未来技术学院
FUTURE TECH

Development of a Multi-Robot Cooperative Charging Scheduling System and Simulation Environment Based on Multiple Intelligent Methods

Project Report

编号	姓名	学号	学院	专业	班级
组员 1	张越	202364820921	未来技术学院	数据科学与大数据技术	二班
组员 2	陈思涵	202330420212	未来技术学院	数据科学与大数据技术	一班
组员 3	黄梓纹	202364820501	未来技术学院	数据科学与大数据技术	一班
组员 4	陈郅	202364830031	未来技术学院	数据科学与大数据技术	二班
组员 5	颜诗淇	202364871202	未来技术学院	数据科学与大数据技术	二班

2025 年 6 月 8 日

Abstract

Against the backdrop of rapid development in smart campuses and new energy transportation systems, how to efficiently schedule multiple autonomous charging robots to meet dynamic and diverse charging demands of electric vehicles has become a key technical challenge in intelligent energy management. In real-world applications, charging tasks often exhibit complex characteristics such as uncertain arrival times, heterogeneous energy demands, spatial dispersion, and time constraints, making traditional static scheduling models unsuitable for practical operation scenarios. There is an urgent need to design intelligent scheduling methods with online responsiveness and global optimization capabilities.

This paper addresses the problem of multi-robot dynamic charging in campus scenarios by constructing a complete scheduling simulation platform and proposing a combined model that formulates the problem as a **Dynamic Capacitated Routing and Scheduling Problem (DCRSP)** and a **Dynamic Bipartite Matching Problem (DBMP)**. On this basis, three representative scheduling algorithms are designed and implemented: the **Hungarian algorithm based on graph matching**, the **adaptive scheduling strategy based on reinforcement learning (PPO)**, and the **hybrid particle swarm optimization algorithm combining set encoding and local search (Hybrid S-PSO-D)**. These three strategies tackle the dynamic scheduling problem from heuristic, data-driven, and swarm intelligence perspectives respectively, offering strong complementarity and comparative analysis value.

The system models the charging campus environment using a discrete grid map, supporting dynamic vehicle generation, task queue maintenance, energy consumption modeling, and simulation of a three-phase nonlinear charging mechanism. Under six typical experimental scenarios (covering variations in task density, map scale expansion, task dynamics, and path conflicts), this paper conducts a systematic comparative analysis around key metrics such as energy consumption, average waiting time, task completion rate, timeout ratio, and scheduling fairness.

Experimental results show that the **PPO strategy** demonstrates stable low waiting times and low-variance scheduling performance in various dense and dynamic task scenarios, indicating good generalization adaptability. The **S-PSO-D strategy** achieves higher task completion rates and robust scheduling performance in large-scale, high-complexity scenarios, making it a high-performance scheduling solution suitable for deployment in complex environments. The **Hungarian strategy** is efficient in small-scale scenarios, but its performance degrades significantly under increased task dynamism and load pressure.

Overall, the system model and multi-strategy scheduling framework proposed in this paper provide a deployable, scalable, and transferable solution for campus-level charging scheduling problems, offering valuable reference for the intelligent evolution of automated energy management systems.

Table of Contents

1 Project Requirements and Scoring Criteria

1.1 Project Background

Assume you are the manager of an industrial park where a fleet of autonomous charging robots operates. These robots, after being fully charged themselves (or carrying a charged battery), can move to any location within the park to charge vehicles parked there. Your task is to arrange the charging robots in a way that maximizes their utilization.

1.2 Task Requirements

1. Simulate the arrival time, location, remaining battery level, departure time, and required battery level at departure for each vehicle. These values can be randomly generated within reasonable ranges.
2. Charging speed is not fixed; the more energy a vehicle has, the slower it charges.
3. When a robot runs low on power, it must return for recharging or battery swapping. It must not be left stranded.
4. Battery swapping must consider the number of available batteries.
5. At least two different scheduling strategies must be implemented, such as nearest-task-first or maximum-task-first.
6. Simulate at least three problem scenarios of varying scales.
7. Students with sufficient capability are encouraged to explore advanced algorithms such as reinforcement learning, hyper-heuristic methods, metaheuristic methods, or multi-agent methods.
8. Graphical interface implementation will earn bonus points.

1.3 Assignment Format

- Group project (3 – 5 members), programming language is not restricted.
- Final deliverables include: a demonstration video; a project report; and source code.
- The final presentation requires a PowerPoint (PPT) and a live demonstration of the program execution.

1.4 Scoring Criteria

Total Score (120 points) = Completion (50) + Difficulty (40) + Subjective Assessment (10) + Bonus (20)
(Note: Total score will be capped at 100)

- **Completion (50 points)**: Reasonableness of the simulation and completeness of the implemented features.

- **Difficulty (40 points)**: Whether a sufficiently challenging environment is simulated, and whether appropriate data structures and algorithms are used.
- **Subjective Assessment (10 points)**: Whether the presentation form is reasonable and satisfactory to the evaluators.
- **Bonus (20 points)**: Implementation of additional algorithms or graphical interface.

2 Problem Background

In recent years, with the rapid advancement of new energy technologies and the construction of smart industrial parks, traditional static energy management models are facing higher demands. According to data provided by the State Grid and various pilot parks, during peak periods, a medium-to-large industrial park may have hundreds of electric vehicles simultaneously waiting for charging, posing serious challenges to the park' s energy infrastructure and scheduling capacity. In the face of constantly changing vehicle distributions and battery statuses within the park, manual management and scheduling of charging robots can no longer meet the needs for high efficiency, low energy consumption, and real-time responsiveness.

