

# Intelligent Campus Sanitation: Scheduling and Path Planning for Autonomous Cleaning Vehicles

## Summary

As campus sizes expand, traditional manual cleaning methods struggle to meet the demands for efficiency and quality. **Autonomous cleaning vehicles**, with their numerous advantages, offer a viable alternative. However, efficiently managing and scheduling these vehicles, along with planning their cleaning routes based on varying needs, presents a significant challenge. This article aims to optimize the scheduling and path planning of cleaning vehicles to enhance cleaning efficiency and quality.

Initially, a **spectral clustering model** was established to segment the campus into distinct cleaning zones, laying the groundwork for subsequent path planning and dynamic scheduling. The campus map was abstracted into an undirected connected graph, with road dirtiness as edge weights. The spectral clustering algorithm then divided the campus into **three cleaning areas** based on road dirtiness. Cleaning vehicles were allocated to each area, with **quantities of 3, 2, and 2**, based on the weighted average dirtiness of each zone.

Subsequently, the **Dynamic A\* Path Planning and Multi-Mode Vehicle Dispatch Model** was developed to rationally schedule cleaning paths and dynamically adjust to unforeseen events, updating route planning in real time. The model detects emergencies and, in their absence, operates in regular cleaning mode. It **prioritizes cleaning roads with higher dirt levels**, dynamically updating dirt levels within the graph and reassessing paths at each time step. A **path protection mechanism** is integrated to prevent frequent route changes, only replanning after completing the previous target. When unexpected events occur, an event response mechanism **dynamically reassigns cleaning vehicle resources** to meet the cleaning demands of affected areas. This model ensures the continuity and efficiency of cleaning operations, adapting to environmental changes and enabling flexible dynamic scheduling.

Finally, a **benefit evaluation model** measures operating costs and cleaning effects to assess the overall benefit of automatic cleaning vehicles in campus management. It ensures optimal resource allocation and cost control, calculates a cleanliness index from the weighted average of road dirt and length, and introduces a **Cost-Effectiveness Ratio (CER)** to measure the cost improvement in cleaning effect. By comparing marginal costs and benefits, the model determines the optimal number of vehicles, proving that automatic cleaning vehicles have higher cost-effectiveness and cleaning performance than traditional manual cleaning.

**Keywords:** autonomous cleaning vehicles; spectral clustering model; dynamic A\* algorithm; path planning; dynamic scheduling

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Problem Background . . . . .	3
1.2	Restatement of the problem . . . . .	4
1.3	Our Work . . . . .	4
<b>2</b>	<b>Assumptions and Notations</b>	<b>5</b>
2.1	Assumptions . . . . .	5
2.2	Notations . . . . .	5
<b>3</b>	<b>Data Source</b>	<b>6</b>
<b>4</b>	<b>Optimal Cleaning Area Division</b>	<b>6</b>
4.1	Construction of an Undirected Connected Graph . . . . .	6
4.2	Spectral Clustering Model for Partitioning . . . . .	6
<b>5</b>	<b>Allocation of Autonomous Cleaning Vehicles</b>	<b>9</b>
<b>6</b>	<b>Campus Cleaning Vehicle Operations</b>	<b>9</b>
6.1	Path planning for campus cleaning vehicle . . . . .	9
6.2	Scheduling of campus automatic cleaning vehicles . . . . .	12
<b>7</b>	<b>Benefit Evaluation Model</b>	<b>14</b>
<b>8</b>	<b>Sensitivity Analysis</b>	<b>16</b>
<b>9</b>	<b>model Assessment</b>	<b>16</b>
9.1	Strengths . . . . .	16
9.2	Weaknesses . . . . .	17
	<b>Appendices</b>	<b>19</b>

# 1 Introduction

## 1.1 Problem Background

With the development of society, the scale and area of university campuses in China have been expanding, leading to increased difficulties in environmental sanitation management. Traditional manual cleaning methods are not only inefficient but also susceptible to objective factors such as weather. To alleviate this issue, Chongqing University's Huxi Campus has opted to introduce autonomous cleaning vehicles.

These vehicles, integrated with advanced sensors and navigation systems, can automatically identify cleaning areas, avoid obstacles, plan efficient cleaning routes, and complete sweeping tasks. They possess significant advantages such as round-the-clock operation, reduced labor costs, and improved work efficiency. As technology continues to evolve, autonomous cleaning vehicles have begun to be widely applied in some universities and are gradually becoming an important means to enhance the efficiency of campus cleaning management. However, with the increase in the number of vehicles, effectively scheduling multiple cleaning vehicles, reasonably planning their cleaning routes, and ensuring that each area is timely swept at different times still face numerous challenges.



Figure 1: Pictures of Automatic Cleaning Vehicle in Operation

Against this backdrop, establishing mathematical models for campus zoning, task allocation for cleaning vehicles, planning their driving routes, and scheduling vehicles for emergencies becomes particularly important. Through reasonable map partitioning, route planning, and vehicle scheduling algorithms, not only can cleaning costs be reduced, but cleaning efficiency can also be improved. By solving the aforementioned problems through mathematical modeling and evaluating operational costs and benefits, we can enhance the overall efficiency of campus environmental

sanitation management through intelligent management, providing a cleaner and more comfortable learning and living space for teachers and students.

## 1.2 Restatement of the problem

Our task is to establish a mathematical model to assist Chongqing University Huxi Campus in managing and scheduling automatic cleaning vehicles, and planning reasonable driving paths. As the campus map can be simplified into a weighted map, the problem can be transformed into the following parts:

- Divide the campus map into several areas based on terrain characteristics, pedestrian flow, and other factors, and assign corresponding priority and weight values.
- Assign corresponding numbers of automatic cleaning vehicles to each region.
- Design a path planning algorithm for the cleaning vehicle to maximize the completion of more cleaning tasks within the specified time and ensure timely cleaning of high priority areas.
- Design a dynamic scheduling algorithm for automatic cleaning vehicles in response to unexpected situations.
- Design a benefit evaluation algorithm to assess the operating costs and cleaning effectiveness of introducing automatic cleaning vehicles.

