Comparisons of RRT and MCTS for Safe Assured Path Planning in Urban Air Mobility

Pengcheng Wu*
University of California San Diego, La Jolla, CA, 92093
San Diego State University, San Diego, CA, 92182

Jun Chen†
San Diego State University, San Diego, CA, 92182

Urban air mobility (UAM) using electrical vertical take-off and landing (eVTOL) aircraft is an emerging way of air transportation within metropolitan areas. However, the ability to navigate eVTOL aircraft through congested urban air environment poses a great challenge for path planning. For the successful operations of autonomous aerial vehicles in UAM, it is of great importance to take the collision-free requirement into consideration. It is also necessary to consider the presence of various forms of uncertainty in path planning. To address these issues, in this paper the path planning problem is first formulated as a Markov Decision Process (MDP) and then solved by an online algorithm of Monte Carlo Tree Search (MCTS) incorporating chance constraints of uncertainty. For the sake of illustration, a high-density free flight airspace simulator is created to test the performance of this proposed algorithm. Numerical simulation results demonstrate that this proposed algorithm outperforms the Chance Constrained Rapidly-exploring Random Tree (CCRRT) Algorithm, a path planning algorithm we proposed before, in terms of many aspects.

I. Introduction

The applications of urban air mobility (UAM) using electrical vertical take-off and landing (eVTOL) aircraft are increasingly drawing great attention [1], [2]. However, it is still faced with many challenges, among which safe flying is a critical concern for the successful operations. Ideally, for safe and reliable flying, the requirement of collision avoidance should be fulfilled. However, various kinds of uncertainty may be present in realistic scenarios, which may highly increase the chance of infeasibility for operations based on the current collision avoidance system. Alternatively, it is expected that probabilistically guaranteed feasible paths can be identified at a given risk level of collision for the eVTOL subject to distinct forms of uncertainty. Lower risk leads to higher computational intensity and longer planning time and trajectories. Therefore, it is important to achieve a balance between the planning conservatism and the risk of infeasibility entailed by collision.

Several different forms of uncertainty are found in common path planning scenarios involving collision avoidance [3]–[6]. One named model uncertainty is derived from every vehicle itself. Another one is derived from the sensing uncertainty of environmental obstacles that need to be evaded by all the vehicles in the team. Prentice et al. considered a linear system subject to Gaussian process noise to achieve the desired probability of feasibility [7]. Pepy et al. sought guaranteed feasibility for a nonlinear system subject to bounded state uncertainty of the model [8]. In addition to model uncertainty, many existing works have also focused on the environmental sensing uncertainty. Luders et al. studied probabilistic robustness to both process noise and uncertain dynamic obstacles following deterministic trajectories [9], [10]. Aoude et al. discussed the path planning for autonomous robots operating amidst dynamic obstacles with uncertain motion patterns [11]. In this paper, we will consider the location uncertainty for all the aircraft in a multi-agent system.

For an eVTOL operating in a complex environment, one effective way to achieve the balance between planning conservatism and the risk of infeasibility is through formulating the probabilistic bound for collision with obstacles as the chance constraints, meaning that the probability of constraint violation does not exceed a prescribed value [12]. Through a transformation, it's possible to convert probabilistic constraints into equivalent deterministic ones. That way the desired probability of feasibility for path planning can be achieved. Blackmore et al. proposed a chance constraints

*Joint PhD Student, Department of Aerospace Engineering, Student Member AIAA, pwu@sdsu.edu,pcwupat@ucsd.edu
†Assistant Professor, Department of Aerospace Engineering, Member AIAA, jun.chen@sdsu.edu (Corresponding Author)

formulation for a system subject to Gaussian noise, assuming all environmental obstacles have static, deterministic locations [13], [14]. Toit et al. developed the chance constrained framework to consider other types of uncertainty [15]. Luders et al. extended the chance constraints formulation to allow dynamic, probabilistic obstacles [10]. However, such formulations only work for convex polygon obstacles and are also very conservative. In contrast, we propose a chance constraints formulation corresponding to Gaussian distribution, which is more generic and accurate than existing ones.

Decades of research have explored a variety of approaches for designing collision avoidance systems for aircraft, which can be categorized as centralized and decentralized ways. In centralized methods, the conflicts between aircraft are resolved by a central supervising controller. Under such scenario, the state of each aircraft, the obstacle information, the trajectory constraint as well as the terminal condition are known to the central controller (thus centralized methods are always cooperative), and the central controller in return designs the whole individual trajectory for all aircraft before the flight, typically by formulating it to an optimal control problem. However, these formulations can become computationally intensive when the state space becomes large or high-dimensional. To address this issue, sample-based planning algorithms like Rapidly-exploring Random Tree (RRT) are proposed [3], [16]. On the other hand, decentralized methods scale better with respect to the number of agents and are more robust since they do not possess a single point of failure [17]. In decentralized methods, all the conflicts are resolved by each aircraft individually. With the help of machine learning and reinforcement learning [18]–[22], collision avoidance algorithm can have a promising performance, but usually needs much time to train. Using the Monte Carlo Tree Search algorithm (MCTS) to solve this problem [23] does not need model training. In this paper, we formulate this guidance and collision avoidance system as a Markov Decision Process (MDP) and solve this MDP using the online algorithm MCTS with chance constraints, where the chance constraints are introduced to account for both model and environmental uncertainty. The aircraft can adopt continuously different actions at each time step.