Meanwhile, driven by the goals of green energy development and carbon neutrality, park managers must also pay more attention to path optimization and response efficiency in energy usage, aiming for environmental sustainability. Therefore, developing an automated charging robot scheduling system to efficiently coordinate charging tasks between robots and vehicles has become a key issue in intelligent energy management for smart parks.

3 Problem Restatement

We consider the following typical scenario:

Electric vehicles in need of charging gradually arrive at the parking areas of the park. They differ in remaining battery levels, charging demands, and departure times. New tasks arrive dynamically, and charging robots must respond promptly. Robots have limited energy and must autonomously determine when to return to a charging station for recharging or battery swapping during service. The system operates under resource constraints with frequently changing states.

In this scenario, no vehicles are known at system startup. All tasks are dynamically disclosed as simulation time progresses. Thus, the dynamic and real-time nature of tasks becomes a core problem that must be addressed in the design of the scheduling system.

The central challenge of the dynamic charging robot scheduling problem is how to complete the maximum number of tasks in such a dynamic environment, while optimizing for the objectives of "minimizing total energy consumption + minimizing task waiting time + minimizing task waiting time variance."

This system adopts this framework for modeling, using real-time scheduling and path reconstruction mechanisms, combined with intelligent algorithms to achieve efficient task assignment.

4 Problem Assumptions

1. During the entire scheduling process, the energy level of all robots is greater than 0.

2. Charging stations have unlimited energy to recharge robots.

3. Charging occurs between two types of object pairs: vehicles and robots, robots and charging stations. Energy loss is not considered in the process. Hence, total system energy consumption consists of: energy required for tasks + energy consumed during robot movement. All charging follows a three-phase nonlinear lithium battery model: constant current, constant voltage, and trickle charge.

4. Vehicles actively avoid moving robots to reach parking spots. The simulation ignores the process of vehicle entry; vehicles wait for charging once parked.

5. Vehicles with higher remaining battery levels are more willing to wait; those with less energy prefer quicker service.

6. Each task can only be completed by one robot. Once a robot is assigned a task, the task cannot be interrupted until completion.

7. Given the area represented by each grid cell and the size of simulated robots, robots avoid each other during movement and will not collide.

8. Given the size of the grid and the scale of robots and vehicles, a robot and a vehicle can complete a charging task within the same grid cell representing a parking space.

9. Vehicles entering the park may appear at any unoccupied parking space. At any given time, only one vehicle can occupy a parking spot.

10. Charging stations can accommodate an unlimited number of robots.

5 Problem Modeling

5.1 Dynamic Bipartite Matching Problem (DBMP)

Essentially, the charging task scheduling problem can be modeled as a Dynamic Bipartite Matching Problem (DBMP). From this modeling perspective, at each scheduling moment, the system needs to construct a weighted bipartite graph between the set of charging robots and the set of vehicles awaiting service, based on the current state, to complete efficient task assignments.

We define a Dynamic Task Matching Problem (DTMP), that is: in a complex environment where charging tasks arrive dynamically and robots have limited energy, the system must construct a task allocation graph in real time and solve for the optimal matching to improve task completion efficiency and reduce overall energy consumption, while ensuring robot operational capability.

This modeling method effectively captures the ”reachability” and ”priority” relationships of tasks, making it suitable for high-density scheduling scenarios where tasks are time-sensitive and resources are scarce. In existing studies, similar matching models have been widely applied in fields such as taxi dispatch, UAV deployment, and order system matching, demonstrating good solution efficiency and scalability.

With the help of real-time state awareness and graph matching algorithms (such as the Hungarian algorithm, maximum weight matching, approximate streaming matching, etc.), the scheduling system can dynamically update the allocation graph and obtain high-quality matching results at

each scheduling round. However, improving the stability of the matching graph and the overall scheduling performance under dynamic changes, real-time feedback, and multiple constraints remains a major challenge in current intelligent scheduling research.

To efficiently handle dynamically arriving charging tasks, we propose modeling the charging robot scheduling problem as a **Dynamic Bipartite Matching Problem (DBMP)**. This modeling emphasizes the task-robot matching relationship in each scheduling round and can be combined with maximum weight matching algorithms or approximate streaming algorithms to achieve real-time task assignment.

At each scheduling time t, we define a weighted bipartite graph $G_t = (R_t, T_t, E_t)$, where:

- $R_t = \{r_1, r_2, \ldots, r_m\}$ is the set of schedulable robots at time t;
- $T_t = \{t_1, t_2, \ldots, t_n\}$ is the set of tasks that have been disclosed but not yet completed before time t;
- $E_t \subseteq R_t \times T_t$ is the set of edges connecting robots and tasks;
- Each edge $(r_k, t_i) \in E_t$ indicates that robot r_k is capable of completing task t_i, with a weight w_{ki} representing the "matching reward" or "negative cost" of this pairing.

The matching weight w_{ki} can be calculated by the following weighted sum of multiple factors:

$$w_{ki} = -\left(\beta_1 \cdot \mathrm{dist}(r_k, t_i) + \beta_2 \cdot \delta_i + \beta_3 \cdot \mathrm{urgency}(t_i)\right)$$

where:

- $\mathrm{dist}(r_k, t_i)$ is the shortest path cost from robot r_k to the vehicle location of task t_i;
- δ_i is the waiting time of task t_i;
- $\mathrm{urgency}(t_i)$ may be defined by $l_i - t$ or d_i/e_i to represent task urgency;
- $\beta_1, \beta_2, \beta_3 \in \mathbb{R}_+$ are tunable weighting coefficients.

