Journal of Electronics and Information Science (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/jeis.2025.100105
ISSN 2371-9524 Vol. 10 Num. 1

Optimisation of efficient soil sampling scheme based on machine learning and approximation algorithm

Xuecheng Luo*^,#^, Shouyin Xiao#, Mingyu Wang#, Zhicheng Xu, Sijia Wang, Hongfan Chai

School of Management, Shanxi medical University, Jinzhong, China
#Co-first author: Xuecheng Luo, Shouyin Xiao, Mingyu Wang
**Corresponding author: luoxuecheng@sxmu.edu.cn*

Keywords: Soil Sampling, Machine Learning, Approximation Algorithms, Path Optimisation, Efficiency, Workload Balancing

Abstract: The aim of this study is to optimise the soil sampling scheme through machine learning and approximation algorithms to improve the efficiency of soil sampling work. The research methodology included calculating straight-line distances between sampling points using Haversine's formula and applying a clustering model and path-planning algorithm to determine the most efficient sampling routes. The main results show that the use of Haversine's formula, K-means clustering algorithm and Christofides' algorithm can significantly reduce the time required for soil sampling and increase the efficiency of the work. In addition, the paths were optimised for equalisation by the greedy algorithm to ensure that the daily working time was more balanced and within the allowed time frame. The study concludes that the proposed method not only optimises the soil sampling schedule but also ensures that the working hours are balanced and within the stipulated time limits.

1. Introduction

Soil survey is a crucial basic work for national food security. By conducting a comprehensive soil check-up on various types of agricultural land such as arable land, garden land, forest land, grassland and other land that has not been fully utilised [1], we can fully understand the actual situation of soil quality and soil resources, which provides a strong scientific basis for guaranteeing food production and guarding the red line of arable land [2, 3]. However, in the face of the vast and complex geographic environment, how to efficiently complete the collection of soil in multiple locations under the double constraints of time and resource conditions has become an urgent problem [4].

2. Reference Review

Research on shortest paths has focused on both static and dynamic paths [5]. Static shortest path algorithms are used in situations where the network topology is stable, while dynamic shortest path algorithms are used in situations where the topology changes frequently. In environments where the network topology changes frequently, it is too expensive to reconstruct the shortest path tree using

Dijkstra's algorithm, so a series of dynamics are proposed on the basis of the static Dijkstra's algorithm. The DSPT algorithm proposed in paper [6] is more efficient than the static Dijkstra algorithm, but lacks algorithmic analysis and experimental data. Reference [8] proposes an algorithm to reconstruct SPT based on the current SPT based on [7]. Among them, the algorithm proposed in reference only applies to the change of one edge, and the algorithm proposed in reference [8] addresses the case of multiple edges with reduced weights. The algorithm based on Ball-and-String model proposed in reference is the best dynamic update algorithm available.

3. Optimal Path Planning Programme

In order to better organise the programme, we will carry out the sampling task based on the eight points (31, 44, 61, 83, 100, 115, 147, 158) arranged by the office for the working group. Assuming that these points are passable in a straight line and maintain a speed of 20km/h, we will find the optimal path and the shortest working time of the working group for this day.

We first need to calculate the straight line distance between the points. Here we use Haversine's formula for its calculation and the formula is as follows:

$$d_{ij} 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_j-\varphi_i}{2}\right)+\cos(\varphi_i)\cos(\varphi_j)\sin^2\left(\frac{\theta_j-\theta_i}{2}\right)}\right)$$

Where r represents the radius of the Earth, φ_i and φ_j represent the latitude of the points p_i and p_j , and θ_i, θ_j represents the longitude of the points p_i and p_j. In the calculation process, we approximate the spherical plane of the Earth as a plane and calculate the Euclidean distances, i.e., straight-line distances, between the eight points using Haversine's formula. Finally, we visualise the data using Python, as shown in Figure 1.

