2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

An LLM-driven Framework for Multiple-Vehicle Dispatching and Navigation in Smart City Landscapes

Ruiqing Chen[1], Wenbin Song[1], Weiqin Zu[1], ZiXin Dong[1], Ze Guo[2], Fanglei Sun[3*], Zheng Tian[1], Jun Wang[4]

***Abstract*— In the context of smart cities, autonomous vehicles, such as unmanned delivery vehicles and taxis are gradually gaining acceptance. However, their application scenarios remain significantly fragmented. Typically, an Autonomous Multi-Functional Vehicle (AMFV) is not engaged in other scenarios when idle in a specific one. Currently, a unified system capable of coordinating and using these resources efficiently is lacking. Moreover, there is an absence of an advanced navigation algorithm for facilitating coordinated navigation among Heterogeneous Vehicles (HVs). To address these issues, we propose the LLM-driven Multi-vehicle Dispatching and navigation (LiMeda) framework. It comprises an LLM-driven scheduling module that facilitates efficient allocation considering task scenarios and vehicle information, which addresses the issue of incompatible vehicle resources across various smart city scenarios. And the other is a navigation module, founded on the Heterogeneous Agent Reinforcement Learning (HARL) framework we previously proposed, which can effectively perform cooperative navigation tasks among heterogeneous agents, assisting the cooperative task completion by HVs in a smart city. Experimental results show our method outperforms both traditional scheduling algorithms and Reinforcement Learning navigation algorithms in metric terms. Additionally, it shows remarkable scalability and generalization under varying city scales, vehicle numbers, and task numbers.**

I. Introduction

What does the city of the future look like? Autonomous Multi-Functional Vehicles (AMFV), traversing throughout the cityscape, are an embodiment of our vision for a smart city, made possible through the advancements in Autonomous Driving (AD) [1], Internet of Things (IoT) [2], next-gen communication [3], and sensor technologies. These vehicles transcend their original, singular function to carry the capacity for multi-functional tasks, thereby becoming highly efficient and valuable tools for societal services. Smart cities can serve to three types of users: individuals, for whom vehicles can rapidly transport emergency patients to hospitals; businesses, where temporary passenger and freight vehicles can be dispatched from the respective companies; and public services, such as fire emergencies at markets, where road cleaning vehicles could promptly be engaged for assistance.

Despite significant breakthroughs in the freight and passenger carrying scenarios achieved by AD, accelerated by cutting-edge technological advancements, their deployment remains scattered. For instance, some AD vehicle are used solely for freight transportation, while others are restricted to passenger transport. However, these vehicles can be utilized for other critical tasks such as emergency medical services, firefighting, or cargo transportation when necessary. This raises the following question: how do we achieve efficient dispatching of these heterogeneous vehicles to execute cross-scenario tasks, while also maximizing vehicle resource utilization?

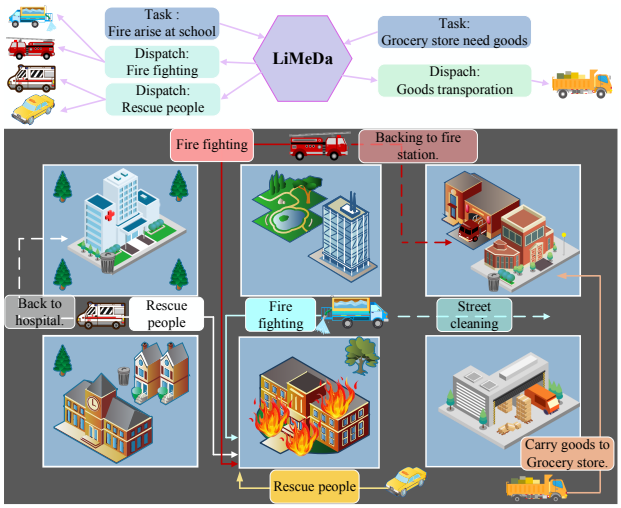


Fig. 1: LiMeDa for Smart City. The task highlighted with semi-transparent coloring denotes that which is presently being executed by the vehicle, alternatively, tasks that are non-transparent are those scheduled to be performed by the vehicle after dispatching by LiMeDa. Likewise, the dashed lines illustrate the route for the executing task before scheduling, whereas the solid lines indicate the path of the vehicle after scheduling.

When smart cities obtained dispatch result for Heterogeneous Vehicles (HVs) with varied control ranges in scenarios, how to facilitate cooperative navigation for HVs remains a problem. Despite Reinforcement Learning (RL) methods being widely applied in navigation research for its advantage in dealing with complex scenarios, Single Agent Reinforcement Learning (SARL) [4] methods have not optimized agents to avoid collisions collaboratively. While Multi Agent Reinforcement Learning (MARL) [5] has the potential to boost the cooperation of agents, it suffers limitations [6] regarding experience sharing and the same action space for each agent. Therefore, a comprehensive multi-agent cooperative navigation algorithm is essential for HVs to properly execute smart city dispatching results.

