Online Dynamic Path Planner for UAVs

Manisha Wadhwa[1], Neelima Gupta[2], and Sanjay Madria[3(✉)]

[1] Department of Computer Science, Ram Lal Anand College, University of Delhi, New Delhi, India
mwadhwa@cs.du.ac.in
[2] Department of Computer Science, University of Delhi, New Delhi, India
ngupta@cs.du.ac.in
[3] Department of Computer Science, Missouri University of Science and Technology, Rolla, USA
madrias@mst.edu

Abstract. Unmanned Aerial Vehicles (UAVs) flight path generation that passes through specified waypoints while satisfying spatio-temporal task points (delivering goods or taking aerial pictures) is an important application in battlefield and disaster management among others. Most of the previous work on UAV path planning has been on finding a shortest distance path from source to destination. However, the shortest distance does not always mean fastest route due to congestion or other environmental conditions such as rain, storm, wind etc, passing by aerial objects and no fly zone that may affect the speed/direction of the UAV. Moreover most of these approaches are offline (using A* type algorithms) and hence cannot respond to the dynamically changing environmental conditions when the number of dynamic task points increases. In this paper, we present a greedy Online Dynamic Path Planning algorithm (ODP) considering the dynamically changing requests of tasks, and the environmental conditions. We model the problem as an online multiple choice knapsack problem where-the task points have rewards (earned on completing tasks on the path) and the aim is to maximise the sum of reward points earned on completing the tasks on the path while satisfying the way points with a limit on the total traverse time. To analyze the efficiency of ODP (an online approach), we compare its performance empirically with that of the optimal offline (brute force) algorithm. It was found that ODP provides a competitive ratio (competitive ratio is defined as the ratio of performance of an online algorithm to that of the optimal offline algorithm in the context of algorithm analysis) of 0.8 indicating a near-optimal performance. That is, ODP returns a path that obtains about 80% of the optimal rewards.

Keywords: Path Planning · Knapsack · UAV · Online

1 Introduction

Various industries have emerged to fulfill the demand for services provided by Unmanned Aerial Vehicles (UAVs) such as delivering goods and collecting

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 594, pp. 294–315, 2024.
https://doi.org/10.1007/978-3-031-63992-0_20

imagery data at various locations (e.g., for battlefield and disaster applications). Thus, UAV path planning is an important problem where time-constrained waypoints (in flight path) have been defined and UAVs have to efficiently satisfy the tasks at different locations requested by the ground station while traversing the waypoints. Since the tasks may be spread over a wider space, one needs to plan the path of the UAV in such a way to finish as many tasks as possible satisfying the waypoints constrained with location and time.

UAV path planning is one of the most fundamental problems addressed in the literature [6,10,17,20,21]. Most of the previous work on UAV path planning has been on finding a shortest distance path from source to destination using A* algorithm [8]. However, the shortest distance does not always mean fastest route due to congestion or other environmental conditions such as rain, storm, wind etc, passing by aerial objects and no fly zone that may affect the speed/direction of the UAV from time to time. Improved A^* [21] handles bad weather conditions predicted in advance using climate models and consider them as no-fly zones and thus do not perform well in real-time (still an off-line scheme). In this paper, we present an online algorithm considering the dynamically changing requests of tasks, and the environmental conditions where the bad environmental conditions (which can affect the speed of the UAV) are encountered on-the-go. Hence, an off-line algorithm that provides a distance-efficient path cannot be modified to obtain a time-efficient path.

[12] maximizes the number of tasks performed, given a trajectory defined by a sequence of waypoints without deviating too much from the trajectory. They do capture task priorities but they do not consider dynamically changing environmental conditions as their approach is offline in the sense that they solve an optimization problem with the pre-defined waypoints and the newly generated task points as inputs. They have considered their problem in open space meaning UAVs can reach to any point.

In this work, each task has some reward value that one gets on completing the task. Thus, we consider the problem of finding a path, following a trajectory defined by the waypoints, that maximizes the rewards associated with tasks. We would like to consider such task points so long as they can be covered within the time constraint. A weighted directed graph is used to establish an initial structured representation of the environment, with vertices representing the three dimensional waypoints and task points and, edges representing the connectivity between them and the weights on the edges represent the distance between the two end points; the UAV can travel only along the edges of the graph.

Sometimes a task point is difficult to reach as it may be located in a congested area or in a difficult terrain. Thus, we also associate a quality-of-task (QoT) with each task, $0 \leq QoT \leq 1$; a smaller value indicating difficult to reach task point. A point located in such an area may cost us more in terms of fuel consumption or take more time and we may want to avoid it even if it has high rewards.

Formally, we model the problem as a multiple-choice Online Knapsack Problem [15] where we have a set of task points with associated reward points r_i and the time to complete the task tc_i with a knapsack capacity of kc on the total time taken by the UAV on completing all the tasks. In addition to the

time constraint, other constraints such as turning angle and no-fly zone have also been considered in this work. At every node v, for each outgoing edge, we check the feasibility in terms of turning angle and no-fly zone. For each feasible edge (v, w), we compute the speed with which the UAV can fly depending upon several environmental factors such as weather conditions, other aerial objects crossing the path to avoid collision and, difficult terrain. Thus the time to reach does not only depend upon the distance of w from v (given by the weight on the edge (v, w)) but also on the dynamically changing environmental conditions. We then greedily visit a feasible out-neighbour w of v that maximizes r_w/t_w, where t_w is the sum of the time to reach the task point and the time to complete the task. The algorithm works like a multiple-choice online algorithm in the sense that, at a node v, only the task points that are out-neighbours of v are added to our known set of task points and we have to select the best from them.

To the best of our knowledge, this is the first work that considers the dynamic UAV path planning by modeling it as a greedy algorithm using the online multiple-choice knapsack problem [5], referred to as ODP (Online Dynamic Path Planning). The online aspect of the problem comes from the sequential presentation of tasks and the algorithm has to choose at most one task point from the set, where the decision is based upon the current state of the knapsack and the available information at that moment. We do not have the prior knowledge of future tasks when making decisions about the current task. As a result, the algorithm makes irrevocable choices based on incomplete information and the decision to include a task in the knapsack must be made immediately.

Based on the experimental results, the ODP algorithm is shown to provide a competitive ratio of about 0.8 when compared to the brute force approach which suggests that the online algorithm (ODP) is able to achieve a solution that is very close to the offline algorithm (brute force) for a wide range of input scenarios. This implies that the online algorithm is effective in completing high reward tasks while operating in real-time or in online settings where complete information is not available in advance.