The remainder of this article is organized as follows. In section II, we come up with the statement of the problem, chance constraints formulation, and MDP formulation. In section III, we develop MCTS and CCRRT algorithms for path planning. In section IV, the performance of the two proposed algorithms is compared through the simulation study. Finally, we conclude this article in section V.

II. Mathematical Modeling

A. Problem Statement

This paper is proposed to navigate a multi-agent system consisting of multiple eVTOL aircraft to fly from their departure points to respective destinations through a series of control actions with the intent to avoid conflicts among them. This is a sequential decision-making problem and can be formulated as a Markov Decision Process (MDP) problem. In addition, instead of the MDP formulation which considers a single aircraft only, we formulate Multi-agent Markov Decision Process (MMDP) as an extension of MDP to fit the setting of the multi-agent system presented in this paper. In the MMDP formulation, for every decision, the aircraft's action is determined by its state which contains all the information to decide the optimal action for the current state.

To achieve the goal, an algorithm which acts on every aircraft in the multi-agent system is required. Two algorithms are developed based on the method of Chance Constrained Rapidly-exploring Random Tree (CCRRT) and Monte Carlo Tree Search (MCTS) respectively. In this paper, a high-density free flight airspace scenario is considered: every aircraft is equipped with the two algorithms and will try to avoid conflicts with other aircraft in the system. To control the aircraft, all the aircraft in the system are assumed to fly at the same altitude and thus only horizontal actions need to be considered. In addition, the location uncertainty of the aircraft is considered in this paper, and the performance comparisons of the two algorithms are made.

The objectives of this specific MMDP problem are presented as follows: 1) To avoid potential conflicts among all the aircraft as much as possible; 2) To navigate all the aircraft in a multi-agent system to reach their respective destinations as soon as possible. The details of the MMDP problem will be introduced next.

B. LOCCS Formulation

A Loss of Separation (LOS) event is defined to be an event when the distance between two aircraft is less than a minimum distance, i.e.,

$$\|\boldsymbol{x}_j - \boldsymbol{x}_i\| \leq r_j + r_i \tag{1}$$

where r_j and r_i are the required safety ranges, and $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$ are the locations of the aircraft j and the aircraft i at time t, respectively. In this paper, for simplicity, r_j and r_i are assumed to be constant at any time t. Due to location uncertainty, it's impossible to find paths that absolutely avoids LOS events. Therefore, we attempt to find safe assured paths for all the aircraft to reach their goal positions. That is, the probability of the occurrence of LOS event between any two aircraft in the system at any time is less than a certain level, i.e.,

$$\Pr(\text{LOS event}) \leq \alpha \tag{2}$$

where α is a given risk level.

Given unknown location uncertainty of the aircraft, both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in Eq. (1) should be treated as random variables. Thus it's impossible to evaluate the real distance between two aircraft with the information of their states only. To quantify the probability of an aircraft avoiding conflicts with other aircraft considering location uncertainty as presented in Eq. (2), we first define the risk domain of a d-dimensional Gaussian random variable. That is, a set $\mathcal{D} \subset \mathbb{R}^d$ that satisfies

$$\Pr(\boldsymbol{X} \in \mathcal{D}) \geq 1 - \alpha \tag{3}$$

is called a risk domain at risk level α of a random variable $\boldsymbol{X}$. Especially, when the aircraft location $\boldsymbol{c}_{ti}$ is assumed to obey Gaussian distribution and the safety range is ignored (i.e. $r_i = 0$), the boundary of the corresponding risk domain $\mathcal{D}$ for $\boldsymbol{c}_{ti}$ only can be proved to be a circle or an ellipse, according to our previous work [24]. To be specific, let $\boldsymbol{X} \in \mathbb{R}^d$ be a d-dimensional random variable that obeys a d-dimensional Gaussian distribution $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the following set

$$\mathcal{D} := \{\boldsymbol{X} \mid (\boldsymbol{X} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{X} - \boldsymbol{\mu}) \leq F_d^{-1}(1 - \alpha)\} \tag{4}$$

specifies the risk domain of $\boldsymbol{X}$ at risk level α, where F_d^{-1} is the inverse mapping of the cumulative distribution function (CDF) of χ^2 distribution with d degrees of freedom.

When it comes to conflict checking, the radius of the safety range for the particular aircraft and the other aircraft i are r_* and r_i respectively, and the location of the aircraft i is $\boldsymbol{c}_{ti}$ obeying a Gaussian distribution. We can first obtain the risk domain $\mathcal{D}_{ti}$ for the location $\boldsymbol{c}_{ti}$ of aircraft i at time t, assuming that $r_i = 0$. Next, when considering the safety range r_i of aircraft i, the corresponding risk domain for aircraft i at time step t is in fact the enveloping area of a series of circles whose centers are inside or on the boundary of $\mathcal{D}_{ti}$. We denote this actual risk domain as $\mathcal{D}_{ti}^{range}$, which is a larger area than $\mathcal{D}_{ti}$, as illustrated in Fig. 1.