## 1.3 Our Work

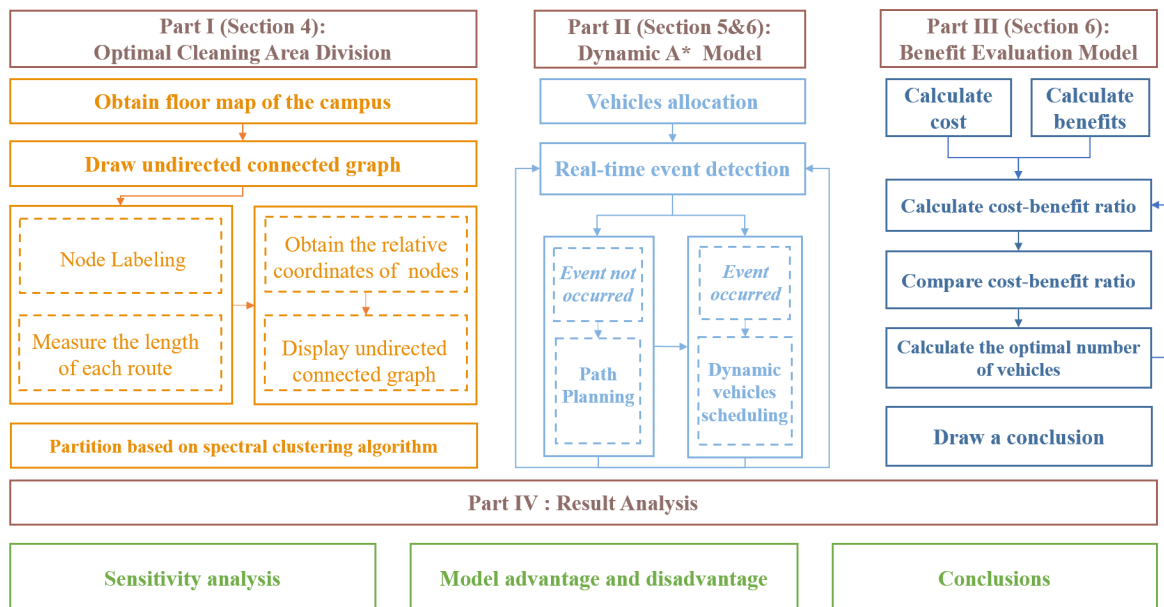


Figure 2: Framework of Our Work

## 2 Assumptions and Notations

### 2.1 Assumptions

Some fundamental assumptions are listed below.

- The map of Chongqing University Huxi Campus is abstracted into an undirected weighted graph, with the roads constituting edges and intersections serving as the nodes.
- Weather conditions affect all areas of the campus equally, with uniform start and end times.
- Autonomous cleaning vehicles exhibit standardized performance characteristics, encompassing uniform power and velocity parameters, while the possibility of malfunctions is neglected.
- Throughout the operational phase of autonomous cleaning vehicles, parameters including pedestrian density and the dust accumulation coefficient within each area remain invariant.
- During periods marked by public activities or road construction, the associated areas and roads are deemed to be off-limits to autonomous cleaning vehicles.

### 2.2 Notations

Here is an explanation of some important symbols.

Table 1: Significant symbols in this paper

Symbols	Description
$G = (V, E)$	The campus graph structure
$D(e, t)$	The dirtiness of edge $e$ at time step $t$
$a(e)$	Growth coefficient of dirtiness of edge $a$
$C = \{C_1, C_2, \dots, C_K\}$	Cleaning car collection
$P_i$	Path of cleaning car $C_i$
$S = \{S_1, S_2, \dots, S_J\}$	Cleaning area assembly
$c(e)$	The dirtiness reduction of the cleaning vehicle
$n_{\text{clusters}}$	The number of campus areas divided into cleaning zones
$target\_edge_i$	Target edge for cleaning car $C_i$
$adj\_matrix$	Adjacency matrix of campus map
$cluster\_id$	Identifier of the cleaning area to which the cleaning vehicle belongs
$A$	A collection of all scheduled activities on campus
$R_j$	Scope of Activity $A_j$
$E_{A_j}$	The set of edges blocked by activity $A_j$
$D_{\text{threshold}}$	Threshold of dirtiness
$\lambda$	Dirtiness threshold level

### 3 Data Source

Table 2: Data Sources and Websites

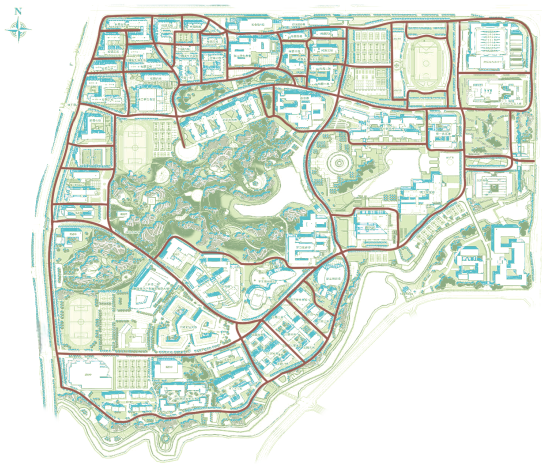
Data Source	Website
Bing Map	<a href="https://cn.bing.com/maps">https://cn.bing.com/maps</a>

## 4 Optimal Cleaning Area Division

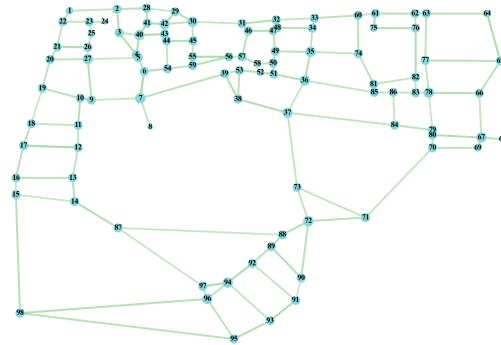
### 4.1 Construction of an Undirected Connected Graph

The geographical layout of Chongqing University's Huxi Campus has been meticulously translated into a graph-theoretic representation where the campus roads are abstracted as edges (highlighted in brown on the map) and the points of intersection as nodes. This transformation yields a simplified campus map, which is visually represented in fig. 3a. Each node within this graph has been systematically numbered from 1 to 98, facilitating a clear and organized reference system.