2 Related Work

[8] introduced the A* search algorithm for graph traversal by combining elements of Dijkstra's algorithm and heuristic search to find optimal path while considering both the cost to reach a node and an estimate of the remaining cost to the goal but the trade-off between the optimality and computational efficiency is not explicitly explored. [3] proposed a two step path-planning algorithm for UAVs using Voronoi polygons [7] by assigning length cost to edges. This work does not address the practical constraints relevant to UAV path planning, such as energy consumption, uncertainties or mission-specific requirements. [16] presented two methods of path planning of UAV using A* algorithm, and genetic algorithm where the path is composed of the lines on the Voronoi diagram with the edge cost. The UAV's turning angle constraint was considered, path was smoothed by a geometry method. This work lacks the comparison with existing approaches

to determine the effectiveness of proposed approach. [22] used Ant Colony Optimization (ACO) algorithm for UAV path planning by dividing flying area into grids and optimizing path between grid point and destination point. Their simulation in two dimension proved that ACO can effectively and quickly accomplish UAV path planning but since it includes a series of connected points, it still need smoothness processing in order to be applied for real UAV flying. [17] relates to A* type algorithm in finding a path from source to destination keeping no fly/exclusion zones and turning angle in consideration while addressing the issue of (deconfliction) avoidance of collisions or conflicts with other UAVs, obstacles, or airspace regulations. They formulate the problem as a multi-objective optimization task, where the objectives include finding the shortest path while ensuring deconfliction with other UAVs and obstacles in three dimension. They have not discussed about the performance of the approach developed by them moreover it can not be used in real world UAV path planning as it does not take environmental conditions into account.

[1,10,11] and [13] provide a comprehensive review of path planning techniques for unmanned aerial vehicles (UAVs) and discuss their strengths, limitations, and challenges. They also discuss the impact of various factors such as environment modeling, obstacle avoidance, optimization criteria, and real-time considerations on path planning algorithms. They also provide insights into different path planning algorithms commonly used in UAV applications.

[6] proposed A* type algorithm by adding direction parameter considering the barriers between source and destination and focused on finding an optimal flight path in two dimension. This work does not consider the flight parameters and environmental constraints such as limited or no flying range into account.

Some researchers [6,17,21] have attempted to find a paradigm using A* algorithm to find cost optimum path from source to destination but it finds the result in exponential time; not feasible for real time scenarios. Also, the constraints - QoS, trajectory, uncertainties like thunder, rain, lightning etc., variable speed, limited driving range are not taken into account. Some researchers [12] have not considered these constraints but finds a path by going forward and choosing the node that seems best at that particular point of time. It would give a solution in a feasible time adhering to real time constraints on total time due to limited driving range, QoS, angle component and timestamps.

3 System Model and Nomenclature

Here we discuss the system model, basic terminology and notations used.

Task Points: Each task point is a 3D point where UAV performs some task (such as delivering some goods or clicking aerial pictures). Thus, a task point is specified by the tuple $<id, x, y, z, r, tc, QoT>$ where id = task id, x = latitude, y = longitude, z = altitude, r = reward associated, tc = Time to complete (in seconds) and QoT = Quality of Task point as explained below.

Reward: Each task point tp has a reward r associated with it ($tp.r$). For example, in case of delivering food packets, it could mean a bigger order; for a TV

channel collecting pictures from a prime event may fetch them more viewership leading to higher Television Rating Point (TRP).

Quality of Task Point: Each task point tp also has quality of accessibility QoT associated with it ($tp.QoT$). It refers to the accessibility and reachability of the task points. An easy-to-reach task point has higher value of QoT as compared to another task point that is difficult to reach. It takes values in the range from 0 to 1. [19] discusses that the locations lying in the open space can be easily covered by UAV in comparison to congested areas. We have assigned QoT values with different numerical values. Interpretation is as follows:

0.05: Difficult Terrain - task point in an area with challenging terrain.

0.10: Congested Areas - task point in a heavily congested area, which may pose challenges for navigation and maneuvering.

0.15: High-rise areas - task point in an area with tall buildings or structures, potentially affecting line-of-sight communication or requiring special consideration for aerial navigation.

0.20: Delivery point on the main road - task point corresponds to a delivery location on a primary road or a major intersection.

0.40: Delivery point in streets - task point represents a delivery location situated within smaller streets or narrower pathways.

0.50: Metropolitan Areas - task point in a densely populated urban area, typically associated with cities or metropolitan regions.

0.60: Delivery point on a terrace - task point corresponds to a delivery location situated on the rooftop or terrace of a building.

1.00: Task point in Open Area - task point located in a region of high quality of service, typically indicating an open area without significant obstacles.

Waypoints: 3D points which define the flight path trajectory. Each waypoint is defined by the tuple $<id, x, y, z, ts>$ where id = waypoint id, x = latitude, y = longitude, z = altitude, ts = timestamp. Time stamps are defined as $[yyyy, mm, dd, hh, mn, ss, ms]$ where $yyyy$ = year, mm = month, dd = day, hh = hour, mn = minutes, ss = seconds, ms = milliseconds. Each waypoint has time stamp associated with it that need to be covered on or before by the UAV.

Angle Component (θ): Let θ be the angle by which UAV needs to turn to reach a task. The angle between vector **ab** and **bc** is described as:

$$\theta = \arccos\left(\frac{\mathbf{ab} \cdot \mathbf{bc}}{|\mathbf{ab}||\mathbf{bc}|}\right)$$

where $\cdot$ denotes the dot product between vectors **ab** and **bc**. arccos is the inverse cosine function. $|\mathbf{ab}|$ and $|\mathbf{bc}|$ represent the magnitudes (lengths) of vectors **ab** and **bc** respectively. A UAV has a limit on the angle by which it can rotate. Thus, a task point is feasible if $\theta \leq \tau$. The value of θ must be lesser in order to provide a smooth path and at the same time it should allow the UAV to explore a wide variety of task points. Hence an optimum value of τ is chosen to be 60° in order to balance the trade-off as discussed in [4] and [14].

Uncertainty Parameter (μ_e): At any point of time, μ_e denotes if one or more bad weather condition is present along an edge e or if the weather is clear along e.

[18] lists down various weather conditions affecting the speed of UAV according to their severity. According to the paper, thunderstorm affects the UAV most severely whereas extreme temperatures may not pose immediate danger to the UAV. A high value of μ_e thus may denote the presence of thunderstorm on e whereas a low value may denote extreme temperatures and a value of 0 denotes clear weather. We assign following values to various weather conditions according to their severity as discussed in [18]:

0.95: Thunderstorm - It can impact visibility and may pose risks to the operation of UAV.

0.90: Fog/Clouds - presence of haze or fog in the area can reduce visibility and impact the operation of UAV.

0.85: Snow - Icing can affect the UAVs aerodynamics and performance, potentially leading to control problems and instability.

0.80: Rain and Hail - hail and rain in the area can cause damage to UAV and pose risks to safety.

0.70: Windy - Strong winds can affect a UAV's stability and control.

0.60: Passing by aerial vehicle - presence of another aerial vehicle in the vicinity may introduce additional uncertainity or safety concerns.

0.50: Moisture and Humidity - Exposure to moisture and water can cause the drone's flight path to be disrupted due to potential electronics malfunctions or loss of control.

0.40: Temperature Extremes - Extreme cold or heat can impact the UAVs battery performance and overall functionality, but it might not immediately pose as much danger as other conditions.

0.00: No significant uncertainties - no adverse conditions present in the area.

