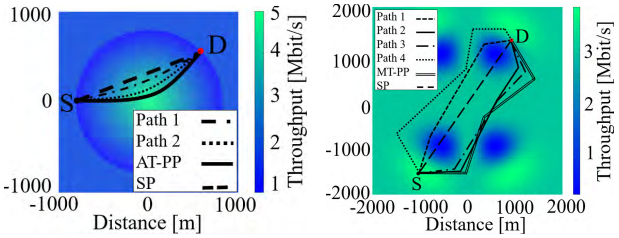

FIGURE 12. The effect of energy budget on the AT-PP (left), and MT-PP (right).

MT-PP finds the path 1 in the first iteration. The energy consumption limit is set to 91 kJ in this experiment. Since the energy consumption is not reached the limit, it continues to increase the threshold up to the path 4 which exceeds the energy limit. Then the threshold is decreased and the final MT-PP path is found. The path of each iteration is shown in Fig. 12 (right). In Scenario C, each resulting path is the output of AT-PP with a specified energy limit. It is obvious in Fig. 12 (left) that as long as we set a greater energy limit, the path will be longer and closer to the high throughput region. It is notable to mention from Table 5 that the energy consumption of MT-PP and AT-PP paths are clearly more than the SP, but they meet the energy budget constraint. In this experiment, the angular rotation speed (ω_{turn}) is considered $2.1 rad/s$, and the power consumed during the rotations (P_{turn}) is considered $225W$. Power consumption of the drone (P) in all the experiments in this work is considered $200W$ [27].

VI. ALGORITHM INTEGRATION INTO THE QGROUNDCONTROL PATH PLANNER

Our approach can be utilized in practice to plan an optimal path for a drone in the real-world situations. As validation, we integrated our algorithms in a popular, open-source, offline path planner like QGroundControl (QGC) station [46]. To achieve a seamless integration, we designed a *proxy* between the QGC and the autopilot. All the data sending from the QGC towards the drone and sending from the drone to the QGC, are passing through the proxy. From the point of view of QGC, the proxy acts as a standard drone; from the point of view of the drone, the proxy acts as QGC. This allows to achieve a transparent behavior, which does not require any modification in QGC and the drone (except for proper configuration settings).

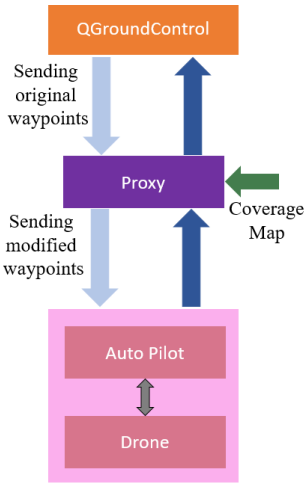

FIGURE 13. Proof of concept for the path planner.

The proxy acts as a server, creating a TCP connection with the QGC, and as a client, creating another TCP connection with the drone. TCP connection is employed to provide reliable transport of drone configuration. Fig. 13 demonstrates the adopted architecture.

In a standard situation (when the proxy is not present), once the user is defining a mission on the QGC, the way-points are transferred to the drone by uploading the mission and the UAV starts the mission as long as a command is received from the QGC side by the user. In this case, the drone flies between each two way-points using the shortest path (a direct path). Now, by considering the proxy between the QGC and the autopilot, all the data sending and receiving from both sides are passing through the proxy. The proxy relays transparently all the data through, except if the data type is MISSION_ITEM. Each MISSION_ITEM contains the GPS coordinates of a way-point and all the other information (e.g., system ID) that are needed by the drone. These data are packed based on the MAVLINK telemetry protocol. Therefore, in the case that data are not way-points or not related to the start or end of way-point transmission, the proxy acts transparently and passes the data equivalent to a pipe. But, if the data are involved in defining way-points, the proxy acts double-faced. Whenever QGC is sending way-points, the proxy acts as the drone. In this case, it sends requests to the QGC for the way-points and receives them one by one and sends the acknowledgment to the QGC when all of the way-points are received. Then, the proxy feeds the original way-points to our algorithms and a new path is planned between each two original way-points. The new path is typically different from the default path (shortest path) and is designed according to our proposed network coverage aware approach. As shown in Fig. 13, the coverage map is fed to the proxy. In the event that the proxy requires to send the new way-points to the drone, it acts as QGC. In this case, the proxy asks for sending the new way-points and answers to the requests of the drone for the way-points. When the