To quantify the spatial relationships within this graph, the lengths of the roads, which correspond to the edges, were meticulously measured using the distance measurement tool provided by Baidu Maps. This step ensures that the graph accurately reflects the actual distances that would be traversed within the campus. Additionally, the relative coordinates of each of the 98 nodes were precisely determined through the digitalization feature within the Origin software. This process allows for the accurate placement of each node within the graph, enhancing the model's geometric fidelity to the campus's actual layout. With this comprehensive data, an undirected connected graph was constructed, abstracting the original map, which is represented in fig. 3b.



(a) The Simplified Huxi Campus Map



(b) The Undirected Connected Graph

Figure 3: The Map of Huxi Campus

### 4.2 Spectral Clustering Model for Partitioning

Spectral clustering algorithm, grounded in spectral graph theory, abstracts objects to be clustered as vertices in a graph and translates the similarity between objects into the weights of edges

connecting these vertices, thereby constructing a weighted undirected graph. This graph-theoretic partitioning strategy endows the spectral clustering algorithm with a distinct advantage in capturing the hierarchical and spatial structures within complex datasets, offering an efficient method for regional division. The concept of "dirtiness" refers to the accumulation of grime on roads or edges. Within the graph, "dirtiness" has been quantified and assigned as the weights of edges connecting these vertices.

The dirtiness( $D$ ) of each road follows a logarithmic growth pattern over time, which can be mathematically articulated as:

$$D(t) = D_0 + a \ln(t + 1) \quad (1)$$

$D(t)$  denotes the level of dirtiness on the road at time  $t$ ; Initial dirtiness ( $D_0$ ) refers to the baseline level of contamination present on each road segment at the beginning of the observation period, which can be determined through methodologies such as stochastic assignment; Dirtiness Change Coefficient ( $a$ ) encapsulates the rate of alteration in dirtiness levels, indicating how dirtiness changes over time in a specified pattern.

It is evident that "dirtiness" of the road decreases when the cleaning vehicle passes by. It is hypothesized that the reduction in dirtiness exhibits variability, oscillating within the interval of 10%–20%. Concurrently, the proportion by which dirtiness is diminished during the cleaning process is generally characterized as a stochastic value, falling within the bounds of [0.1, 0.2]. This phenomenon can be quantitatively captured by the equation:

$$D_{\text{new}} = D_{\text{old}} - r \cdot D_{\text{old}} \quad (2)$$

$D_{\text{new}}$  represents the dirtiness level subsequent to the cleaning vehicle's passage;  $D_{\text{old}}$  is the dirtiness level antecedent to the cleaning intervention;  $r$  is the reduction ratio, which is a stochastic variable uniformly distributed within the interval [0.1,0.2]

By integrating the effects of both the cleaning vehicle's passage and the time-dependent accumulation of dirtiness, the updated dirtiness level is articulated by the following formula:

$$D_t = \max(0, (D_0 + a \ln(t + 1)) \cdot (1 - r)) \quad (3)$$

This formula synthesizes the initial dirtiness, the logarithmic progression of dirtiness over time, and the mitigating effect of the cleaning vehicle's intervention, providing a comprehensive framework for understanding the dynamics of dirtiness on campus roads.

In consideration of the connectivity data and numerical attributes—such as dirtiness and distance—associated with the edges within the graph, the spectral clustering algorithm has been elected to execute the partitioning of regions.

For any node  $v_i$  in the graph, its degree  $d_i$  is defined as the sum of the weights of all edges connected to it, where the term "dirtiness" refers to the weight of the edges.

$$d_i = \sum_{j=1}^n D_{ij} \quad (4)$$

Based on the definition of "dirtiness"(D) mentioned above, a degree matrix is obtained. By utilizing D between all points, an adjacency matrix can be derived. In addition to this, for a subset  $A \subseteq V$ , the following is defined:

$$|A| : \text{the number of points in subset } A \quad (5)$$

$$vol(A) = \sum_{j \in A} d_j \quad (6)$$

The goal is to minimize  $Ncut(A_1, A_2, \dots, A_k)$ .

$$Ncut(A_1, A_2, \dots, A_k) = \sum_{i=1}^k h_i^T L h_i = \sum_{i=1}^k (H^T L H)_{ii} = tr(H^T L H) \quad (7)$$

That is to calculate

$$\arg \min_H tr(H^T L H) \quad (8)$$

$$s.t. \quad H^T D H = I \quad (9)$$

Here is the pseudo-code for the algorithm we used algorithm 1.

---

**Algorithm 1:** Spectral Clustering based on Dirtiness Matrix

---

**Input:**  $G \leftarrow$  Graph, containing nodes and edges

$E_d \leftarrow$  Dirtiness matrix of edges

$n_{clusters} \leftarrow$  Number of clusters to divide the graph

**Output:**  $C \leftarrow$  Cluster labels for each node

```

1 Initialize the set of nodes  $V \leftarrow$  All nodes in graph  $G$ 
2 for  $u \leftarrow 1$  to  $|V|$  do
3   for  $v \leftarrow 1$  to  $|V|$  do
4     if  $(u, v) \in G$ 's edges then
5        $A[u][v] \leftarrow E_d[(u, v)]$ 
6     end
7   end
8 end
9 Perform spectral clustering on matrix  $A$  with  $n_{extclusters}$ 
10  $C \leftarrow$  Cluster labels from the clustering result
11 return  $C$ 

```

---

According to the spectral clustering algorithm [3], a map of campus cleaning area division fig. 4 was obtained.

In the context of campus environments, variations in foot traffic, event scheduling, and usage frequency across different areas may lead to a highly stochastic and uneven distribution of "dirtiness" data. It is in such scenarios that the spectral clustering algorithm demonstrates its unique strengths. The algorithm adapts to the random distribution of  $D$  data across various campus regions, thereby achieving more accurate and efficient regional partitioning.