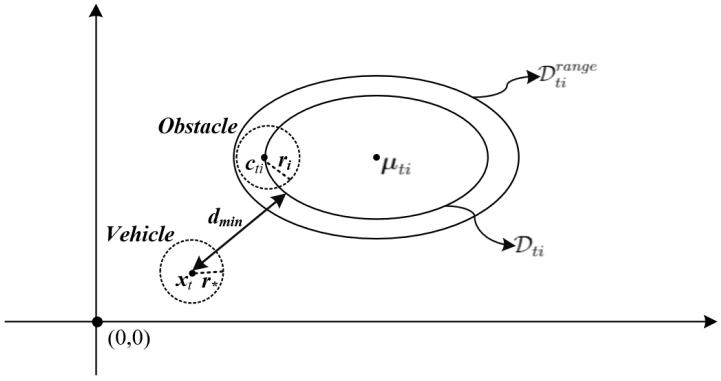


Fig. 1 Diagram of LOCCS formulation

Since there is no explicit expression of the actual risk domain $\mathcal{D}_{ti}^{range}$, we will check the feasibility of the chance constraints by measuring the minimum distance $d_{\min}$ between the particular aircraft's location $\boldsymbol{x}_t$ and the risk domain $\mathcal{D}_{ti}$ for the location $\boldsymbol{c}_{ti}$ of aircraft i at time t. In fact, $d_{\min}$ can be treated as the distance between a point and an ellipse, which can be evaluated explicitly according to [25]. Then we check the feasibility of the deterministic constraints using the inequality

$$d_{\min}(\boldsymbol{x}_t, \mathcal{D}_{ti}) > r_i + r_* \tag{5}$$

and if it holds, then

$$\begin{aligned} &\Pr(\text{LOS event} \mid \text{Eq. (5) holds}) \\ &= \Pr(\|\boldsymbol{x}_t - \boldsymbol{c}_{ti}\| \leq r_i + r_* \mid \boldsymbol{x}_t \notin \mathcal{D}_{ti}^{range}) \\ &\leq \Pr(\boldsymbol{c}_{ti} \notin \mathcal{D}_{ti}) = \alpha \end{aligned} \tag{6}$$

We can quantitatively define the concept of Loss of Chance Constrained Separation (LOCCS) event. That is, an LOCCS event occurs when

$$d_{\min}(\boldsymbol{x}_t, \mathcal{D}_{ti}) \leq r_i + r_* \tag{7}$$

Above all, to handle the location uncertainty, we introduce the concept of LOCCS event. Given a risk level α which denotes the probabilistic bound on the chance that the particular aircraft conflicts with any other aircraft i at any time step t, we can convert the stochastic constraint Eq. (6) into a deterministic constraint Eq. (5) equivalently, in consideration of the location uncertainty of aircraft. Through the formulation of LOCCS, we learn that whether the probability of LOS event occurrence is smaller than a given risk level or not can be judged through evaluating the distance between a point and an ellipse and then checking whether that distance is larger than a level, which depends on the given risk level [24].

C. MMDP Formulation

In a traditional MDP formulation, one agent may choose any action a that is available based on current state s at each time step. The process responds at the next time step by proceeding to a new state s' with certain transition probability, and gives the agent a corresponding reward r. An MDP formulation is usually composed of the following components: (1) The state space $\mathcal{S}$ which consists of all the possible states; (2) The action space $\mathcal{A}$ which consists of all the actions that the agent can take; (3) Transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$ which describes the probability of arriving at state s_{t+1}, given the current state s_t and action a_t; (4) The reward function $\mathcal{R}(s_t, a_t, s_{t+1})$ which decides the reward received after transitioning from state s to state s' according to action a.

In an MDP problem, a policy π is a mapping from the state to a specific action:

$$\pi : \mathcal{S} \rightarrow \mathcal{A} \tag{8}$$

The goal of an MDP formulation is to find an optimal policy π^* that, if followed from any initial state, maximizes the expected cumulative rewards over all the future steps:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[\sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1})|\pi] \tag{9}$$

Q-function and value function are two important concepts in MDP. The optimal Q-function $Q^*(s, a)$ means the expected cumulative reward received by an agent starting in state s and picks action a, then will behave optimally afterwards. Therefore, $Q^*(s, a)$ is an indication of how good it is for an agent to pick action a while being in state s. The optimal value function $V^*(s)$ denotes the maximum expected total reward when starting from state s, which can be expressed as the maximum of $Q^*(s, a)$ over all possible actions:

$$V^*(s) = \max_a Q^*(s, a) \quad \forall s \in \mathcal{S} \tag{10}$$

Based on the discussions of traditional MDP formulation, MMDP formulation can be extended to fit the setting of a multi-agent system. The MMDP formulation presented in this paper is made up of the following five components.

(1) Continuous State Space: The state includes the necessary information for the algorithm to perform actions on every single aircraft in the system: the position (x, y), speed v, heading angle ψ, and goal position (g_x, g_y) for every aircraft in the system. Capturing the information of every aircraft in the system, the state for the MMDP formulation turns out to be an $n \times 6$ matrix, where n is the number of aircraft and each row of the matrix represents the information for one single aircraft in the system.

It is worth noting that in this paper the state space being discussed is continuous. For instance, all the components of a state can take continuous values. In general, for an MDP with continuous state variables, it is not clear how to best represent the policy, because it is impossible to enumerate all possible state–action mappings. For previous MDP-based algorithms to solve conflict avoidance problems, some possible approaches were proposed to represent the policy include using a grid-based discretization of the state space $\mathcal{S}$ and the action space $\mathcal{A}$, or using policy compression techniques [26]. The advantage of the MCTS algorithm used in this paper is that it does not need to discretize the state space. For each state, the MCTS algorithm will generate actions for the aircraft to follow in real time.