To solve the smart city problem with both upstream data integration and decision making (scheduling) and down-

[1] ShanghaiTech University, Shanghai, China
[2] Harbin Institute of Technology, Harbin, China
[3] University of Shanghai for Science and Technology, Shanghai, China
[4] University College London, London, United Kingdom
*Corresponding author. sunfanglei1@gmail.com

stream task completion (navigation), we introduce the **L**LM-dr**i**ven **M**ulti-v**e**hicle **D**ispatching and n**a**vigation framework (LiMeda), which mainly include two modules: LLM-driven module for information parsing and output dispatch result, and RL-based navigation module for improving the collision problem in the navigation process of HVs. Our contributions mainly lie in the following areas:

- The LiMeDa framework has been introduced as a reliable and robust solution for enhancing autonomous vehicle transportation within the context of smart cities, depicted in Fig. 1.
- An LLM-driven module has been developed to facilitate efficient allocation, taking into account task scenarios and vehicle information while handling the issue of incompatible vehicle resources across diverse smart city scenarios.
- Based on our previously proposed HARL framework [7], the Heterogeneous Agent Reinforcement Learning Navigation (HARLN) module has been devised. This module is capable of effectively executing multi-agent cooperative navigation tasks among HVs, contributing to the collaborative completion of tasks by HVs within a smart city.
- In the context of smart city scenarios, our framework outperforms the State-Of-The-Art (SOTA) both dispatching and navigation algorithm, maintaining performance stability as the city scales and the number of vehicles and tasks vary.

II. Related Work

A. Smart city navigation and dispatching

Numerous algorithms tackle navigation challenges within smart cities. The Max-Flow Min-Cost model [8] effectively measures the distance between patients and ambulances, allowing efficient dispatch of relevant ambulances for emergency services. However, its applicability is limited by its reliance on static city maps. An alternative strategy involves propagating information to other vehicles via Road Side Units (RSU) [9], optimizing their routes and, consequently, reducing their travel time. Despite its effectiveness, this approach is constrained by the scalability of hardware devices RSU and might not adapt well to rapidly expanding cities. Traffic signal lights, used to decrease ambulance response time [10], face similar scalability problems.

The Model Predictive Control (MPC) algorithm [11] addresses long wait times for customers in the Autonomous Mobility-on-Demand (AMoD). However, as the scale of vehicle and customer demand increases, solution time grows significantly. To dynamically accommodate new requests in AMoD, an online re-optimization strategy is proposed [12]. Although with scalability, it is still a challenging task to handle effectively when involving numerous vehicles.

B. LLM

Many fields have witnessed innovative progress through the support of LLM. The generative agents devised in AI Town [13], which incorporates a self-reflective mechanism allowing realistic simulation of human behavior. Regarding AI in the game, LLM has been utilized to develop an information processing structure in Minecraft [14], [15], its performance and computational cost significantly outperform RL algorithms. Similarly, LLM can also participate in the RL training process, replacing the human in the loop, allowing RL to generate behavior that is more consistent with common sense [16], and enhancing the performance of human-AI collaboration [17].

In the robotics field, VoxPoser [18] leverages the LLM code writing ability and combines it with the visual Language Model (VLM) to map language instructions to a 3D value map, giving it the ability to synthesize trajectories in a zero-shot manner. Similarly, [19] and [20] both use LLM to parse and complete human instructions, significantly improving the human-machine interaction experience. To the best of our knowledge, there seems to be a lack of research on LLM-based applications in the multi-vehicle navigation within smart cities.

C. RL for navigation

RL has been comprehensively researched within the navigation domain. For single-robot navigation, asynchronous Deep Reinforcement Learning (DRL) algorithms have been designed to address the mapless navigation issue [4], enabling mobile robots to arrive at the target location while avoiding collisions. Furthermore, DRL-based robots designed for socially aware motion planning scenarios crowded with pedestrians [21], achieving a human walking speed. In addition, incorporating the Long Short-Term Memory (LSTM) within the DRL framework equips the algorithm with the ability to adapt to fluctuating numbers of robots [22].

When considering multi-robot navigation, the Hierarchical Navigation Reinforcement Network (HNRN) [23] divides the navigation into two targets: a goal-driven system and a collision-avoidance DRL system, ensuring the accomplishment of the goal with avoiding collision. For map-based multi-robot movements, the Proximal Policy Optimization (PPO) algorithm employs the positions of other robots coupled with surrounding obstacle sensing data as inputs [24]. However, given its direct application of an SARL algorithm in multi-robot navigation scenarios, robots are not trained to effectively cooperate to prevent collisions. The Multi-agent Graph-Enhanced Commander-EXecutor (MAGE-X), merges MARL with Graph Neural N etworks (GNN) [25]. This combination helps in constructing a crucial partner-exclusive subgraph for each agent, which markedly enhances the scalability of multi-robot cooperation. Moreover, a communication protocol between agents is researched that leverages shared information to facilitate collaborative task completion in navigation scenarios [26].

