2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Dynamics-Constrained Graph Learning Approach for Hypersonic Vehicle Path Planning Problem

Yuan Zhang, Ran Zhang, and Huifeng Li

Abstract—**This paper proposes a new dynamics-constrained path planning method for hypersonic vehicles. Due to vehicles' limited maneuverability, path and dynamics are tightly coupled. To ensure executability, it is required to consider dynamic constraints and trajectory indexes in path planning. This leads to a more intricate problem formulation, and the path search space is high-dimensional and sparse. Existing methods mainly target low-velocity vehicles and rely on simple motion models, making them cannot be directly applied here. We address the problem with a graph learning method. This method begins by modeling the graph-search Markov decision process (MDP) on a Graph Attention Network (GAT) to find a path in the topological graph, which guides the subsequent trajectory generation. On top of the path, it uses a three-dimensional waypoint-crossing navigation (3D WCN) law to generate a trajectory under full dynamics and uncertainties. The GAT is trained using reinforcement learning, where the devised cost function includes both trajectory and path indexes. The trajectory index punishes trajectories that fail safety checks to ensure both the performance and executability of the path. The path index, aimed at eliminating the erratic impact of implicit trajectory representations and uncertainties, is calculated from the mean square error between the path and an optimal reference, thereby improving learning efficiency. Simulation results show superior optimality, adaptability, and millisecond-level computation speed.**

I. Introduction

This paper focuses on a type of unpowered hypersonic vehicle with a flight velocity far exceeding Mach 5. In near-space mission environments, the vehicle will encounter obstacles such as no-fly zones [1]. These obstacles have random and scattered distribution, presenting the necessity in global path planning to ensure safe flight. For the hypersonic vehicle path planning problem, it is required to consider dynamic constraints and trajectory indexes. That is because these vehicles, characterized by high velocity and unpowered flight, have limited maneuverability determined by dynamics [2]. Failure to consider these constraints could result in an inexecutable path and cannot generate an associated trajectory. However, early works for path planning mainly focuses on low-velocity aircraft or robots [3][4], relying on simple trajectory representations. They cannot guarantee the complex dynamic constraints and thus cannot resolve the hypersonic vehicle path planning. In a word, dynamics-constrained hypersonic vehicle path planning is an emerging problem.

In the literature, there are two general strategies to achieve global path planning, both aimed at obstacle avoidance by searching for a sequence of waypoints from start to target within a topological graph while using an easy-to-handle motion model. The first strategy is to transform the path planning problem into a mathematical programming problem, in which the main work is to model obstacle avoidance constraints as integer constraints and model trajectory-associated constraints as continuous constraints. Afonso et al. [5] transformed the problem into a mixed integer linear programming problem. The obstacle avoidance constraints were modeled as waypoint constraints by introducing binary variables for nodes on the graph, and the motion equations were modeled as linear constraints. Similarly, Dutta et al. [6] defined the binary variables as edges on the graph and transformed obstacle avoidance constraints into directional constraints. Trajectory information is defined by continuous variables, which form quadratic constraints. These methods can intuitively implement path planning but have limitations in terms of solution efficiency. To improve efficiency, another strategy is to design heuristic algorithms, and make decisions sequentially on the graph. The trajectory is then generated separately using either geometrical curves or simple motion equations. Kool et al. [7] designed a graph attention network (GAT) to model the MDP of graph search, training it for shortest path planning by reinforcement learning. This method assumes that the agent can move in straight lines. Luo et al. [8] designed a forward-looking model based on Johnson's curve for obstacle identification, and the avoidance path is searched by the genetic algorithm (GA). Penicka et al. [9] used the informed RRT* algorithm to fast search the paths and trajectories under simple motion equations. These methods design or learn heuristic rules to accelerate solution efficiency, but they neglect motion under full dynamics and deal with trajectory generation separate from path planning.

Using existing methods to solve the dynamics-constrained hypersonic vehicle path planning problem is challenging. On the one hand, this problem requires consideration of dynamic constraints and trajectory indexes. The trajectory of the hypersonic vehicle has strong nonlinearity and implicit representations [2], resulting in a more intricate problem formulation compared to benchmark problems. So, it is difficult for algorithms to keep stable performance. On the other hand, due to the limited maneuverability, path planning and trajectory generation are tightly coupled and cannot be separated [10]. When constructing the topological graph, the sampling space becomes high-dimensional, and the node distribution is sparse, resulting in low search efficiency. Therefore, the existing methods cannot be directly applied to the hypersonic vehicle path planning problem.

Our previous work first combines hypersonic vehicle path

This work was supported by the National Natural Science Foundation of China under Grant 62103014.