transmission is finished, it receives the acknowledgement from the drone. Now, the drone receives a path which is passing by all the defined way-points and the throughput of the path is maximized considering the on-board energy budget and the effect of wind. Fig. 3 demonstrates the resulting path which is planned by MT-PP algorithm. The orange path is the default path that the drone passes through, in the usual direct connection (i.e., absence of the proxy) with the QGC. The red path is the one which designed by MT-PP and passes through the original way-points and high throughput regions. Depending on the algorithm (MT-PP or AT-PP), the red path may be different.

VII. CONCLUSIONS

Inspired by providing cellular connectivity for UAVs, in order to guarantee the requirements of QoS in the communication for video streaming, we investigated the path optimization problem for an autonomous drone. In the study, the cellular coverage map, the drone budget of energy, and the possible presence of wind is considered. Two algorithms are presented to maximize either the worst-case throughput (MT-PP) or the average throughput (AT-PP) along the path. In addition, a novel path smoothing method outperforming the classical one in both terms of path length and computation time is proposed. It is shown, through a numerical evaluation of various scenarios, that our communication-aware approach leads to ameliorate the throughput of the path comparing with the classical approaches that are absolutely oblivious of the cellular coverage maps, with a profitable impact on video streaming applications. We evaluated the performance of the MT-PP and AT-PP algorithms, in presence and absence of the wind. We investigated the effect of the energy budget on the resulting paths of MT-PP and AT-PP. Finally, we validated experimentally our approach by integrating our algorithms in a popular offline path planner (QGC). We proved that our algorithms can be practically utilized in real-world path planners.

REFERENCES

[1] *Enhanced LTE Support for Aerial Vehicles*, document 3GPP TR 36.777, 2017. [Online]. Available: ftp://www.3gpp.org/specs/archive/36_series/36.777

[2] *Qualcomm and AT&T to Trial Drones on Cellular Network to Accelerate Wide-Scale Deployment*. Accessed: Sep. 2018. [Online]. Available: https://about.att.com/story/qualcomm_and_att_to_trial_drones_on_cellular_network.html

[3] *Study on Enhanced LTE Support for Aerial Vehicles*, document RP-170779, 2017. Accessed: Dec. 14, 2017. [Online]. Available: http://www.3gpp.org/ftp/tsg_ran/tsg_ran/TSGR_75/Docs/RP-170779.zip

[4] A. Mardani, M. Chiaberge, and P. Giaccone, "Communication-aware UAV path planning," in *Proc. 6th IEEE Int. Conf. Wireless Space Extreme Environments (WiSEE)*, Dec. 2018, pp. 12–17.

[5] L. Yang, J. Qi, J. Xiao, and X. Yong, "A literature review of UAV 3D path planning," in *Proc. IEEE WCICA*, Jun. 2014, pp. 2376–2381.

[6] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 1–8.

[7] A. Nash, S. Koenig, and C. Tovey, "Lazy theta*: Any-angle path planning and path length analysis in 3D," in *Proc. 3rd Annu. Symp. Combinat. Search*, 2010, pp. 1–8.

[8] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *Proc. AAAI*, 2007, pp. 1–7.

[9] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proc. ICAPS*, Jun. 2005, pp. 262–271.

[10] X. Song and S. Hu, "2D path planning with dubins-path-based A*algorithm for a fixed-wing UAV," in *Proc. ICCSSE*, Aug. 2017, pp. 69–73.

[11] A. Nash and S. Koenig, "Any-angle path planning," *AI Mag.*, vol. 34, no. 4, pp. 85–107, 2013.

[12] J. Zeng, X. Zhang, and X. Guan, "Path planning for general aircrafts under complex scenarios using an improved NSGA-II algorithm," *J. Comput. Inf. Syst.*, vol. 9, no. 16, pp. 6545–6553, 2013.

