Hierarchical Clustering-Based State Grouping Reinforcement Learning for Switching Decision of Autonomous Vehicles

Ouyang, Wenjie; Jiao, Yiwen; Liu, Yang; Li, Yang; Hu, Manjiang; Qin, Hongmao

DOI
10.1109/ICUS58632.2023.10318413

Publication date
2023

Document Version
Final published version

Published in
Proceedings of 2023 IEEE International Conference on Unmanned Systems, ICUS 2023

Hierarchical Clustering-based State Grouping Reinforcement Learning for Switching Decision of Autonomous Vehicles

Wenjie Ouyang
College of Mechanical and Vehicle Engineering
Hunan University
Changsha, China
oywjhnu@hnu.edu.cn

Yiwen Jiao
College of Computer Science and Electronic Engineering
Hunan University
Changsha, China
jiaoyiwen001@hnu.edu.cn

Yang Liu
College of Electrical, Mechanical and Computer Science
Delft University of Technology
Delft, The Netherlands
y.liu-29@outlook.com

Yang Li*
State Key Laboratory of Advanced Design and Manufacturing Technology for Vehicle, College of Mechanical and Vehicle Engineering
Hunan University
Changsha, China
lyxc56@gmail.com

Manjiang Hu
State Key Laboratory of Advanced Design and Manufacturing Technology for Vehicle, College of Mechanical and Vehicle Engineering
Hunan University
Changsha, China
manjiang_h@hnu.edu.cn

Hongmao Qin[1,2]
1. State Key Laboratory of Advanced Design and Manufacturing Technology for Vehicle, College of Mechanical and Vehicle Engineering
Hunan University
Changsha, China
2. Wuxi Intelligent Control Research Institute of Hunan University
Wuxi, China
qinhongmao@hnu.edu.cn

***Abstract*—Reinforcement learning (RL) has gained wide attention, but its implementation in autonomous vehicles is still limited by insufficient sample efficiency and heavy training costs. The training efficiency of RL agents is influenced by the dimension of the state space, which can be partitioned to reduce the complexity of sampling and computation. This study proposes a hierarchical clustering-based state grouping reinforcement learning (HCSG-RL) method for the switching decision of autonomous vehicles. First, we partition the base state space into groups and generate a hierarchical tree of state space groups. Then, we train multiple sub-agents for each node in the hierarchical tree. Finally, we add these trained-well sub-model into master policy. This method allows us to fully explore all state spaces and improve the training efficiency of individual agents, which handles the "long-tail" issue and the curse of dimensionality issue. We conduct experiments in a simulation environment and results show that the proposed method has 16-72% reward improvement compared to the tree model in different road length.**

***Keywords*—reinforcement learning, hierarchical clustering, state grouping, autonomous switching**

I. Introduction

Vehicle automation has the potential to enhance transportation efficiency and improve traffic safety. Studies suggest that implementing automated vehicle technology and assistance systems could lead to a reduction in road crashes and fatalities of 25-90% [1]. However, when vehicle control is shared by human drivers and automation, the driver's task changes from an active to a passive, supervisory role, which may reduce attention and situational awareness, leading to boredom, mode confusion, and performance degradation [2]. Additionally, automated entities do not always outperform human drivers under all circumstances, especially under specific road and environmental conditions such as bright light, heavy rain, and poor quality of road and traffic signs [3]. Thus, the issue of where control authority is shared and how it is switched between the vehicle and the human driver is a special focus of this study.

Traditional rule-based solutions are widely employed but incapable to account for real-time driver conditions and fast-moving obstacles. However, it is incapable to account for real-time driver conditions like fatigue or distraction and requires a more intelligent and adaptive strategy. Learning-based techniques focus on two capabilities: the cooperation between the human and the system, and the system's ability to self-learn from these collaborative actions [4]. However, two limitations are present in learning-based techniques: first, how the user can validate correct learning based on their input, and second, how the system can integrate new competencies while maintaining validation and ensuring that it behaves correctly under all driving conditions. [5] proposed an automated driving transition framework, defining static states for primary driving task control, driver monitoring, and driving state transitions. However, their model overlooks the learning process between the vehicle and driver. [6] developed a learning-based model for automated driving, enhancing autonomy but lacks integration of new capabilities in varied driving conditions.

Reinforcement learning (RL) has been applied to various autonomous driving tasks, including path planning, motion planning, and developing high-level driving policies [7]. Despite remarkable advancements, DRL also suffers from the issue of "long-tail" data imbalance distribution [8], [9]. Various techniques have been proposed to tackle long-tailed instance segmentation, such as bi-level sampling strategies [10], dynamic curriculum learning [11], and feature augmentation and sampling adaption [12]. Data-centric methods such as re-weighting and re-sampling are used to address the "long-tail" data imbalance issue. Re-weighting adjusts the weights of samples based on frequency [13], loss values [14] and prediction probabilities [15] to alleviate performance bias. Re-sampling, on the other hand, alters the data distribution to solve the data imbalance problem.