III. Method

Our proposed LiMeDa framework consists mainly of two modules: an LLM-based dispatcher, responsible for the integration of upstream information, process it and dispatching

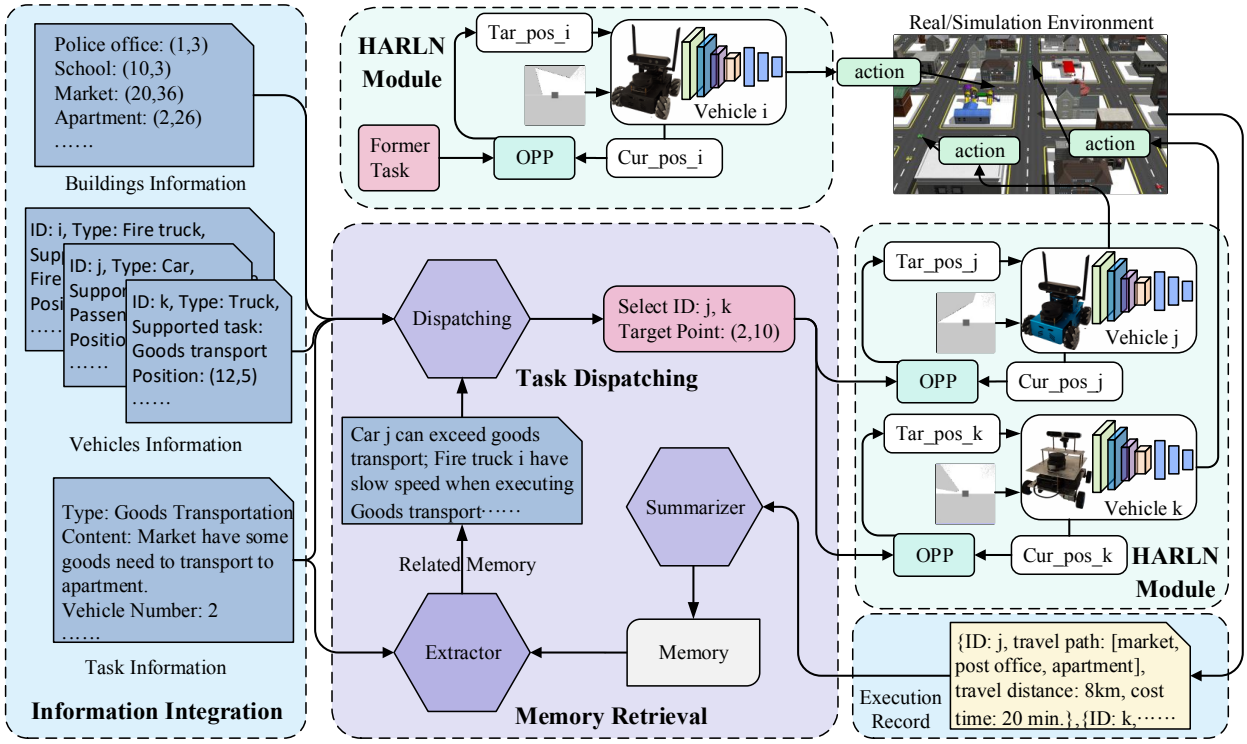


Fig. 2: The Structure of LiMeDa. The *Information integration* consolidates the task, building, and vehicle data. The *task dispatch* is responsible for parsing the integrated information, conducting decision-making, and producing vehicle dispatch results. The *Memory retrieval* extract related memory to aid in decision making. The *HARLN module* starts the corresponding vehicles based on the dispatch results. To determine the current target position (Tar_pos_x), it executes an Optimal Path Planning (OPP) method using the destination and the vehicle's current position (Cur_pos_x). Target position along with the laser information, input into the vehicle model to generate control information and execute it. After completing a task, the system collects execution records, summarizes, and stores them in memory.

task, and an RL-based cooperative multi-agent navigation planner, which executes downstream navigation tasks based on the dispatching results. The executing pipeline of the LiMeDa framework is exhibited in Fig. 2.

A. LLM-based Dispatcher

With the context of smart city scenarios, the dispatcher plays a pivotal role that mainly include three sub-modules: information integration, memory retrieval, and task dispatching. The information process of these modules is shown in Fig. 3. The details of the submodules are presented as follows.