Yuan Zhang, Ran Zhang, and Huifeng Li are with School of Astronautics, Beihang University, Beijing, China (e-mail: zyuan@buaa.edu.cn, zhangran@buaa.edu.cn, lihuifeng@buaa.edu.cn, corresponding author: Ran Zhang).

planning with trajectory information and index based on graph learning [11]. GAT is used to plan waypoints, kinematic trajectory provides performance indexes, and network parameters are trained based on these indexes. However, this work neglects the impact of dynamic constraints on the optimality and safety of the path. As an extension, this paper proposes an improved graph learning-based hierarchical solution framework to achieve dynamics satisfaction. This framework consists of three parts: path planning network, trajectory generation simulator, and cost function. The novelty lies in the latter two parts. Specifically, the path planning network is designed based on a GAT that matches the graph-search MDP, enabling sequential decision-making of waypoints. On top of the waypoint sequence, the trajectory generation simulator generates a trajectory under uncertainties and full dynamics, which is achieved by designing a 3D WCN law. The cost function for GAT learning is designed based on the trajectory index and safety check, penalizing trajectories that do not meet constraints. In addition, to eliminate the erratic impact of implicit trajectory representations and uncertainties, the cost function also includes a path index. This path index is the mean squared error of the path compared to an optimal reference, which improves learning efficiency compared to using only trajectory indexes. Simulation results show the potential of our method as its superior optimality, adaptability to environments, and computing efficiency at the millisecond level.

II. Problem Formulation

The mission scenario is shown in Fig. 1, where there are many obstacles in near space. To safely reach the target, the hypersonic vehicle needs to avoid these obstacles. The path planning problem contains two goals: decide a sequence of waypoints with the minimum total control effort and ensure the waypoint sequence executable by trajectory generation constrained by full dynamics.

As the cornerstone, a global topological graph $G = (\tilde{V}, \tilde{A})$ is established to cover the feasible paths, where $\tilde{V}$ is the nodes set and $\tilde{A}$ is the edges set. The basic idea of graph modeling is laying out the nodes located above and below the no-fly zone respectively, and connecting directed edges if there is no no-fly zone in the line-of-sight direction between two nodes. For specific details, refer to literature [10]. Path planning requires additional decision variables to encode the path and account for directional constraints, result in a sequence of waypoints. Specifically, we define the binary integer variables $x_{ij} = 1$ if the edge from node i to j in $\tilde{A}$ is on the path, 0 otherwise, $i, j = 1, 2, \ldots, n$, n is the number of nodes in $\tilde{V}$. Further, we ensure that each node, except for the start and target node, has an in-degree equal to its out-degree, with the out-degree being at most 1. Assuming that the start and target nodes are at indexes 1 and n, and there are no subtours present in the path, the directional constraints can be formulated as follows [12]:

$$\begin{aligned}&\sum\nolimits_j x_{1j} - \sum\nolimits_j x_{j1} = 1\\&\sum\nolimits_j x_{ij} - \sum\nolimits_j x_{ji} = 0,\ 2 \le i \le n-1\\&\sum\nolimits_j x_{nj} - \sum\nolimits_j x_{jn} = -1\end{aligned} \tag{1}$$

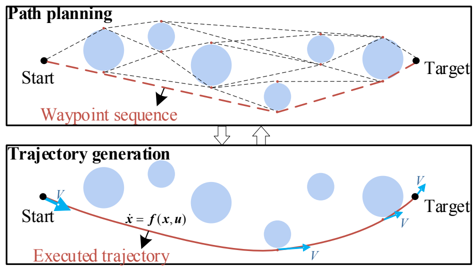

Figure 1. Diagram of dynamics-constrained path planning problem.

Constraints in (1) serve to guarantee that the path starts at node 1 and ends at node n. This ensures the connectivity, and that every node is visited at most once. Note that if the out-degree of node i except for the start node is 1, this node is a waypoint on the path, which can be formulated as follows:

$$\sum\nolimits_j x_{ij} = 1 \tag{2}$$

Different from typical graph search problems, the hypersonic path planning problem also includes dynamics introduced trajectory index and associated constraints, such as dynamic constraints, path constraints, as well as corner point constraints from the coupling with waypoints.