*Corresponding Author

This work was supported by National Key R&D Program of China (2021YFF0501102), National Natural Science Foundation of China (52272415), Hunan Provincial NSF of China (2023JJ40155), Science and Technology Innovation Program of Hunan Province (2022RC1033), and Science and Technology Development Program of Wuxi (K20221059).

Wenjie Ouyang and Yiwen Jiao should be considered as co-first authors, as they contributed equally to this work.

979-8-3503-1630-8/23/$31.00 ©2023 IEEE

2023 IEEE International Conference on Unmanned Systems (ICUS) | 979-8-3503-1630-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICUS58632.2023.10318413

The curse of dimensionality is also a significant issue in RL. As the size of the state space increases, the budget required grows exponentially. To tackle the curse of dimensionality, deep methods, such as Deep Q-networks (DQNs) and Double-DQNs (DDQNs), use deep neural networks (DNNs) to approximate parameterized value functions [16], [17]. State abstraction is an approach that condenses the dimension of the state space and groups states that exhibit similarities together and most methods use k-means clustering for state grouping [18]. However, these state abstraction methods may lose some important information and result in suboptimal solutions.

In this paper, we propose a Hierarchical Clustering-based State Grouping Reinforcement Learning (HCSG-RL) method. This method focuses on improving the performance of RL agents on the switching decision from two aspects: decomposing the state space for RL agents to learn effectively in terms of computation time and memory, and balancing the training to maintain/improve general performance of the model. We demonstrate the performance of the method in the Mediator environment, which integrates the best capabilities of driver and vehicle automation systems to maximize driving safety and comfort. Our method shows improved performance compared to traditional methods in this complex environment. Our main contributions are as follows:

1) A hierarchical clustering-based state grouping method is proposed to determine the switching among different autonomous levels when the driver is distracted.

2) The proposed method partitions the state space into small subsets as the starting point for training the sub-agents and groups the sub-agents by a master policy, to handle the issues of the "long-tail" data imbalance and the curse of dimensionality.

3) The results of experiments show that the proposed method achieves the best performance, alleviate driver's unfitness and prevents emergency stop.

II. Framework

In this paper, we proposed a HCSG-RL method for the switching decision of autonomous vehicles. We enhance the exploration mechanism of the agent by partitioning the expansive state space, and constructed a hierarchical state tree instead of training the agent on all state spaces. Each leaf node of the state tree represents a sub-state space with distinct state characteristics.

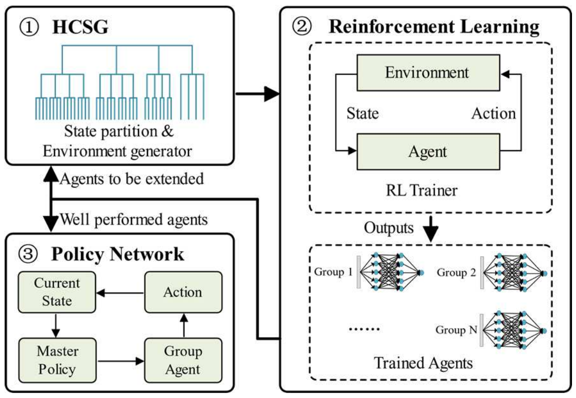


Fig. 1. Framework of the HCSG-RL algorithm.

Fig. 1 shows the framework of the HCSG-RL method, which contains three parts: HCSG, reinforcement learning and policy network. State partition is carried out in the HCSG section, where sub-state spaces are placed into the RL section for RL agents training. Well-performing agents are integrated into the policy network, while those with poor performance are sent back to the HCSG section for further state partition. The final model consists of a master policy and several well-performing sub-agents.

III. Methodology

A. Mediator Environment

In this study, we focus on the situation where the mediator needs to make decisions of switching control among different autonomous levels once the driver distraction is detected, as shown in Fig. 2. The controller receives observations from the driver and the car, determining actions and sending signals to the driver accordingly. In terms of interaction with the driver, the mediator controller takes the driver's distraction level as one of the observations and corrects the driver's distraction by sending signals when necessary. Additionally, the mediator controller considers the driver's competence and willingness when making decisions to ensure optimal support. For example, if the distraction correction is ineffective for an extended period of time, the mediator controller will take an "emergency stop" action rather than continuing with a "distraction correction" action. Regarding car interaction, the mediator controller takes into account the car's level of autonomy and capabilities as the rest of the observations. The mediator controller assesses whether switching to a different level of autonomous driving is appropriate, and if so, sends signals to the car to facilitate the transition. The responsibility of the mediator controller is to determine the most suitable autonomous driving level given the current observations within the car capacities. The mediator controller will sample all observations and determine the next course of action based on both the driver's and car's states. Once the next action has been determined, the corresponding signal will be sent to either the driver or the car accordingly.