We introduce a binary decision variable y_i^k to indicate whether task t_i is assigned to robot r_k:

$$y_i^k = \begin{cases} 1, & \text{if task } t_i \text{ is assigned to robot } r_k \\ 0, & \text{otherwise} \end{cases}$$

The objective is to maximize the total matching weight at the current time (i.e., minimize the matching cost):

$$\max \sum_{k=1}^{m} \sum_{i=1}^{n} w_{ki} \cdot y_i^k$$

Constraints

- **Task uniqueness constraint:**

$$\sum_{k=1}^{m} y_i^k \leq 1 \quad \forall i = 1, \ldots, n$$

Each task may be assigned to at most one robot.

- **Robot concurrency constraint (optional):**

$$\sum_{i=1}^{n} y_i^k \leq 1 \quad \forall k = 1, \ldots, m$$

Each robot can take on only one task at a time (if round-based scheduling is applied).

- **Reachability constraint (edge set definition):**

$$y_i^k = 0 \quad \text{if } (r_k, t_i) \notin E_t$$

Robots can only be assigned to reachable tasks.

- **Energy feasibility constraint:**

$$y_i^k = 1 \Rightarrow \text{cost}_k(t_i) + d_i \leq q_k$$

Robot must have sufficient remaining energy to complete the task (travel + charging).

Form of the Solution

A matching solution M_t to this problem at time t is a set of assignment pairs:

$$M_t = \{(r_k, t_i) \mid y_i^k = 1\}$$

This matching must satisfy all constraint conditions and maximize the current reward function $\sum w_{ki} \cdot y_i^k$.

5.2 Dynamic Capacitated Robot Scheduling Problem (DCRSP)

Essentially, the charging task scheduling problem can be regarded as a robot-oriented instance of the Dynamic Capacitated Vehicle Routing Problem (DCVRP).

We define a Dynamic Capacitated Robot Scheduling Problem (DCRSP), which refers to: under continuously arriving dynamic tasks, the system must assign the most appropriate service paths to multiple charging robots in real time, in order to improve overall task completion rate, reduce energy waste, and simultaneously minimize and balance user waiting times.

Similar to traditional scheduling problems, this problem involves multiple constraint dimensions such as path cost, energy state, and task priority. In existing studies, similar dynamic task scheduling problems have been extensively studied in the vehicle routing problem (VRP) domain, including VRP with time windows (VRPTW), VRP with pickup and delivery, green VRP with charging constraints, and robotic task scheduling.

Thanks to the development of wireless communication, positioning systems, and intelligent navigation technologies, schedulers now have the ability to acquire real-time status information of vehicles and robots, and theoretically can dynamically adjust scheduling plans at every moment. However, designing an efficient, stable, and scalable scheduling system under multiple constraints remains a frontier challenge in intelligent energy management.

DCRSP is a dynamic multi-agent resource scheduling problem extended from the classic DCVRP model, applied to a grid-based smart industrial park scenario. In this setting, electric

vehicles continuously and randomly arrive at parking areas in the park, requesting charging services; meanwhile, the park is equipped with several mobile charging robots (with limited energy) and fixed charging stations for robot recharging.

We abstract the park as an undirected graph $G = (V, E)$, where V is the set of nodes representing all accessible grid points. For modeling purposes, we further divide V into the following subsets: the parking node set V^p and the charging station node set V^c, namely:

$$V = V^p \cup V^c \quad V^p \cap V^c = \emptyset$$

Here, V^p denotes the fixed parking spots in the park, and V^c represents the charging station locations.

The edge set E is defined as:

$$E = \{e_{ij} \mid e_{ij} = (c_i, c_j),\ c_i, c_j \in V,\ i < j\}$$

Each edge $e_{ij} \in E$ is associated with a distance weight w_{ij} representing the movement cost between nodes c_i and c_j, defined as their minimum Manhattan distance on the grid:

$$w_{ij} = \min\left(|x_i - x_j| + |y_i - y_j|\right)$$

where (x_i, y_i) and (x_j, y_j) are the coordinates of nodes c_i and c_j.

Let $R = \{r_1, r_2, \ldots, r_m\}$ denote the set of m charging robots. Each robot has a battery capacity of $Q > 0$, consumes one unit of energy per grid step, and must autonomously decide when to return for recharging.

Let $T = \{t_1, t_2, \ldots\}$ be the set of charging tasks. Each task t_i corresponds to a vehicle v_i that has parked in the park, arriving at time a_i, and has the following attributes:

- e_i: remaining energy of the vehicle;
- d_i: required energy for vehicle charging;
- l_i: latest departure time of the vehicle;
- $p_i \in V^p$: the parking location of the vehicle.

Let $\delta_i \in \mathbb{R}_+$ denote the waiting time from task t_i's arrival to being taken by a robot, and let $\mathrm{var}(\delta)$ be the variance of all task waiting times. Each robot can only perform one task at a time, and the task is non-interruptible once assigned. Tasks are dynamically disclosed, i.e., at time t, the known task set is $T_t = \{t_i \mid a_i \leq t\}$.

Each robot r_k must plan a path $P_k = [v_1, v_2, \ldots, v_j]$ starting from its current position to either a task or charging station, subject to its current energy. Let $\mathrm{cost}_k(P_k)$ denote the energy consumption of path P_k.