To be specific, we define the continuous vectors $\boldsymbol{x} = [r; \theta; \varphi; V; \gamma; \psi]$ as the flight state and $\boldsymbol{u} = [\alpha; \sigma]$ as the control of the hypersonic vehicle. Where r is the radial distance, θ is the longitude, φ is the latitude, V is the velocity, γ is the flight path angle, ψ is the heading angle, α is the angle of attack, and σ is the bank angle. We define the constants θ_i and φ_i are the longitude and latitude of node i in G, variable t_i is the time at which the vehicle passes through, $i = 1, 2, \ldots, n$. The motion is governed by gravity g_0, aerodynamic lift L, and aerodynamic drag D. The translation dynamics in consideration of a spherical and rotating Earth refer to the literature [2]. The compact form with respect to time t can be formulated as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t),\ \boldsymbol{u}(t)\right) \tag{3}$$

Constrained by the waypoints on the path, the corner point constraints of the trajectory can be formulated as follows:

$$\begin{aligned}&\theta(t_i) - \theta_i + M(1 - \sum\nolimits_j x_{ij}) \ge 0, \theta_i - \theta(t_i) + M(1 - \sum\nolimits_j x_{ij}) \ge 0\\&\varphi(t_i) - \varphi_i + M(1 - \sum\nolimits_j x_{ij}) \ge 0, \varphi_i - \varphi(t_i) + M(1 - \sum\nolimits_j x_{ij}) \ge 0\end{aligned} \tag{4}$$

Constraints in (4) serve to guarantee that node i is on the trajectory if it is on the path, where M is a large enough number. Specifically, if condition (2) is satisfied, indicating that node i is a waypoint on the path, then it must hold that $\theta(t_i) = \theta_i$, $\varphi(t_i) = \varphi_i$.

The performance index for dynamics-constrained path planning problem is to minimize the total control effort. In all, the problem can be formulated in the following compact form:

Problem 1. *Dynamics-constrained path planning problem.*

$$\begin{aligned}
\min \quad & J=\int_{t_0}^{t_f}\boldsymbol{u}(t)^2dt && \text{(Performance index)}\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t)=\boldsymbol{f}\left(\boldsymbol{x}(t),\ \boldsymbol{u}(t)\right) && \text{(Dynamic constraint)}\\
& \boldsymbol{g}\left(\boldsymbol{x}(t),\ \boldsymbol{u}(t)\right)\le \boldsymbol{g}_{\max} && \text{(Path constraint)}\\
& \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\ \boldsymbol{x}(t_f)=\boldsymbol{x}_f && \text{(Boundary constraint)}\\
& \boldsymbol{k}(x_{ij},\ \boldsymbol{x}(t_i))\ge 0,\ \forall i && \text{(Corner point constraint)}\\
& \boldsymbol{h}\left(x_{ij}\right)=0,\ \forall i && \text{(Directional constraint)}
\end{aligned} \tag{5}$$

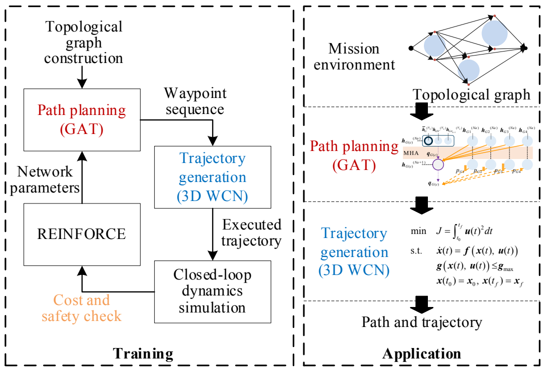

Figure 2. Dynamics-constrained graph learning framework.

where t_0 is the given initial time, t_f is the free terminal time. $\boldsymbol{g}_{\max}$ is the upper bound of path constraints. $\boldsymbol{x}_0/\boldsymbol{x}_f$ are the initial/terminal values.

III. Dynamics-Constrained Graph Learning Approach

This paper proposes a graph learning-based hierarchical framework, while maintaining the dynamics-associated index and constraints, as illustrated in Fig. 2. A sequential waypoint decision method based on the GAT is designed to search the path. The GAT matches the graph-search MDP into the network structure, enabling the network to automatically generalize the obstacle distribution. Then, we introduce a 3D WCN law, generating trajectories under paths with varying waypoint combinations. The coupling between path and trajectory is realized by the interaction between the two layers, in which the cost function design and safety check make the selected path meet both optimality and dynamic constraints.

A. Path Planning Graph Attention Network

The path planning problem is described as a sequential waypoint decision on the global topological graph. Variations in the obstacle distribution led to different graph structures and sequential decision models, creating distinct MDPs. Consequently, applying learning algorithms directly is not feasible. To address this challenge, a GAT-based network is designed to inherently match the graph-search MDP.

The designed GAT includes an encoder and a decoder. The encoder generates embeddings for each node based on the node coordinates $\boldsymbol{x}_{wi}$, $i = 1, 2, \ldots, n$, and adjacency matrix $A = \{a_{ij}\}_{n\times n}$ of the graph G. The decoder generates nodes step by step until the target node n is output, forming the path composed of waypoint sequence $\boldsymbol{\pi} = [\pi_0, \pi_1,\ldots, \pi_m]$. The network structure is illustrated in Fig. 3. The specific network layers and parameters can be found in [7][11].