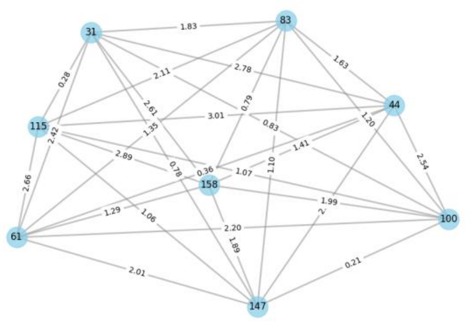

Figure 1 Straight-line distances between eight points

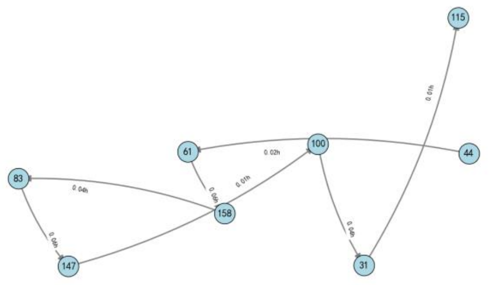

Figure 2 Optimal route after screening

We construct the TSP problem using the distances between points as inputs. We find the most efficient work route without considering the travelling time of the working group from the site to the first working point and the last working point back to the site. The implementation steps are as follows:

STEP 1. Determine the location of the point at which the working group needs to carry out its tasks: this is usually represented by a node.

STEP 2. Determine the distance between work point locations: usually expressed in terms of side weights.

STEP 3. Conditions for establishing the optimal path: each point works and works only once, and each point is passable in a straight line.

STEP 4. Path optimisation: shortest total distance/shortest total working time.

We have derived a complete map between the eight working points based on the TSP problem.

Due to the small amount of data and to avoid errors caused by calling the library, we use an exhaustive approach to filter the best routes. In order to get the shortest time to complete the work on the same day, we first calculate the travelling time between the points based on the travelling speed. In order to get the shortest working time to complete the work on the same day, we calculate the travelling time between points according to the required travelling speed. Where V represents the travelling speed, $V = 20\text{km/h}$. It is assumed that the travelling time t_{ij} is proportional to the distance d_{ij} and the travelling speed v is a constant:

$$t_{ij} = \frac{d_{ij}}{v}$$

We should also consider the working time of the working group at each point, assuming that the working time of each point p_i is ω_i and the total working time T is the sum of the working time and travelling time of all the points, and finally the formula for the total time is as follows:

$$T = \sum_{i=1}^{n} \omega_i + \sum_{i=1}^{n-1} t_{i,i+1}$$

Combined with the above analysis, we use Python to solve the above model and visualise the optimal path as shown in Fig. 2. The optimal path is 44->61->158->83->147->100->31->115, and the total working time for sampling according to the optimal path is 6.68 hours.

4. Holistic path planning based on K-means clustering model

Based on the principle of proximity and the condition of working on eight points per day, we need to divide all sampling points. In order to improve the efficiency, we need to give the optimal paths of the working group for each day and the corresponding working time, and then compare the working time of all the optimal paths to get the maximum and minimum values.

We firstly calculate the distance between each point according to the latitude and longitude of each point, combined with Haversine's formula, and we randomly select some representative data for display, as shown in Table 1

Table 1 Distances between points

Point A	Point B	Gap	Point A	Point B	Gap	Point A	Point B	Gap
1	2	4.55	99	181	16.27	38	181	26.41
2	50	14.01	112	146	2.03	42	74	18.94
7	42	0.16	129	210	2.05	63	96	5.56
19	145	8.28	146	159	1.65	75	217	9.22

From some of the data presented in the table above, it can be seen that there are large differences in the length of the distances between the various points, some of which are very close to each other, only a few hundred metres apart, and some of which are very far away from each other, being able to reach more than 20 kilometres. Then, we calculated the time taken to pass each path:

$$t_{ij} = \frac{d_{ij}}{v}$$

Next, we divide the data according to the proximity principle and using the eight points as a benchmark. We choose to use the K-means clustering algorithm, the key of this algorithm lies in the determination of the initial centre of mass in the dataset, and then allocate the data according to the distance between each data and the centre of mass. By calculating the centre of mass and assigning data several times, iterative optimization results in the best cluster division. We draw the flow chart shown in Figure 3:

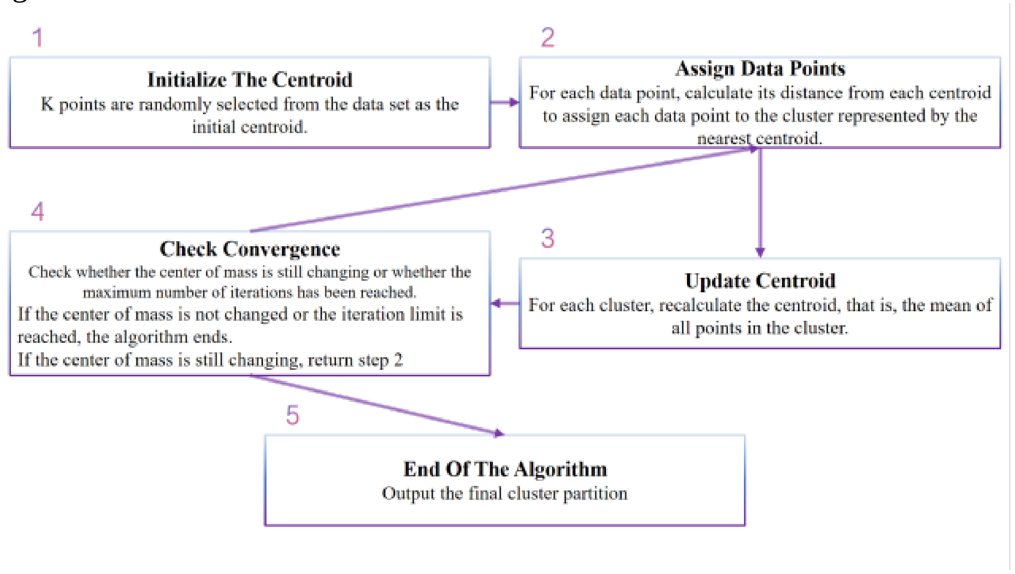
1
Initialize The Centroid
K points are randomly selected from the data set as the initial centroid.
2
Assign Data Points
For each data point, calculate its distance from each centroid to assign each data point to the cluster represented by the nearest centroid.
3
Update Centroid
For each cluster, recalculate the centroid, that is, the mean of all points in the cluster.
4
Check Convergence
Check whether the center of mass is still changing or whether the maximum number of iterations has been reached.
If the center of mass is not changed or the iteration limit is reached, the algorithm ends.
If the center of mass is still changing, return step 2
5
End Of The Algorithm
Output the final cluster partition

Fig. 3 Flowchart of K-means clustering algorithm

Working with 8 points per day, i.e. each cluster contains 8 data points, since there are 222 points in total, we determined 27 initial centres of mass to divide the points according to the proximity principle. Based on the previously calculated distance between two points, we used Python to implement the K-means algorithm to divide the points and get the clustering results. And the data visualisation is shown in Figure 4.

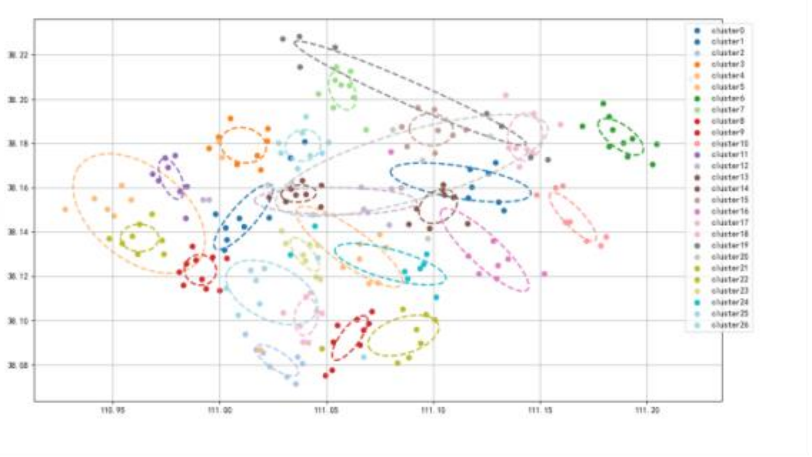

Fig. 4 Classification of clustering results for each site

The calculated 27 optimal cluster divisions are shown in Figure 3, the clustering results are divided based on latitude and longitude, different colours represent different categories, we also use dotted circles to indicate the approximate range of the distribution of points in different categories. Overall, some of the clusters contain more than 8 points, which is due to a total of 222 points, and the remaining 6 points are classified into neighbouring clusters in close proximity. It is easy to find that there exist some categories in which the distribution of points is more concentrated and the clustering density is higher, while some categories have a larger range of dashed circle contours of points, which are loosely distributed and the clustering density is lower.