(2) Continuous Action Space: At each time step, the aircraft can choose to turn its heading angle at a certain rate. Each aircraft has a set of continuous actions $\mathcal{A} = \{a \in \mathbb{R} | -5\,^\circ/\mathrm{s} \leq a \leq 5\,^\circ/\mathrm{s}\}$, where the plus and minus sign corresponds to right and left turn, respectively. This paper assumes that the aircraft is flying with cruise speed at

190 km/h [26]. At each time step, an action will be generated from the action set for the aircraft based on the current state, and the action will maintain the same until next step.

(3) Dynamic Model with Location Uncertainty: Built on the information from current state and action, the following kinematic model is used to find state transition for every aircraft in the system:

$$\dot{x} = v\cos\psi \tag{11}$$

$$\dot{y} = v\sin\psi \tag{12}$$

$$\dot{\psi} = a_{\psi} \tag{13}$$

where v is the cruise speed, ψ is the heading angle, and a_{ψ} is the selected action describing the changing rate of heading angle for the aircraft, whose cruise speed is set as 190 km/h [26].

After an aircraft executes an action, the aircraft speed is held constant during one time step. In addition, we also consider some disturbance that influences the aircraft speed and its changing rate of the heading angle. This ultimately accounts for the location uncertainty of the aircraft, which can be modeled as a Gaussian distribution centered at the expected position of the aircraft after one time step with a certain covariance. The disturbance here aims to capture the uncertainty from environment and aircraft dynamics respectively.

(4) Terminal State and Reward Function: Based on the aforementioned separation requirements, the terminal state of the MMDP formulation includes the following two different types of states, which can be determined directly from the state information: 1) An LOCCS event occurs, i.e., the probability that an LOS event occurs is larger than a given risk level (referred to as an LOCCS state in the following); 2) the aircraft reaches the goal position (referred to as a goal state in the following). In addition to the terminal states listed above, all the other scenarios encountered can be stacked together and defined as a non-terminal state.

(5) The objective of our computational guidance system is to achieve the dual objectives of maintaining safety while guiding the aircraft to their destinations as rapidly as possible. These two objectives can be captured in the reward function defined as follows:

$$r(s) = \begin{cases} 1, & \text{if } s \text{ is a goal state} \\ 0, & \text{if } s \text{ is an LOCCS state} \\ 1 - \frac{d(o,g)}{\max d(a,g)}, & \text{if } s \text{ is a non-terminal state} \end{cases} \tag{14}$$

where $d(o, g)$ denotes the distance from the aircraft to its goal position and $\max d(o, g)$ is the maximum distance from an aircraft to its goal position, which is the diagonal distance of the map. (If the map has an irregular convex shape, we can use the diameter of this convex shape.) In this way, if an aircraft is not at the LOCCS state (which has reward 0), this aircraft will get a positive reward between 0 and 1, depending on how far this aircraft is from the goal position.

After we have the reward function definition for one aircraft, the reward function for the MMDP is defined to be the sum of reward for each aircraft:

$$R(s) = \sum_{i} r_i(s) \tag{15}$$

With this reward setting, when we solve the formulated MMDP by maximizing the reward, all the aircraft will try to select action leading to a state that is closer to the goal position (which has a positive reward) and avoid any LOCCS states (which has a reward of 0).

III. Solution Methods

The algorithm of path planning which acts on every aircraft is required to identify the safe assured paths for the multi-agent system. In this paper, two algorithms are developed based on the method of Chance Constrained Rapidly-exploring Random Tree (CCRRT) and Monte Carlo Tree Search (MCTS) respectively. In this article, we take into account the scenarios of high-density free flight airspace. we will run the proposed algorithms on every individual aircraft in the multi-agent system with the purpose of avoiding potential inter-agent conflicts. All the aircraft in the system are assumed to fly at the same altitude (flight level) and thus only horizontal actions need to be considered. The results of the two proposed algorithms will also be compared.

A. Monte Carlo Tree Search (MCTS) Algorithm

Monte Carlo Tree Search (MCTS) is a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results [27], [28].

The basic MCTS process is conceptually easy to understand, where a tree is built in an incremental and asymmetric manner, as shown in Fig. 2 (from [28]). For each iteration of the algorithm, a tree policy is used to find the most urgent node of the current tree. The tree policy attempts to balance considerations of exploration (look in areas that have not been well sampled yet) and exploitation (look in areas which appear to be promising). A simulation is then rolled out from the selected node and the search tree updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node and an update of the statistics of its ancestors. Moves are made during this simulation according to some default policy, which in the simplest case is to make uniformly random moves. A great benefit of MCTS is that the values of intermediate states do not have to be evaluated, as for depth-limited minimax search, which greatly reduces the amount of domain knowledge required. Only the value of the terminal state at the end of each simulation is required.