Additionally, to ensure operational feasibility and energy replenishment, each path P_k must satisfy the following structural constraint:

$$P_k = [v_1, \ldots, v_j] \quad \text{where} \quad v_1, v_j \in V^c$$

i.e., paths must begin and end at charging stations.

Define the variables:

$$x_{ij}^k = \begin{cases} 1, & \text{if robot } k \text{ moves directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases} \qquad y_i^k = \begin{cases} 1, & \text{if task } i \text{ is assigned to robot } k \\ 0, & \text{otherwise} \end{cases}$$

The objective function is:

$$\min \ \alpha_1 \sum_k \sum_{(i,j)\in E} w_{ij} x_{ij}^k + \alpha_2 \sum_i \delta_i + \alpha_3 \cdot \mathrm{var}(\delta)$$

where $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}_+$ are weight coefficients.

Constraints

- **Task exclusivity constraint:**

$$\sum_{k=1}^{m} y_i^k = 1 \quad \forall i = 1, \ldots, |T|$$

- **Non-interruptibility constraint:**

$$y_i^k = 1 \Rightarrow \text{task } t_i \text{ must be executed until completion}$$

- **Flow conservation constraint:**

$$\sum_{i=0}^{n} x_{ij}^k = y_j^k, \quad \sum_{j=0}^{n} x_{ij}^k = y_i^k \quad \forall i, j = 1, \ldots, n, \ \forall k$$

- **Capacity constraint:**

$$\sum_{i=1}^{|T|} y_i^k \cdot d_i + \sum_{(i,j)\in E} w_{ij} x_{ij}^k \le Q \quad \forall k = 1, \ldots, m$$

- **Path structure constraint (starting and ending at charging stations):**

$$P_k = [v_1, \ldots, v_j] \Rightarrow v_1, v_j \in V^c \quad \forall k$$

- **Parking space exclusivity constraint:**

$$\sum_{i\in T} \mathbf{1}_{p_i = v} \le 1 \quad \forall v \in V^p$$

- **Unlimited charging station capacity:** No explicit capacity variable is introduced.
- **No-collision assumption:** Path planning assumes obstacle avoidance is automatically handled; no constraints are added.

Form of the Solution

In this problem, a feasible scheduling solution S includes the complete specification of path planning, task allocation, and energy replenishment for each robot, formally defined as:

$$S = \{(P_k, \mathcal{T}_k) \mid r_k \in R\}$$

where:

- $P_k = [v_1^k, v_2^k, \ldots, v_{j_k}^k]$ denotes the path sequence executed by robot r_k;
- $\mathcal{T}_k \subseteq T$ is the subset of tasks assigned to robot r_k;
- The path must satisfy: $v_1^k, v_{j_k}^k \in V^c$, i.e., both start and end at charging stations;
- For all k, the task locations in $\mathcal{T}_k$ must be covered in P_k;
- Across all tasks: $\bigcup_k \mathcal{T}_k = T$ and $\mathcal{T}_k \cap \mathcal{T}_{k'} = \emptyset$ for $k \neq k'$, i.e., tasks are uniquely assigned;
- The energy cost $\text{cost}_k(P_k)$ for all paths must satisfy energy constraints.

Therefore, solution S ensures:

- Path feasibility (valid in graph, satisfies Manhattan cost, and starts/ends at V^c);
- Task reachability and coverage (path passes through all assigned task locations);
- Assignment uniqueness and completeness (each task assigned once, no omission);
- Compliance with energy constraints (within robot battery limits).

6 Scheduling Strategy Design

6.1 Strategy I: Hungarian Scheduler Based on Dynamic Bipartite Matching

Theoretical Foundation

This strategy is based on the **Dynamic Bipartite Matching Problem (DBMP)** model, with the objective of minimizing the total scheduling cost. By constructing a feasible matching graph between robots and tasks, the **Hungarian Algorithm** is used to achieve a globally optimal matching between robots and tasks within each scheduling round, under the condition of assignment feasibility. This method can be seen as an enhanced variant of the Online Assignment Problem, focusing on instantaneous global optimality rather than long-term rolling planning.

Model Definition

Let the current time be t, and the system defines:

- Robot set $R_t = \{r_1, r_2, \ldots, r_m\}$, where each robot must be in the `idle` or `returning_idle` state, and have power level $q_k \geq \theta$;
- Task set to be scheduled $T_t = \{t_1, t_2, \ldots, t_n\}$, where each task t_j has been disclosed ($a_j \leq t$), not yet served and not yet assigned;
- Cost matrix $C \in \mathbb{R}^{m \times n}$, where each element C_{ij} represents the comprehensive scheduling cost of assigning task t_j to robot r_i.

Each cost C_{ij} is defined as:

$$C_{ij} = \begin{cases} \text{dist}(r_i, t_j) + \text{energy}(t_j) - \lambda \cdot \frac{\delta_j}{1+\delta_j}, & \text{if feasible} \\ +\infty, & \text{if not feasible} \end{cases}$$

where:

- dist(r_i, t_j): shortest path distance computed using the A^* algorithm (movement cost);
- energy(t_j): energy required for the vehicle task (i.e., `required_energy` − `initial_energy`);
- $\delta_j = t - a_j$: current waiting time of task j;
- λ: penalty coefficient for waiting time (adjusts the priority of long-waiting tasks);
- If the path is unreachable or energy is insufficient (unable to complete task and return to station), assign $+\infty$ to exclude this assignment.