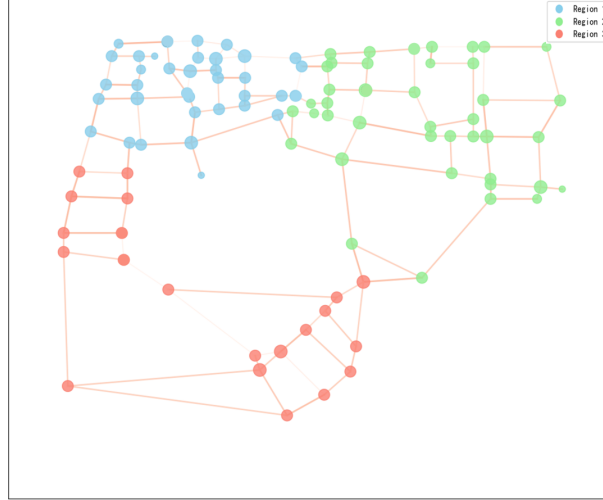


Figure 4: Campus Cleaning Area Division Map

## 5 Allocation of Autonomous Cleaning Vehicles

Following the delineation of distinct zones, a rational allocation of vehicles is imperative to optimize resource utilization. The dirt levels within each zone can be ascertained by computing the weighted average of the dirtiness of the roads encompassed, which in turn dictates the allocation of a greater number of cleaning vehicles to zones with higher dirt levels. The weighted average dirtiness of the roads within the three delineated zones, with  $N$  denoting the total number of roads (in this case,  $N = 7$ ), has been calculated individually. Employing eq (10) in conjunction with the initial dirt levels of the roads, the allocation of cleaning vehicles to the three zones is determined to be 3, 2, and 2, respectively.

$$n_k = \left\lceil \frac{\sum_{e \in S_k} D(e, t)}{\sum_{k=1}^K \sum_{e \in S_k} D(e, t)} \times N \right\rceil \quad (10)$$

## 6 Campus Cleaning Vehicle Operations

### 6.1 Path planning for campus cleaning vehicle

Once the distribution of cleaning vehicles is finalized, it becomes feasible to chart out rational cleaning paths for the vehicles assigned to each area.

Accounting for the fact that  $D$  of a road decreases by a certain proportion after each cleaning session by a cleaning vehicle, and considering that roads in certain areas may be temporarily inaccessible due to unforeseen events such as activities or construction, the variation in  $D$  of roads exhibits a high degree of randomness, and cleaning routes may need to be altered at any time. In light of these considerations, the **Dynamic A\* Path Planning and Multi-Mode Vehicle Dispatch Model** [2] has been chosen. This model is adept at dynamically updating routes during the planning process to adapt to the impacts of real-time environmental changes, such as weather, activities, and road construction, without the need to recalculate the global path each time, thereby enhancing the efficiency and real-time adaptability of the path planning process.

The establishment process of Dynamic A \* Path Planning and Multi Mode Vehicle Dispatch Model is shown in the fig. 5.

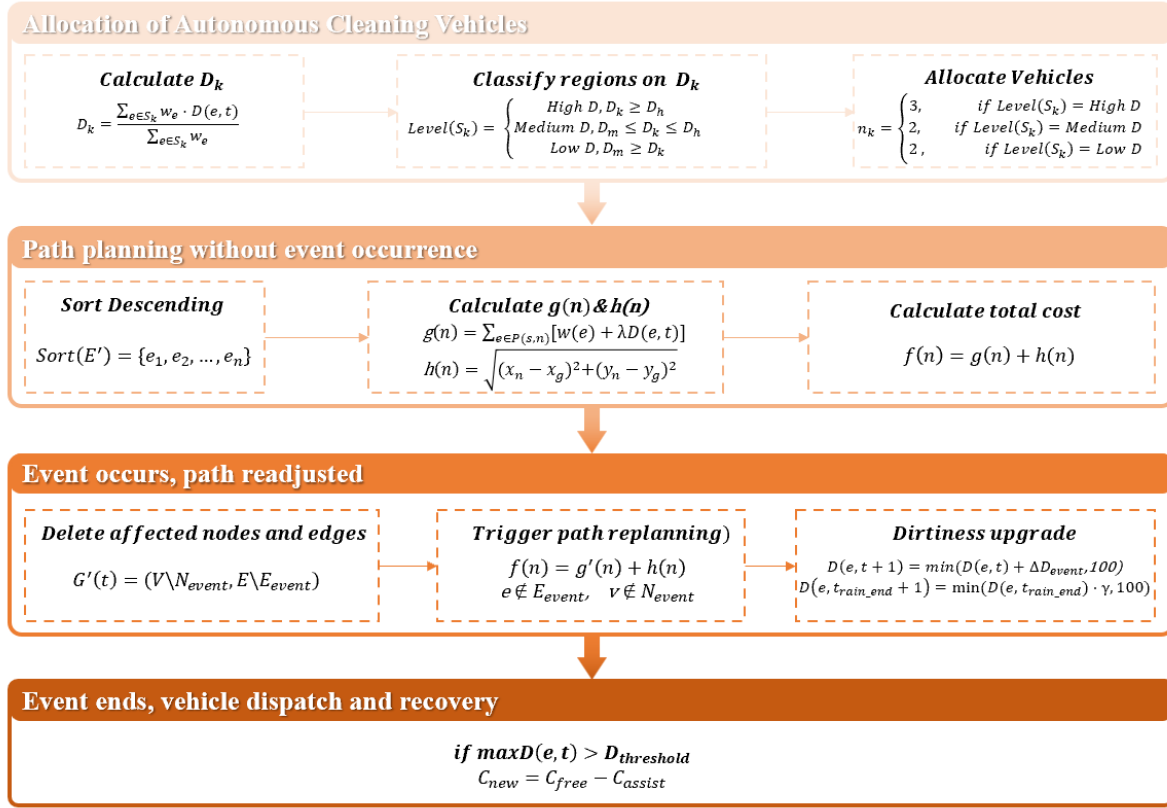


Figure 5: Framework of Dynamic A\* Model

In scenarios where no emergencies arise, cleaning vehicles proceed to their standard cleaning mode. Under these circumstances, it is essential to develop a cost function and constraints aimed at achieving the goal of prioritizing the cleaning of roads with higher dirt levels using the dynamic A\* algorithm. Initially, the edges  $E'$  that require cleaning are sorted in descending order of their dirt levels  $D(e, t)$ , and the cleaning vehicle selects, in sequence, the road with the highest dirt level that has not yet been cleaned as the next target for cleaning.