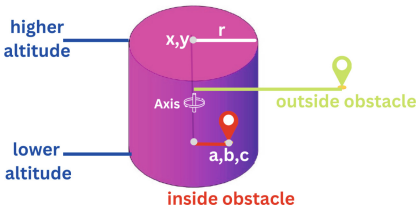


Fig. 1. Obstacle/No-fly zone

Speed (u_x): A UAV flies at the maximum attainable speed (mentioned by [2] and [9], and similar predetermined value used in our simulations) and it's speed is affected by some environmental condition or passing by vehicle on the flight path. We assume that the UAV changes its speed only at the task points and waypoints and not along the edges. Let u_x denote the speed of the UAV at location x i.e., while leaving the location x. Note that the effective speed of the UAV along an edge may be lesser in presence of bad environmental conditions. We denote the effective speed along an edge e by $effu_e$. Then $effu_{(x,j)} = u_x * (1 - \mu_{(x,j)})$.

Obstacles/No-Fly Zones (NFZ): Exclusion area where the UAVs can not fly. For example the airspace around airports is NFZ for UAVs to prevent any interference with manned aircrafts take off and landing. We have simulated the obstacles with the help of 3D cylindrical shapes. A UAV is not allowed to choose a node which is present inside the 3D shaped cylindrical obstacle (Fig. 1). Let $NFZ = \{NFZ_1, NFZ_2, NFZ_3....NFZ_m\}$ be the set of m no-fly zones known before the mission where NFZ_i is a vertical cylinder specified by its higher altitude, lower altitude, axis and radius. Axis is specified by its x and y coordinates. Thus $NFZ_i = <xa, ya, radius, la, ha>$ for all $i = 1, 2, \ldots, m$ with $xa = x$ coordinate of the axis, $ya = y$ coordinate of the axis, $radius =$ radius of cylinder, $la =$ lower altitude of cylinder and $ha =$ higher altitude of cylinder. A task point is feasible if it does not lie in the no fly zone.

4 Problem Statement

We can formally define our problem via tuple $<g,\ s,\ d,\ r,\ tc,\ ts,\ kc,\ \mu_e,\ QoT,\ NFZ>$ where $g = (v, e)$ is a graph with v as the set of task points and waypoints in $3D$ space. e is the set of edges. s is the source, d is the destination and time constraint kc to reach the destination is the knapsack capacity. NFZ is the set of no-fly zones known before-hand. Each task point is associated with reward r, Quality of task point QoT and time to complete tc. Each way point is associated with the timestamp ts. Each edge has environmental uncertainty value μ_e associated with it, which in turn affects the speed of the UAV. Aim is to find a path from s to d such that it maximizes the reward (by finishing the tasks on its route) and takes no more than kc units of time from s to d. Waypoints define the trajectory of the path and a UAV has to cover the waypoints on the path on or before their respective timestamps to reach. The source and the destination are also considered as waypoints with timestamps 0 and kc respectively. We assume that there is a path from each task point to the destination following the given trajectory. In this paper, we have considered the above problem in an online setting. i.e. the task points, the way points and the edges arrive online. At any point of time, we are exposed to the next waypoint along with its time stamp, a set of task points along with their attributes and a set of edges along with the environmental conditions on its route. The next waypoint is exposed only when we have reached the previous waypoint.

4.1 Proposed Online Dynamic Path Planning Approach

We propose Online Dynamic Path Planning (ODP) approach modelled as an optimization problem using knapsack paradigm given that:

$$\textbf{Maximize: } \sum_{i=1}^{n} r_i \cdot s_i$$

$$\textbf{Subject to: } \sum_{i=1}^{n} t_i \cdot s_i \leq kc$$

$$s_i \in \{0, 1\} \quad \forall i = 1, 2, \ldots, n$$

each task from a set of n task points is associated with time t_i, the sum of time to reach the task point from the current location, and the time to complete the task, and reward r_i. Let s_i be a binary decision variable that indicates whether task i is selected ($s_i = 1$) or not selected ($s_i = 0$). We need to choose a subset of task points that maximize the reward such that knapsack capacity is satisfied. Its a classical knapsack problem where each task is selected at most once.

4.2 ODP Approach

An online greedy multiple-choice approach is used to solve the knapsack problem with the knapsack capacity kc. Suppose we are at location x called the current location at any point of time. The current location x can be represented by the tuple $<T_x, w_x, \mu_e>$ where T_x is the set of task points that are out-neighbours of x; some or all of them may become available online; edges to the vertices in T_x are available online, w_x is the next waypoint to reach from x; this may be same as that at the previous location if x is a task point or may become available online if x is a waypoint. u_x denote the speed of the UAV at location x i.e., while leaving the location x. μ_e denotes one or more bad weather condition or if the weather is clear along an edge e.

We consider the task points in T_x, and extract the set of feasible points from them and decide to visit the task point tp that gives maximum efficiency. A task point is considered feasible if (i) $\theta \leq \tau$, (ii) it does not lie in a no fly zone (iii) UAV can reach the next way point within the time limit.

Efficiency of a feasible point is computed as reward per unit time. However, while computing the reward, we also consider the ease of reaching a task point. A task point that is hard to reach may consume more fuel and hence is less useful for us. Thus, we compute what we call as the effective reward denoted by $effr_{tp}$. Effective reward of a task is the product of the reward and the quality of the task. i.e. $effr_{tp} = tp.r * tp.QoT$; lower value of QoT indicates that it is more difficult to reach the task point thereby reducing the significance of the task point for us. Time is the sum of time-to-reach ($t_{(x,tp)}$) and time-to-complete ($tp.tc$) a task point. Thus, the efficiency of a task point tp is:

$$eff_{tp} = \frac{effr_{tp}}{tp.tc + t_{(x,tp)}}$$

The environmental uncertainties can affect the effective speed of the UAV and hence the time-to-reach a task point. Effective speed, denoted by $effu_{(x,tp)}$ is computed as $effu_{(x,tp)} = u_x * (1 - \mu_{(x,tp)})$ indicating the impact of higher uncertainties on slowing down the UAV. And, $t_{(x,tp)} = \frac{d_{(x,tp)}}{effu_{(x,tp)}}$ where $d_{(x,tp)}$ is the distance to reach from x to tp (calculated as the sum of horizontal and vertical distance, where the vertical distance is calculated as the absolute difference of the altitude of two $3D$ points and the horizontal distance is calculated as the Euclidean distance between them in a horizontal plane.

Whether a task point lies in a no fly zone is checked as follows: If the altitude of the task point is greater than the lower altitude and lesser than the higher

altitude of the cylinder, then we calculate the distance of the task point from the axis. If the distance calculated is less than the radius of the cylinder, then the task point lies in the no fly zone otherwise the point doesn't lies in the no fly zone. For a task point tp, we also estimate the time to reach the next way point via tp. This is the sum of time-to-reach the task tp from the current location, time-to-complete the task tp and time to reach the waypoint from tp i.e. $(t_{(tp,w_x)})$ where $t_{(tp,w_x)} = \frac{d_{(tp,w_x)}}{effu_{(tp,x_x)}}$. We say that the task point is feasible if this estimate is at most the remaining available time.

