Tsinghua Science and Technology
Doi: 10.26599/TST.2026.9010042

Research Article

PaE: A deep reinforcement learning-based two-level online solution for uncertain moving-target traveling salesman problems

Zhao Zhang, Yihan Zhong, Chen Chen (✉), Bin Xin, and Fang Deng

Abstract This paper investigates an uncertain Moving-Target Traveling Salesman Problem (MT-TSP), in which the future motion trajectories of targets are uncertain. The objective is to maximize the number of visited targets within a bounded region while minimizing the total travel distance. To address the challenges introduced by dynamic target maneuvers, we propose a two-level online solution framework, termed PaE (Plan and Execute), which leverages the current observations of targets to solve the problem in real time. The high-level module plans the next target to visit, while the low-level module executes the corresponding access process. We develop a deep reinforcement learning–based algorithm at the high level, which employs an attention-based model to effectively exploit both the current environment state and historical visiting information when selecting the next target. At the low level, rapid target access is achieved through a lightweight prediction mechanism with continuous state updates. Extensive experiments in simulated environments demonstrate that PaE is capable of extracting key problem features to capture dynamic changes, leveraging rich contextual information to guide decision-making, and generating high-quality solutions within extremely short computation times.

Keywords moving-target traveling salesman problem, deep reinforcement learning, online solving, uncertain target trajectory

1 Introduction

The Moving-Target Traveling Salesman Problem (MT-TSP) [1] is a generalization of the classical TSP in which targets are no longer stationary but move continuously. An agent typically travels faster than the targets in order to visit or intercept them as illustrated in Fig. 1. This problem arises in many real-world scenarios, including surveillance and monitoring [2, 3], disaster response [4, 5], marine harvesting [6, 7], and missile defense [8]. Target mobility makes the travel cost between cities dependent on the current city distribution or on the elapsed time after the tour begins, which significantly increases the problem's complexity.

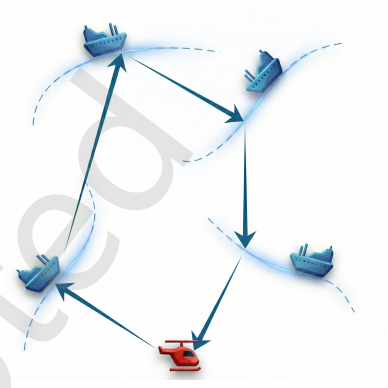


Fig. 1 Illustration of the MT-TSP.

Previous studies have proposed approximation algorithms [1, 9], exact algorithms [10, 11], and various heuristic methods [12–15] to address the MT-TSP. Reference [16] performed quantitative and qualitative comparisons of three different heuristic algorithms. The results showed that the genetic algorithm performed best in solution quality and computation time, compared with the simulated annealing and ant colony algorithm. However, in these studies, targets generally move along straight trajectories with constant velocities, or their trajectories are assumed to be known in advance. In practical settings, by contrast, target motion is often uncertain or difficult to predict.

Because the uncertain MT-TSP more closely reflects real-world scenarios, it has substantial research value, yet it remains relatively underexplored. Reference [17] modelled the uncertain MT-TSP as a two-stage stochastic programming problem and proposed an iterative randomized dynamic programming algorithm that converges to the global optimum with probability one. By pre-sampling a finite set of stochas-

• Zhao Zhang, Yihan Zhong, and Bin Xin are with the School of Automation, Beijing Institute of Technology, Beijing, 100081, China, and also with the State Key Laboratory of Autonomous Intelligent Unmanned Systems, Beijing, 100081, China. E-mail: bitzhangzhao@163.com, 3120255722@bit.edu.cn, brucebin@bit.edu.cn.
Chen Chen and Fang Deng are with the School of AI, Beijing Institute of Technology, Beijing, 100081, China, and also with the State Key Laboratory of Autonomous Intelligent Unmanned Systems, Beijing, 100081, China. E-mail: xiaofan@bit.edu.cn, dengfang@bit.edu.cn.
Manuscript received: 2026-01-05; revised: 2026-04-09; accepted: 2026-04-21

Just Accepted

tic scenarios and their associated probabilities, this method optimizes the expected objective and can handle system uncertainty. However, such approaches typically require significant computation time and rely on pre-sampled stochastic scenarios of targets that behave passively and non-adversarially. This limitation reduces their ability to respond in real time to dynamic stochastic environments. When targets exhibit adversarial behavior, their abrupt maneuvers invalidate the pre-sampled stochastic scenarios. Consequently, during execution, it becomes essential to rapidly capture system changes and dynamically optimize the routing strategy to address real-time target motion.

Recently, integrating Reinforcement Learning (RL) with Deep Neural Networks (DNNs) has emerged as a promising approach for addressing complex online decision-making problems involving uncertainty and dynamism [18–20]. RL learns decision policies directly from reward feedback through continuous interaction with the environment, while DNNs provide powerful function approximation and representation learning capabilities. This Deep Reinforcement Learning (DRL) paradigm enables the learned policies to adapt to environmental uncertainty and dynamic changes without relying on accurate prior models, while optimizing long-term cumulative performance. Consequently, in dynamic routing problems, by formulating node selection as a sequential decision-making process, DRL methods can be naturally applied to achieve online and adaptive optimization. Reference [21] highlighted the potential of RL methods to effectively tackle dynamic Vehicle Routing Problems (VRPs) by leveraging stochastic information. Once trained, RL-based approaches can generate online tours within a short computation time. [22] proposed a neural network model with an encoder–decoder architecture for solving the Dynamic Traveling Salesman Problem with Time-Dependent (DTSP-TD).

This paper is the first to explore solving the uncertain MT-TSP online using DRL method. We propose a two-level online solution framework, termed PaE (Plan and Execute), as illustrated in Fig. 2. The high-level module selects the next target to visit, while the low-level module executes the access to that target. At the high level, we modify the Group Relative Policy Optimization (GRPO) algorithm [23] and train an encoder–decoder model based on attention mechanism to decide which target should be visited next. Because targets are moving, the agent does not know the exact interception point even after receiving a high-level instruction. Therefore, the low-level module is responsible for carrying out the instruction by predicting target motion and updating the plan in real time. We implement rapid target access through a lightweight prediction mechanism with continuous state updates.

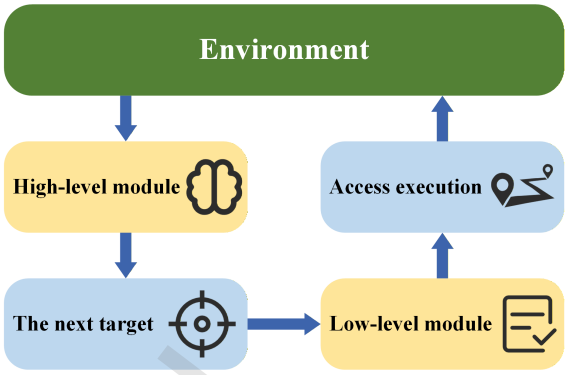


Fig. 2 Illustration of PaE. The high-level module selects the next target to visit, while the low-level module executes the access to that target.

Our main contributions are summarized as follows:

(1) We propose a two-level online solution for real-time decision-making in the uncertain MT-TSP, where the high-level module plans the target sequence and the low-level module executes the corresponding access.

(2) We design an encoder–decoder architecture based on attention mechanism and enhance it with a Long Short-Term Memory (LSTM) network to better capture historical trajectory context.

(3) We modify the GRPO algorithm by removing the KL-divergence penalty term and incorporating an entropy regularization term to encourage early-stage exploration, which enables more effective training of the proposed model.

(4) We develop a surrogate model to approximate the outcomes of target-selection decisions during agent–environment interactions, thereby significantly accelerating the training process.