[13] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Citeseer, Tech. Rep., 1998.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[15] H. Sharma and P. Balamuralidhar, "A communication approach for aerial surveillance of long linear infrastructures in non-urban terrain," in *Proc. NCC*, Jan. 2015, pp. 1–6.

[16] F. Jiang and A. L. Swindlehurst, "Optimization of UAV heading for the ground-to-air uplink," *J. Sel. Areas Commun.*, vol. 30, no. 5, pp. 993–1005, Jun. 2012.

[17] K. Anazawa, P. Li, T. Miyazaki, and S. Guo, "Trajectory and data planning for mobile relay to enable efficient Internet access after disasters," in *Proc. IEEE GLOBECOM*, Dec. 2015, pp. 1–6.

[18] Z. Han, A. L. Swindlehurst, and K. R. Liu, "Optimization of MANET connectivity via smart deployment/movement of unmanned air vehicles," *IEEE Trans. Veh. Technol.*, vol. 58, no. 7, pp. 3533–3546, Sep. 2009.

[19] S. Kim, H. Oh, J. Suk, and A. Tsourdos, "Coordinated trajectory planning for efficient communication relay using multiple UAVs," *Control Eng. Pract.*, vol. 29, pp. 42–49, Aug. 2014.

[20] F. Jiang and A. L. Swindlehurst, "Dynamic UAV relay positioning for the ground-to-air uplink," in *Proc. IEEE GC Wkshps*, Dec. 2010, pp. 1766–1770.

[21] P. Zhan, K. Yu, and A. L. Swindlehurst, "Wireless relay communications with unmanned aerial vehicles: Performance and optimization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 3, pp. 2068–2085, Jul. 2011.

[22] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.

[23] Q. Wu and R. Zhang. (2018). "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration." [Online]. Available: https://arxiv.org/abs/1801.00444

[24] N. Zhao *et al.*, "Caching UAV assisted secure transmission in hyper-dense networks based on interference alignment," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 2281–2294, May 2018.

[25] F. Cheng *et al.*, "UAV trajectory optimization for data offloading at the edge of multiple cells," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6732–6736, Jul. 2018.

[26] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 42–63, May 2016.

[27] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of UAVs," in *Proc. IEEE ICARSC*, Apr. 2015, pp. 111–117.

[28] N. Bezzo, K. Mohta, C. Nowzari, I. Lee, V. Kumar, and G. Pappas, "Online planning for energy-efficient and disturbance-aware UAV operations," in *Proc. IEEE IROS*, Oct. 2016, pp. 5027–5033.

[29] S. Primatesta, E. Capello, R. Antonini, M. Gaspardone, G. Guglieri, and A. Rizzo, "A cloud-based framework for risk-aware intelligent navigation in urban environments," in *Proc. IEEE ICUAS*, Jun. 2017, pp. 447–455.

[30] D. Palossi, M. Furci, R. Naldi, A. Marongiu, L. Marconi, and L. Benini, "An energy-efficient parallel algorithm for real-time near-optimal UAV path planning," in *Proc. ACM Int. Conf. Comput. Frontiers*, 2016, pp. 392–397.

[31] N. Ganganath, C.-T. Cheng, and K. T. Chi, "Multiobjective path planning on uneven terrains based on NAMOA," in *Proc. IEEE ISCAS*, May 2016, pp. 1846–1849.

[32] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017.

[33] K. Li, W. Ni, X. Wang, R. P. Liu, S. S. Kanhere, and S. Jha, "Energy-efficient cooperative relaying for unmanned aerial vehicles," *IEEE Trans. Mobile Comput.*, vol. 15, no. 6, pp. 1377–1386, Jun. 2016.