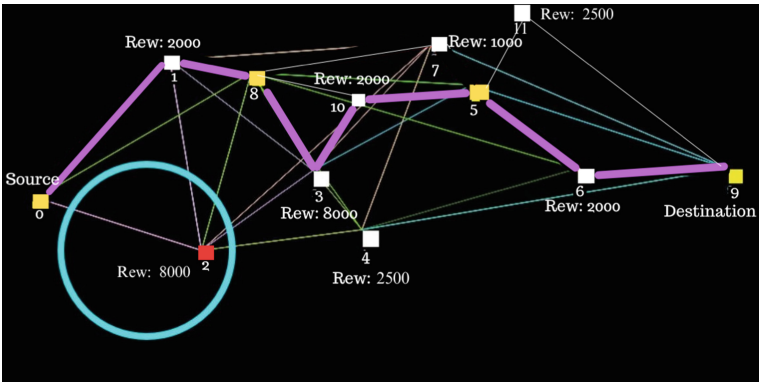


Fig. 2. A Simulated Graph (Color figure online)

Figure 2 depicts a simulation graph using some synthetic data where node numbered 0 is the source and node numbered 9 is the destination. The task points are coloured white, and waypoints are coloured yellow. The no-fly zones is represented by blue coloured circle. The purple line is the flight path taken by UAV to move from source to destination. The path taken by UAV is 0− >1− >8− >3 − >10− >5− >6− >9. We can see that the UAV started from node 0 and since node 2 was in no-fly zone, so it covered node 1 with reward 2000 and reached the waypoint (node 8). Then it has node 3, node 7 and 10 as choices. So UAV deviated in order to fetch a good reward of 8000 instead of covering node 10 (that was nearer) with reward 2000. At node 3, 4 and 10 are the available choices. 4 turns out to be infeasible as the time to reach 5, the next waypoint, via 4 is more. Since 10 turns to be a feasible task point, node 10 is covered before moving to 5. At node 5, 6 and 11 are the available choices. Since for node 11 turning angle is more than τ. So, node 6 is covered before reaching destination.

5 ODP Traversal Algorithms

In order to obtain the flight path in real-time, an online multiple-choice algorithm modeled as Knapsack is developed based on a greedy approach where the path is generated without going back to the previously visited nodes thus eliminating the overhead of repeated task of path planning from a certain set of points.

Algorithm 1 describes our greedy approach where starting from the current location x at source (line 2), it gets the next set T_x of task points (line 6) along with the uncertainties μ_e (line 7) on their way from x and uses Algorithm 2 (line 10) to obtain the most efficient task point (with maximum rewards per unit time) from them. It moves to the most efficient task point (line 12) with the remaining knapsack capacity. We remember the previous node as we move to the next node (lines 3 and 11); this is used to compute the angle of rotation required to reach next location. In case x is a waypoint, we also get the next waypoint w_x in line 9. The process is repeated (line 5) until we reach the destination.

Algorithm 2 finds the most efficient out-neighbour of the current node x considering flight parameters (turning angle at lines 6, 7), no-fly zones (line 8) and environmental constraints. At lines 9 and 10, it computes the impact of environmental uncertainties on the effective speed of the UAV. In addition to the time-to-reach and time-to-complete a task point, we also estimate the time-to-reach the next waypoint from the task point under the impact of environmental uncertainties (lines 13 and 14). If the total time is within the available time (line 15), we compute the efficiency (reward per unit time, line 16) of the task point. Lines 16–20 maintain the efficiency and time-to-reach the most efficient task point. At line 27, we compute the time that will be consumed in reaching the most efficient task point and completing it; this time is subtracted from the knapsack capacity to compute the remaining available time. If no feasible out-neighbour of x is found (line 29), then we go to the next waypoint.

Algorithm 1 : Online Dynamic Path Planner for UAVs
Input : s, d, Knapsack Capacity kc
Output : Path from s to d

1: **procedure** ODP(s, d, kc)
2: $x \leftarrow s$
3: $prev \leftarrow NULL$
4: $resKC \leftarrow kc$
5: **while** $(x! = d)$ **do**
6: getT_x
7: get μ_e for all T_x
8: getu_x
9: If $x \in W getnextw_x$
10: $(maxeffpt, resKC) \leftarrow mep(x, T_x, w_x, \mu_e, u_x, prev, resKC)$
11: $prev \leftarrow x$
12: $x \leftarrow maxeffpt$
13: **end while**
14: **end procedure**

Algorithm 3 calculates the distance between any two 3D points $p1$ and $p2$ as the sum of horizontal distance (calculated as the Euclidean distance between the two points (line 3) and vertical distance (calculated as the absolute difference of the altitude of two points (line 2).

Algorithm 2 : Most Efficient Point(MEP)
Input : current node x, T_x, w_x, μ_e, $prev$, Knapsack Capacity kc
Output : returns the next maximum efficient point

1: **procedure** MEP(x, T_x, w_x, μ_e, u_x, $prev$, kc)
2: $current \leftarrow x$
3: $maxeffpoint \leftarrow null$
4: $maxefficiency \leftarrow 0$
5: **for** tp `in` T_x **do**
6: $\theta \leftarrow angle(prev, current, tp)$
7: **if** $\theta \leq \tau$ **then**
8: **if** $!(inNFZ(tp, NFZ))$ **then**
9: $effu_{(x,tp)} \leftarrow (u_x * (1 - \mu_{(x,tp)}))$
10: $effu_{(tp,w_x)} \leftarrow (u_x * (1 - \mu_{(tp,w_x)}))$
11: $d_{(x,tp)} \leftarrow distance(current, tp)$
12: $t_{(x,tp)} \leftarrow \frac{d_{(x,tp)}}{effu_{(x,tp)}}$
13: $d_{(tp,w_x)} \leftarrow distance(tp, w_x)$
14: $t_{(tp,w_x)} \leftarrow \frac{d_{(tp,w_x)}}{effu_{(tp,w_x)}}$
15: **if** $tp.tc + t_{(x,tp)} + t_{(tp,w_x)} \leq kc$ **then**
16: $effr_{tp} \leftarrow tp.r * tp.QoT$
17: $eff_{tp} \leftarrow \frac{effr_{tp}}{tp.tc + t_{(x,tp)}}$
18: **if** $eff_{tp} > maxefficiency$ **then**
19: $maxeffpoint \leftarrow tp$
20: $maxefficiency \leftarrow eff_{tp}$
21: $t \leftarrow t_{(x,tp)}$
22: $tpc \leftarrow tp.tc$
23: **end if**
24: **end if**
25: **end if**
26: **end if**
27: **end for**
28: $usedCapacity \leftarrow t + tpc$
29: **if** $maxeffpoint == null$ **then**
30: $maxeffpoint \leftarrow w_x$
31: $d_{(x,w_x)} \leftarrow distance(x, maxeffpoint)$
32: $effu_{(x,w_x)} \leftarrow (u_x * (1 - \mu_{(x,w_x)})$
33: $t_{(x,w_x)} \leftarrow \frac{d_{(x,w_x)}}{effu_{(x,w_x)}}$
34: $usedCapacity \leftarrow t_{(x,w_x)}$
35: **end if**
36: $resCapacity \leftarrow kc - usedCapacity$
37: **return** $maxeffpoint, resCapacity$
38: **end procedure**

Algorithm 4 is used to calculate the angle between any three data-points. Vectors ab and bc are obtained by using the coordinates of a, b and c respectively (lines 2 and 3). Magnitudes (lengths) of vectors ab and bc is calculated using the Euclidean distance formula (lines 4 and 5). The inverse cosine ($acos$) function is

calculated using the dot product and magnitude of the vectors and is converted to degrees (line 6).