The remainder of this paper is organized as follows. Section 2 formulates the problem as a Markov Decision Process (MDP). Section 3 details the models and specifics in PaE. Section 4 describes the reinforcement learning framework. Section 5 reports the numerical experimental results. Finally, Section 6 concludes the paper and discusses future work.

2 Problem Formulation in MDP

This paper investigates the uncertain MT-TSP. A set of targets $\{1, 2, ..., n\}$ enter a bounded region with side length B_r, move within it for a period of time, and eventually leave. Their trajectories inside the region are uncertain. An agent departs from a depot, intercepts the targets, and then returns to the depot. The objective is to maximize the number of accessed targets while minimizing the total travel distance.

Just Accepted

We assume that the agent can observe target positions in real time, and that only the 10 most recent trajectory points of each target with an interval of $\Delta\tau$ are stored. We formulate the uncertain MT-TSP as an MDP [24] with multiple decision steps, defined as follows.

- **Current state** s_t. The current state includes
  (1) real-time observations of target positions materialized as updated historical trajectories $x_i = [L_i^{-9}, L_i^{-8}, ..., L_i^0, D_i]$, where L_i represents the stored historical coordinates of target i and D_i represents the distance that target i will move to reach the boundary at its current speed;
  (2) the dynamically growing set of historical access points $\{l_0, l_1, ..., l_c\}$ visited by the agent;
  (3) the visitation status of all targets, represented as a mask indicating which targets have already been visited.
- **Action** a_t^π. The action is the next target to visit selected by policy π.
- **State transition**. The transition from the current state s_t to the next state s_{t+1} corresponds to the execution of the selected action. In our PaE framework, this is realized by the low-level module carrying out the high-level decision.
- **Objective function**. The objective is to maximize the number of accessed targets while minimizing the total travel distance. Since the number of accessed targets is significantly more important, we combine the two objectives through a weighted format.

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{n-1}(N_t - 1/B_r \times L_t)\right]. \tag{1}$$

where N_t and L_t represent the access score and travel distance, respectively, after the state transition at the t-th decision step. If the visit is successful, N_t is 1; otherwise, it is 0. The episodic reward is defined accordingly. With the weight $1/B_r$, the reward contribution of successfully accessing one target is approximately equivalent to traversing a distance equal to the side length of the region. Since the region size is significantly larger than inter-target distances, this scaling encourages the agent to prioritize visiting more targets rather than excessively minimizing travel distance.

By modeling the uncertain MT-TSP as an MDP, we can apply reinforcement learning to learn a near-optimal policy that generates decisions in real time, thereby enabling an online solution to this problem.

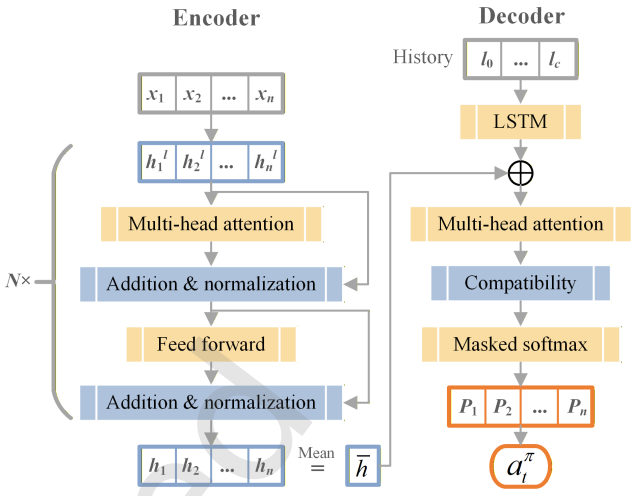


Fig. 3 Structure of our encoder–decoder model. The encoder processes the input node states through N attention layers to generate a graph embedding. Historical visitation information is encoded by an LSTM to produce a context embedding. These two embeddings are then combined and fed into the decoder, which outputs the selection probabilities for all nodes.

3 Models and Specifics in PaE

This section provides a detailed description of the PaE framework. Section 3.1 presents the encoder–decoder model used in the high-level module, while Section 3.2 introduces the execution process implemented in the low-level module.

3.1 Encoder–decoder model in the high-level module

We define a problem instance q as a graph containing n targets, where each target i is associated with a feature vector x_i representing its historical trajectory coordinates, as described in Section 2. A tour $\mathbf{a} = \{a_0, ..., a_{n-1}\}$ is defined as an ordered sequence of targets. Our attention-based encoder–decoder model specifies a stochastic policy $\pi(\mathbf{a}|q)$ that generates a tour $\mathbf{a}$ given an instance q. The policy is parameterized by θ and, by the chain rule, can be expressed as the product of the conditional probabilities at each decision step.

$$\pi_\theta(\mathbf{a}|q) = \prod_{t=0}^{n-1} \pi_\theta(a_t|q, a_{0:t-1}). \tag{2}$$

The structure of the encoder–decoder model is illustrated in Fig. 3. Following the architecture used in [25], the encoder first generates initial feature embeddings for all input nodes based on the current observations of the targets through a learned linear projection with parameters $\mathbf{W}^X$ and $\mathbf{b}^X$.

$$h_i^1 = \mathbf{W}^X x_i + \mathbf{b}^X. \tag{3}$$

These embeddings are then processed through N attention layers, each consisting of a multi-head attention (MHA) layer

Just Accepted

followed by a feed-forward (FF) layer, with a skip connection and a batch normalization (BN) applied after each component. We denote with h_i^l the node embeddings produced by layer $l \in \{1, ..., N\}$.

$$\hat{h}_i^l = \text{BN}^l(h_i^l + \text{MHA}_i^l(h_1^l, ..., h_n^l)). \tag{4}$$

$$h_i^{l+1} = \text{BN}^l(\hat{h}_i^l + \text{FF}^l(\hat{h}_i^l)). \tag{5}$$

The final node embeddings are averaged to produce a graph embedding, which is passed to the decoder.

$$\bar{h} = \frac{1}{n}\sum_{i=1}^{n} h_i. \tag{6}$$

For the decoder input, we also incorporate the agent's historical visited points to provide contextual information and enrich the state representation. Since the set of visited points grows as the decision process progresses, we employ an LSTM network [26] to cope with this variable-length sequence and capture long-range temporal dependencies. At each decision step, the LSTM takes as input the agent's current position (i.e., the location reached after the previous access, or the depot at the first step) and updates its hidden and cell states to produce a context embedding. This context embedding is then combined with the graph embedding to form a query vector q_c.

$$q_c = \bar{h} + \text{LSTM}(l_0, ..., l_c). \tag{7}$$

A multi-head attention layer processes this query to generate a new query vector q_{c+1}, which is similar to the glimpse described by [27].

$$q_{c+1} = \text{MHA}_c(q_c). \tag{8}$$

Then we use the q_{c+1} to compute compatibility score u_i for each target. Because previously visited targets cannot be selected again and do not affect future decisions, we mask out all visited nodes. The result is clipped within $[-C, C]$ using tanh.

$$u_i = \begin{cases} -\infty, & \text{if masked} \\ C \cdot \tanh\left(\frac{q_{c+1}^T k_i}{\sqrt{d_k}}\right), & \text{otherwise.} \end{cases} \tag{9}$$

where k_i is the key and d_k is the dimension of key. Finally, a softmax layer transforms the compatibility scores into a probability distribution p_i over the unvisited targets, from which the next target is selected either greedily or by sampling.

$$p_i = \pi_\theta(a_t = i | s_t, a_{0:t-1}) = \frac{e^{u_i}}{\sum_i e^{u_i}}. \tag{10}$$

3.2 Execution process in the low-level module

After the high-level module selects a target, the low-level module begins executing the access process to that target. The low-level module also receives real-time observations of targets. We adopt a lightweight prediction–update mechanism that enables rapid target access. We found that this approach performs well and effectively handles uncertain and dynamically maneuvering target trajectories.