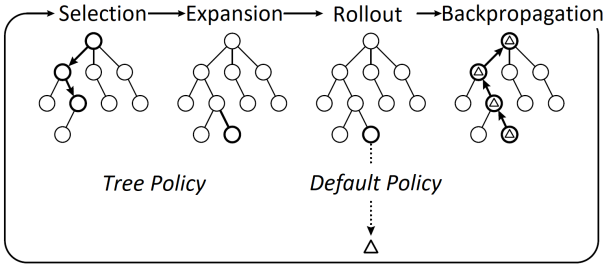


Fig. 2 One iteration of general MCTS approach [28]

Applying the tree search algorithm to continuous action case causes tension between exploring the larger set of candidate actions to cover more actions, and exploiting the current candidate actions to evaluate them more accurately through deeper search and more execution outcomes. In this paper, we use GP Regression to generalize action value estimates over the entire parameter space, which can guide the agent to add the most promising action from the action space to the search tree.

We now describe the technical details of the MCTS algorithm with continuous action space. The overall steps of the algorithm are similar to [26], and in this paper we only describe the difference in the selection step due to the incorporation of continuous action space. MCTS algorithm with continuous action space, similar to MCTS algorithm with discrete action space, selects actions by searching ahead. In the step of selection, the current node will select a node (corresponding to an action) from all of its child nodes as next state, which maximizes the function of Upper Confidence Bound Applied to Trees (UCT) introduced in Eq. (16). The UCT function consists of two terms: a mean action value $\overline{X}_j$, and an uncertainty addition [28], [29]:

$$UCT = \overline{X}_j + 2C\sqrt{\frac{2\ln n}{n_j}} \tag{16}$$

The first term $\overline{X}_j$ can be understood as an exploitation term derived from the formula:

$$\overline{X}_j = Q_j/n_j \tag{17}$$

where n_j is the number of the visits to the child node s^{k-1} and Q_j is the total reward of all rollouts that passed through this node. The second term can be understood as an exploration term where n is the the number of visits to the current node and C is a weighting parameter which seeks to balance exploitation and exploration of tree search.

For continuous action space, it is unlikely any more to figure out all the UCT scores for all the candidate actions. Under this circumstance, the strategy of progressive widening can address the continuous action space through limiting the number of actions in the search tree according to the times the node has been visited before and the slow-growing discrete set for sampling actions. Once the attribute of the best available is well estimated, we can take into account the additional actions. To be specific, for action selection, MCTS algorithm runs through either the improvement of value estimation for current child actions by selecting an action of highest UCT score, or the exploration of untried actions by

attaching a new action to the current node. The decision can be made according to maintaining the quantity of child actions for a node which is bounded by a sub-linear function $m(s)$ of the number of visits to the current node $n(s)$:

$$m(s) = B \cdot n(s)^{\beta} \tag{18}$$

where B and $\beta \in (0, 1)$ are two parameters that achieve a trade-off between covering more actions and improving the estimate of less actions. At each step of action selection, if the quantity of child actions under the current node s is smaller than $m(s)$, a new action will be attached to the current node s as a child action. Otherwise, an action which owns the highest UCT score will be selected from the existing child actions. On the one hand, UCT score assures that the depth of the search tree grows more rapidly in its promising parts; on the other hand, the progressive widening strategy tells that the search tree also grows broader to explore more actions in some parts of the search tree.

During each step of action selection, the simulation starts from the initial current state s^0, selecting actions repetitively before arriving at an unexpanded node (which means it has no child actions). For each time step $k = 1, \cdots$, a decision is made according to the comparison between the quantity of child actions $|\mathcal{A}_{s^{k-1}}|$ for the node s^{k-1} and the value $m(s^{k-1})$ in Eq. (18). If $|\mathcal{A}_{s^{k-1}}| \geq m(s^{k-1})$, an action a^k will be determined by maximizing the score of UCT presented in Eq. (16); otherwise, the agent will select a new action from the action space, attach it to the search tree, and expand this new edge.

In this paper, assuming the Q value is continuous in a, we use GP Regression to fit the Q value from the tree search result, which can inform us to select the most promising action. More specifically, we assume that the current node s has child actions $a_1, \cdots, a_p$ with Q value $Q(s, a_1), \cdots, Q(s_p)$. We use GP to fit the observation data $X = \{(a_i, Q(s, a_i)|i = 1, ..., p\}$. Then we sample m actions $a_1, \cdots, a_m$ according the Gaussian policy distribution. After evaluating these m actions, the proposed algorithm will attach the action with the maximum value to the search tree.

In order to evaluate the reward of a selected action, it is required to conduct conflict checking between two aircraft. Through introducing LOCCS formulation, we can propose an equivalent deterministic constraint instead of evaluating the probability of LOS event occurrence between two aircraft. Once in compliance with the proposed deterministic constraint, we can guarantee that the probability of LOS event occurrence between two aircraft should be less than a prescribed risk level.

When conducting conflict checking between two aircraft considering location uncertainty at each time step, the location of one aircraft can be represented as an ellipse while the other aircraft can be also represented as another ellipse. Note that the direction of the long axis of the ellipse aligns with the heading orientation of the aircraft. We can obtain the LOCCS constraint based on Eq. (7). If such a constraint is satisfied, it means the probability of LOS event occurrence really exceeds a prescribed risk level. Thus, the aircraft is at the state of LOCCS and the reward $r(s)$ of the current state should be set to be $r(s) = 0$, as presented in Eq. (14).