1) Information Integration: In smart cities, multi-functional vehicles and tasks are the primary information entities that influence dispatching. The process of information integration is distributed in the process of dispatching. First, at the start of dispatching, task-related information will be sent to the memory retrieval module, and task-related memory will be extracted for dispatching. Next, a combination of building coordinate information, vehicle information, and task information is utilized to generate scheduling results. Finally, upon task completion, the execution record is forwarded to the summarizer for experience summary.

2) Memory Retrieval: The memory management mechanism plays a vital role in dispatching, as it summarizes the experiences of past execution records and applies this information to future task scheduling. For instance, a execution record might include the electricity/fuel consumption, duration, and route of a certain vehicle performing a certain type of task. When similar types of dispatch tasks occur, the memory data can assist in determining which type of vehicle is more suitable for completing the task and deducing the preconditions required for task completion, such as electricity/fuel consumption.

We have designed a memory manager divided into three sub-modules: *summarizer*, *simplifier* and *extractor*. The *summarizer*, based on the execution records, deduces experiences that can aid other scheduling tasks. For instance, it assesses the rationality of former scheduling result by considering the vehicle's driving paths and speed, provides an evaluation of the execution results, and infers the conditions required to carry out such tasks based on the consumption of the vehicle's electricity/fuel. Considering the restricted input and memory storage capacity of the LLM model, a *simplifier* is devised to manage the space of memory. This system activates when stored experiences exceed the memory limit, triggering the elimination of duplicate experiences and the merging of compatible ones. In the absence of duplicates or mergeable experiences, the *simplifier* selectively removes the experiences that contribute the least information. The *extractor* facilitates the preprocessing of information before task dispatching. It identifies and extracts relevant experiences based on the details of the current task, which are subsequently used for task scheduling.

3) Task Dispatching: Task scheduling is divided into two stages. In the first stage, i.e. information parsing, deduces the number and coordinates of destinations, priority of tasks,

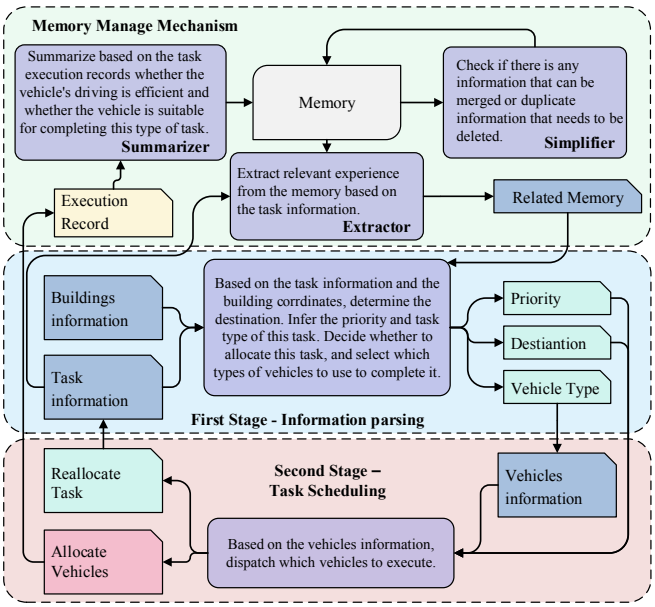


Fig. 3: The information process and structure of LLM-based Dispatcher

and identifies the types of vehicles to be deployed. This is accomplished through a comprehensive analysis of task details, building information, priority settings, and vehicle information. The *extractor* is utilized to extract experiences related to the current task and aid in decision making. In the second stage, i.e. task scheduling, according to the selected vehicle types in the first stage to extract detailed vehicle status information. By combining the parsed destination, priority, and the related experience obtained by *extractor*, the final scheduling result is produced.

In the process of task execution, when higher priority assignments emerge (such as rescuing people or firefighting), the dispatcher may redirect a nearby vehicle engaged in lower priority tasks (e.g., road cleaning or freight transport) to execute these urgent tasks. The previous assignment of the vehicle will be subsequently re-allocated by the dispatcher. This rescheduling process similar to the original scheduling approach, but also requires additional consideration of the time cost about task modification. To minimize the impact of these modifications, vehicles that are in the process of executing assignments are excluded from the target of reallocation.

B. HARLN module

Assuming that the long-distance optimal route from the starting point to the destination in a smart city can be obtained using a conventional Optimal Path Planning (OPP) algorithm such as A* [27] or RRT [28]. Based on our previously proposed HARL framework [7], our navigation algorithm concentrates on analyzing collision avoidance and path planning for local HVs scenarios.

1) Preliminaries: We consider a multi-agent decentralized partially observable Markov Decision Process (Dec-POMDP) [29], represented by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, R, Z, \mathcal{O}, p, \gamma \rangle$. In this formulation, $\mathcal{N} = \{1, \ldots, n\}$ corresponds to the agent set; $\mathcal{S}$ signifies the finite state space, and $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}^i$ illustrates the joint action space, a composition of the actions of each agent. The transition probability function is given by $p : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$, where $\Delta_{\mathcal{S}}$ represents the distribution of states. The reward function is denoted as $R(s, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and $\gamma \in [0, 1)$ is the discount factor.