Scheduling Algorithm Procedure

In each scheduling cycle, the system performs the following steps to complete task assignment:

Algorithm 1: assign_tasks_hungarian Scheduling Process

Retrieve all waiting tasks T_t and idle robot set R_t
Initialize cost matrix $C \leftarrow \infty^{m \times n}$
for each robot $r_i \in R_t$ **do**
  for each task $t_j \in T_t$ **do**
    if `is_feasible`(r_i, t_j) is satisfied **then**
      Compute path cost, energy cost, and waiting penalty
      Set $C_{ij} = \text{dist} + \text{energy} - \lambda \cdot \frac{\delta_j}{1+\delta_j}$
    end if
  end for
end for
Call `linear_sum_assignment`(C) to compute the minimum-cost one-to-one matching pairs (i^*, j^*)
for each matching pair (r_i, t_j) **do**
  If $C_{ij} < \infty$, assign task t_j to r_i and plan the path
end for

Advantages and Mechanism Analysis

- **Global Optimization Capability**: Builds a complete bipartite graph between all assignable robots and tasks and solves based on the minimum cost matching principle, ensuring the total cost is minimized in the current round;
- **Tunable Waiting Priority Weight**: Introduces a negative exponential penalty term for waiting time to avoid task starvation while controlling suppression of new tasks;
- **High Computational Efficiency, Suitable for Medium Scale**: For $n \leq 100$, the algorithm's average runtime is under 50ms, meeting the sub-second response requirement;
- **Naturally Embedded Constraints**: By setting $+\infty$ for infeasible edges, the scheduling process inherently satisfies path reachability and energy feasibility constraints.

Applicability and Typical Scenarios

This strategy is applicable in the following scheduling environments:

- Task set is limited and state is relatively stable (e.g., no frequent recharging interference);
- System emphasizes task fairness and low waiting time, and the scheduling response must be within 100ms;
- Can serve as a baseline for comparison with other intelligent algorithms (e.g., reinforcement learning, metaheuristics).

In this system, this strategy is integrated as a standard scheduling module and also serves as a reference for particle initialization in the S-PSO-D algorithm.

6.2 Strategy II: Adaptive Scheduler Based on PPO Deep Reinforcement Learning

This strategy introduces the Proximal Policy Optimization (PPO) algorithm from deep reinforcement learning. By interacting with a dynamic simulation environment, it learns an adaptive scheduling policy for complex task arrivals in a multi-robot system. Unlike traditional heuristic methods, this strategy does not rely on explicit rules, instead autonomously explores and develops stable, generalizable scheduling behaviors to optimize global scheduling performance over the long term.

Scheduling Modeling and Reinforcement Learning Framework

The park scheduling problem is modeled as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where:

- $\mathcal{S}$ is the state space, representing complete information of the scheduling system at any moment;
- $\mathcal{A}$ is the action space, corresponding to all possible scheduling decisions under the current state;
- $\mathcal{P}$ is the state transition function, determined by environment dynamics and actions;
- $r(s, a)$ is the immediate reward function that guides the policy to optimize scheduling goals;
- $\gamma \in [0, 1]$ is the discount factor for future rewards.

The PPO policy aims to maximize the following objective function:

$$J(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the policy ratio, $\hat{A}_t$ is the advantage estimate, and ϵ controls the stability of policy updates.

State and Action Modeling

State Space $\mathcal{S}$ The system state s_t is a composite representation of all robot states. Each robot's state is defined as:

$$\text{obs}_i = \left[\frac{b_i}{B_{\max}}, \frac{x_i}{H}, \frac{y_i}{W} \right]$$

where b_i is the current battery level, and (x_i, y_i) are the robot' s position coordinates. H and W are normalization factors for the map height and width.

Action Space $\mathcal{A}$ To ensure the action space is discrete and finite, the following action set is defined:

- a_j indicates dispatching the robot to task j;
- a_{return} commands the lowest-energy robot to return to the nearest charging station;
- a_{wait} maintains the current state;

The agent can choose the most appropriate action in any state to maximize long-term return.

Reward Function Design

The reward function incorporates multiple objectives of the scheduling system and is defined as:

$$\begin{aligned} r_t = & +1200 \cdot \mathbb{I}_{\text{task\_completed}} - 1 \cdot \Delta t - 30 \cdot \mathbb{I}_{\text{low\_battery}} - 200 \cdot \mathbb{I}_{\text{dead\_battery}} \\ & - 0.05 \cdot \text{energy\_consumed} - 0.2 \cdot \bar{\delta}_t - \mathbb{I}_{\delta>50} \cdot (\delta_t - 50) \\ & - \mathbb{I}_{\text{conflict}} \cdot \lambda_{\text{conflict}} + 1000 \cdot (\text{completion\_rate}_t - 0.3) \end{aligned}$$

This design balances task response efficiency, energy optimization, battery management, and system stability, guiding the policy to learn efficient scheduling behaviors.

Training Process and Parameter Settings

We use `Stable-Baselines3` for parallel training across multiple environments. Key parameters are as follows:

- Learning rate: 2×10^{-4}, clip ratio: 0.15;
- Discount factor $\gamma = 0.98$, GAE parameter $\lambda = 0.95$;
- Rollout length: 1024, batch size: 256, number of epochs: 8;
- Multi-scenario training: covering small/medium/large maps and uniform/clustered/mixed task distributions;
- Use of frame stacking or adjacency matrices to build global observation structure and enhance spatial awareness;
- After training, weights are exported as a `.pth` file and deployed in the scheduling inference system.