$$Sort(E') = e_1, e_2, \dots, e_n, \quad D(e_1, t) \geq D(e_2, t) \geq \dots \geq D(e_n, t) \quad (11)$$

In the traditional A\* algorithm, the heuristic function  $h(n)$  is typically the Euclidean distance. However, to prioritize the cleaning of roads with higher dirt levels, it is necessary to incorporate a weighing of  $D$  into the actual cost  $g(n)$ . Assuming the path from the starting point  $s$  to the target node  $g$  is  $P$ , the total cost is defined as:

$$f(n) = g(n) + h(n) \quad (12)$$

Here,  $g(n)$  represents the actual cost from the starting point to node  $n$ , defined as the path

distance plus a dirt-weighted term. For example, the formula is as follows:

$$g(n) = \sum_{e \in P} w(e) + \lambda \sum_{e \in P} D(e, t) \quad (13)$$

where  $w(e)$  is the length of edge  $e$ ,  $D(e, t)$  is the dirt level, and  $\lambda$  is a parameter that adjusts the weight of the dirt.

$h(n)$  represents the heuristic cost, using the Euclidean distance as a simple estimate:

$$h(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} \quad (14)$$

In scenarios devoid of events, the graph structure remains constant, yet the dirt levels evolve with time. The dynamic A\* algorithm reassesses the path at each temporal increment. To circumvent the pitfalls of frequent route alterations for the cleaning vehicle, which could diminish cleaning efficacy and squander computational resources, a path protection mechanism is instituted. Consequently, the cleaning vehicle is programmed to recalculate its route solely upon the fulfillment of its preceding objective.

The simulated situation obtained is shown in the fig. 6.

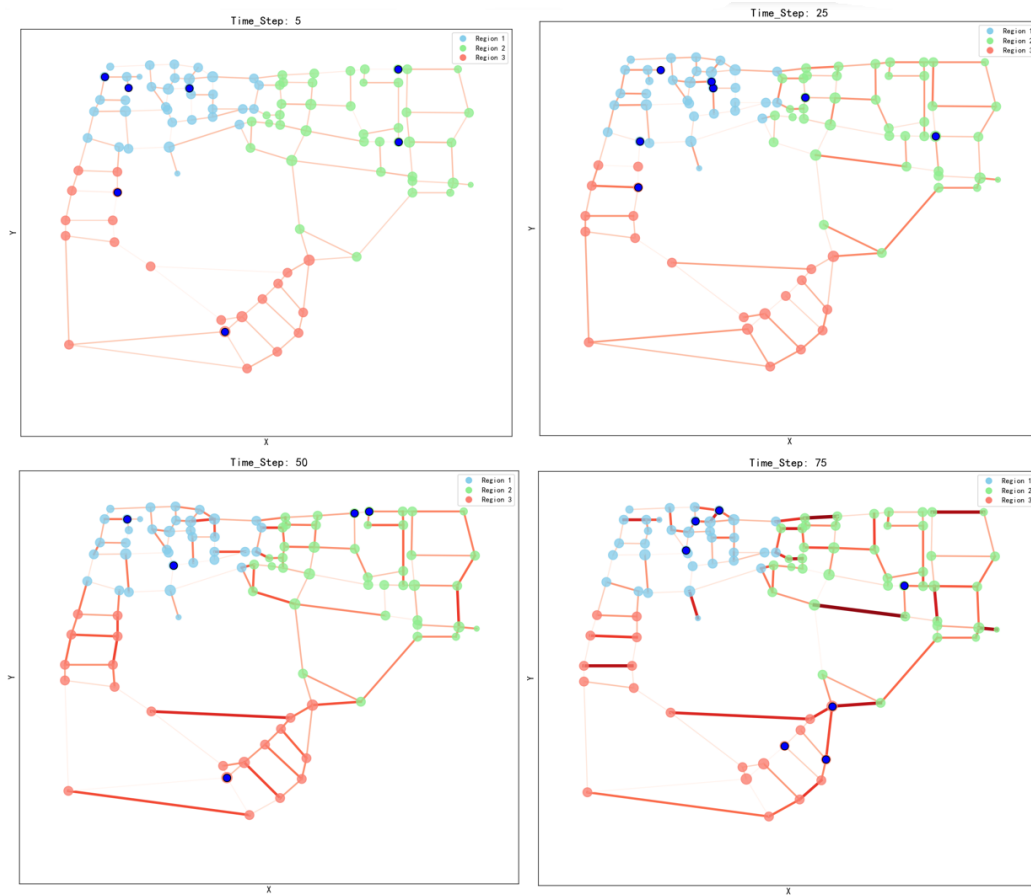


Figure 6: Simulation Scenario With No Events Occurs

Deep blue represents the cleaning car, and the path color from dark to light represents an increase in dirtiness.

## 6.2 Scheduling of campus automatic cleaning vehicles

However, when unforeseeable events occur, the execution of cleaning tasks by campus cleaning vehicles cannot be ignored. In the event of emergencies, to ensure that areas with heightened dirt levels resulting from such incidents are promptly addressed, real-time detection of these incidents is imperative.

At each time step  $t$ , the system evaluates the current campus environmental status based on the event set  $O(t)$ . Should an activity/construction event (A) occur, certain nodes and edges become temporarily unavailable, indicating that these nodes and edges are temporarily removed from the graph structure during the event period. In the case of a weather event (W), such as rainfall, cleaning vehicles are to suspend their cleaning tasks and wait in place; during the rain, paths are not updated, and the position of the cleaning vehicles remains unchanged, with only the dirt levels increasing over time according to  $a(e) \ln(t)$ , where  $\Delta t$  represents the number of time steps the rain persists.