In Dec-POMDP, each agent i obtains an observation, denoted $o_i \in \mathcal{O}$, given by the observation mapping: $Z(s, \boldsymbol{a}) \to \mathcal{O}$. We consider each agent's decision as a stationary policy $\pi_i : \mathcal{O} \to \Delta_{\mathcal{A}}$, with $\Delta_{\mathcal{A}}$ representing the action distribution. Following a protocol, agents interact with their environment. At a time step $t \in \mathbb{N}$, the agents exist at a state denoted $s_t \in \mathcal{S}$, with each agent i perceiving o_t^i. Then, agent i takes an action $a_t^i \in \mathcal{A}^i$, decided by its policy $\pi_i(\cdot \mid o_t^i)$. Collectively with the actions of other counterparts, these form a joint action represented by $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^n) \in \mathcal{A}$, defined by the joint policy $\boldsymbol{\pi}(\cdot \mid s_t) = \prod_{i=1}^{n} \pi_i(\cdot \mid o_t^i)$. In conclusion, agents achieve a team reward, $r_t = R(s_t, \boldsymbol{a}_t) \in \mathbb{R}$, and transition to the subsequent state, $s_{t+1} \sim p(\cdot \mid s_t, \boldsymbol{a}_t)$. we define the state action value function and the state value function as $Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}\right]$ and $V_{\boldsymbol{\pi}}(s) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ respectively. Therefore, the advantage function is expressed as $A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - V_{\boldsymbol{\pi}}(s)$.

2) Training Stage: Smart Cities feature a diversity of vehicle types, each with its unique action space, such as speed and steering range. While traditional MARL algorithms excel in managing homogeneous agents, they are failed when applied to heterogeneous agents. Consequently, to address this gap, our study leverages the advantageous capabilities of the HARL [7] framework which excels with heterogeneous agents. To mitigate the computational resources requisite for training, whilst concurrently accommodating both discrete and continuous action spaces, our multi-vehicle cooperative navigation training is founded on the HARL updating pipeline.

We parameterize the policy of each agent π_i by θ_i, and the joint policy $\boldsymbol{\pi}_\theta$ by $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$, the global critic as V_ϕ, ϕ_k denotes the parameter of critic V at k-th iteration. The training target of global critic is:

$$\phi_{k+1} = \arg\min_{\phi_k} \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=0}^{T} \left(V_{\phi_k}(s_t) - \hat{R}_t \right)^2, \quad (1)$$

where B denote batch size, T is episode length, k represents iteration number.

To achieve monotonic improvement for heterogeneous agents, we adopt a sequential updating approach for each agent same as HARL, wherein the training goal for each agent is defined as follows:

$$G_{\theta_i^k} = \min \left(\rho_{\theta_i^k} A_{\pi_{\theta_i^k}} (\boldsymbol{s}_t, \boldsymbol{a}_t), \text{clip} \left(\rho_{\theta_i^k} A_{\pi_{\theta_i^k}} (\boldsymbol{s}_t, \boldsymbol{a}_t) \right) \right)$$
$$\theta_i^{k+1} = \arg\max_{\theta_i^k} \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=0}^{T} G_{\theta_i^k} \quad (2)$$

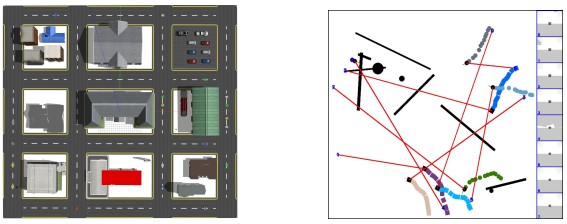
where the operation $\text{clip}(\cdot) = \text{clip}(\cdot, 1 - \epsilon, 1 + \epsilon)$, $\rho_{\theta_i^k} =$

$\prod_{l=1}^{i} \frac{\bar{\pi}_{\theta_l^k}(a_t \| o_t)}{\pi_{\theta_l^k}(a_t \| o_t)}$, $\bar{\pi}$ represent the candidate policy for updating. θ_i^k represents the parameter of agent i at iteration k.

IV. Experiment

For a comprehensive evaluation of the performance of our proposed framework, we have established a simulated environment for conducting experiments on various performance metrics. In addition, a real-world scenario has been established to verify the deployment of our framework.

A. Experiment in Simulation



(a) 4x4 streets SCVSS, indicates that there are 4 horizontal streets and 4 vertical streets within the city.

(b) Training Scenarios with heterogeneous agents.