Algorithm 5 checks if a task point lies in the no-fly zones. If the altitude of the task point is greater than the lower altitude la and lesser than the higher altitude of the cylinder ha (line 6), then we calculate the distance of the task point from the axis (line 12). If the distance calculated is less than the radius of the cylinder (line 13), then the task point lies in the no fly zone otherwise not. The same step is repeated for each no fly zone (line 2).

6 Experimental Set up

This section explains the numerical simulations performed, experimental analysis, results obtained and the performance of the proposed algorithm.

6.1 Numerical Simulations

We have taken a canvas of 1500 * 800 for all the experiments. The coordinates of the points range from 10 to 1400 pixels for the x-axis, 10 to 700 pixels for y-axis and from 3 to 17 for the z-axis. The task points are randomly distributed over the region to simulate the task points covering various cities. We conducted our experimental studies on two types of synthetic data sets:

- 10 data sets, each consisting of 10 randomly generated $3D$ points (first type of data set - used to compare the performance of ODP with brute force).
- 10 data sets, each consisting of 400 randomly generated $3D$ points (second type of data set - to study the effect of parameters under different scenarios).

Each data set is in the form of a graph where the next waypoint and set of tasks appear in the online fashion as explained in the approach i.e. the tasks can be issued anytime. For both types of the data sets, each set of data points, edges and other input parameters are generated randomly from the values as explained next: the values of reward for the task points are in the range from 100 to 1000, for tc the values are taken between 100 and 1000 s, the QoT values, the value of τ and the values of the uncertainty parameters are chosen from the values as discussed in Sect. 3. For the scenarios considering variable speed, we have taken the value of speed (u_x) in the range: $Smin$ is the minimum limit on the UAV speed i.e. 3.6 km/hr and $Smax$ is the maximum limit on the UAV speed. i.e. 18 km/hr. Points with the smallest and the largest x-coordinates are chosen as the source and the destination respectively.

Algorithm 3 : To calculate distance
Input : Two 3D points $p1$ and $p2$
Output : Distance between $p1$ and $p2$

1: **procedure** DISTANCE($p1.x, p1.y, p1.z, p2.x, p2.y, p2.z$)
2: $vd \leftarrow abs(p2.z - p1.z)$
3: $hd \leftarrow \sqrt{(p2.y - p1.y)^2 + (p2.x - p1.x)^2}$
4: **return** $vd + hd$
5: **end procedure**

The ids for task points and waypoints are the unique numbers from 0 to 9 for the first type of data set and from 0 to 399 for the second type of data set.

6.2 Comparison with Offline Optimal

We have used brute force algorithm to obtain a feasible path with the maximum reward points offline. Brute force algorithm finds all the paths from the source to the destination and from there, we select all the feasible paths (those paths which covers all the waypoints including the destination, at the given time and order). From all the feasible paths, we select the best path (i.e., having maximum reward). In case we have one or more such path, one path is selected arbitrary. Thus, brute force approach gives the optimal solution. The path obtained by ODP for all the sixteen scenarios on ten different data sets is compared with the brute force.

Algorithm 4 : To calculate angle
Input : Any three 3D points
Output : Angle between the three 3D points

procedure ANGLE(a, b, c)
  $ab \leftarrow [b.x - a.x, b.y - a.y, b.z - a.z]$
  $bc \leftarrow [c.x - b.x, c.y - b.y, c.z - b.z]$
  $|\mathbf{ab}| \leftarrow sqrt(ab[0] * ab[0] + ab[1] * ab[1] + ab[2] * ab[2])$
  $|\mathbf{bc}| \leftarrow sqrt(bc[0] * bc[0] + bc[1] * bc[1] + bc[2] * bc[2])$
  $\theta \leftarrow \arccos\left(\frac{\mathbf{ab} \cdot \mathbf{bc}}{|\mathbf{ab}||\mathbf{bc}|}\right)$
  $a \leftarrow \theta * 180.0/3.141592653589793$
  return a
end procedure

We have used the following metrics to compare the performance of our approach with the brute force algorithm: competitive ratio, reward per unit traverse time, reward per unit distance traveled, reward per unit computational time and theoretical time complexity. The values reported are average over the 10 data-sets for each scenario.

Competitive Ratio: Let *opt* denote the reward obtained by an optimal offline algorithm or the benchmark solution that has access to complete information about the input sequence in advance. Let *alg* denote the online algorithm that makes decisions without having complete knowledge of the future input. For any input sequence, let $alg(i)$ represent the cost or performance achieved by the online algorithm *alg* on input sequence i, and let $opt(i)$ represent the cost or performance achieved by the optimal offline algorithm *opt* on the same input sequence i. The competitive ratio, $cr(alg)$ of *alg* with respect to *opt* is given as:

$$cr(alg) = sup(\frac{alg(i)}{opt(i)})$$

for all possible input sequences i where *sup* denotes the supremum or the least upper bound. Simply, the competitive ratio measures the performance of the

Algorithm 5 : To find out if a task point lies in no-fly zones
Input : Task point tp, NFZ
Output : returns true if the task point lies in a no-fly zone

```
1: procedure inNFZ(tp, NFZ)
2:     for i in NFZ do
3:         if tp.z ≥ i.la && tp.z ≤ i.ha then
4:             distance ← sqrt((tp.x − i.x)^2 + (tp.y − i.y)^2)
5:             if distance ≤ i.radius then
6:                 return true
7:             end if
8:         end if
9:     end for
10:    return false
11: end procedure
```

online algorithm compared to the optimal offline algorithm over all possible input sequences. It quantifies the ratio of the cost or performance of the online algorithm to the cost or performance of the optimal offline algorithm.

Our Results: Table 1 shows that in more than 55% of the scenarios (i.e., 9 out of 16 scenarios), ODP gives a competitive ratio of more than 0.9 i.e. in more than 55% scenarios our algorithm is able to fetch about 90% of the maximum obtainable rewards. In all the sixteen scenarios, ODP gives a competitive ratio of more than 0.8. i.e. in all the scenarios our algorithm is able to fetch more than 80% of the maximum obtainable rewards. It can be observed that in presence of the restriction of turning angle, the competitive ratio achieved is less than 0.9 whereas in all other cases it is more than 0.9. Thus, our approach handles the environmental uncertainties with a competitive ratio of more than 0.9 in the absence of restriction imposed on the turning angle and with a competitive ratio of more than 0.8 in the presence of restriction imposed on the turning angle.

Reward/Traverse Time: Though brute force gives more reward points than our algorithm, our approach provides faster routes in more than 65% of the cases. Since time is important to save fuel consumption, reward obtained per unit of fuel spent (in terms of per unit of time) is an important parameter to compare any two approaches. In more than 60% of the scenarios reward per unit traverse time is more in our case as compared to that of the brute force as displayed in Fig. 4.

Reward/Traverse Distance: In 50% of the scenarios, the traverse distance of the ODP approach is lesser than that of brute force and the reward per unit traverse distance is comparable in all the scenarios as displayed in Fig. 5 (of 10 data-sets).