Here, we describe the proposed adjacency matrix mask in the attention mechanism. The attention mechanism transfers message through interactions between nodes. The purpose of the mask is to check the connections between individual nodes to determine the eligibility of message transfer. If two nodes are not connected, the message will be masked, preventing the selection in the sequential decision. This process aligns with the Markovian nature of the graph, ensuring the directional constraints (1).

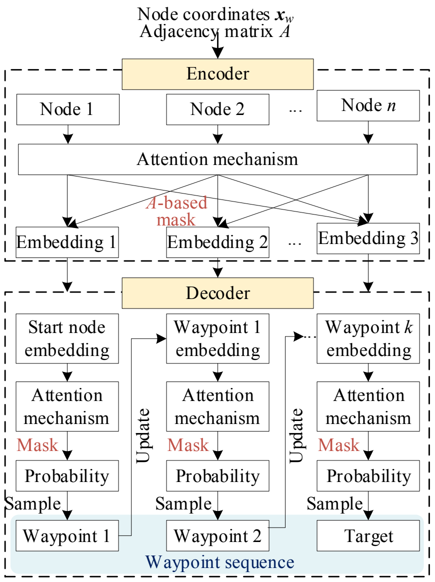

Figure 3. A graphical of the path planning GAT.

Specifically, assuming the node embedding for the input of the attention mechanism is denoted as $\boldsymbol{h}_G$, the three variables, query $\boldsymbol{q}_G$, key $\boldsymbol{k}_G$, and value $\boldsymbol{v}_G$, are obtained through linear transformations of the node embedding, that is:

$$\boldsymbol{q}_{G(c)}=\boldsymbol{W}^Q\boldsymbol{h}_{G(c)},\ \boldsymbol{k}_{Gi}=\boldsymbol{W}^K\boldsymbol{h}_{Gi},\ \boldsymbol{v}_{Gi}=\boldsymbol{W}^V\boldsymbol{h}_{Gi} \tag{6}$$

where $\boldsymbol{q}_G$ is derived solely from the current waypoint π_{k-1} at step k, while $\boldsymbol{k}_G$ and $\boldsymbol{v}_G$ are obtained from other nodes embeddings interacting with it. W^Q, W^K, and W^V represents the learning parameters of the network. Define the normalization factor d_G, the output of the attention mechanism is given by:

$$\begin{aligned}
\boldsymbol{h}'_{G(c)} &= \sum_{j=1}^{n}\operatorname{softmax}\left(u_{(c)j}\right)\boldsymbol{v}_{Gj}\\
u_{(c)j} &= \begin{cases}\dfrac{\boldsymbol{q}_{G(c)}{}^{\mathrm{T}}\boldsymbol{k}_{Gj}}{\sqrt{d_G}}, & a_{\pi_{k-1}j}=1 \text{ and } j\neq \pi_{k'}, \forall k'<k\\ -\infty, & \text{else}\end{cases}
\end{aligned} \tag{7}$$

In (7), when $a_{ij} = 1$, node i and j are adjacent, else if $a_{ij} = 0$, or node j has been visited before step k, the decoder masks the message, setting the compatibility $u_{(c)j} = -\infty$. The decoder samples the next waypoint based on the probability obtained by applying a SoftMax function to compatibilities $u_{(c)j}$ of the current waypoint with other nodes.

The network's ability to generalize across different obstacle distributions can be attributed to two factors. Firstly, the sequential decision adapts to various graph structures through the design of an adjacency matrix mask. Secondly, all projections in the GAT are node-wise, with shared parameters for different nodes, so the forward propagation adapts to various obstacle numbers.

B. Trajectory Generation Simulator by 3D WCN Law

In this subsection, a 3D WCN law is designed to execute the dynamics-constrained path. The trajectory generation simulator needs to show adaptability because of the frequent generation of paths with different numbers and combinations of waypoints when training the GAT. This paper introduces a waypoint-crossing law to obtain controls $\boldsymbol{u} = [\alpha; \sigma]$.

1) Waypoint-Crossing Navigation logic

Navigation is to determine the direction of flight $\boldsymbol{\Phi} = [\gamma; \psi]$, including vertical navigation and horizontal navigation.

Vertical navigation is to determine the vertical direction of the flight γ to achieve the required terminal altitude, the command γ_{com} is:

$$\dot{\gamma}_{\text{com}} = k\dot{q} \tag{8}$$

where q is the vertical line-of-sight angle, k is the proportional coefficient.