Assume that the low-level module is instructed to access target i. At this moment, the observation x_i contains the most recent 10 trajectory points of target i. We compute the velocity at each recorded time $\mathbf{v}_i = [v_i^{-8}, ..., v_i^0]$ by taking the difference of x_i and dividing by the time interval $\Delta\tau$.

$$\mathbf{v}_i = diff(x_i)/\Delta\tau. \tag{11}$$

We assign a weight $\mathbf{w}_i = [w_i^{-8}, ..., w_i^0]$ to each velocity to reflect its relative importance at different time steps. Since more recent velocities carry greater influence on future motion, we assign higher weights to them. A weighted sum of these velocities yields the predicted velocity of the target.

$$\hat{v}_i = sum(\mathbf{v}_i \mathbf{w}_i). \tag{12}$$

We then assume that the target i will continue moving linearly with this predicted velocity $\hat{v}_i$, while the agent moves along the fastest interception path. Following the prediction process based on dynamics as described in [5], given the current positions of both the target and the agent (L_i^0 and L_a^0), the predicted target velocity $\hat{v}_i$, and the agent's constant speed v_a, we solve the following equation to obtain the moving time τ.

$$(||\hat{v}_i||_2^2 - v_a^2)\tau^2 + ||L_i^0 - L_a^0||_2^2 + 2(\hat{v}_i \cdot (L_i^0 - L_a^0))\tau = 0. \tag{13}$$

Then, we can compute the predicted access point

$$\hat{L}_a^\tau = L_i^0 + \hat{v}_i \tau. \tag{14}$$

we can obtain the unit direction vector towards the predicted access point $\hat{L}_a^\tau$.

$$u_a^0 = \frac{\hat{L}_a^\tau - L_a^0}{||\hat{L}_a^\tau - L_a^0||_2}. \tag{15}$$

The agent moves along this direction vector for a duration of $\Delta\tau$ and arrives at a new position

$$L_a^1 = L_a^0 + u_a^0 v_a \Delta\tau. \tag{16}$$

The agent then updates its observation x_i and repeats the prediction–update cycle until the target enters the capture radius and is successfully visited. Note that if the target leaves the bounded region, the access process is terminated early. Whether the access succeeds directly affects the access score N_t for the current decision step.

Executing the access process for the selected target completes the state transition described in Section 2, after which the system proceeds to the next state and the high-level module selects the next target. After the entire tour $\mathbf{a}$ is completed, the PaE outputs the total access score and the total travel distance.

Just Accepted

4 Reinforcement Learning Algorithm

4.1 Modified GRPO

To address the substantial memory and computational burden caused by training another value model in Proximal Policy Optimization (PPO) [28], DeepSeek introduced GRPO [23], which eliminates the need for additional value function approximation. Instead, it uses the average reward of multiple sampled outputs in response to the same question as a baseline. Their rationale is that in the training of Large Language Model (LLM), rewards are often assigned only after the final token is generated, making it difficult to train a value model that accurately estimates each token. This setting is similar to ours, where the final reward (Equation (1)) is obtained only after an entire tour is completed.

Specifically, for each problem instance q, GRPO samples a group of tours $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{\{\mathbf{a}_g\}_{g=1}^{G} \sim \pi_{\theta_{old}}(\mathbf{a}|q)} \frac{1}{G} \sum_{g=1}^{G} \frac{1}{n} \sum_{t=0}^{n-1} \Big\{ min\big[r_{g,t}\hat{A}_{g,t}, clip(r_{g,t}, 1-\epsilon, 1+\epsilon)\hat{A}_{g,t} \big] - \beta \mathbb{D}_{KL}[\pi_\theta || \pi_{ref}] \Big\}. \tag{17}$$

$$r_{g,t} = \frac{\pi_\theta(a_{g,t}|q, a_{g,0:t-1})}{\pi_{\theta_{old}}(a_{g,t}|q, a_{g,0:t-1})}. \tag{18}$$

where ϵ and β are hyper-parameters. $r_{g,t}$ represents the likelihood ratio between the new and old policies (i.e., the importance sampling weight). GRPO uses the clipping operation on it following PPO to prevent excessively large updates. $\hat{A}_{g,t}$ denotes the relative advantage computed from the rewards of sampled outputs in each group. $\mathbb{D}_{KL}$ is the KL divergence between the trained model and a reference model, used to prevent over-optimization. In GRPO, the reference model π_{ref} is the initial supervised fine-tuned (SFT) model.

In our setting, however, no reference policy exists since there is no SFT stage, so the KL penalty is removed. Unlike Reinforcement Learning from Human Feedback (RLHF) [29] scenarios where KL penalty prevents policy collapse, our task uses a well-defined reward and low-dimensional action space, and stability is ensured via GRPO's clipped objective and reward normalization, allowing safe exploration without KL constraints. Empirically, prior work [30] has reported that GRPO without KL penalty can achieve better performance, suggesting that greater freedom in parameter updates may help the policy explore more effective solutions. Additionally, we include an entropy regularization term with an adaptive coefficient λ to further encourage exploration during early training. λ varies linearly with the training epochs as

$$\lambda = \lambda_{\max} - (\lambda_{\max} - \lambda_{\min}) \times (e_i / e_I), \tag{19}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ denote the maximum and minimum values of λ, respectively; e_i denotes the current epoch, and e_I denotes the total number of epochs. Consequently, we maximize the following modified objective:

$$\mathcal{J}'_{GRPO}(\theta) = \mathbb{E}_{\{\mathbf{a}_g\}_{g=1}^{G} \sim \pi_{\theta_{old}}(\mathbf{a}|q)} \frac{1}{G} \sum_{g=1}^{G} \frac{1}{n} \sum_{t=0}^{n-1} \Big\{ min\big[r_{g,t}\hat{A}_{g,t}, clip(r_{g,t}, 1-\epsilon, 1+\epsilon)\hat{A}_{g,t} \big] - \lambda \pi_\theta log(\pi_\theta) \Big\}. \tag{20}$$

For computing group-relative advantages, we found that outcome supervision performs better than process supervision [23]. Concretely, for each problem q, we sample a group of tours $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_G\}$ from the old policy and compute their rewards $\mathbf{r} = \{r_1, r_2, ..., r_G\}$ using Equation (1) correspondingly. These rewards are normalized by subtracting the group mean and dividing by the group standard deviation. Outcome supervision provides a normalized reward only after a tour is completed, and assigns the same advantage value to all decision steps within that tour.

$$\hat{A}_{g,t} = \tilde{r}_g = \frac{r_g - mean(\mathbf{r})}{std(\mathbf{r})}, \ \forall t \in \{0, ..., n-1\}. \tag{21}$$

The modified GRPO training procedure is summarized in Algorithm 1. At each step, the old model $\pi_{\theta_{old}}$ is updated, and a batch of problem instances $\mathbf{D}_b$ is sampled from problem set $\mathbf{D}$. For each problem q, a group of tours $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_G\}$ is generated using the old policy. Rewards $\mathbf{r}$ for all sampled tours are computed using Equation (1). Group-relative advantages $\hat{A}_{g,t}$ are then assigned to each decision $a_{g,t}$. The policy model is updated by maximizing the modified GRPO objective in Equation (20). After training for I epochs, the final policy model π_θ is obtained.

4.2 Surrogate model approximation

During the RL training process, each state transition after a decision step requires executing the target access process via the low-level module, as described in Section 2. This interaction with the environment is time-consuming and difficult to parallelize, since the time required to access different targets varies across environments. To address this issue, we construct a surrogate model to approximate the environment interaction outcomes and thereby accelerate training. However, for validation and evaluation, we still achieve real-time target access through the low-level module introduced in Section 3.2.