Reward/Runtime: ODP always have hundred times less run time than brute force and always have ten times higher reward per unit run time as displayed in Fig. 6 (of 10 data-sets).

Table 1. Comparison of reward - ODP vs. Brute Force

SNo.	Scenario Type	Reward_ODP	Reward_BF	Competitive Ratio
1	No parameters	53855.2	53855.2	1
2	Turning angle (TA)	40673.9	46293.6	0.8786
3	Environmental uncertainties (EU)	53706.8	53855.2	0.9972
4	QoT	53706.8	53855.2	0.9972
5	Variable speed (VS)	53706.8	53855.2	0.9972
6	TA and EU	39412.9	45723.4	0.8619
7	TA and QoT	38714.7	46293.6	0.8362
8	TA and VS	40426.2	46293.6	0.8732
9	EU and QoT	53706.8	53855.2	0.9972
10	EU and VS	50069.0	53855.2	0.9296
11	QoT and VS	53706.8	53855.2	0.9972
12	TA, QoT and VS	39143.9	46293.6	0.8455
13	TA, QoT and EU	40086.6	45723.4	0.8767
14	EU, VS and TA	37938.1	42269.0	0.8975
15	QoT, VS and EU	48869.8	53855.2	0.9074
16	All parameters	38222.7	42269.0	0.9042

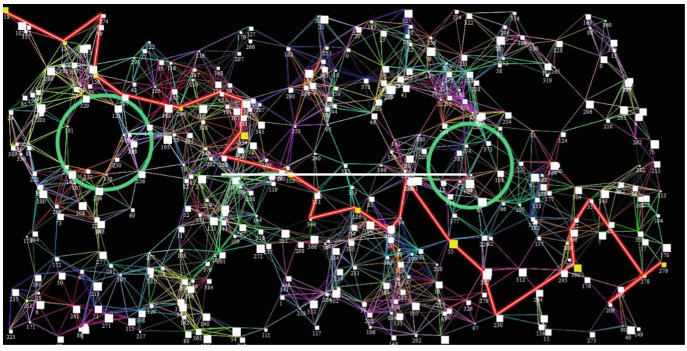


Fig. 3. Flight Path of UAV for 400 Data-points (Red line is flight path), White nodes: task points, yellow nodes: waypoints, Green circles: No Fly Zones (Color figure online)

Theoretical Time Complexity: ODP does constant amount of work along each edge of the graph leading to a complexity of $O(n^2)$ even if the graph is complete whereas as the name suggest, brute force tries out all the paths and selects the one with the maximum rewards leading to exponential time complexity.

Thus, our algorithm is not only dynamic in the sense that it responds to the dynamically changing weather conditions but is also time-efficient obtaining about 80% of the rewards most of the time.

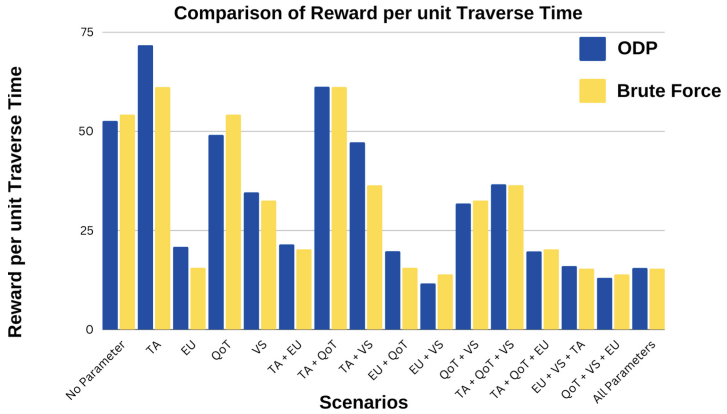


Fig. 4. Reward per unit Traverse Time - Brute Force vs. ODP

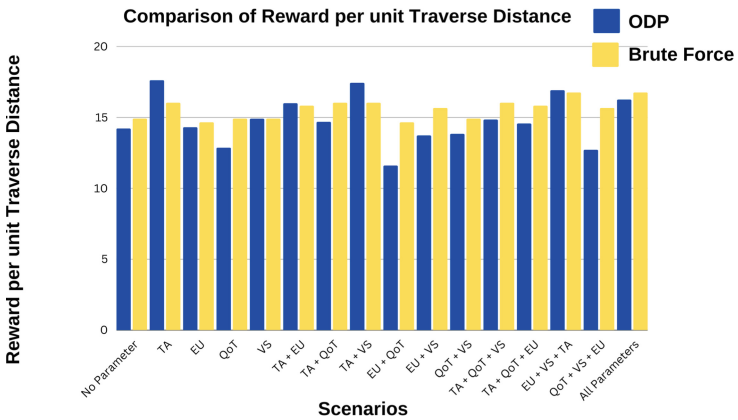


Fig. 5. Reward per unit Traverse Distance - Brute Force vs. ODP

6.3 Experimental Analysis

To study the effect of parameters under different scenarios, we performed simulations on 10 randomly generated data sets each of 400 points (the second type of data-set). Figure 3 shows the simulation for the same. We have used the following metrics for the doing the detailed analysis: reward, reward per unit traverse time, reward per unit distance traveled and reward per unit computational time. The values reported are average over the 10 data-sets for each scenario.

Reward: Figure 7 shows that the total reward earned is getting decreased with the increase in parameters in different scenarios. It can be observed that ODP handles environmental uncertainties with 70.74% of reward as compared to scenario 1 in the absence of restriction imposed on the turning angle and with decrease in the reward by 57.67% in the presence of restriction imposed on the

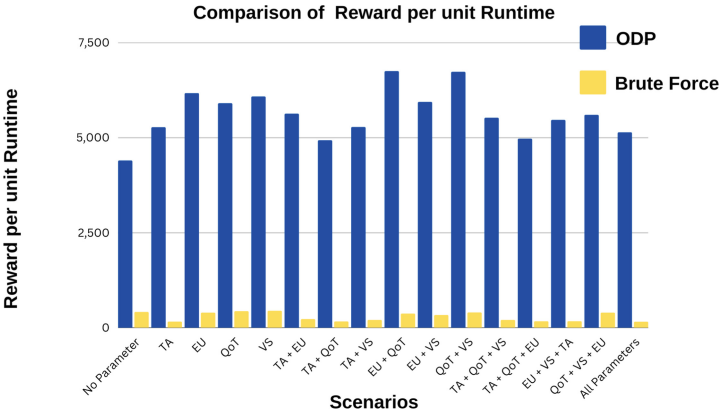


Fig. 6. Reward per unit Runtime - Brute Force vs. ODP

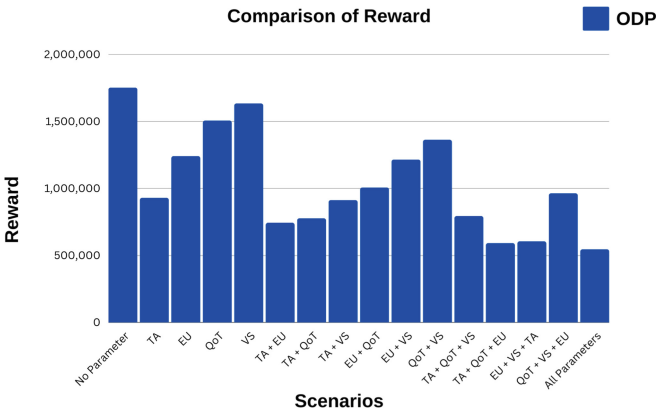


Fig. 7. Reward earned by ODP in different scenarios

turning angle. It can also be observed that when no parameter is considered we got a distance value of 29173.96, a time covered value of 17625.59, and a reward value of 1750181.4, but when all the parameters are considered then the distance covered decreases to 15414.00, time taken decreases to 16917.77 and reward decreases to 543325.4. Without considering the angle, the UAV can choose any point from the space which generates a zig-zag path with sharp turns that are not desired in a real scenario whereas when the angle component is considered, then UAV follows a smooth path directed towards the goal without considering nodes out of coverage area thus decreasing the total distance covered which decreases the total time taken.