Horizontal navigation is to determine the horizontal direction of the flight ψ to achieve the required waypoint sequence and terminal position. According to a minimum control effort waypoint-crossing navigation law [13], define the Lagrange multiplier vector $\boldsymbol{\lambda}=[\lambda_1, \lambda_2,\dots, \lambda_m]^{\text{T}}$, miss distance vector $\boldsymbol{Z}=[Z_1, Z_2,\dots, Z_m]^{\text{T}}$, $i=1,2,\dots,m$, and parameter matrix $R \in \mathbb{R}^{m\times m}$, which satisfy $R\boldsymbol{\lambda} = \boldsymbol{Z}$.

Based on the extended lemma of the Schwartz inequality, the commanded heading angle ψ_{com} is:

$$\dot{\psi}_{\text{com}} = \begin{cases} \sum_{i=1}^{m} \lambda_i (t_{fi} - t)/V \cos\gamma = \sum_{i=1}^{m} k_i \dot{\psi}_{LOS_i}, \ t \le t_{f1} \\ \sum_{i=2}^{m} \lambda_i (t_{fi} - t)/V \cos\gamma = \sum_{i=2}^{m} k_i \dot{\psi}_{LOS_i}, \ t_{f1} < t \le t_{f2} \\ \vdots \\ \lambda_m (t_{fm} - t)/V \cos\gamma = k_m \dot{\psi}_{LOS_m}, \ t_{fm-1} < t \le t_{fm} \end{cases} \tag{9}$$

where ψ_{LOSi} is the horizontal line-of-sight angle relative to the ith waypoint π_i, t_{fi} is the crossing time, $k_1, k_2, \dots, k_m$ are the proportional coefficients obtained by (9) analytically. So, the navigation command can be formulated as follows:

$$\dot{\boldsymbol{\Phi}}_{\text{com}} = \begin{bmatrix} \dot{\gamma}_{\text{com}} \\ \dot{\psi}_{\text{com}} \end{bmatrix} \tag{10}$$

2) Control allocation

Based on the dynamic equations, the required control force $\boldsymbol{L}_{\text{com}}$ can be obtained by the navigation command (10), which can be formulated as follows:

$$\boldsymbol{L}_{\text{com}} = \begin{bmatrix} L_{V\text{com}} \\ L_{H\text{com}} \end{bmatrix} = m_v \begin{bmatrix} V\dot{\gamma}_{\text{com}} + g_0 \cos\gamma \\ V\dot{\psi}_{\text{com}} \cos\gamma \end{bmatrix} \tag{11}$$

where m_v is the mass of the vehicle.

The required lift satisfies:

$$L_{\text{com}} = \sqrt{L_{V\text{com}}^2 + L_{H\text{com}}^2} \tag{12}$$

The commanded controls are allocated by $\boldsymbol{L}_{\text{com}}$, satisfies:

$$\boldsymbol{u}_{\text{com}} = \begin{bmatrix} \alpha_{\text{com}} \\ \sigma_{\text{com}} \end{bmatrix} = \begin{bmatrix} \arg\min\limits_{\alpha_l \le \alpha \le \alpha_u} |L - L_{\text{com}}| \\ \arctan 2(L_{H\text{com}}, L_{V\text{com}}) \end{bmatrix} \tag{13}$$

where α_l and α_u are lower and upper bounds of the angle of attack, respectively. arctan 2 is a four-quadrant inverse tangent function.

Based on control commands $\boldsymbol{u}_{\text{com}} = [\alpha_{\text{com}}; \sigma_{\text{com}}]$, the trajectory under full dynamics along the selected path can be simulated by dynamics integration.

C. Graph Learning-Based Path Planning

A graph learning-based framework is introduced to train the GAT, as depicted in Fig. 4, ensuring that the selected path meets both dynamic constraints and optimal index. To achieve this, a cost function $L(\boldsymbol{\pi})$, including both trajectory and path indexes, is devised as follows:

$$L(\boldsymbol{\pi}) = \begin{cases} -\omega_1 (\boldsymbol{\pi} - \boldsymbol{\pi}^*)^2 + \omega_2 \int_{t_0}^{t_f} \boldsymbol{u}(t)^2 dt, & \text{executable} \\ M, & \text{inexecutable} \end{cases} \tag{14}$$

Where $\boldsymbol{\pi}^*$ is the optimal reference path generated offline, M is a large number, ω_1, ω_2 are the weights.