The schematic image is shown in the fig. 7.

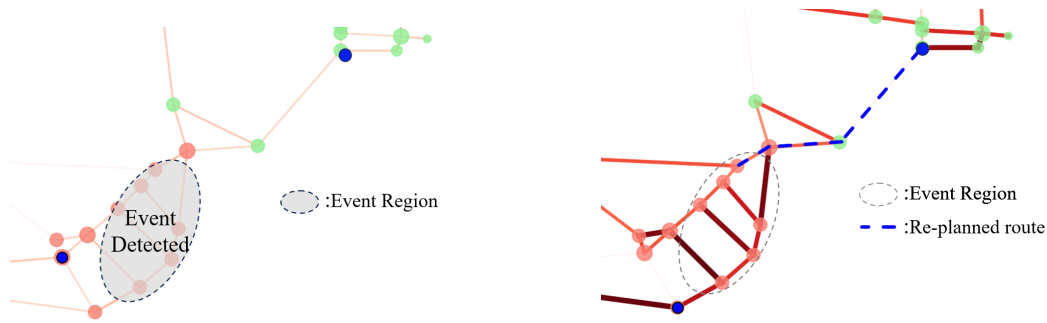


Figure 7: Schematic Image with Event Occurs

The pseudo-code is presented as algorithm 2.

---

**Algorithm 2:** Function check\_events( $t$ )

---

**Input:** Time step  $t$

**Output:** Event status  $O(t)$

```

1  $events \leftarrow O(t)$ 
2 if "Event"  $\in events$  or "Construction"  $\in events$  then
3   |  $N_{event} \leftarrow affected\_nodes(events)$   $E_{event} \leftarrow affected\_edges(events)$ 
3   |  $G'(t) \leftarrow G(t) \setminus (N_{event}, E_{event})$ 
4 end
5 if "Rain"  $\in events$  then
6   | pause_cleaning()
7 end
8 return events

```

---

Upon the detection of an emergency event within the campus, dynamic rescheduling of cleaning vehicles is warranted. It is crucial to distinguish between two broad categories of emergency events: those related to activities and construction, and those related to weather. The management strategies for these two types of events differ, and they are discussed separately below.

When events related to activities or construction occur, certain edges and nodes become temporarily inaccessible. The dynamic A\* algorithm requires real-time updates to the graph structure  $G'(t)$  and the path planning process is reinitiated with the following steps:

- Temporary removal of edges and nodes: Cleaning vehicles will execute the dynamic A\* on  $G'(t)$  during the next path update, ensuring they avoid the removed edges and nodes.
- Dynamic route adjustment of cleaning vehicles: Post-event, cleaning vehicles re-plan their routes based on the new graph structure and dirt distribution. The dynamic A\* algorithm remains consistent with the non-event scenario, but calculations utilize  $G'(t)$ :  $f(n) = g'(n) + h(n)$ , where  $g'(n)$  is calculated similarly to the non-event situation, excluding the removed edges and nodes.
- $D$  increase post-event: After the event,  $D$  of roads in the affected areas is significantly increased by  $\Delta D_{\text{event}}$ , the additional  $D$  increment following the event.

$$D(e, t + 1) = \min (D(e, t) + \Delta D_{\text{event}}, 100) \quad (15)$$

- Reallocate vehicles from other areas for assistance: If an event leads to a substantial increase in  $D$  in a particular area beyond the threshold  $D_{\text{threshold}}$ , a dispatch strategy is invoked to reallocate vehicles from other areas to assist in cleaning.

When weather events occur, cleaning vehicles halt their cleaning operations and adhere to the following rules:

- Cleaning vehicles wait in place without updating their paths.
- Dirt levels increase during the rain according to a specified formula.

$$D(e, t + 1) = \min (D(e, t) + a(e) \ln(\Delta D), 100) \quad (16)$$

- Post-rain, the dirt level of all roads is increased by a specific proportion, where  $\gamma > 1$  is used to amplify the dirt level, achieving a comprehensive increase in dirt post-rain.

$$D(e, t_{\text{rain\_end}} + 1) = \min (D(e, t_{\text{rain\_end}}) \cdot \gamma, 100) \quad (17)$$

After undergoing the aforementioned processes, additional cleaning vehicles are dispatched to the area where the emergency event occurred to expedite the cleaning process, thereby rapidly reducing the dirt level in that area and restoring it to the state previously mentioned, where no emergency events had occurred.

## 7 Benefit Evaluation Model

To fully evaluate the overall benefits of autonomous cleaning vehicles in campus management, a mathematical model is required to measure their operational costs and cleaning effectiveness. This model will assist in determining whether to increase or decrease the number of cleaning vehicles and will facilitate the analysis of the advantages of using autonomous cleaning vehicles compared to traditional manual cleaning methods. Through scientific cost-benefit analysis, it is possible to ensure the optimal allocation of resources, enhance the cleanliness of the campus environment, and control operational costs.

To make the analysis more objective, it is necessary to quantify the costs associated with using cleaning vehicles and hiring cleaning staff.

Firstly, **the cost of using autonomous cleaning vehicles** is calculated. The total operational cost of autonomous cleaning vehicles includes the purchase cost, maintenance cost, energy consumption, and labor cost. The formula is as follows:

$$C_{total}(N) = N \times (C_p + C_m + C_e + C_l) \quad (18)$$

$N$  is the number of autonomous cleaning vehicles;  $C_p$  is the purchase cost per vehicle;  $C_m$  is the annual maintenance cost per vehicle;  $C_e$  is the annual energy consumption cost per vehicle;  $C_l$  is the annual labor cost per vehicle.

Subsequently, **the cost of hiring cleaning staff** to clean the campus must be calculated. The total cost of traditional manual cleaning includes labor wages, equipment depreciation, and maintenance costs. The formula is as follows:

$$C_{manual} = W_h \times R + D_e + C_{em} \quad (19)$$

Where  $W_h$  is the annual working hours;  $R$  is the hourly wage for labor;  $D_e$  is the annual depreciation cost of the equipment;  $C_{em}$  is the annual maintenance cost of the equipment.