[34] E. I. Grøtli and T. A. Johansen, "Path planning for UAVs under communication constraints using SPLAT! and MILP," *J. Intell. Robot. Syst.*, vol. 65, pp. 265–282, Jan. 2012.
[35] Y. Huang, J. Xu, L. Qiu, and R. Zhang. (2018). "Cognitive UAV communication via joint trajectory and power control." [Online]. Available: https://arxiv.org/abs/1802.05090
[36] M. Seleckỳ, P. Váňa, M. Rollo, and T. Meiser, "Wind corrections in flight path planning," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 5, p. 248, 2013.
[37] M. Kothari, I. Postlethwaite, and D.-W. Gu, "UAV path following in windy urban environments," *J. Intell. Robot. Syst.*, vol. 74, pp. 1013–1028, Jun. 2014.
[38] H. G. de Marina, Y. A. Kapitanyuk, M. Bronz, G. Hattenberger, and M. Cao, "Guidance algorithm for smooth trajectory tracking of a fixed wing UAV flying in wind flows," in *Proc. IEEE ICRA*, May 2017, pp. 5740–5745.
[39] T. McGee, S. Spry, and K. Hedrick, "Optimal path planning in a constant wind with a bounded turning rate," in *Proc. AIAA*, 2005, p. 6186.
[40] K. Wu, B. Fan, and X. Zhang, "Trajectory following control of UAVs with wind disturbance," in *Proc. IEEE CCC*, Jul. 2017, pp. 4993–4997.
[41] A. Rucco, A. P. Aguiar, F. L. Pereira, and J. B. de Sousa, "A predictive path-following approach for fixed-wing unmanned aerial vehicles in presence of wind disturbances," in *Proc. Robot 2nd Iberian Robot. Conf.*, 2016, pp. 623–634.
[42] S. U. Ali, M. Z. Shah, R. Samar, and A. Waseem, "Wind estimation for lateral path following of UAVs using higher order sliding mode," in *Proc. IEEE ICISE*, Jan. 2016, pp. 364–371.
[43] C. Thorpe and L. Matthies, "Path relaxation: Path planning for a mobile robot," in *Proc. IEEE OCEANS*, Sep. 1984, pp. 576–581.
[44] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.
[45] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, pp. 533–579, Oct. 2010.
[46] *QGroundControl Drone Control*. Accessed: Oct. 2018. [Online]. Available: http://qgroundcontrol.com/

AFSHIN MARDANI was born in Shiraz, Iran, in 1986. He received the B.S. degree in electronics engineering from Isfahan University, Isfahan, Iran, in 2010, and the M.S. degree in electronics and telecommunications engineering from the Politecnico di Torino, Turin, Italy, in 2014, where he is currently pursuing the Ph.D. degree in electronics and telecommunications. His research interests include the path planning for UAVs and autonomous vehicles, development of path planning algorithms for robots, and connection management in UAVs.

MARCELLO CHIABERGE is currently an Assistant Professor with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy. He is also the Co-Director of the Mechatronics Lab, Politecnico di Torino (www.lim.polito.it), Turin, and the Director and the Principal Investigator of the new Centre for Service Robotics (PIC4SeR, pic4ser.polito.it), Turin. He has authored more than 100 articles accepted in international conferences and journals, and the coauthor of nine international patents. His research interests include hardware implementation of neural networks and fuzzy systems, the design and implementation of reconfigurable computing architectures for hard-real-time control systems, the design and implementation of hybrid control systems based on programmable state-of-the-art devices (DSP and FPGA), and implementation of fault-tolerant communication networks based on plastic optical fibers. Another research field is the design and implementation of non-conventional power stages based on SiC-GaN devices for special functions in automotive, industrial, green energy, and space applications.

PAOLO GIACCONE (M'99–SM'16) received the Dr.Ing. and Ph.D. degrees in telecommunications engineering from the Politecnico di Torino, Torino, Italy, in 1998 and 2001, respectively, where he is currently an Associate Professor with the Department of Electronics. During the summer of 1998, he was with the High Speed Networks Research Group, Lucent Technology-Bell Labs, Holmdel, NJ, USA. From 2000 to 2001, and in 2002, he was with the Information Systems Networking Lab, Electrical Engineering Department, Stanford University, Stanford, CA, USA. His main area of interest is the design of network algorithms.