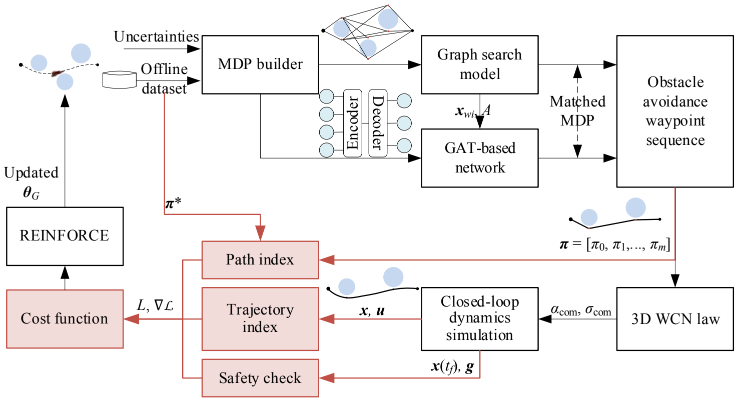
Uncertainties
Offline dataset
MDP builder
Graph search model
Encoder
Decoder
$\boldsymbol{x}_{wi}$, A
GAT-based network
Matched MDP
Obstacle avoidance waypoint sequence
Updated $\boldsymbol{\theta}_G$
$\boldsymbol{\pi}^*$
REINFORCE
$\boldsymbol{\pi} = [\pi_0, \pi_1,\dots, \pi_m]$
Path index
Cost function
L, ∇L
Trajectory index
$\boldsymbol{x}$, $\boldsymbol{u}$
Closed-loop dynamics simulation
α_{com}, σ_{com}
3D WCN law
Safety check
$\boldsymbol{x}(t_f)$, $\boldsymbol{g}$

Figure 4. Dynamics-constrained graph learning algorithm diagram.

We now describe the cost function (14). $L(\boldsymbol{\pi})$ consists of three parts: the mean square error between $\boldsymbol{\pi}$ and the optimal one $\boldsymbol{\pi}^*$, the total control effort of the generated trajectory, and the penalty if the trajectory fails safety checks. Specifically, the first part is the path index, the purpose is to eliminate the erratic impact of implicit trajectory representations and uncertainties when only relying on trajectory indexes [14], thereby improving learning efficiency. The latter two parts are trajectory indexes which are obtained by feedback information from the trajectory generation simulator. The purpose is to optimize path performance while considering full dynamics and uncertainties, enabling effective learning in diverse environments. Additionally, they penalize trajectories that fail safety checks to ensure the path's executability.

Define the loss function as the expectation of $L(\boldsymbol{\pi})$, the gradient of the loss function is defined as follows:

$$\nabla \mathrm{L}(\boldsymbol{\theta}_G \mid G) = \mathrm{E}_{p_{\boldsymbol{\theta}_G}(\boldsymbol{\pi}|G)}\left[\left(L(\boldsymbol{\pi}) - b(G)\right)\nabla \log p_{\boldsymbol{\theta}_G}(\boldsymbol{\pi} \mid G)\right] \quad (15)$$

where G is the training sample, $\boldsymbol{\pi}$ is the selected path, b is the exponential baseline aiming to reduce gradient variance, and $p_{\theta G}$ is the stochastic policy computed by the product of the waypoint probabilities for each step:

$$p_{\boldsymbol{\theta}_G}(\boldsymbol{\pi} \mid G) = \prod_{k=1}^{m} p_{\boldsymbol{\theta}_G}(\pi_k \mid G, \pi_{1:k-1}) \quad (16)$$

We employ the REINFORCE gradient estimator and use Adam [15] as optimizer to update network parameter $\boldsymbol{\theta}_G$.

IV. Simulation Verification

Two types of simulations are conducted to evaluate the effectiveness of the proposed method in common no-fly zone avoidance missions. First, we compare the optimality in comparison to the existing methods. Then, we analyze the adaptability to missions and uncertainties.

The proposed approach is implemented on a desktop PC with Intel Core i5-8265U processor, 8.0 GB memory and 64-bit Windows operation system. The training data consists of 10,000 instances. The central locations and radiuses of no-fly zones uniformly at random in the range of $\theta_o \in (20\,°, 80\,°)$, $\varphi_o \in (-10\,°, 10\,°)$, $r_o \in$ (400 km, 600 km), the number of no-fly zones $N = 3$-7. The initial states of the vehicle are $\theta_0 = 0\,°$, $\varphi_0 = 0\,°$, $h_0 = 64$ km, $V_0 = 7000$ m/s, $\gamma_0 = 0\,°$, $\psi_0 = \psi_{LOS1}$. The terminal states are $\theta_f = 100\,°$, $\varphi_f = 0\,°$, $h_f > 35$ km, $V_f = 3400$ m/s. The average code running time under 100 simulations is 0.067 s.

A. Optimality Verification

For comparison, a traversal method, SBPD algorithm [16], is implemented to generate the "optimal possible path", and the state-of-the-art mixed-integer trajectory optimization method, MITO algorithm [10], is implemented to generate the "optimal possible trajectory".