Just Accepted

Algorithm 1 Modified GRPO

Input: initial policy model $\pi_{\theta_{init}}$; problems $\mathbf{D}$; hyper-parameters ϵ, λ, μ

Output: π_θ

policy model $\pi_\theta \leftarrow \pi_{\theta_{init}}$;
for epoch = 0,...,I **do**
  for step = 0,...,M **do**
    Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$;
    Sample a batch problems $\mathbf{D}_b$ from $\mathbf{D}$;
    Sample G tours $\{\mathbf{a}_g\}_{g=1}^{G} \sim \pi_{\theta_{old}}(\cdot|q)$ for each problem $q \in \mathbf{D}_b$;
    Compute rewards $\{r_g\}_{g=1}^{G}$ for each sampled tour $\mathbf{a}_g$ using Equation (1);
    Compute advantages $\hat{A}_{g,t}$ for each $\mathbf{a}_g$ through group relative advantage estimation;
    for update = 0,...,μ **do**
      Update the policy model π_θ by maximizing the modified GRPO objective in Equation (20);
    end for
  end for
end for

The surrogate model takes as input the agent's current position (2D), scalar speed (1D), and the target's trajectory observation (20D). These components are concatenated into a 23-dimensional input vector o_i. The model consists of three hidden layers with 64, 128, and 64 dimension, respectively. Each layer uses ReLU activation and applies batch normalization and dropout. The model outputs the agent's updated position (2D) after completing the target access process.

It is worth noting that, to maintain generality and avoid introducing environment-specific assumptions, the distance-to-boundary term D_i is not used in the surrogate model or during the training interaction loop. The training process relies on the surrogate model to approximate environment interaction, and the final transition outcomes are computed after the surrogate prediction. This design ensures that omitting D_i from the surrogate input does not affect training fidelity, while preserving the generality and efficiency of the surrogate approximation.

We collect a large amount of simulation data to train the surrogate model. Specifically, for each sample, we randomly select a goal from the problem set $\mathbf{D}$ and randomly initialize the agent's position within the bounded region. Then, we execute the low-level module described in Section 3.2 to visit the target. After completion, we record the input data and the corresponding labels. In total, we obtain 400k samples, which are split into training, validation, and test sets. The surrogate model is trained using the mean squared error (MSE) loss. Fig. A1 in Appendix A illustrates the iteration curve of the training and validation losses during the training process. The model parameters eventually converge to a satisfactory level.

The trained surrogate model is used to replace the low-level module in the RL training process. When the high-level module selects a target, the corresponding input vector o_i is fed into the surrogate model to obtain the agent's updated position. The distance between the old and new positions yields the traveled length L_t. Dividing L_t by the agent's speed gives the moving time τ, and we determine whether the target reaches its terminal state within this interval τ to compute the access score N_t. This process completes the approximate state transition step.

In practice, the surrogate model may introduce approximation errors. However, such errors can be interpreted as a form of environmental noise during training. From a reinforcement learning perspective, moderate noise in state transitions does not necessarily degrade performance; instead, it can improve the robustness and generalization ability of the learned policy by preventing overfitting to a deterministic environment.

Moreover, the final state transition outcomes, including the travel distance L_t and the access score N_t, are computed after the surrogate inference step based on physically meaningful rules. This further mitigates the impact of surrogate approximation errors on the reward signal.

Importantly, the surrogate model significantly accelerates the training process by replacing the computationally expensive low-level simulation, enabling efficient large-scale training that would otherwise be impractical. Empirically, we observe that the trained policy maintains strong performance when evaluated using the true environment dynamics.

5 Experiments

We conduct numerical experiments based on the benchmark dataset generated in Section 5.1 to evaluate the performance of our PaE and baseline algorithms. In the RL training with modified GRPO, we set the learning rate to 1e-6, the batch size to 256, the group size to 32, and the number of update iterations to 10. $\lambda_{\max}$ and $\lambda_{\min}$ are set to 0.1 and 0.01, respectively. Our PaE is programmed with Pytorch, which is executed on a workstation with Intel(R) Xeon(R) Gold 6258R CPU at 2.70 GHz and NVIDIA GeForce RTX 4090 GPU.

5.1 Benchmark dataset

To evaluate the performance of the proposed methods, we construct an MT-TSP benchmark dataset. This dataset contains curve trajectory points of numerous targets moving within

Just Accepted

Table 1 Experimental results of different methods at different problem scales on the test set T00.

	Method	Gurobi	ALNS	NN	AM	PaE-D	PaE-LSTM	PaE-ER	PaE
10	Obj.	**7.72**	7.25	6.91	6.96	6.95	7.08	6.95	7.16
	Gap (%)	**0.00**	-6.11	-10.43	-9.80	-9.90	-8.31	-9.96	-7.15
	Time (s)	335	523	0.05	0.06	0.06	0.06	0.06	0.06
30	Obj.	18.47	13.70	19.28	19.01	19.82	21.18	**22.07**	**22.07**
	Gap (%)	-16.31	-37.94	-12.67	-13.89	-10.20	-4.04	**0.00**	**0.00**
	Time (s)	1017	1261						
50	Obj.	21.16	15.26	30.06	29.98	30.81	33.46	34.11	**34.21**
	Gap (%)	-38.14	-55.39	-12.12	-12.35	-9.95	-2.18	-0.29	**0.00**
	Time (s)	2524	1515						

a $10 \times 10km^2$ area. Fig. B1 in Appendix B illustrates the trajectories of 10 targets. Appendix B details the construction process of the dataset.

We divide the dataset into a training set and a test set. The training set contains trajectory data for 10k targets, while the test set T00 includes 100 targets simulating pre-recorded trajectory data. To further simulate real-world target maneuvers, we rotate the latter portion of each trajectory in the test set T30, T60, and T90 by 30°, 60°, and 90°, respectively, as shown in Fig. B2 in Appendix B. All problem instances used in our experiments are sampled from the above datasets.

5.2 Baseline algorithms

To the best of our knowledge, no existing method directly addresses the uncertain MT-TSP. We therefore introduce two offline and four online baseline algorithms to compare against our PaE framework.

(1) Commercial solver (Gurobi): Gurobi is a state-of-the-art mathematical programming solver. For the pre-recorded test set T00, we use Gurobi to obtain offline optimal or near-optimal solutions.

(2) Adaptive Large Neighborhood Search (ALNS) [5]: ALNS is a state-of-the-art heuristic method for solving MT-TSP. We replace its motion prediction component with our low-level module (Section 3.2) to compute solution fitness. ALNS is applied to the pre-recorded test instances in an offline manner.

(3) Online-Prediction Gurobi (Gurobi-OP): For the test sets (T30, T60, and T90) simulating real-world dynamics, we develop an online version of Gurobi. Whenever the next target must be selected, Gurobi-OP runs Gurobi under a time limit to produce a future visiting sequence, and only the next target in the sequence is executed. The process is then repeated in a rolling-horizon manner. Since Gurobi-OP cannot observe the true future target trajectories, it relies solely on the agent's observations. The future target positions are predicted using Equations (11) and (12) to evaluate solution quality.

(4) Online-Prediction ALNS (ALNS-OP): Similar to Gurobi-OP, we develop an online version of ALNS that operates in a rolling-horizon manner to handle real-world test instances.

(5) Nearest Neighbor (NN): NN is a simple greedy online heuristic that selects, at each step, the nearest unvisited target as the next destination.

(6) Attention Model (AM) [25]: AM is a state-of-the-art DRL-based model capable of solving online routing problems such as TSP and Capacitated Vehicle Routing Problem (CVRP).