Reward/Traverse Time: Figure 8 shows that the scenarios considering environmental uncertainties which decreases the reward per unit traverse time from

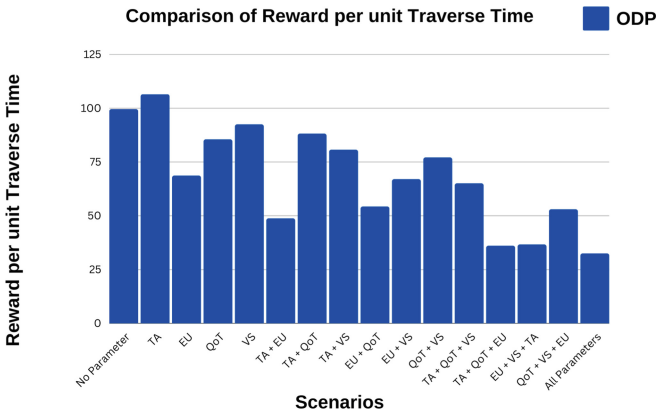


Fig. 8. Reward per unit Traverse Time: ODP

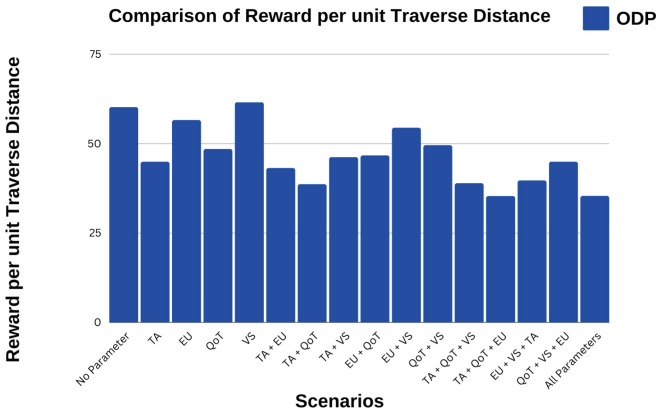


Fig. 9. Reward per unit Traverse Distance: ODP

a range of 32% to 52% as compared to scenario 1. Since environmental uncertainties slows down the speed, traverse time increases.

Reward/Traverse Distance: Figure 9 shows that the scenarios considering turning angle restriction decreases the reward per unit traverse distance from a range of 27% to 37% as compared to scenario 1.

Reward/Runtime: Figure 10 shows that the reward per unit runtime for the scenario considering all the parameters decreases by 45% as compared to scenario 1. Since runtime is higher for the case not considering any parameter in comparison to the case where all the parameters are considered but the reward generated in scenario 1 is higher which in turn results in a greater overall ratio of reward/runtime.

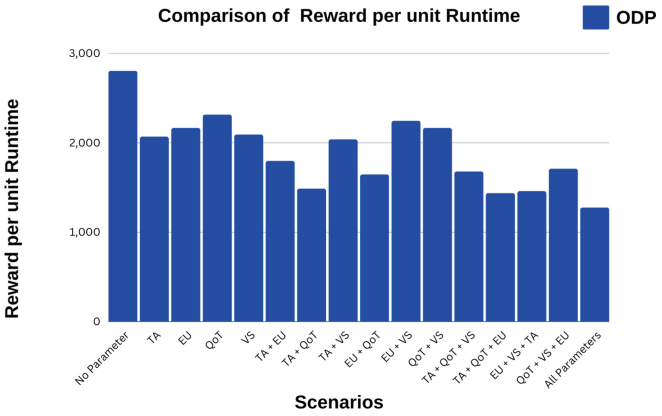


Fig. 10. Reward per unit Runtime: ODP

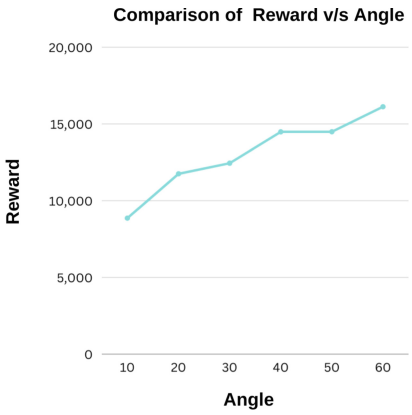


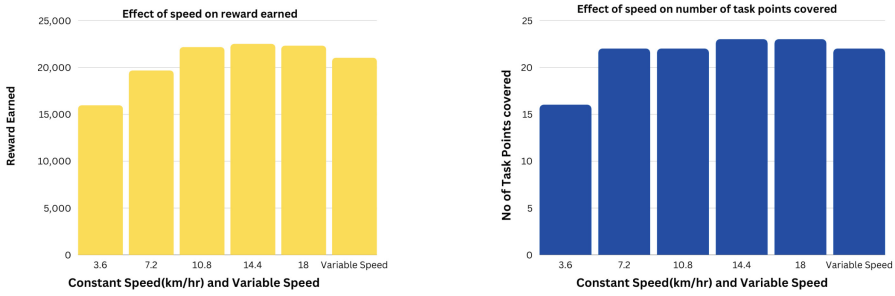
Fig. 11. Line graph for Reward v/s Angle

Turning Angle v/s Reward: In Fig. 11, we evaluate the importance of turning angle value in the process of trajectory generation by analyzing the flight path changes with respect to reward value for the six different angles. As the angle increases, reward earned also increases. This inference can be based on the fact that when a wider angle is provided to the UAV then it is allowed to choose its path from a larger set of task points lead to higher probability of generating rewards.

Speed v/s Reward Earned and Number of Task Points Covered: Figure 12(a and b) shows the implications of constant and variable speed. If we fly the UAV with constant speed throughout the mission (where u_x is not affected by the environmental uncertainties), then the following trend is observed: As the speed increases, there is a relative increase in the total rewards earned and number of task points covered. This can be based on the fact that a UAV with higher

speed is able to cover larger number of points in less time satisfying timestamp constraint. We have also performed ten simulations and recorded the results with variable speed (Scenario 5: where u_x is randomly chosen between S_{min} and S_{max} for every location x and it is not affected by the environmental uncertainties). The algorithm is able to generate the optimum value of reward and cover a good number of task points.

Comparison with A* Algorithm: Since A* algorithm aims for finding the shortest path we cannot provide all the constraints and parameters to A* algorithm as ODP. A* algorithm is not time bound as ODP and thus, it does not work with a prior given trajectory with timestamped task and waypoints.