We test the same 100 instances with $N = 5$ on our method and two existing methods, one is a graph-learning method, GLPD algorithm, without considering dynamics [11], another one is the traditional A* algorithm. The optimality statistics on the test set are shown in TABLE I. Fig. 5 shows an example result. In TABLE I, the cost gap % represents the average cost difference of the paths obtained by our method relative to the SBPD algorithm. It can be observed that the cost of all test results is within a 5 % gap, and the path consistency is much higher than the existing methods. These results show a better learning performance by considering dynamic constraints and introducing both planning and trajectory indexes.

TABLE I. Path Planning Optimality Statistics

Method	**Path Consistency**	**Cost Gap**		
		0.5 %	***2 %***	***5 %***
Proposed	**94 %**	84 %	89 %	100 %
GLPD	67 %	50 %	63 %	78 %
A*	50 %	43 %	49 %	60 %

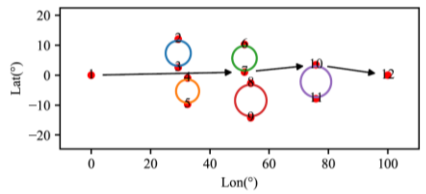

Figure 5. Path planning result.

Besides, the trajectory comparison results of one instance are shown in Fig. 6-7. The results closely match the optimal reference trajectory, with a difference of only 1 % in the total control effort. So, the trajectory generation simulator is valid and can be used to evaluate the performance and executability of the path.

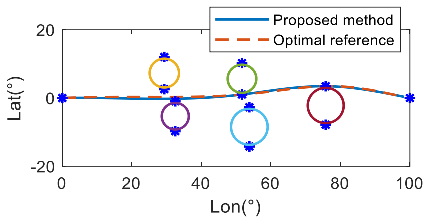

Figure 6. Longitude-latitude comparisons

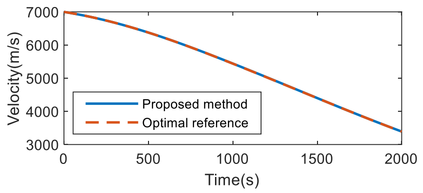

Figure 7. Velocity-time comparisons

B. Adaptability Validation

We separately test our method's performance in different no-fly zone distributions, where $N = 3$-7. At the same time, we consider uncertainties in the flight state and model, where $\Delta\theta_0 = \pm 0.5\,°$, $\Delta\varphi_0 = \pm 0.5\,°$, $\Delta h_0 = \pm 3$ km, $\Delta V_0 = \pm 50$ m/s, $\Delta\gamma_0 = 0.02\,°$, along with 5% aerodynamic deviation and 5% atmospheric density deviation.

The path planning optimality statistics tested on the 100 instances compared to the existing methods are shown in

TABLE II. In all cases, the path consistency is above 85 %, and it is much higher than existing methods. Note that the path consistency of the proposed method decreases as the number of obstacles increases, that is because the number of feasible paths increases, making the planning problem more difficult. Conversely, both the GLPD and A* algorithms exhibit the highest path consistency when N = 5, that is because the former only used instances with $N = 5$ during training, and the latter exhibits randomness in its performance under different grid resolutions and heuristic functions.

Fig. 8-9 illustrates the trajectory generation results of one instance under uncertainties. All trajectories terminate within 5 km of the target. The terminal velocity error is within 45 m/s, and the terminal altitude $h_f > 35$ km. Fig. 10 shows the impact of uncertainties on the trajectory index. The results indicate that it must balance navigation errors by sacrificing trajectory performance, thus having an impact on path planning.

The above results verified that the proposed graph learning method leverages randomization in simulation and uses the cost focus on indexes of both planning and trajectory, enabling the stable learning of a path that is effective in diverse no-fly zone distributions and uncertainties.

TABLE II. PATH CONSISTENCY STATISTICS UNDER UNCERTAINTIES

Method	**Path Consistency**				
	***N* = 3**	***N* = 4**	***N* = 5**	***N* = 6**	***N* = 7**
Proposed	**97 %**	**94 %**	**90 %**	**87 %**	**86 %**
GLPD	54 %	44 %	63 %	59 %	40 %
A*	42 %	40 %	47 %	41 %	35 %

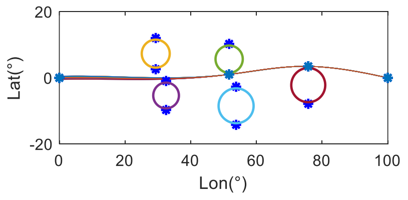

Figure 8. Trajectories under uncertainties.