By comparing the costs of using autonomous cleaning vehicles with those of traditional manual cleaning, one can assess the economic viability of both cleaning methods.

In addition to the cost evaluation of the two cleaning methods, it is also necessary to evaluate the cleaning effectiveness of both approaches.

First, **calculate the cleaning effectiveness of using autonomous cleaning vehicles**. The cleaning effectiveness  $EE$  is defined as the weighted average of road dirtiness and road length.

$$E = \frac{\sum_{e \in E} D(e, t) \times L(e)}{\sum_{e \in E} L(e)} = \frac{\sum_{e \in E} D(e, t) \times L(e)}{W} \quad (20)$$

Where  $E$  is the cleaning effectiveness index, with lower values indicating better cleaning performance;  $L(e)$  represents the length of road  $ee$ ;  $W$  represents the total length of all roads.

Subsequently, **the effectiveness of traditional manual cleaning** is assessed. Similarly, the cleaning effectiveness of traditional manual cleaning, denoted as  $E_m$ , is also defined as the weighted

average of road dirtiness and road length. Although the efficiency and frequency of manual cleaning may differ, the formula for both is the same, as follows:

$$E_{manual} = \frac{\sum_{e \in E} D_{manual}(e, t) \times L(e)}{W} \quad (21)$$

To assess whether the introduction of autonomous cleaning vehicles brings benefits to campus cleaning, the cost-effectiveness ratio (CER) [1] is introduced. The cost-effectiveness ratio is used to measure the improvement in cleaning effectiveness per unit of cost. The formula is as follows:

$$CER = \frac{C_{total}(N)}{E} \quad (22)$$

From the formula, it is understood that a lower CER value indicates better cleaning effectiveness per unit cost. By analyzing the impact of increasing or decreasing one cleaning vehicle on total costs and cleaning effectiveness, it is possible to determine if there is an optimal number of vehicles. Marginal Cost (MC):

$$MC = \frac{\Delta C_{total}}{\Delta N} = C_p + C_m + C_e + C_l \quad (23)$$

Marginal Benefit (MB):

$$MB = \frac{\Delta E}{\Delta N} \quad (24)$$

If  $MB \geq MC$ , then adding one vehicle is beneficial. If  $MB < MC$ , then reducing the number of vehicles is advisable.

Comparing autonomous cleaning vehicles with traditional manual cleaning allows for an assessment of the advantages and disadvantages of both. Comparisons are made based on the following three indicators: Total Cost:

$$\text{Compare Cost} = C_{total}(N) \text{ vs. } C_{manual} \quad (25)$$

Cleaning Effectiveness:

$$\text{Compare Effectiveness} = E \text{ vs. } E_{manual} \quad (26)$$

Cost-Effectiveness Ratio:

$$CER_{auto} = \frac{C_{total}(N)}{E} \text{ vs. } CER_{manual} = \frac{C_{manual}}{E_{manual}} \quad (27)$$

Automatic cleaning vehicles show lower cost-benefit ratio and higher cleaning effect in long-term operation, and have obvious advantages over traditional manual cleaning.

## 8 Sensitivity Analysis

To evaluate the robustness of the dynamic A\* algorithm in our model, we performed a sensitivity analysis by perturbing the key parameter—the dirtiness weight factor ( $\lambda$ ). This parameter controls the importance of road dirtiness in the path planning algorithm, directly influencing the selection of paths for the cleaning vehicles. We conducted the sensitivity analysis by manually varying the value of  $\lambda$  by  $\pm 20\%$ . For each perturbation, the path planning results were recalculated, and the impact on the total cost and cleaning efficiency was recorded. The main goal was to assess how changes in  $\lambda$  affect the overall performance of the system, including cleaning vehicle utilization and path optimization.

Table 3: Sensitivity Analysis of Parameter  $\lambda$

Scenario	Total Cost (USD)	Cleaning Efficiency (E)
$\lambda$ (Baseline)	10,000	0.85
$\lambda + 20\%$	10,200	0.84
$\lambda - 20\%$	9,850	0.87

The comparison results are plotted in table 3. As shown in the figure, the variations in the total operational cost and cleaning efficiency are relatively small with the change in  $\lambda$ , indicating that the model performs stably even with fluctuations in the weight factor. This suggests that the model is robust and not overly sensitive to moderate changes in the importance of road dirtiness during path planning.

The slight changes in total cost and efficiency indicate that while the cleaning vehicle paths can shift slightly with different values of  $\lambda$ , the overall system's performance remains relatively stable. This demonstrates the model's generalizability and confirms its robustness in practical applications, even when the weight factor fluctuates.

## 9 model Assessment

### 9.1 Strengths

- **Highlights**

The concept of "dirtiness" and "dirtiness growth coefficient" is defined, enabling the quantification of road cleaning priorities.

- **Innovation**

A path protection mechanism is added, where autonomous cleaning vehicles only plan a new path after completing the current one.

- **Accuracy**

The dynamic A\* algorithm is used, which offers higher accuracy in path planning compared to the traditional A\* algorithm.

- **Stability**

A spectral clustering model is employed, demonstrating high stability for the relatively random distribution of dirtiness data within the campus.



- **Scalability**

If the campus expands, simply add the corresponding points and edges to the map, assign dirtiness parameters, and rerun the program to obtain new partitions, as well as cleaning vehicle path planning and dynamic scheduling solutions.

## **9.2 Weaknesses**

The dirtiness growth coefficient is an estimated value based on observations and lacks a rigorous basis. The dirtiness growth function is referenced by humans without enough data for fitting.

## References

- [1] K. Nandha Kumar et al. “Methodology for optimizing the number of electric vehicles deployed under a smart grid”. In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. 2013, pp. 4647–4652. DOI: 10.1109/IECON.2013.6699885.
- [2] Zhihong Lv et al. “Research on Global Off-Road Path Planning Based on Improved A\* Algorithm”. In: *ISPRS International Journal of Geo-Information* 13.10 (2024). ISSN: 2220-9964. DOI: 10.3390/ijgi13100362. URL: <https://www.mdpi.com/2220-9964/13/10/362>.
- [3] Hao Wang, Yan Yang, and Bing Liu. “GMC: Graph-Based Multi-View Clustering”. In: *IEEE Transactions on Knowledge and Data Engineering* 32.6 (2020), pp. 1116–1129. DOI: 10.1109/TKDE.2019.2903810.