(a) Speed v/s Reward earned

(b) Speed v/s Task points covered

Fig. 12. Effect of speed on reward earned and number of task point covered

On comparing A* algorithm with the ODP approach like we compared brute force with ODP, the path found by A* (offline algorithm) will not always be feasible (not satisfying the condition of covering waypoints at exact timestamp). Moreover the path returned by A* algorithm would already have been given as one of the path by brute force approach and is already compared with, if that is feasible. If we try to apply A* algorithm between every x and w_x, then the time complexity would be very high, which cannot be used in real world applications. Also, if we consider the scenario where we have 3D task points which are associated with the reward points and we have to find a path from source to destination. The cost function of A* will be $f(n) = (g(n) + h(n))/reward$, where $g(n)$ is the distance from the source to the current node and $h(n)$ is the distance from the current node to the destination. The efficiency measure of ODP works as $efficiency = reward/DistanceToReach$. ODP focuses on generating a flight path in real time without going back to the points previously visited thus eliminating the overhead of repeated task of path planning from a certain set of points whereas A* algorithm re-evaluates the path from the points saved in an open set leading to higher run time and brute force like time complexity in the worst case. Moreover since A* algorithm works with the distance in it's cost function and uses $h(n)$ as the euclidean distance from the current node to

the destination, to compare it with ODP, we need to modify the cost function in terms of time (with varying speed on facing environmental uncertainties) and then make it time-bound, which in turn will modify the algorithm itself.

7 Conclusion and Future Work

This work addresses the problem of generating UAV flight paths that pass through specified waypoints and satisfy spatio-temporal dynamic set of task points with applications in battlefield management and disaster response requiring adaptive flight paths. One of the major differences between our approach and others is that majority of the algorithms are offline (A* type algorithms), and thus do not perform well in real-time as task points are timestamped and are not-predefined. Unlike existing offline algorithms, which are not suitable for real-time applications, ODP incorporates flight parameters and environmental constraints into a single framework, allowing for real-time path planning.

To assess the effectiveness of ODP, it is compared to a brute force offline approach to demonstrate that the online algorithm maximizes the reward within the given knapsack capacity indicating a near-optimal performance close to the optimal offline solution. The online algorithm follows forward exploration and works well in dynamic environment and even in the absence of any specified trajectory where some task need to be performed at waypoint, if necessary. To address the limited travel range of UAVs, instead of minimizing deviation, we set an upper bound on the maximum time for the UAV to travel from the source to the destination to focus on time constraints rather than distances.

For future, we will work on multi-dimensional UAV path planning involving multiple UAVs with applications in a broader range of real-world scenarios.

References

1. Aggarwal, S., Kumar, N.: Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. Comput. Commun. **149**, 270–299 (2020). https://doi.org/10.1016/j.comcom.2019.10.014
2. Ažaltovič, V., Škvareková, I., Pecho, P., Kandera, B.: The correctness and reaction time of piloting the unmanned aerial vehicle. Transp. Res. Procedia **51**, 342–348 (2020)
3. Bortoff, S.A.: Path planning for UAVs. In: Proceedings of the 2000 American Control Conference, ACC (IEEE Cat. No. 00CH36334), vol. 1, no. 6, pp. 364–368. IEEE (2000). https://doi.org/10.1109/ACC.2000.878915
4. Cabreira, T.M., Brisolara, L.B., Ferreira Jr, P.R.: Survey on coverage path planning with unmanned aerial vehicles. Drones **3**(1), 4 (2019)
5. Chakrabarty, D., Zhou, Y., Lukose, R.: Online knapsack problems. In: Workshop on Internet and Network Economics (WINE) (2008)
6. Chen, J., Li, M., Yuan, Z., Gu, Q.: An improved A* algorithm for UAV path planning problems. In: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, pp. 958–962. IEEE (2020)

7. Edelsbrunner, H., Seidel, R.: Voronoi diagrams and arrangements. In: Proceedings of the First Annual Symposium on Computational Geometry – SCG 1985. ACM Press, Baltimore (1985)
8. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
9. Herath MPC Jayaweera and Samer Hanoun: Path planning of unmanned aerial vehicles (UAVs) in windy environments. Drones **6**(5), 101 (2022)
10. Khan, M.T.R., Muhammad Saad, M., Ru, Y., Seo, J., Kim, D.: Aspects of unmanned aerial vehicles path planning: overview and applications. Int. J. Commun Syst **34**(10), e4827 (2021)
11. Luo, J., Wang, Z., Xia, M., Wu, L., Tian, Y., Chen, Y.: Path planning for UAV communication networks: related technologies, solutions, and opportunities. ACM Comput. Surv. **55**(9), 1–37 (2023)
12. Mekala, A.R., Madria, S., Linderman, M.: Aerial vehicle trajectory design for spatio-temporal task aggregation. In 2016 International Conference on Unmanned Aircraft Systems (ICUAS), USA. IEEE (2016)
13. Mohsan, S.A.H., Khan, M.A., Noor, F., Ullah, I., Alsharif, M.H.: Towards the unmanned aerial vehicles (UAVs): a comprehensive review. Drones **6**(6), 147 (2022)
14. Nam, L.H., Huang, L., Li, X.J., Xu, J.F.: An approach for coverage path planning for UAVs. In: 2016 IEEE 14th International Workshop on Advanced Motion Control (AMC), pp. 411–416. IEEE (2016)
15. Padmanabhan, J., Swagath, S.: A study report on solving 0–1 knapsack problem with imprecise data. In: International Conference on Computer Communication and Informatics (ICCCI), pp. 1–5. IEEE (2017)
16. Qu, Y., Pan, Q., Yan, J.: Flight path planning of UAV based on heuristically search and genetic algorithms. In: 31st Annual Conference of IEEE Industrial Electronics Society, IECON 2005, USA, pp. 5–pp. IEEE (2005)
17. Razzaq, S., Xydeas, C., Everett, M.E., Mahmood, A., Alquthami, T.: Three-dimensional UAV routing with deconfliction. IEEE Access **6**, 21536–21551 (2018)
18. Roseman, C.A., Argrow, B.M.: Weather hazard risk quantification for sUAS safety risk management. J. Atmos. Ocean. Technol. **37**(7), 1251–1268 (2020)
19. She, R., Ouyang, Y.: Efficiency of UAV-based last-mile delivery under congestion in low-altitude air. Transp. Res. Part C Emerg. Technol. **122**, 102878 (2021)
20. Tseng, F.H., Liang, T.T., Lee, C.H., Der Chou, L., Chao, H.C.: A star search algorithm for civil UAV path planning with 3G communication. In: 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 942–945 (2014). https://doi.org/10.1109/IIH-MSP.2014.236
21. Zhan, W., Wang, W., Chen, N., Wang, C.: Efficient UAV path planning with multiconstraints in a 3D large battlefield environment. Math. Probl. Eng. **2014**, 1–12 (2014)
22. Zhang, C., Zhen, Z., Wang, D., Li, M.: UAV path planning method based on ant colony optimization. In: 2010 Chinese Control and Decision Conference, Xuzhou, China, pp. 3790–3792. IEEE (2010)