In addition, to demonstrate the effectiveness of our encoder–decoder model and training algorithm, we construct three ablation baselines by removing specific components:

(1) PaE-D: removes the distance-to-boundary feature D_i from the target input representation x_i.

(2) PaE-LSTM: replaces the LSTM context encoder with the commonly used embedding formed in some references [22, 25] by concatenating the first and last access points.

(3) PaE-ER: removes the entropy regularization term from the modified GRPO objective (Equation (20)).

5.3 Experimental results

We first conduct experiments on the test set T00. Since T00 simulates pre-recorded trajectories representing the most likely target motions, sufficient time is available for offline optimization. Table 1 reports the results of two offline methods (Gurobi and ALNS), the online heuristic NN, and other DRL-based approaches. The objective value corresponds to the average of Equation (1) over 50 different instances. The percentage gap relative to the best-performing method is calculated. For online methods, we report only a single line of computation time, as the time required to output the next target is approximately constant and independent of the problem scale.

Just Accepted

Table 2 Experimental results of different methods at different problem scales on the test set T30, T60, and T90.

		Method	Gurobi	ALNS	Gurobi-OP	ALNS-OP	NN	AM	PaE-D	PaE-LSTM	PaE-ER	PaE
		Time (s)			3.92	0.79	0.04	0.06	0.06	0.06	0.06	0.06
10	T30	Obj.	**7.26**	5.90	5.81	6.58	6.12	6.39	6.38	6.57	6.57	6.60
		Gap (%)	**0.00**	-18.82	-20.07	-9.38	-15.71	-12.05	-12.12	-9.55	-9.58	-9.12
	T60	Obj.	5.51	3.95	4.88	5.35	5.19	5.20	5.27	5.62	5.77	**5.79**
		Gap (%)	-4.72	-31.80	-15.74	-7.56	-10.33	-10.19	-9.01	-2.85	-0.37	**0.00**
	T90	Obj.	3.60	2.82	4.21	4.17	4.08	3.92	4.21	4.31	4.36	**4.56**
		Gap (%)	-21.19	-38.21	-7.74	-8.75	-10.54	-14.13	-7.76	-5.67	-4.46	**0.00**
30	T30	Obj.	17.59	9.67	7.40	11.48	18.42	19.23	19.16	20.60	**21.96**	21.92
		Gap (%)	-19.87	-55.96	-66.28	-47.73	-16.11	-12.41	-12.72	-6.20	**0.00**	-0.16
	T60	Obj.	15.43	6.02	7.21	8.54	15.34	15.03	15.93	18.61	19.30	**19.58**
		Gap (%)	-21.18	-69.27	-63.15	-56.37	-21.64	-23.23	-18.64	-4.97	-1.44	**0.00**
	T90	Obj.	10.61	4.47	5.97	6.30	11.91	11.15	12.51	15.01	14.14	**15.21**
		Gap (%)	-30.26	-70.58	-60.75	-58.59	-21.66	-26.67	-17.74	-1.32	-7.05	**0.00**
50	T30	Obj.	22.09	5.26	8.55	12.71	27.93	28.10	29.24	30.62	31.05	**31.71**
		Gap (%)	-30.33	-83.43	-73.04	-59.93	-11.92	-11.39	-7.79	-3.42	-2.09	**0.00**
	T60	Obj.	17.34	2.97	8.19	9.53	23.19	22.40	23.62	25.42	25.95	**27.00**
		Gap (%)	-35.79	-88.99	-69.65	-64.71	-14.12	-17.04	-12.53	-5.85	-3.90	**0.00**
	T90	Obj.	12.98	1.90	6.92	6.80	18.38	16.26	18.88	22.33	22.24	**23.06**
		Gap (%)	-43.72	-91.74	-70.02	-70.52	-20.30	-29.50	-18.12	-3.17	-3.58	**0.00**

As shown in Table 1, for problem instances with 10 targets, Gurobi is able to compute superior solutions within the given time limit. However, as the problem scale increases, the solver fails to produce high-quality solutions even with extended time limits, highlighting the inherent complexity of the problem. In contrast, our PaE achieves the best performance at larger scales and remains competitive with offline methods at the small scale, while requiring only approximately 0.06 seconds to determine the next target. This demonstrates both the effectiveness and efficiency of our online MT-TSP solution framework. The heuristic method ALNS performs poorly, especially at larger scales, because it struggles to balance exploration and exploitation and thus cannot identify high-quality solutions within limited computation time.

For the ablation baselines, PaE-ER exhibits a slight performance degradation compared to PaE, indicating that entropy regularization enhances early-stage exploration and guides the policy toward more promising optimization directions. PaE-LSTM suffers a more pronounced performance drop, underscoring the importance of using an LSTM to encode historical trajectory context. By leveraging LSTM-based context encoding, the agent can better capture intrinsic relationships among historical trajectories, starting points (i.e., upcoming destinations), and future targets, leading to more informed decisions. Notably, PaE-D, which removes the distance-to-boundary feature from the target representation, experiences a substantial performance drop, degrading to the level of AM and even NN. This result highlights that the distance-to-boundary is a critical handcrafted feature. Agents equipped with this information are more inclined to prioritize targets that are about to exit the operational region, thereby increasing the number of successfully visited targets. Without this feature, PaE-D fails to identify such urgent targets, resulting in frequent target loss and only marginally outperforming NN.

However, the pre-recorded test set T00 inherently favors offline algorithms such as Gurobi and ALNS. These methods exploit deterministic trajectory information and can repeatedly evaluate fitness using our low-level module (Section 3.2) during offline optimization. In real-world scenarios, target trajectories are uncertain and may deviate significantly from pre-estimated paths, rendering offline-computed fitness inaccurate and leading to poor execution performance. Therefore, we further evaluate all methods on three additional test sets: T30, T60, and T90, which simulate target maneuvers by rotating the latter portions of trajectories in T00 by 30°, 60°, and 90°, respectively. The experimental results across different problem scales and test sets are summarized in Table 2. The results for Gurobi and ALNS correspond to directly executing their offline solutions obtained from T00 on the new test sets; therefore, no computation time is reported for these methods. For other online approaches, the reported computation time corresponds to a single decision step for selecting the next target.

The results indicate that, except for small-scale problems with mild trajectory changes (30° rotation), where Gurobi's

Just Accepted

offline solution remains applicable, PaE consistently outperforms baselines across almost all other scenarios. This demonstrates that our online solution framework effectively handles trajectory uncertainty and dynamic target maneuvers. In contrast, the execution performance of offline solutions from Gurobi and ALNS deteriorates sharply as trajectory deviations increase. Their online-prediction variants perform reasonably at small scales but become ineffective at larger scales due to insufficient computation time, resulting in unacceptable solution quality.

Consistent with Table 1, both PaE-ER and PaE-LSTM show moderate performance degradation, while PaE-D exhibits the most severe drop. These results further confirm the positive contributions of entropy regularization and LSTM-based context encoding, as well as the critical importance of the distance-to-boundary feature.

It is also worth noting that, for all methods, solution quality degrades as trajectory deviations increase across all problem scales. This observation highlights the extreme difficulty of the uncertain MT-TSP: stronger target maneuverability significantly complicates the task of maximizing the number of visited targets while minimizing travel distance under limited computation time. At the current stage, PaE appears to be a viable answer to this challenge.

Fig. 4 visualizes the accessing process for a representative instance with 10 targets. The left panel shows the solutions obtained by Gurobi and PaE on the test set T00, while the right panel illustrates their execution on the test set T30. Gurobi's offline solution performs well when target trajectories remain unchanged in T00. However, when trajectories deviate in T30, its execution fails to visit Target 4, which exits the bounded region due to dynamic maneuvering. In contrast, although PaE does not achieve the optimal solution on T00, it rapidly identifies targets that are about to leave the region and prioritizes them accordingly. As a result, PaE successfully visits all 10 targets on T30 and even attains a higher objective value in this scenario.This example clearly demonstrates that PaE can effectively handle dynamic target maneuvers when solving the uncertain MT-TSP in an online manner.