# Appendices

Here is the Python code used in our model.

## Input python source:

---

```
def dynamic_A_star(graph, cleaner, target_node, k, t):
    def heuristic(node, target_node):
        x1, y1 = graph.node_positions[node]
        x2, y2 = graph.node_positions[target_node]
        return math.hypot(x2 - x1, y2 - y1)

    open_list = []
    heapq.heappush(open_list, (0 + heuristic(cleaner.current_node, target_node
        ↪ ), 0, cleaner.current_node))

    came_from = {}
    g_costs = {cleaner.current_node: 0}
    f_costs = {cleaner.current_node: heuristic(cleaner.current_node,
        ↪ target_node)}

    closed_set = set()

    while open_list:
        _, g, current_node = heapq.heappop(open_list)

        if current_node in closed_set:
            continue

        closed_set.add(current_node)

        if current_node == target_node:
            path = []
            while current_node in came_from:
                path.append(current_node)
                current_node = came_from[current_node]
            path.append(cleaner.current_node)
            return path[::-1]

        for neighbor in graph.graph.neighbors(current_node):
            edge_key = (min(current_node, neighbor), max(current_node,
                ↪ neighbor))
            distance = graph.graph[current_node][neighbor].get("weight", 1)

            dynamic_factor = k * D((current_node, neighbor), t)
            tentative_g = g + distance + dynamic_factor

            if neighbor in closed_set:
                continue

            if neighbor not in g_costs or tentative_g < g_costs[neighbor]:
                came_from[neighbor] = current_node
                g_costs[neighbor] = tentative_g
```

```
f_costs[neighbor] = tentative_g + heuristic(neighbor,  
    ↪ target_node)  
heapq.heappush(open_list, (f_costs[neighbor], tentative_g,  
    ↪ neighbor))
```

```
return []
```

---

```
def main_loop():  
    t = 0  
    while system_running:  
        events = check_events(t)  
  
        if "rain" in events:  
            update_dirtiness_rain_mode(G, a, t)  
        else:  
            if "event" in events or "construction" in events:  
                G_prime = update_graph_for_events(G, N_event, E_event)  
            else:  
                G_prime = G  
  
            for cleaner in C:  
                if not cleaner.path:  
                    target_edge = select_high_dirtiness_edge(G_prime, D)  
                    lambda_param = choose_lambda_param()  
                    path = dynamic_A_star(G_prime, cleaner, target_edge,  
                        ↪ lambda_param, t)  
                    cleaner.set_path(path)  
                else:  
                    cleaner.move_along_path()  
                    update_dirtiness(G, cleaner)  
  
            if event_ended:  
                restore_graph(G_prime, G)  
                increase_dirtiness_post_event(G, E_event, deltaD_event)  
                if need_reallocate_vehicles(G, D_threshold):  
                    reallocate_vehicles(C, G, D, D_threshold)  
  
    t += 1
```

---

## **Report on Use of AI**

### 1. OpenAI ChatGPT(Dec 6, 2024 version, ChatGPT-4o)

Query1: We are working on path planning for campus cleaning vehicles. Can you suggest some keywords for literature search related to this topic?

Output: Here are some relevant keywords for literature search in the field of campus cleaning vehicle path planning:

- Autonomous Path Planning
- Algorithm in Path Planning\*
- Mobile Robot Path Optimization
- Greedy Pathfinding Algorithms
- Vehicle Routing Problems (VRP)
- Multi-Robot Path Planning

These keywords should help guide your literature search and provide a foundation for understanding the various approaches in this domain.

### 2. OpenAI ChatGPT(Dec 6, 2024 version, ChatGPT-4o)

Query1: We have obtained the lengths of the campus roads through an electronic map and measuring function. Now we want to use these edge data to construct an undirected connected graph. What format should we process this data into for ease of future modeling?

Output: To construct an undirected connected graph from the campus road data, you can format the data as an edge list or an adjacency matrix.

### 3. OpenAI ChatGPT(Dec 6, 2024 version, ChatGPT-4o)

Query1: We categorize unexpected events into two types: activity-construction events and special weather events. What are the differences in handling these two types of events in the model?

Output: In the model, unexpected events are treated differently based on their nature:

1.Activity-Construction Events: These events temporarily block certain paths or areas, so the model will mark these paths as impassable (i.e., remove them from the graph). The cleaning vehicle will then re-plan its path to avoid these blocked areas.

2.Special Weather Events: For events like rain, which don't physically block paths, the model may adjust the road's cleanliness level, increasing the dirtiness of the roads and possibly triggering additional cleaning tasks. Instead of blocking paths, weather events typically lead to changes in the environment's status, which influence cleaning priorities.

### 4. OpenAI ChatGPT(Dec 6, 2024 version, ChatGPT-4o)

Query: The dynamic A\* algorithm in our current version causes the cleaning vehicle to get stuck in place. Debugging revealed that after each timestamp update, the vehicle recalculates the path, causing the path to become chaotic. After around ten steps, the vehicle freezes. I would like to add a path protection mechanism, where the vehicle will only recalculate the path once it has completed the previous cleaning task. Please help me improve the Python code.

Output: In order to fix the issue of the vehicle getting stuck, we can implement a path protection mechanism that ensures the vehicle only re-plans its path after completing the previous task. Below is a sample Python code to modify the path planning logic:

---

```
def update_path(cleaning_vehicle,current_task,path,target edge):  
    # Check if the vehicle has completed the current task  
    if cleaning_vehicle.current_edge == target_edge:  
        #Recalculate the path if the task is complete  
        new path =calculate new path(cleaning vehicle)  
        return new path  
    else:  
        # Keep the existing path until the task is coplete  
        return path
```

---

This code checks if the vehicle has reached the target edge before recalculating the path. If not, the current path is maintained until the task is completed.