Next, we use Christofides algorithm to solve the optimal path. Firstly, we apply Prim or Kruskal algorithm to prefer the edges with small weights to cover all the vertices, so as to obtain a minimum spanning tree MST. Then we pick out all the fixed points in the mst that have an odd degree, and then we use Hungarian or Blossom algorithm to match the vertices with the minimum weight, and use Fleury or Hierholzer algorithm to find the Eulerian circuits, and finally converting them to Hamiltonian circuits. To facilitate a systematic understanding of the Christofides algorithm, we have drawn the flowchart as shown in Figure 5 below

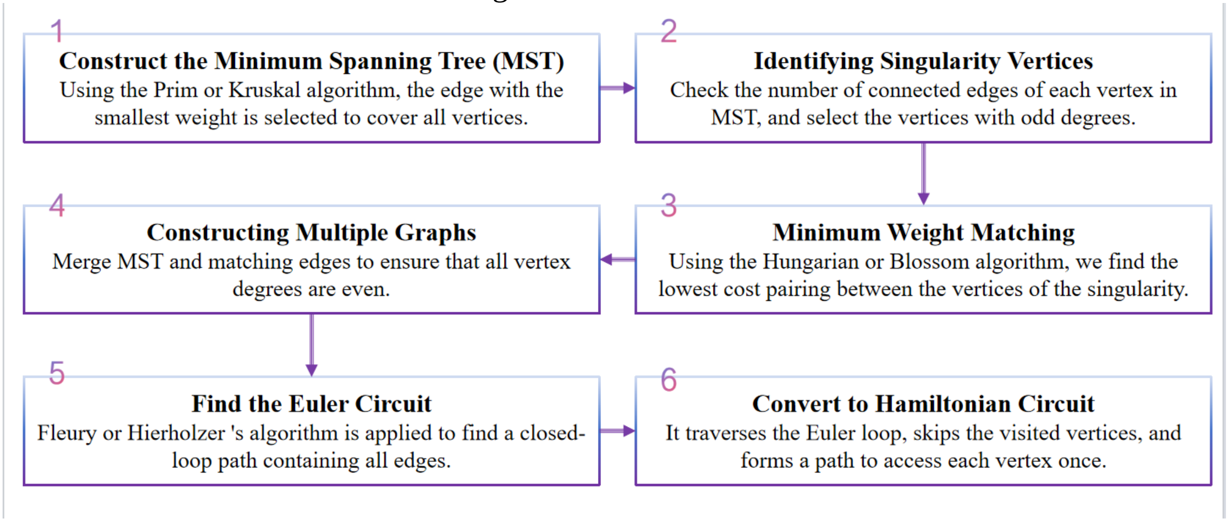
1
Construct the Minimum Spanning Tree (MST)
Using the Prim or Kruskal algorithm, the edge with the smallest weight is selected to cover all vertices.
2
Identifying Singularity Vertices
Check the number of connected edges of each vertex in MST, and select the vertices with odd degrees.
3
Minimum Weight Matching
Using the Hungarian or Blossom algorithm, we find the lowest cost pairing between the vertices of the singularity.
4
Constructing Multiple Graphs
Merge MST and matching edges to ensure that all vertex degrees are even.
5
Find the Euler Circuit
Fleury or Hierholzer 's algorithm is applied to find a closed-loop path containing all edges.
6
Convert to Hamiltonian Circuit
It traverses the Euler loop, skips the visited vertices, and forms a path to access each vertex once.

Fig. 5 Flowchart of Christofides algorithm

We combined the time to pass on the switching road between the points obtained above, the working paths on each point and the 27 optimal cluster divisions obtained through clustering, coded and solved using Python software. We obtained the route planning, i.e., the optimal path division, for each point in each cluster and solved to obtain the required time, and some of the computational results are shown in Table 2

Table 2 Optimal daily path, i.e., working hours (partial)

Cluster ID	Optimal Route	Total Time (hours)
0	194 -> 159 -> 96 -> 112 -> 177 -> 37 -> 81 -> 135	7.22
3	97 -> 190 -> 195 -> 191 -> 25 -> 143 -> 128 -> 13	6.49
6	58 -> 36 -> 48 -> 98 -> 121 -> 65 -> 109 -> 117 -> 108 -> 64	8.62
11	3 -> 45 -> 131 -> 214 -> 213 -> 39 -> 175 -> 24	6.81
16	178 -> 114 -> 180 -> 220 -> 66 -> 95 -> 219 -> 212	7.41
21	179 -> 57 -> 1 -> 110 -> 78 -> 218 -> 11 -> 217	7.12
24	202 -> 19 -> 20 -> 203 -> 204 -> 200 -> 210 -> 150	7.26