The aforementioned procedure is summarized in **Algorithm 1** as follows. In these pseudo codes, we use v to denote the node and s to denote the state information of node v. $s(v)$ indicates the state of a node and $v(s)$ means the node created from state s. $Q(v)$ is the total reward of all playouts that passed through the node v and $N(v)$ is the times that the node v has been visited before. $d(v)$ represents the search depth of the node v.

B. Chance Constrained Rapidly-exploring Random Tree (CCRRT) Algorithm

We review CCRRT algorithm presented by us in [24]. For the safe flying of a multi-agent system, the possibility of collision between any two aircraft needs to be considered. This can be addressed by decomposing the problem into a set of single-agent path planning sub-problems, with each sub-problem solved in a pre-fixed iterative order by the algorithm of path planning.

At the beginning of every new time step, when we are ready to extend the trajectory for a particular aircraft in the multi-agent system, the possible trajectories for any other aircraft within the same time step may have yet to be determined. In view of this, for the particular aircraft being prepared to be planned, the positions of other aircraft within current time step are uncertain. Thus, when we perform CCRRT algorithms given in [24] to generate new path for a particular aircraft in the system, all the other vehicles can be formulated as dynamic obstacles obeying a stochastic distribution. In doing do, the multi-agent path planning problem is reduced to a set of single-agent sub-problems. To further reduce the risk for one aircraft to collide with the other aircraft in the system, we consider the uncertainties of aircraft at not only the current time step but also the next few steps.

The steps of CCRRT algorithm for a multi-agent system are presented in **Algorithm 2** (Take a system consisting of three vehicles a, b, c as a demo).

Algorithm 1 MCTS Algorithm

function SEARCH(s_0)
  create root node v_0 with state s_0
  while Runtime $\leq T_{max}$ **do**
    $v_l, reward \leftarrow$ TREEPOLICY (v_0)
    BACKUP $(v_l, reward)$
  return a(BESTCHILD$(v_0, 0)$)

function TREEPOLICY(v)
  while v nonterminal **and** $d(v) \leq d$ **do**
    if v not fully expanded **then**
      return expand v
    else
      $v \leftarrow$ BESTCHILD (v, C)
  return v

function BESTCHILD(v, C)
  return $\underset{v' \in \text{ children of } v}{\text{argmax}} \frac{Q(v')}{N(v')} + C\sqrt{\frac{2 \ln N(v)}{N(v')}}$

function BACKUP($v, reward$)
  while v is not null **do**
    $N(v) \leftarrow N(v) + 1$
    $Q(v) \leftarrow Q(v) + reward$
    $v \leftarrow$ parent of v

Algorithm 2 CCRRT Algorithm

Start point $a_0 \leftarrow a_{\text{start}}$, $b_0 \leftarrow b_{\text{start}}$, $c_0 \leftarrow c_{\text{start}}$;
Current time step $t \leftarrow 0$;
while $t < max$ **do**
  $path\_a$.append(a_t), $path\_b$.append(b_t),
  $path\_c$.append(c_t);
  $ObstacleList\_a \leftarrow [b_t, b_{t+1}, b_{t+2}, c_t, c_{t+1}, c_{t+2}]$;
  $[a_{t+1}, a_{t+2}, a_{t+3}] \leftarrow$ CC-RRT$(ObstacleList_a)$;
  $ObstacleList\_b \leftarrow [a_{t+1}, a_{t+2}, a_{t+3}, c_t, c_{t+1}, c_{t+2}]$;
  $[b_{t+1}, b_{t+2}, b_{t+3}] \leftarrow$ CC-RRT$(ObstacleList_b)$;
  $ObstacleList\_c \leftarrow [a_{t+1}, a_{t+2}, a_{t+3}, b_{t+1}, b_{t+2}, b_{t+3}]$;
  $[c_{t+1}, c_{t+2}, c_{t+3}] \leftarrow$ CC-RRT$(ObstacleList_c)$;
  if $\|a_{t+1} - a_{\text{goal}}\| < \delta$ **or** $\|b_{t+1} - b_{\text{goal}}\| < \delta$ **or**
    $\|c_{t+1} - c_{\text{goal}}\| < \delta$
      break;
  end if
  $t \leftarrow t + 1$;
end while

IV. Simulation Results

In this section, we will make comparisons between the performance of MCTS and CCRRT algorithm, which are proposed in last section.

To validate the performance of the proposed algorithms in real world applications, we will simplify the Urban Air Mobility network by following a generic city model. In this generic city model, seven vertiports are distributed in a hexagonal pattern of "six around one". As shown in Fig. 3, one vertiport is at the center of the hexagon and located equidistant from the other six vertiports at a distance of 16 km. The safety ranges for all the aircraft are defined uniformly to be 250 m. The covariance of an aircraft is set to be $\mathbf{\Sigma} = [\frac{1}{24}\ 0;\ 0\ \frac{1}{96}]$. All the tests conducted in this section are implemented in Python 3.8 and were run on an Intel Xeon Silver 4210 CPU 2.20GHz desktop with 32GB RAM.