Fig. 4: The simulation environment and training scenarios.

1) Environment: We have constructed a Smart City Vehicle Simulation System (SCVSS) within Gazebo. To assist in OPP and navigation algorithms, we have quantized city buildings and preserved roads in an adjacency matrix. The SCVSS includes a random task generator that publishes a randomly type task with random destinations and vehicle requirements after a set duration. The tasks involve emergency incidents (e.g. firefighting, patient transport), commercial tasks (e.g. passenger and goods transport), and social services (e.g. road cleaning, rubbish transport). We preset the priority of different task types following the societal rule. The vehicles within SCVSS are furnished with RPLIDAR A1 radar for avoiding collision and Mecanum wheels for motion, similar to the control and perception interface of real vehicles, a SCVSS with 4x4 scale is depicted in Fig. 4a.

Our HARLN algorithm was trained in a simulator built by OpenCV [30], with an Nvidia RTX 2080Ti GPU and a Intel 28-core CPU. The training environment step is $5e5$ and the learning rate is set as $1e-4$. Once trained, the model can be deployed for navigation within the SCVSS environment. The simulator of training agents show in Fig. 4b.

2) Evaluation Metrics: To comprehensively evaluate the reasonableness of the scheduling algorithm, we have defined the average task response time T_{atr} and the weighted task response time T_{wtr} as follows:

$$T_{atr} = \frac{1}{n}\sum_{i=1}^{n} t_i^r,\ T_{wtr} = \frac{1}{n}\sum_{i=1}^{n} p_i t_i^r, \tag{3}$$

where t_i^r denotes the response time of task i from publish until being dispatched successfully, $p_i \in (0,1]$ represents the normalized priority of task i, n is the total number of completion tasks.

The performance difference with the navigational algorithm is reflected in the completion time of the task (the task is completed in less time with fewer collisions), and hence, we have defined the average task completion time T_{atc} and the weighted task completion time T_{wtc} as:

$$T_{atc} = \frac{1}{n}\sum_{i=1}^{n} t_i^c,\ T_{wtc} = \frac{1}{n}\sum_{i=1}^{n} p_i t_i^c, \tag{4}$$

where p_i and n are consistent with the definitions mentioned above, t_i^c is the completion time span from the initial execution to completion of task i.

3) Dispatching Performance: The performance of our dispatching algorithm is rigorously assessed against four prevalent heuristic algorithms, namely: Distance First (DF), Idle First (IF), Priority First (PF) and Mixed First (MF). The DF algorithm assigns tasks to the vehicle closest to the current task location. The IF algorithm schedules tasks to any idle vehicles. The PF algorithm schedules tasks using an idle vehicle or a vehicle engaged in executing a task with lower priority than the unallocated task. Lastly, the MF algorithm conducts task allocation, proceeding consecutively based on closeness through vehicles. If a vehicle is idle, it immediately assigns the task. Otherwise, if the vehicle is already executing a task that has a lower priority than the unallocated task, the algorithm reallocates its executing task and allocates the unallocated task to the vehicle. If this is not the case, the algorithm continues to visit the next vehicle.

City Scale	Metric(/s)	LiMeDa	DF	IF	PF	MF
3x3	T_{atr}	8.3	**4.5**	13.2	9.9	7.9
	T_{wtr}	5.2	**4.6**	12.8	8.1	7.6
	T_{atc}	**74.8**	120.3	96.5	85.5	75.2
	T_{wtc}	**61.6**	148.3	108.2	76.1	73.0
4x4	T_{atr}	18.1	**5.1**	30.3	23.3	21.0
	T_{wtr}	14.7	**5.3**	35.6	21.1	18.7
	T_{atc}	**115.2**	246.0	153.9	170.3	121.6
	T_{wtc}	**97.9**	232.9	168.7	148.8	114.0
5x5	T_{atr}	32.7	**5.5**	55.7	49.9	43.8
	T_{wtr}	25.1	**5.2**	48.3	40.6	39.1
	T_{atc}	155.8	319.4	215.8	197.5	**152.4**
	T_{wtc}	**138.0**	328.5	203.6	173.1	144.5

TABLE I: Dispatching algorithm in the SCVSS with different city scale, configure task number as 30 and vehicle number as 20.

According to the results in Table I, DF has the fastest response time, but frequent scheduling would lead to continuous task switching, significantly increasing the time it takes to complete tasks. Both IF and PF lack consideration of vehicle distance, and MF lacks consideration for vehicles performing tasks across scenarios. Our algorithm is superior to other algorithms in task completion time and has a tolerable task response time. Moreover, with the growth of city scale, our scheduling algorithm can still maintain stable task completion time.