Fig. 5 depicts the iteration curve of the batch-averaged reward over training step for problem size 10. The curve shows that the modified GRPO algorithm enables stable convergence of the model parameters. After a slight drop around 17,000 steps, the reward continues to increase and converges to a higher value, indicating that enhanced exploration in the early training stage facilitates convergence to better policies.

In summary, the proposed PaE framework effectively extracts key problem features to capture dynamic changes, and encodes rich historical trajectory context to guide decision-making. Trained with the modified GRPO algorithm, PaE benefits from improved early exploration and converges toward higher-quality policies. Extensive experimental results demonstrate that PaE is both efficient and robust in handling trajectory uncertainty and dynamic target maneuvers, making it a strong solution for the uncertain MT-TSP.

Table 3 Experimental results of different trained models at different problem scales on the test set T00 .

	Model	PaE-10	PaE-30	PaE-50
10	Obj.	**7.16**	7.13	7.07
	Gap (%)	**0.00**	-0.51	-1.28
30	Obj.	19.97	**22.07**	21.15
	Gap (%)	-9.55	**0.00**	-4.18
50	Obj.	30.73	32.73	**34.21**
	Gap (%)	-10.17	-4.32	**0.00**

5.4 Generalization analysis

In practical scenarios, a trained policy is expected to generalize across different problem scales to handle varying numbers of targets. To evaluate the scalability and generalization capability of PaE, we conduct experiments by applying PaE models trained on a specific problem scale to instances with other scales.

The experimental results on test set T00 for different problem sizes are summarized in Table 3, where the objective value is averaged over 50 instances. PaE-10, PaE-30, and PaE-50 denote PaE models trained on problem sizes of 10, 30, and 50 targets, respectively. As shown in the table, each model exhibits varying degrees of performance degradation when applied to other problem scales, reflecting the inherent dependence of RL methods on the training problem domain.

Moreover, we observe that the greater the discrepancy between the training and testing scales, the more severe the performance degradation. For example, PaE-50 performs worse than PaE-30 when solving instances with 10 targets, while PaE-10 underperforms PaE-30 on instances with 50 targets. This phenomenon can be attributed to the fact that the optimal policy spaces differ across problem scales. RL training tends to overfit to the state distribution and dynamic characteristics specific to the training scale, which leads to reduced generalization, especially when the scale mismatch becomes large.

Despite this limitation, PaE-30 and PaE-50 still outperform all other methods reported in Table 1 across all tested scales. In addition, when solving problems ranging from 10 to 50 targets, PaE-30 provides a more robust trade-off between performance and generalization. These results suggest that,

Just Accepted

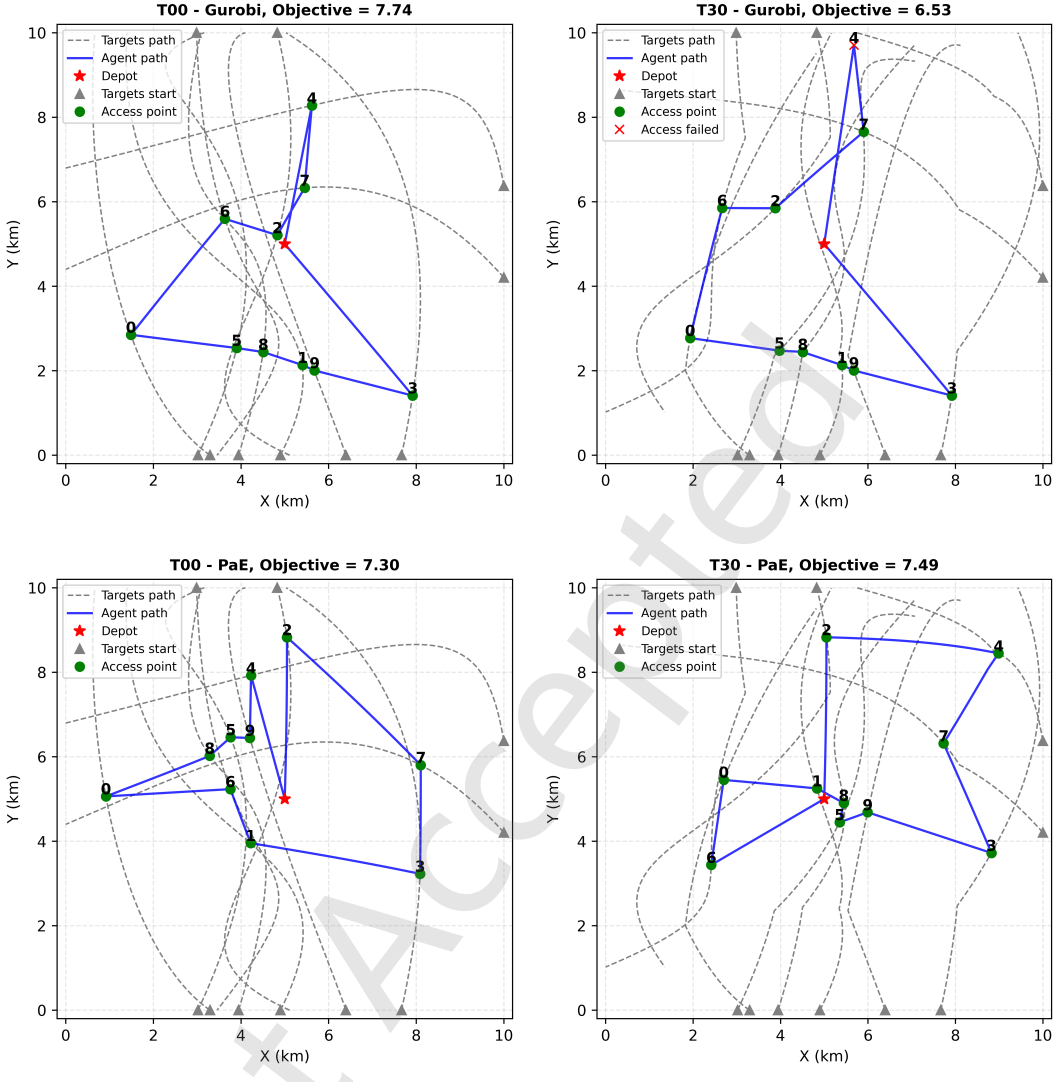


Fig. 4 Accessing process for a representative instance with 10 targets. The left panel shows the solutions obtained by Gurobi and PaE on the test set T00, while the right panel illustrates their execution on the test set T30. When Gurobi's offline solution is executed in T30, it fails to visit Target 4, which exits the bounded region due to dynamic maneuvering. In contrast, PaE successfully visits all 10 targets on T30.

in practical applications, deploying a policy trained on a problem scale close to the expected target size is a prudent and effective strategy.

6 Conclusions

This paper addresses an uncertain MT-TSP and proposes a two-level online solution framework, PaE, to tackle the challenges induced by dynamic target maneuvers. PaE operates in real time by leveraging current observations of targets, where the high-level module plans the next target to visit and the low-level module executes the corresponding access process.

At the high level, we develop an attention-based encoder–decoder model and introduce an LSTM to enhance the extraction of contextual information from historical trajectories. The model is trained using a modified GRPO algorithm. At the low level, rapid target access is achieved through a lightweight prediction mechanism with continuous state updates. In addition, we construct a surrogate model to approximate decision–environment interactions, significantly accelerating the training process.