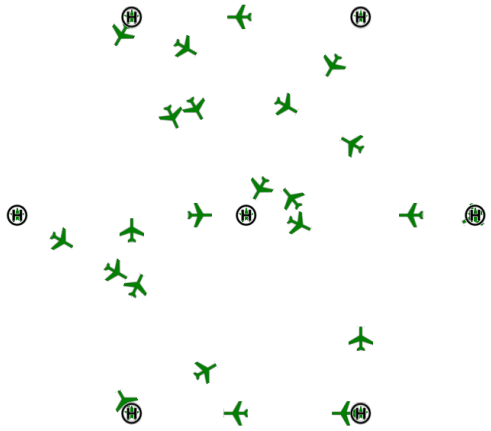


Fig. 3 Configuration of seven vertiports

Based on the setup above, we conduct the following several tests to compare the performance of two proposed algorithms. The performance of the two proposed algorithms is compared in terms of three different aspects: goal probability, average running time, and flight time.

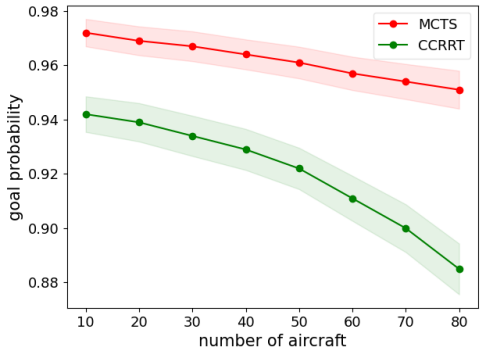
Fig. 4a shows that given the confidence level 90%, the goal probability of both MCTS and CCRRT decreases as the number of aircraft increases from 10 to 80 with incremental step 10. It also illustrates that compared with CCRRT, MCTS enables the aircraft to reach the goal with a higher probability. Fig. 4b shows that the average running time for both algorithms increases at the confidence level 90% as the number of aircraft grows, while the average running time of CCRRT increases faster than MCTS. Both MCTS and CCRRT can well identify safe assured paths for a multi-agent system.

As shown in Fig. 5, the performance of MCTS and CCRRT like goal probability and average running time at the confidence level of 95% follows the similar trends with Fig. 4. However, through comparing Fig. 4 and Fig. 5, we can find that when the level of confidence is raised, the goal probability for both MCTS and CCRRT algorithm increases at the cost of longer average running time.

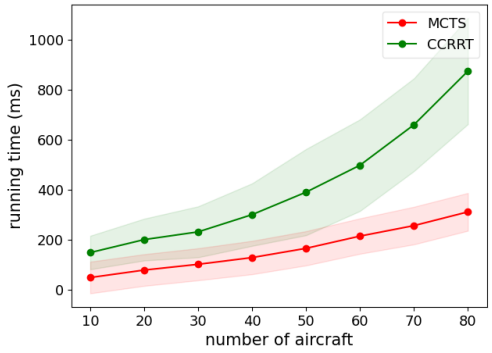
To compare the en route flight time using MCTS and CCRRT respectively, we can divide the route option into three categories based on the distance between the start vertiport and target vertiport, as shown in Fig. 6. In this vertiport map, route 1 is the shortest (e.g., from vertiport 1 to vertiport 2), route 2 has the medium length (e.g., from vertiport 3 to vertiport 7), and route 3 has the longest path (e.g., from vertiport 3 to vertiport 6).

Table 1 shows the en route flight time for different route options. From this table we can find that, the flight time ratio of MCTS to CCRRT ranges from 97.9% to 93.0%, which indicates that given any route, the flight time of aircraft using MCTS is always shorter than CCRRT. Also, as the length of route grows, the flight time difference between MCTS and CCRRT becomes more significant, which suggests that as the length of route increases, the flight time of aircraft using MCTS will be shorter and shorter than CCRRT.

The en route flight time comparisons across different covariances are recorded in Table 2. It illustrates that for any covariance, the flight time using MCTS is always shorter than CCRRT. In addition, the ratio of MCTS to CCRRT varies from 98.0% to 88.4% as the covariance increases. Thus, MCTS outperforms CCRRT in flight time, especially when the covariance is quite large.



(a) goal probability



(b) average running time

Fig. 4 The comparisons between MCTS and CCRRT at confidence level 90%

Table 1 Flight time comparisons for routes in seconds

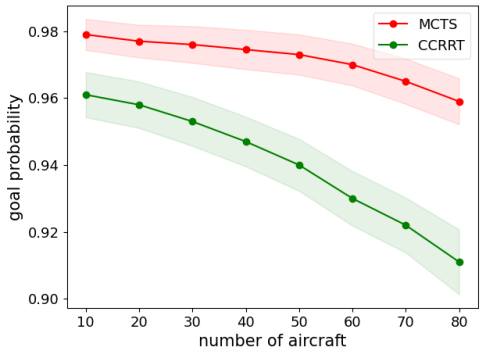
Route	1	2	3
MCTS	342.1 ± 6.8	607.8 ± 9.3	687.1 ± 10.4
CCRRT	349.5 ± 7.9	632.4 ± 10.2	738.8 ± 11.7
Ratio	97.9%	96.1%	93.0%

Table 2 Flight time comparisons for covariances in seconds

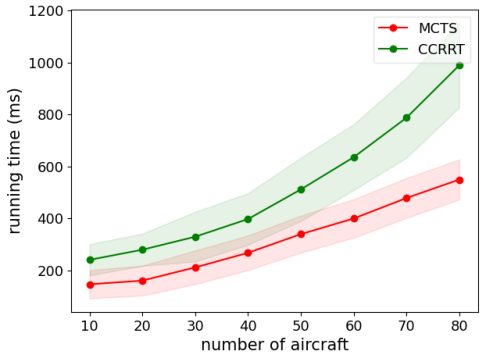
Covariance	1/4Σ	Σ	4Σ
MCTS	314.7 ± 6.0	342.1 ± 6.8	361.6 ± 7.4
CCRRT	321.1 ± 7.2	349.5 ± 7.9	409.3 ± 9.7
Ratio	98.0%	97.9%	88.4%