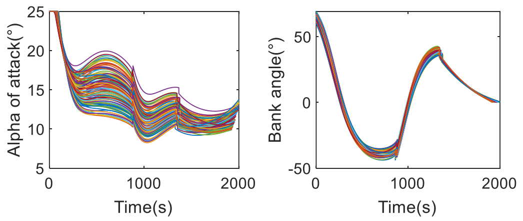

Figure 9. Control commands under uncertainties.

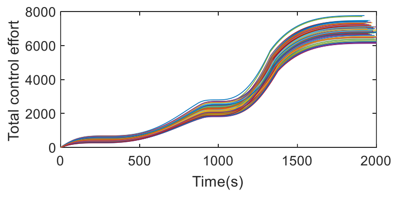

Figure 10. Trajectory performances under uncertainties.

V. CONCLUSION

In this paper, we discuss the path planning problem focusing on a hypersonic vehicle. We propose a dynamics-constrained graph learning method for this problem. The main idea is to decompose this intricate problem and integrate information from both path planning and trajectory generation into the cost function and learning algorithm. Simulation results demonstrate a more stable learning effect in diverse environments. This is because we add a mean square error index of the path, which reduces the erratic impact of implicit trajectory representations and uncertainties. Compared to existing methods, the method shows superior optimality and adaptability, proving the unignorable impact of dynamics and uncertainties on path planning. Additionally, with reliable and millisecond-level computations, this approach holds promise for practical applications in engineering.

REFERENCES

[1] R. Zhang and N. Cui, "Entry trajectory optimization with general polygonal no-fly zone constraints," *IEEE Trans. Aero. Electr. Syst.*, vol. 59, no. 6, pp. 9205–9218, Dec. 2023.
[2] X. Liu, Z. Shen, and P. Lu, "Entry trajectory optimization by second order cone programming," *J. Guid. Control Dyn.*, vol. 39, no. 2, pp. 227–241, Aug. 2016.
[3] Z. Chen, "On dubins paths to a circle," *Automatica*, vol. 117, pp. 108996, Jul. 2020.
[4] L. Winterer, S. Junges, R. Wimmer, N. Jansen, U. Topcu, J. -P. Katoen, and B. Becker, "Strategy synthesis for POMDPs in robot planning via game-based abstractions," *IEEE Trans. Automat. Contr.*, vol. 66, no. 3, pp. 1040–1054, March. 2021.
[5] R. J. M. Afonso, R. K. H. Galvão, and K. H. Kienitz, "Waypoint trajectory planning in the presence of obstacles with a tunnel-MILP approach," in *Proc. European Control Conf.*, Zurich, Switzerland, 2013, pp. 1390–1397.
[6] S. Dutta, N. Wilde, and S. L. Smith, "Informative path planning in random fields via mixed integer programming," in *Proc. 1st IEEE Conf. Decis. Control*, Cancun, Mexico, 2022, pp. 7222–7228.
[7] W. Kool, H. V. Hoof, and M. Welling, "Attention, learn to solve routing problems," *in Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–25.
[8] N. Luo, H. Wang, S. Huang, W. Gao, B. Zhong, Y. Huang, and B. Li, "Multi-UUV dynamic cooperative task planning method based on multi-objective genetic algorithm," in *Proc. 62nd IEEE Conf. Decis. Control*, Singapore, Singapore, 2023, pp. 8836–8843.
[9] R. Penicka and D. Scaramuzza. "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5719–5726, Jan. 2022.
[10] Y. Zhang, R. Zhang, and H. Li. "Mixed-integer trajectory optimization with no-fly zone constraints for a hypersonic vehicle," *Acta Astronaut.*, vol. 207, pp. 331–339, 2023.
[11] Y. Zhang, R. Zhang, and H. Li, "Online path decision of no-fly zones avoidance for hypersonic vehicles based on a graph attention network," *IEEE Trans. Aero. Electr. Syst.*, vol. 59, no. 5, pp. 5554–5567, Oct. 2023.
[12] D. K. Smith, "Network flows: theory, algorithms, and applications," *J Oper Res Soc*, vol. 45, no. 11, pp. 1340-1340, Aug. 1994.
[13] S. He, C.-H. Lee, H.-S. Shin, and A. Tsourdos, "Minimum-effort waypoint-following guidance," *J. Guid. Control Dyn.*, vol. 42, no. 7, pp. 1551–1561, Feb. 2019.
[14] Y. Song, A. Romero, M. Müller. V, Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Sci. Robot.*, vol. 8, no. 82, pp. 1462, Sep. 2023.
[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
[16] Y. Zhang, R. Zhang, and H. Li, "Graph-based path decision modeling for hypersonic vehicles with no-fly zone constraints," *Aerosp Sci Technol*, vol. 116, pp. 106857, Jun. 2021.