Analysing the above table, we conclude that the maximum value of the optimal path is path 6 (58 -> 36 -> 48 -> 98 -> 121 -> 65 -> 109 -> 117 -> 108 -> 64), which takes 8.62 hours, and the minimum value is path 3 (97 -> 190 -> 195 -> 191 -> 25 -> 143 -> 128 -> 13) with a working time of 6.49 hours. We can find that there are some differences in the total time consumed by different paths, but the overall distribution is around 6-8 hours, which also confirms that our calculation process and results are more correct and reasonable.

5. Optimisation scheme for path equalisation based on greedy algorithm

In reality, the working time was limited to 8 hours and due to the remoteness of the soil sampling sites, the working time was extended to a maximum of 8.5 hours. Now, by removing the limitation on the number of sites, we can once again achieve a balanced solution for efficient sampling and keep the working hours within the time limit.

Above, the maximum value of working hours is 8.62 hours, which is obviously out of the permitted range, indicating that there are cases of working hours overtime. Meanwhile, in order to visualise the fluctuation of the daily working hours, we firstly present the working hours of 27 days in the form of a line graph, as shown in Figure 6.

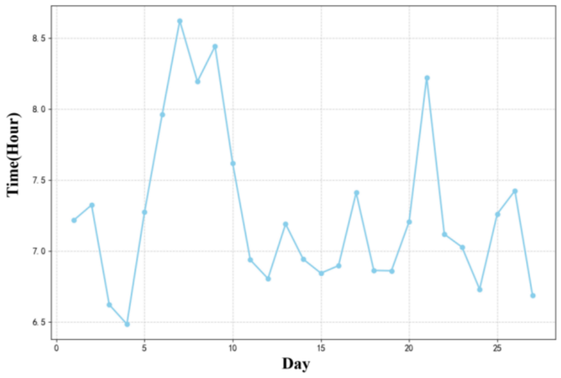

Fig. 6 Line graph of working time before optimisation

The line graph shows that there is a significant fluctuation in working hours. We chose to quantify this fluctuation by calculating the variance of the working time, which was calculated to be 0.31, indicating that the above scenario has a significant imbalance in working time, which affects the overall efficiency of the working group.

We may assume that the time spent on point work is increased as a percentage of the working time, and minimising the time spent on switching between points to pass on the road is the key to ensure the efficiency of the work. Therefore, in the selection of points, based on the clustering results, the greedy algorithm is used to continuously coordinate and improve the optimal path.

A greedy algorithm is an algorithm that selects the optimal or best possible solution for each step carried out to get the best possible result. This problem uses the greedy algorithm to select a point as the starting point, and through continuous iteration, the unvisited nearest nodes are added to the path, and finally a more balanced path solution is optimised. Part of the data results are shown in Table 3 below

Table 3 Optimised daily routes and working hours

Day	**Optimal Route**	**Total Time (hours)**
1	1 -> 57 -> 110 -> 78 -> 218 -> 11 -> 217 -> 179	7.2
2	2 -> 142 -> 157 -> 4 -> 164 -> 141 -> 132 -> 119	6.6
3	3 -> 45 -> 131 -> 24 -> 175 -> 39 -> 213 -> 214	6.8
4	5 -> 10 -> 123 -> 71 -> 100 -> 147 -> 56 -> 52	6.7
5	6 -> 21 -> 106 -> 148 -> 12 -> 99 -> 144 -> 127	6.9
6	7 -> 42 -> 23 -> 172 -> 50 -> 62 -> 171 -> 28	7.0
7	8 -> 54 -> 82 -> 137 -> 68 -> 76 -> 222 -> 221	7.2
8	9 -> 32 -> 107 -> 22 -> 207 -> 34 -> 208 -> 193	7.4

The maximum value of the optimal path working time in this scenario is the 10th day, 8.5 hours, which meets the maximum allowable working time In addition, we again visualise the data in the form of a line graph of the daily working time in the scenario, as shown in Figure 7.