V. Conclusions

In this paper, we proposed a path planning algorithm with the ability of collision avoidance for autonomous free flight operations, which can navigate the eVTOL aircraft in UAM to its goal state through the control of heading angles. The problem is formulated as a Markov Decision Process (MDP) with uncertainty from the dynamics of aircraft and the environment. The MDP problem is solved by Monte Carlo Tree Search (MCTS) algorithm with an estimated value function to reduce the computation time. Compared with the algorithm of CCRRT, simulation results demonstrate that the proposed algorithm of MCTS with chance constraints has promising performance and outperforms the algorithm of CCRRT in the scenarios of dense air transportation. This proposed algorithm offers a promising solution framework for



(a) goal probability



(b) average running time

Fig. 5 The comparisons between MCTS and CCRRT at confidence level 95%

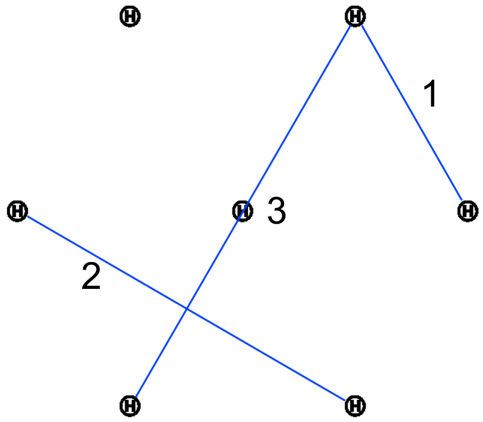


Fig. 6 Three different routes in the above vertiport setting

autonomous free flight operations in UAM.

Acknowledgment

We would like to thank the National Science Foundation (NSF) under Grants CMMI-2138612 for the support of this work.

References

[1] L. Gipson, *Nasa embraces urban air mobility, calls for market study*, `https://www.nasa.gov/aero/nasa-embraces-urban-air-mobility`, Accessed: 2020-02-15, 2017.

[2] J. Holden and N. Goel, "Fast-forwarding to a future of on-demand urban air transportation," *San Francisco, CA*, 2016.

[3] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[4] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[5] Y. Yang, J. Zhang, K.-Q. Cai, and M. Prandini, "Multi-aircraft conflict detection and resolution based on probabilistic reach sets," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 309–316, 2016.

[6] P. Wu, L. Li, J. Xie, and J. Chen, "Probabilistically guaranteed path planning for safe urban air mobility using chance constrained rrt*," in *AIAA AVIATION 2020 FORUM*, 2020, p. 2914.

[7] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear pomdps by factoring the covariance," in *Robotics research*, Springer, 2010, pp. 293–305.

[8] R. Pepy and A. Lambert, "Safe path planning in an uncertain-configuration space using rrt," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 5376–5381.

[9] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference*, 2010, p. 8160.

[10] B. D. Luders, G. S. Aoude, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns," Tech. Rep., 2011.

[11] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[12] P. Li, M. Wendt, and G. Wozny, "A probabilistically constrained model predictive controller," *Automatica*, vol. 38, no. 7, pp. 1171–1176, 2002.

[13] L. Blackmore, "Robust path planning and feedback design under stochastic uncertainty," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6304.

[14] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.

[15] N. Du Toit, "Robot motion planning in dynamic, cluttered, and uncertain environments: The partially closed-loop receding horizon control approach," PhD thesis, California Institute of Technology, 2010.

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

[17] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi, "Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, 2006, pp. 2448–2453.

[18] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 528–535.

[19] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 285–292.

[20] D.-T. Pham, N. P. Tran, S. Alam, V. Duong, and D. Delahaye, "A machine learning approach for conflict resolution in dense traffic scenarios with uncertainties," 2019.

[21] S. Li, M. Egorov, and M. Kochenderfer, "Optimizing collision avoidance in dense airspace using deep reinforcement learning," *arXiv preprint arXiv:1912.10146*, 2019.

[22] J. Hu, X. Yang, W. Wang, P. Wei, L. Ying, and Y. Liu, "Uas conflict resolution in continuous action space using deep reinforcement learning," in *AIAA AVIATION 2020 FORUM*, 2020, p. 2909.

[23] X. Yang and P. Wei, "Autonomous on-demand free flight operations in urban air mobility using monte carlo tree search," in *8th International Conference on Research in Air Transportation (ICRAT)*, ICRAT, 2018.

[24] P. Wu, J. Xie, and J. Chen, "Safe path planning for unmanned aerial vehicle under location uncertainty," in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, IEEE, 2020, pp. 342–347.

[25] V. A. Zorich, *Mathematical analysis II*. Springer, 2016.

[26] X. Yang and P. Wei, "Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1473–1486, 2020.

[27] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*, Springer, 2006, pp. 72–83.

[28] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[29] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, Springer, 2006, pp. 282–293.