4) Navigation Performance: To maintain the consistency of the RL update algorithm, this study utilizes PPO [31] from SARL and Multi Agent Proximal Policy Optimization (MAPPO) [32] from MARL as performance evaluation benchmarks. The HVs in SCVSS have different linear and angular velocities. Each PPO and MAPPO model is trained to accommodate one type of HVs, therefore, multiple models

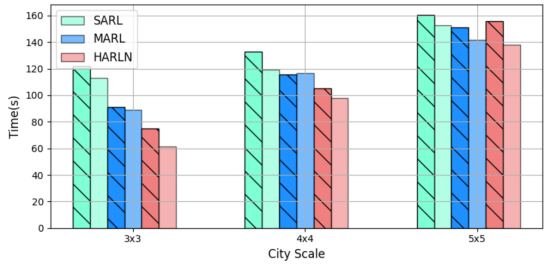
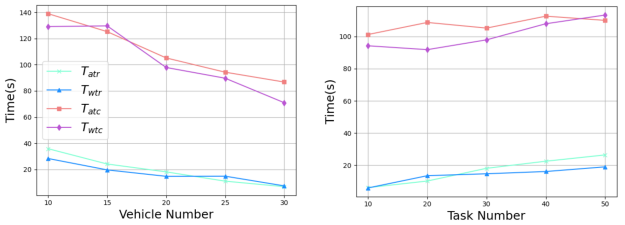


Fig. 5: RL navigation algirithms in the SCVSS for different city scale with 30 task number and 20 vehicle number. Cylinders of the same color, those with shadows (diagonal lines) represent T_{atc}, those without shadows represent T_{wtc}.

need to be trained and employed in different type vehicles. However, HARLN incorporates HVs within a single scenario, allowing one-time training for deployment on all HVs. To maintain control over the variables, all dispatching algorithms uniformly utilized the dispatching module sourced from LiMeDa.

Fig. 5 illustrates that in scenarios of small city scale, vehicular cooperation is crucial for navigation and collision avoidance. SARL fails to optimize multi-vehicle cooperative navigation, and MARL lacks the knowledge about the coordinated navigation of HVs, therefore, the task completion times of them are longer. Our algorithm, trained in the context of HVs, effectively handles the cooperative navigation of HVs, and thereby significantly outperforms other algorithms in terms of task completion time.

5) Robustness and Adaptability: To analyze the robustness and adaptability of dispatching algorithms, we conducted a study from the perspectives of vehicle and task number, drawing on the method of control variables. The baseline city configuration is set as (4x4, with 30 tasks and 20 vehicles). The vehicle number ranges from 10 to 30, and the task number varies from 10 to 50. The experimental results are displayed in Fig. 6a and 6b.



(a) Performance with different vehicle number.

(b) Performance with different task number.

Fig. 6: The performance of LiMeDa framework in varying task and vehicle number.

The experimental findings indicate that, because our navigation module can cooperate to avoid collisions, as the number of vehicles increases, closer vehicles can be dispatched to complete tasks. The values of T_{atc} and T_{atr} gradually decrease, tasks with higher priority receive more timely responses, and the values of T_{atc} and T_{wtc} correspondingly decrease. As the number of tasks increases, the values of T_{atr} and T_{wtr} increase, but the values of T_{atc} and T_{wtc} are relatively stable and not greatly affected. In light of these findings, our framework has exhibited strong robustness and adaptability in handling the variation in the number of vehicles and tasks.

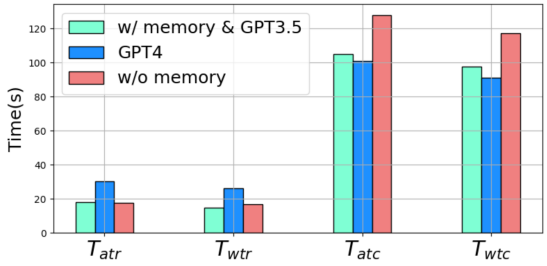


Fig. 7: Ablation of memory module and evaluation for LLM ability.

6) LLM Ability and Ablation: To investigate the impact of the LLM ability and the memory module on scheduling results, we executed experiments utilizing both GPT3.5 and 4.0 models, with and without the memory module. The experiment was conducted in a city area configured as (4x4, 30 tasks, 20 vehicles). The results are displayed in Fig. 7. The experimental results show that without the memory module, the scheduling algorithm cannot utilize the experience gained before, and the task completion time increases. Although using the stronger LLM model (GPT4) can obtain better scheduling results (less T_{wtc}), due to the longer inference time, T_{atr} and T_{atr} will also increase, and it may fail when many emergent tasks arise simultaneously.