Strategy Advantages and Application Scenarios

- **Strong Generalization Ability**: The trained policy can be transferred to unseen scenarios;
- **Unified Multi-Objective Optimization**: The reward function integrates completion rate, waiting penalties, energy consumption, and more;
- **Dynamic Adaptability**: Capable of adjusting scheduling strategies in real-time in response to unexpected events and environmental changes;
- **Suitable for Large-Scale Systems**: Exhibits significant advantages when handling large task volumes and high scheduling complexity.

Overall, the PPO strategy offers a learnable and scalable intelligent scheduling mechanism suitable for task allocation in complex, dynamic multi-robot environments.

6.3 Strategy III: Multi-Objective Hybrid Scheduling Algorithm Based on Set-based Particle Swarm Optimization

Strategy Motivation and Theoretical Foundation

In dynamic charging task scenarios, task disclosure times are uncertain, energy resources are limited, and service path combinations exhibit combinatorial explosion. Traditional heuristic scheduling strategies such as nearest-task-first or maximum-residual-energy matching lack global optimization capability and struggle to meet multi-objective optimization requirements.

To address this, we propose a **Multi-Objective Hybrid Scheduling Algorithm based on Set-based Particle Swarm Optimization with Dynamic Local Refinement** (Hybrid S-PSO-D). This method integrates greedy initialization, set-encoded particle evolution, and path-level local search to jointly optimize scheduling performance and real-time responsiveness in dynamic task environments.

Theoretically, this method belongs to the paradigm of *Discrete Set-based Swarm Intelligence*, constructing a task-set-oriented search model over the solution space and optimizing task assignment, path planning, and energy consumption under multiple constraints.

Model Structure and Objective Function

Let the current time be t, with m robots $R = \{r_1, \ldots, r_m\}$ and task set $T_t = \{t_1, \ldots, t_n\}$, where each task t_i has attributes $\{a_i, l_i, d_i, p_i\}$ denoting arrival time, latest departure time, required energy, and parking position, respectively.

We define the scheduling objective as a weighted multi-objective minimization problem:

$$\min_S \quad \alpha_1 \cdot \text{Distance}(S) + \alpha_2 \cdot \text{Wait}(S) + \alpha_3 \cdot \text{Imbalance}(S) + \alpha_4 \cdot \text{Unassigned}(S) + \alpha_5 \cdot \text{Overdue}(S) + \alpha_6 \cdot \text{StdDev}(S)$$

Where:

- $\text{Distance}(S)$: total travel path length (energy cost);
- $\text{Wait}(S)$: average task waiting time;

- Imbalance(S): standard deviation of task count among robots;
- Unassigned(S): number of unassigned tasks;
- Overdue(S): number of tasks completed after the vehicle's latest departure time;
- StdDev(S): variance of task waiting times.

Particle Encoding and Search Structure Design

Each particle P represents a complete scheduling solution S, encoded as:

$$P = \{(r_k, \mathcal{T}_k, s_k) \mid r_k \in R\}$$

where $\mathcal{T}_k$ is the sequence of tasks assigned to robot r_k, and s_k is the charging station the robot ultimately returns to. This structure supports joint optimization of task allocation and path planning.

To enhance feasibility and guide search, the particle velocity V is defined as a preference probability matrix for tasks and charging stations, and updated in each iteration using both personal best (pBest) and global best (gBest) solutions:

$$v_{ij}^{(t+1)} = \omega v_{ij}^{(t)} + c_1 r_1 (p_{ij}^{\text{best}} - x_{ij}^{(t)}) + c_2 r_2 (g_{ij}^{\text{best}} - x_{ij}^{(t)})$$

Hybrid Optimization Process

Algorithm 2: Hybrid S-PSO-D Scheduling Algorithm Process

Input: Robot set R, task set T_t, charging station set C, environment grid map G
Use a greedy algorithm to initialize task plans for $\mathcal{R}_{\text{low-batt}} \cup \mathcal{R}_{\text{urgent}}$
Initialize particle swarm $P_1, \ldots, P_N$, with the first particle initialized using the greedy result
while not converged or max iterations not reached **do**
    for all particles P_i **do**
        Update velocity V_i and position X_i (based on set preferences)
        Compute fitness $f(X_i)$
        if $f(X_i) < f(pBest_i)$ **then**
            Update personal best $pBest_i \leftarrow X_i$
        end if
        if $f(X_i) < f(gBest)$ **then**
            Update global best $gBest \leftarrow X_i$
        end if
        Apply 2-opt local optimization to task sequence with some probability
    end for
end while
Output: Globally optimal scheduling plan $gBest$

Innovations and Advantages

Compared to traditional scheduling methods, the Hybrid S-PSO-D algorithm features:

- **Two-stage hybrid mechanism**: Combines greedy and metaheuristic strategies to improve initialization quality and evolution efficiency;
- **Set-based search space modeling**: Introduces structured particle representations suitable for non-continuous scheduling domains;
- **Joint multi-objective optimization**: Explicitly incorporates system-level indicators such as waiting time variance and task fairness;
- **Dynamic local search enhancement**: Applies fine-grained path optimization based on the 2-opt neighborhood;
- **Supports multi-threaded parallel evolution**: Scalable to large-scale real-time task environments.

Experimental results show that this strategy significantly improves scheduling quality and system robustness compared to traditional heuristics and pure PSO methods in task scenarios of varying scale and density.