Extensive simulation experiments across diverse testing environments demonstrate that PaE is able to extract key problem features to capture dynamic changes, exploit rich contextual information to guide subsequent decisions, and generate high-quality solutions with very low computational latency.

Despite these promising results, some limitations remain. First, some components of the framework, such as reward

Just Accepted

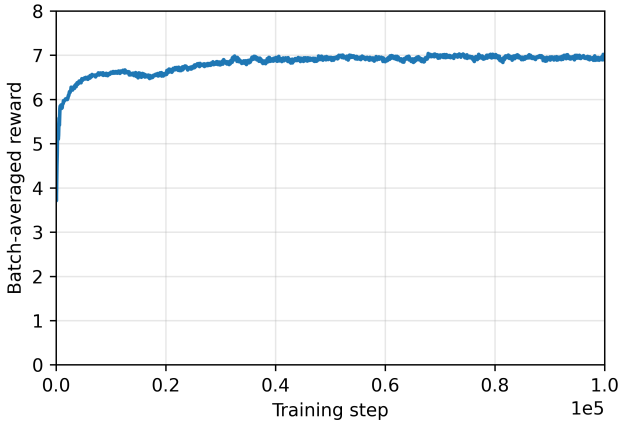


Fig. 5 Iteration curve of the batch-averaged reward over training step for problem size 10.

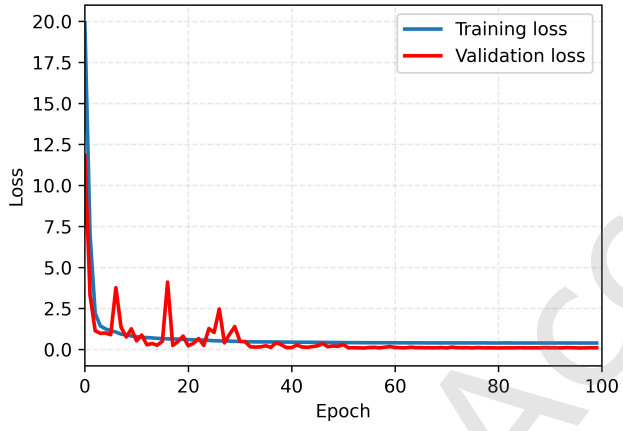


Fig. A1 Iteration curve of the training and validation losses of the surrogate model.

weighting and feature design (e.g., distance-to-boundary), rely on problem-specific insights, which may limit generalization. Second, the current framework is designed for a single-agent setting and does not explicitly consider interactions among multiple decision-makers.

In the future, we plan to extend this work to the Moving-Target Multiple Traveling Salesman Problem (MT-MTSP) [8, 31], where multiple agents collaboratively visit targets. In such settings, addressing the non-stationarity [32, 33] arising from interactions among agents will pose a significant challenge. Additionally, improving the stability and scalability of multi-agent reinforcement learning [34, 35], as well as designing more generalizable model architectures that can adapt to varying problem scales and dynamics, remains an important direction for future research.

Appendix A

We construct a surrogate model for approximating environment interactions. Fig. A1 illustrates the iteration curve of the

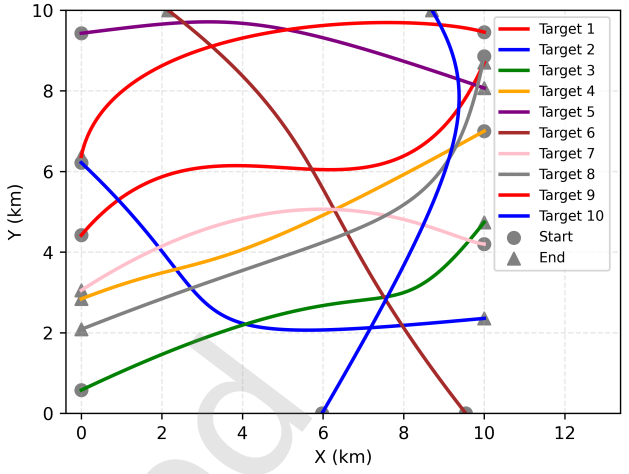


Fig. B1 Illustration of trajectories of 10 targets within the bounded region. The gray circles denote start points and gray triangles denote end points.

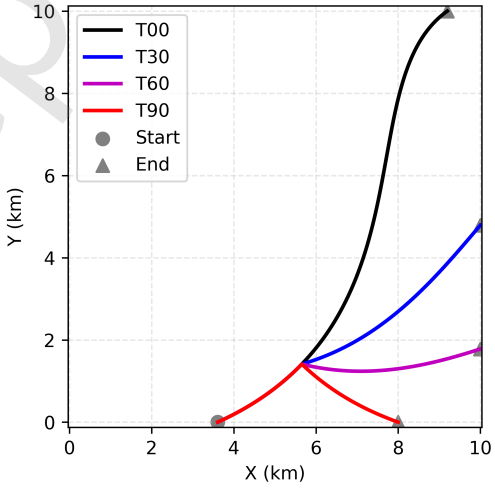


Fig. B2 Illustration of trajectory of a target before and after rotation.

training and validation losses during the training process. As shown in the figure, the training loss decreases steadily and converges to a low level. The validation loss exhibits several spikes, which occur when the model parameters temporarily reach states with poor generalization capability. As training continues and more epochs are performed, the validation loss eventually decreases to a satisfactory level and stabilizes.

Appendix B

We construct an MT-TSP benchmark dataset. This dataset contains curve trajectory points of numerous targets moving within a $10 \times 10km^2$ area, with an interval of 5 meters. Assuming that each point is recorded once per second, the corresponding target speed is approximately 18 km/h. Fig. B1 illustrates the trajectories of 10 targets within the bounded region, where the gray circles denote start points and gray

Just Accepted

triangles denote end points. The trajectory generation process is carried out as follows: first, start and end points (p_0 and p_3) are randomly selected from a pair of opposite edges of the region. Next, two control points (p_1 and p_2) are randomly chosen along the line connecting the start and end points, with the constraint that a minimum distance is maintained among all four points. A cubic Bézier curve is then constructed based on these four points

$$B(t) = p_0(1-t)^3+3p_1(1-t)^2t+3p_2(1-t)t^2+p_3t^3. \quad \text{(B1)}$$

Finally, trajectory points are sampled every 5 meters along the curve, resulting in a smooth trajectory confined within the bounded region. Using this method, a training set containing 10k target trajectories and a test set T00 containing 100 target trajectories are generated.

To simulate real-world target maneuvers, we rotate the latter portion of each trajectory in the test set T30, T60, and T90 by 30°, 60°, and 90°, respectively. Fig. B2 illustrates the trajectory of a target before and after rotation. It can be observed that by performing a higher level of maneuverability, the target is able to exit the bounded region more rapidly. This reflects real-world adversarial scenarios such as maritime surveillance [2, 3]. Stronger target maneuverability significantly complicates the task of maximizing the number of visited targets while minimizing travel distance under limited computation time. This constitutes the core difficulty of the uncertain MT-TSP.

Acknowledgements

This work is supported by the National Natural Science Foundations of China (NSFC) under Grants 62273044, 62088101, and 62425304.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Electronic Supplementary Material

The electronic supplementary material (Pytorch code) is publicly available in https://github.com/BIT-Zhangzhao/PaE_PLan_and_Execute. It includes the attention model, training program, surrogate model, low-level environment, benchmark dataset, etc.

References

[1] C. S. Helvig, G. Robins, and A. Zelikovsky, The moving-target traveling salesman problem, *Journal of Algorithms*, vol. 49, no. 1, pp. 153–174, 2003.

[2] S. K. İşleyen, U. Uçar, and F. Balo, A new solution approach for maritime surveillance operation: The case of aegean sea, *Mathematical Problems in Engineering*, vol. 2019, no. 1, p. 8575219, 2019.