B. Deployment in Real World

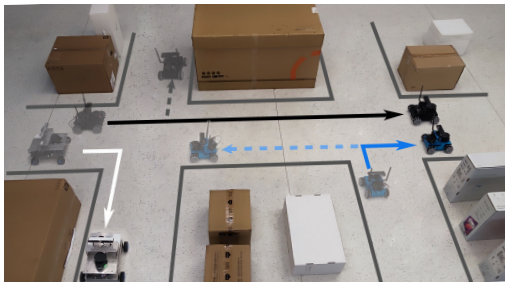


Fig. 8: LiMeDa in real Micro-city. The dotted line represents the destination and route of the vehicle's previous task, while the solid line represents the route taken to perform the task after task scheduling has occurred. In this scenario, the previous task of the black and blue vehicle will be reallocated.

To assess the deployability of our framework, we constructed a micro-city in a real world, scnarios show in Fig. 8. We assign the navigation tasks to multiple heterogeneous mobile robots, as illustrated in the demo video. The result indicates the scalability and application of our framework in real-world smart city scenarios.

V. Conclusion

The LiMeDa framework, proposed to address vehicle dispatching challenges across various situations in smart cities and to cooperativelly avoid HVs collisions, incorporates both LLM-driven dispatching and RL-based HARLN module. The former integrate data from all vehicles contributing to optimized dispatch, while the latter effectively prevent collisions between different vehicle types. The experimental results indicate that the LiMeDa framework outperforms the baseline, demonstrating its robustness and strong adaptability. In the future, systems in smart cities will incorporate end-effectors like robotic arms and drones, facilitating collaboration to execute intricate tasks.

REFERENCES

[1] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi, and W. Cho, "A review on autonomous vehicles: Progress, methods and challenges," *Electronics*, vol. 11, no. 14, p. 2162, 2022.

[2] S. Madakam, V. Lake, V. Lake, V. Lake, *et al.*, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[3] M. H. Alsharif, A. H. Kelechi, M. A. Albreem, S. A. Chaudhry, M. S. Zia, and S. Kim, "Sixth generation (6g) wireless networks: Vision, research activities, challenges and potential solutions," *Symmetry*, vol. 12, no. 4, p. 676, 2020.

[4] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[5] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[6] Z. Zhou, G. Liu, and Y. Tang, "Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges," *arXiv preprint arXiv:2305.10091*, 2023.

[7] J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang, "Trust region policy optimisation in multi-agent reinforcement learning," *arXiv preprint arXiv:2109.11251*, 2021.

[8] J. Lee, J. Park, and E. Park, "Ambulance dispatch scheme based on maximum flow minimum cost models," in *2023 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2023, pp. 1–2.

[9] Y. Moroi and K. Takami, "A method of securing priority-use routes for emergency vehicles using inter-vehicle and vehicle-road communication," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, 2015, pp. 1–5.

[10] Y. Tanaka, H. Yamada, S. Tamasaku, and H. Inaba, "The fast emergency vehicle pre-emption system improved the outcomes of out-of-hospital cardiac arrest," *The American journal of emergency medicine*, vol. 31, no. 10, pp. 1466–1471, 2013.

[11] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of autonomous mobility-on-demand systems," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1382–1389.

[12] C. Li, D. Parker, and Q. Hao, "Optimal online dispatch for high-capacity shared autonomous mobility-on-demand systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 779–785.

[13] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," *arXiv preprint arXiv:2304.03442*, 2023.

[14] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[15] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, *et al.*, "Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory," *arXiv preprint arXiv:2305.17144*, 2023.

[16] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas, "Guiding pretraining in reinforcement learning with large language models," *arXiv preprint arXiv:2302.06692*, 2023.

[17] H. Hu and D. Sadigh, "Language instructed reinforcement learning for human-ai coordination," *arXiv preprint arXiv:2304.07297*, 2023.

[18] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[19] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[20] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," *arXiv preprint arXiv:2307.04738*, 2023.

[21] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[22] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[23] W. Ding, S. Li, H. Qian, and Y. Chen, "Hierarchical reinforcement learning framework towards multi-agent navigation," in *2018 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2018, pp. 237–242.

[24] S. Yao, G. Chen, L. Pan, J. Ma, J. Ji, and X. Chen, "Multi-robot collision avoidance with map-based deep reinforcement learning," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 532–539.

[25] X. Yang, S. Huang, Y. Sun, Y. Yang, C. Yu, W.-W. Tu, H. Yang, and Y. Wang, "Learning graph-enhanced commander-executor for multi-agent navigation," *arXiv preprint arXiv:2302.04094*, 2023.

[26] M. K. Abdelaziz, M. S. Elbamby, S. Samarakoon, and M. Bennis, "Cooperative multi-agent learning for navigation via structured state abstraction," *arXiv preprint arXiv:2306.11336*, 2023.

[27] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[28] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[29] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[30] Q. Qiu, S. Yao, J. Wang, J. Ma, G. Chen, and J. Ji, "Learning to socially navigate in pedestrian-rich environments with interaction capacity," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 279–285.

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

[32] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.