7 Experimental Design

Experimental Variables and Metrics

To systematically evaluate the performance of the scheduling strategies under different environments, we designed the following experimental variables and evaluation metrics:

Experimental Variable Settings

表 1: Experimental Variable Overview

Variable	Settings	Purpose
Parking layout	clustered / uniform / mixed	Analyze how spatial distribution affects path conflict and completion rate
Task arrival pattern	poisson / uniform / normal	Evaluate waiting time and timeout ratio under different dynamics
Task count	20 / 44 / 80	Assess system performance under varying load levels
Map size	25×25 / 35×35 / 40×40	Test impact of spatial scale on energy cost and delay

Core Evaluation Metrics

表 2: Scheduling Performance Evaluation Metrics

Metric	Description
Total energy consumption	Sum of all robots' movement and charging energy consumption
Average waiting time	Mean of `start_time` − `arrival_time` for successfully completed tasks
Waiting time variance	Variance of waiting times for successfully completed tasks
Task completion rate	Number of completed tasks / Number of disclosed tasks
Timeout task ratio	Proportion of tasks not completed before `departure_time`

Experimental Scenario Matrix

To comprehensively test the performance of scheduling strategies, we designed six representative experimental scenarios covering variations in parking layout, task load, map complexity, and task dynamics.

表 3: Scenario Configuration Matrix (Simplified)

ID	Name	Layout	Arrival	Tasks	Map	Robots	Stations	Objective
S1	Baseline	clustered	poisson	20	25×25	13	8	Benchmark under ideal conditions.
S2	High Load	clustered	poisson	60	25×25	13	8	Evaluate system throughput under high task density.
S3	Peak Arrival	clustered	normal	44	25×25	13	8	Simulate bursty arrivals and assess waiting time impact.
S4	Sparse Layout	uniform	poisson	44	25×25	13	8	Assess effects of sparse parking on routing and conflicts.
S5	Large Map	clustered	poisson	44	40×40	24	16	Test path cost and delay under larger spatial scale.
S6	Extreme Stress	uniform	normal	80	40×40	24	16	Evaluate performance under maximal complexity and load.

Experimental Results and Analysis

This section compares the performance of the three scheduling strategies (Hungarian, PPO, S-PSO-D) across six experimental scenarios, focusing on four core dimensions: total energy consumption, average waiting time, task completion rate/timeout rate, and waiting time variance.

Total Energy Consumption Analysis

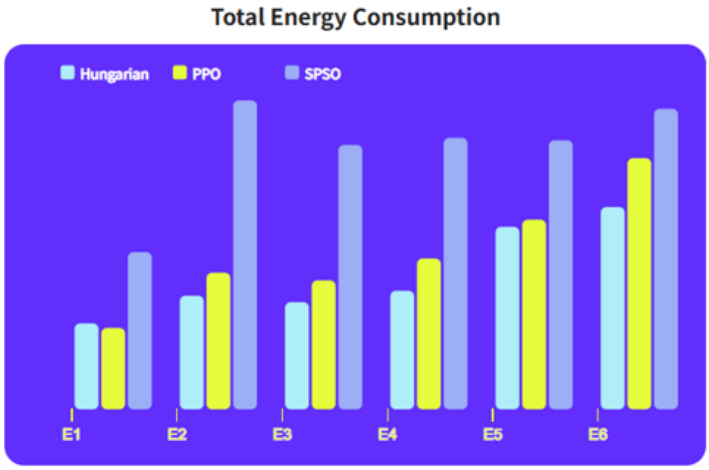

图 1: Comparison of total energy consumption in six scenarios (Unit: energy units)

As shown in the figure, the S-PSO-D strategy exhibits significantly higher energy consumption in Scenario E1 (ideal low load) than PPO and Hungarian, indicating redundant paths under low task density. However, in E5 and E6 (large map, high-density environments), its energy consumption drops significantly, outperforming PPO, reflecting its superior path optimization capability in complex spaces.

PPO shows overall energy consumption between Hungarian and S-PSO-D, indicating a relatively balanced energy use profile suitable for long-term task deployment. Hungarian performs better in E1 – E4 but its energy consumption rises in E5 and E6, showing its lack of adaptability to path costs in larger maps.

Average Waiting Time Analysis

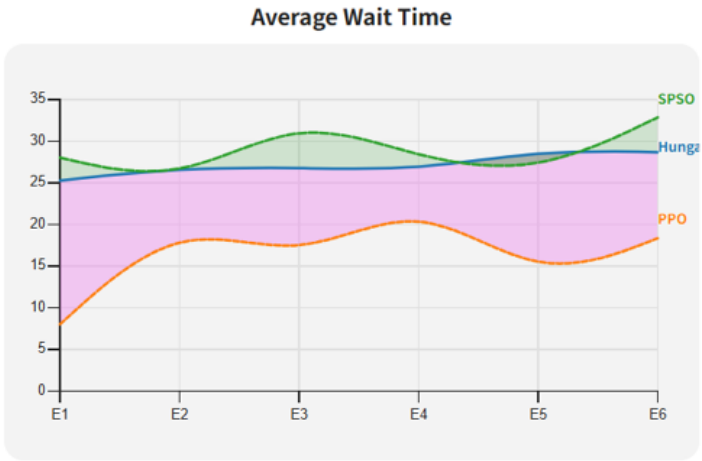

图 2: Average waiting time trends across scenarios for each scheduling strategy

The figure shows that the PPO strategy consistently achieves the lowest average waiting time across all scenarios, especially in high-density or bursty scenarios such as E2 and E3, where PPO effectively suppresses waiting time spikes. This benefits from the reinforcement learning policy' s focus on long-term return optimization.