[3] Y. Wang and N. Wang, Moving-target travelling salesman problem for a helicopter patrolling suspicious boats in antipiracy escort operations, *Expert Systems with Applications*, vol. 213, p. 118986, 2023.

[4] K. Sriniketh, A. V. Le, R. E. Mohan, B. J. Sheu, V. D. Tung, P. Van Duc, and M. B. Vu, Robot-aided human evacuation optimal path planning for fire drill in buildings, *Journal of Building Engineering*, vol. 72, p. 106512, 2023.

[5] Z. Zhang, C. Chen, L. Wang, Y. Ding, and F. Deng, A two-phase planner for messenger routing problem in uav-ugv coordination systems, *IEEE Transactions on Automation Science and Engineering*, 2025.

[6] C. Groba, A. Sartal, and X. H. Vázquez, Integrating forecasting in metaheuristic methods to solve dynamic routing problems: Evidence from the logistic processes of tuna vessels, *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 55–66, 2018.

[7] I. Granado, L. Hernando, Z. Uriondo, and J. A. Fernandes-Salvador, A fishing route optimization decision support system: The case of the tuna purse seiner, *European Journal of Operational Research*, vol. 312, no. 2, pp. 718–732, 2024.

[8] A. Stieber and A. Fügenschuh, Dealing with time in the multiple traveling salespersons problem with moving targets, *Central European Journal of Operations Research*, vol. 30, no. 3, pp. 991–1017, 2022.

[9] Hammar and Nilsson, Approximation results for kinetic variants of tsp, *Discrete & Computational Geometry*, vol. 27, no. 4, pp. 635–651, 2002.

[10] A. G. Philip, Z. Ren, S. Rathinam, and H. Choset, A mixed-integer conic program for the moving-target traveling salesman problem based on a graph of convex sets, in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 8847–8853.

[11] ——, C*: A new bounding approach for the moving-target traveling salesman problem, *IEEE Transactions on Robotics*, 2025.

[12] N. S. Choubey, Moving target travelling salesman problem using genetic algorithm, *International Journal of Computer Applications*, vol. 70, no. 2, 2013.

[13] C. Groba, A. Sartal, and X. H. Vázquez, Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices, *Computers & Operations Research*, vol. 56, pp. 22–32, 2015.

[14] U. Ucar and S. K. Işleyen, A meta-heuristic solution approach for the destruction of moving targets through air operations. *International Journal of Industrial Engineering*, vol. 26, no. 6, 2019.

[15] Z. Zhang, L. Wang, and C. Chen, A dual-archive memetic algorithm with motion prediction for solving dynamic multiple

Just Accepted

traveling salesmen problem, in *2024 43rd Chinese Control Conference (CCC)*. IEEE, 2024, pp. 2177–2182.

[16] R. S. de Moraes and E. P. de Freitas, Experimental analysis of heuristic solutions for the moving target traveling salesman problem applied to a moving targets monitoring system, *Expert Systems with Applications*, vol. 136, pp. 392–409, 2019.

[17] A. Maskooki and M. Kallio, A bi-criteria moving-target travelling salesman problem under uncertainty, *European Journal of Operational Research*, vol. 309, no. 1, pp. 271–285, 2023.

[18] C. Zhou, J. Ma, L. Douge, E. P. Chew, and L. H. Lee, Reinforcement learning-based approach for dynamic vehicle routing problem with stochastic demand, *Computers & Industrial Engineering*, vol. 182, p. 109443, 2023.

[19] D. Chen, C. Imdahl, D. Lai, and T. Van Woensel, The dynamic traveling salesman problem with time-dependent and stochastic travel times: A deep reinforcement learning approach, *Transportation Research Part C: Emerging Technologies*, vol. 172, p. 105022, 2025.

[20] S. Kadyrov, A. Azamov, Y. Abdumajitov, and C. Turan, Deep reinforcement learning for dynamic vehicle routing with demand and traffic uncertainty, *Operations Research Perspectives*, p. 100351, 2025.

[21] F. D. Hildebrandt, B. W. Thomas, and M. W. Ulmer, Opportunities for reinforcement learning in stochastic dynamic vehicle routing, *Computers & operations research*, vol. 150, p. 106071, 2023.

[22] Z. Zhang, H. Liu, M. Zhou, and J. Wang, Solving dynamic traveling salesman problems with deep reinforcement learning, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 4, pp. 2119–2132, 2021.

[23] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu *et al.*, Deepseekmath: Pushing the limits of mathematical reasoning in open language models, *arXiv preprint arXiv:2402.03300*, 2024.

[24] R. Bellman, A markovian decision process, *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[25] W. Kool, H. Van Hoof, and M. Welling, Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[26] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, Neural combinatorial optimization with reinforcement learning, *arXiv preprint arXiv:1611.09940*, 2016.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347*, 2017.

[29] T. Kaufmann, P. Weng, V. Bengs, and E. Hüllermeier, A survey of reinforcement learning from human feedback, *Transactions on Machine Learning Research*, 2024.

[30] L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, S. Huang, K. Rasul, and Q. Gallouédec, Trl: Transformer reinforcement learning, https://github.com/huggingface/trl, 2020.

[31] A. G. Philip, Z. Ren, S. Rathinam, and H. Choset, A mixed-integer conic program for the multi-agent moving-target traveling salesman problem, *arXiv preprint arXiv:2501.06130*, 2025.

[32] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, Dealing with non-stationarity in multi-agent deep reinforcement learning, *arXiv preprint arXiv:1906.04737*, 2019.

[33] B. Peng and C. Papadimitriou, The complexity of non-stationary reinforcement learning, in *International Conference on Algorithmic Learning Theory*. PMLR, 2024, pp. 972–996.

[34] Z. Ning and L. Xie, A survey on multi-agent reinforcement learning and its application, *Journal of Automation and Intelligence*, vol. 3, no. 2, pp. 73–91, 2024.

[35] J. Chen, Y. Wang, Y. Zhang, Y. Lu, Q. Shu, and Y. Hu, Extrinsic-and-intrinsic reward-based multi-agent reinforcement learning for multi-uav cooperative target encirclement, *IEEE Transactions on Intelligent Transportation Systems*, 2025.



Zhao Zhang received the B.Eng. and M.Eng. degrees from Beijing Institute of Technology (BIT), Beijing, China, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree in control science and technology with the State Key Laboratory of Autonomous Intelligent Unmanned Systems. His research interests include reinforcement learning, evolutionary computation, and operations research.



Yihan Zhong received the B.Eng. from Beijing Institute of Technology (BIT), Beijing, China, in 2025. She's currently working toward the Ph.D. degree in control science and technology with the State Key Laboratory of Autonomous Intelligent Unmanned Systems, BIT. Her research interests include operations research and evolutionary computation.



Chen Chen received the Ph.D. degree in control science and engineering from Beijing Institute of Technology, Beijing, China, in 2009. She is currently a Professor with Beijing Institute of Technology. Her research interests include multi-agent game and machine learning, intelligent optimization, and decision-making of complex systems.

Just Accepted



Bin Xin received the B.S. degree in information engineering and the Ph.D. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2004 and 2012, respectively. From 2011 to 2012, he was an Academic Visitor with the Decision and Cognitive Sciences Research Centre, The University of Manchester, Manchester, U.K. He is currently a Professor with the School of Automation, Beijing Institute of Technology. His research interests include search and optimization, evolutionary computation, combinatorial optimization, and multi-agent systems.



Fang Deng received the B.E. and Ph.D. degrees in control science and engineering from Beijing Institute of Technology, Beijing, China, in 2004 and 2009, respectively. He is currently a Professor with Beijing Institute of Technology. His research interests include intelligent robots, intelligent information processing, and smart wearable devices.

Just Accepted