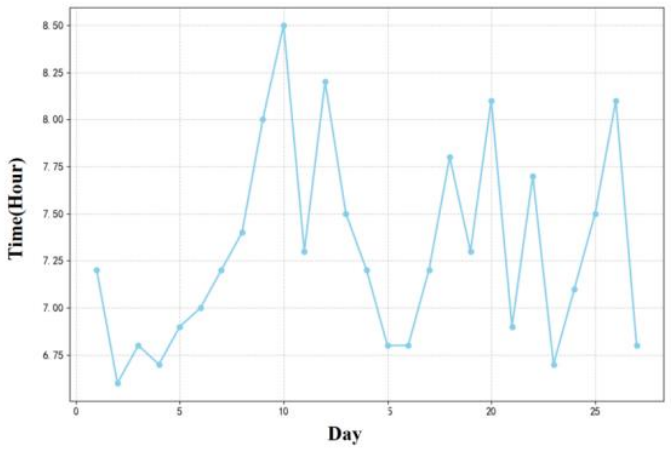

Fig. 7 Line graph of optimised working time

Observing Fig. 7, we can see that although there are still some fluctuations in the daily working time, the degree of fluctuation and the degree of fluctuation are both slowed down to a certain extent, and the extreme variance is also reduced from 2.1 to 1.9, which means that the gap between the daily working time after optimisation becomes smaller. In order to accurately verify the degree of work time balance after the optimisation, we calculate that the variance of daily work time after the optimisation is 0.27, which is reduced to a certain extent compared with that before the optimisation, indicating that the optimisation scheme significantly improves the work balance. Moreover, none of the optimised working time is overtime, and most of the working time is not more than 8 hours, which is in line with the regulations for special cases.

6. Conclusion

In this study, an efficient soil sampling scheme optimisation method is proposed based on machine learning and approximation algorithms, and the efficiency of soil sampling is improved by calculating the straight-line distance between points and applying the clustering model and path planning algorithm. The results show that the Haversine formula for calculating the distance between points, the K-means clustering algorithm for dividing the points, and the Christofides algorithm for solving the optimal paths can significantly reduce the time required for sampling and improve the work efficiency. In addition, the paths were optimised for equalisation by the greedy algorithm, which resulted in a more balanced daily working time that was within the allowed time frame.

Despite the results of this study, there are some limitations. For example, this study assumes that movement between points can be approximated as a straight line, whereas terrain and road conditions may affect this assumption in real situations. In addition, this study mainly analysed specific point locations, and the generalisability of its results may be limited. Future studies could consider the actual road network and terrain features with the support of more sophisticated geographic information system (GIS) data to obtain more accurate travel distance and time estimates. At the same time, a more detailed analysis of soil sampling conditions at different points could be conducted to further improve the adaptability and practicality of the scheme.

References

[1] Diveev A, Konstantinov S, Shmalko E, et al. Machine learning control based on approximation of optimal trajectories[J]. Mathematics, 2021, 9(3): 265.

[2] Maoudj A, Hentout A. Optimal path planning approach based on Q-learning algorithm for mobile robots[J].

Applied Soft Computing, 2020, 97: 106796.
[3] Narvaez P, Siu K Y, Tzeng H Y. New dynamic SPT algorithm based on a ball-and-string model[J]. IEEE/ACM transactions on networking, 2001, 9(6): 706-718.
[4] Yan C, Xiang X, Wang C. Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments[J]. Journal of Intelligent & Robotic Systems, 2020, 98: 297-309.
[5] Pritzkoleit M, Knoll C, Röbenack K. Reinforcement Learning and Trajectory Planning based on Model Approximation with Neural Networks applied to Transition Problems[J]. IFAC-PapersOnLine, 2020, 53(2): 1581-1587.
[6] Wang Q, Tang C. Deep reinforcement learning for transportation network combinatorial optimisation: a survey[J]. Knowledge-Based Systems, 2021, 233: 107526.
[7] Hu J, Wang M, Zhang C, et al. Path Planning for Unmanned Vehicles Based on Value Function Approximation Algorithm[C]//2019 IEEE 15th International Conference on Control and Automation (ICCA). IEEE, 2019: 272-277.
[8] Mordatch I, Todorov E. Combining the benefits of function approximation and trajectory optimization[C]//Robotics: Science and Systems. 2014, 4: 23.