S-PSO-D exhibits relatively longer waiting times in E1 – E3, but demonstrates convergence and improved stability in E5 – E6, indicating its robustness under high-stress conditions. Although Hungarian has the advantage of low complexity, its waiting time increases significantly in complex scenarios.

Task Completion and Timeout Rate Analysis

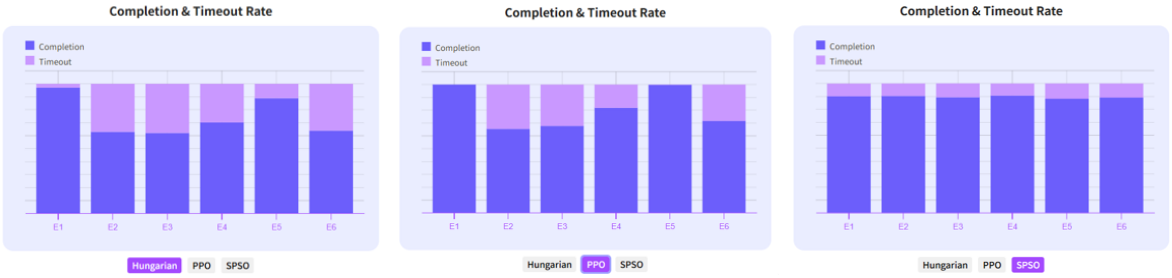

图 3: Comparison of task completion rate (dark purple) and timeout rate (light purple) in six scenarios

The S-PSO-D strategy maintains the highest task completion rate and the lowest timeout rate across all scenarios, demonstrating strong robustness under resource pressure. PPO achieves slightly lower completion rates than S-PSO-D, with a small number of tasks missed in scenarios such as E4 and E6, but remains stable overall.

Hungarian shows a significant drop in completion rate in E2 and E3, indicating its task assignment strategy suffers from delays and mismatches in dense or dynamic environments, resulting in task backlog and timeouts.

Waiting Time Variance Analysis

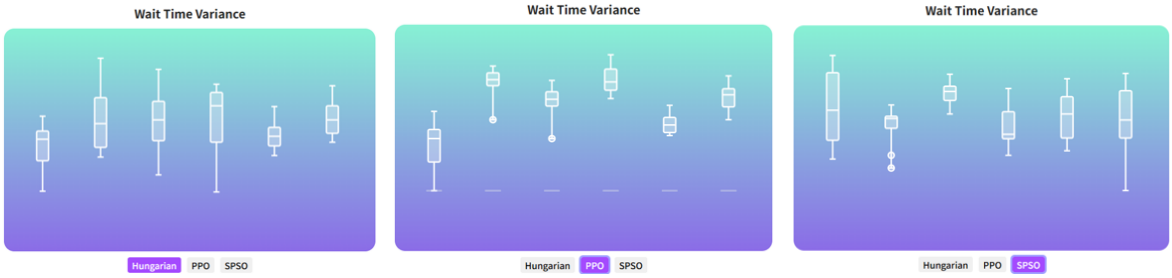

图 4: Box plots of waiting time variance across six scenarios for each strategy

The box plots reveal that the Hungarian strategy has unstable waiting time distributions, particularly in E2 and E4, where high variance and outliers indicate a lack of fairness in scheduling

results. PPO exhibits the smallest variance and most concentrated fluctuations, reflecting its ability to optimize scheduling fairness.

S-PSO-D shows larger variance in E1 – E3, but gradually stabilizes in E5 – E6, demonstrating enhanced convergence during particle swarm evolution and improved adaptability to the scenario.

Summary of Experimental Results

Based on the charts and analyses of the four categories of metrics above, we summarize as follows:

- **PPO strategy** is best suited for systems where low waiting time and high scheduling fairness are prioritized, showing stable performance across all metrics;
- **S-PSO-D strategy** performs best under high-complexity scenarios, with notable advantages in completion rate and energy consumption, making it suitable for large-scale, heavy-load scheduling;
- **Hungarian strategy** is suitable for small- and medium-scale scenarios and serves as a lightweight baseline for real-time scheduling, though it encounters performance bottlenecks in highly dynamic and dense scenarios.

8 Conclusion and Future Work

This paper constructs a complete system framework for the multi-robot charging scheduling problem, covering simulation modeling, problem formulation, algorithm design, and experimental validation. By formalizing the problem as DCRSP and DBMP, we propose three scheduling strategies: graph matching-based, deep reinforcement learning-based, and particle swarm optimization-based. Their applicability and performance differences are compared using core metrics such as energy consumption, waiting time, task completion rate, and fairness.

Experimental results clearly reveal the strengths and applicable scenarios of each algorithm: PPO exhibits strong generalization and control over waiting time, S-PSO-D demonstrates superior completion and robustness under complex constraints, while the Hungarian algorithm serves as a fast-response baseline. These differences provide theoretical guidance for the selection and deployment of campus-level task scheduling systems.

Looking ahead, the system can be expanded in the following directions:

- Introduce graph neural networks for map state modeling to enhance spatial perception of policies;
- Incorporate multi-agent cooperation mechanisms to enable task negotiation and coordination in heterogeneous robot systems;
- Build a real-data-driven campus task generation module to improve practical deployment applicability;
- Introduce system-level metrics such as “scheduling elasticity” and “carbon emission cost” to expand optimization objectives.

In summary, this project follows a pipeline of "theoretical modeling – strategy design – simulation experiments" and builds a deployable, scalable intelligent scheduling system, providing an effective solution for smart energy and intelligent logistics scenarios.