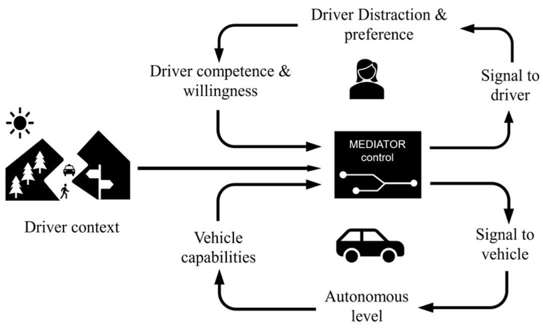


Fig. 2. Interaction process illustration of the mediator environment.

B. MDP Models

In this section, the Markov Decision Process (MDP) model for the mediator environment is introduced as below:

1) State definition: State space S = [L, D, TTDU, M]. L (Automation Level) stores the current level of automation that the vehicle is in, which includes L0 (no automation), L2 (partial automation), L3 (conditional automation), L4 (full automation). D (Distraction Level) describes the driver's fatigue level, ranging from F0 (alert) to F3 (sleepy). TTDU (Time to Driver Unfit) describes the time until the driver becomes unavailable. M (Max Automation Level) is the maximum available automation level.

2) Action definition: Action space A = [DN, CD, SSLx, ESLx, ES]. DN (Do Nothing) is the default action. CD (Correct Distraction) will provide corrections when the driver becomes distracted. SSLx (Suggest Shift Lx) will suggest the user to shift to automation level Lx, while ESLx (Enforce Shift Lx) will shift directly to automation level Lx without asking the user. ES (Emergency Stop) will execute an emergency stop.

3) State transition: P = $P(s'|s,a)$ is the conditional transition probability from state s to state s' at action a. In the action space, ESLx and ES are deterministic actions with a transition probability of 1. While the success probability of SSLx and CD actions depends on whether the user accepts the system's notifications, and the transition probabilities are defined within the mediator environment.

4) Reward design: R = $R(s,a)$, which mentions in Fig. 3. First, the scenario is assessed for criticality based on TTDU. In critical scenarios (higher TTDU), the driver is in a fatigued state, increasing the automation level or making an emergency stop would yield better benefits. Conversely (lower TTDU), maintaining normal vehicle operation and promptly correcting distractions are the encouraged actions.

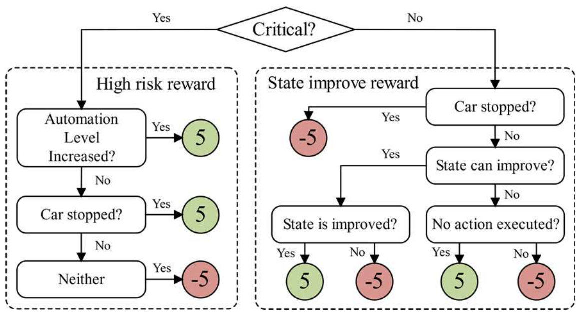


Fig. 3. Reward model of the Mediator environment.

C. Hierarchical Clustering-Based State Grouping

The HCSG-RL algorithm segment the whole state space based on specified metrics to generate a hierarchical tree of state space groups, followed by top-down sub-state space training for each leaf node. The effectiveness of the training is assessed by comparing the performance of the overall model before and after training. Upon successful training, the corresponding sub-model and sub-state are integrated into the primary policy. If not, the sub-state space is skipped, and training proceeds to the next leaf node's sub-state space. The detailed process is as follows:

To evaluate the success of training on a sub-state space, we compare the performance of the model before and after training. Specifically, we conduct scenario testing in the sub-state space with a certain number of rounds, and calculate the average reward based on (1)

$$R = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{T_n}\sum_{t=1}^{T_n} r(t) \tag{1}$$

where N is the total testing rounds, T_n is the total epochs in the n-th round, and $r(t)$ denotes the reward obtained at each step. If the model outperforms after training, we consider the training to be successful and fit into the main policy network.

Consider that each action not only has an immediate impact, but may also have long-term consequences, we initialize the environment with the sub-state space's state so that the agent can discover what choices produce the best long-term benefits when beginning in the present sub-state. This method evaluates the long-term consequences of each action and helps to prevent suboptimal options.

Algorithm 1: Hierarchical clustering based state grouping

```
Data: Maximum state dimension $d_{max}$ and training episodes $L$
Result: Master Policy $M$ and sub-state groups $G_M$
1  Initialize master policy and state cluster $s_0$= all;
2  Train base agent $A_0$ on environment $env_0$ with whole state space for $L$ episodes;
3  Add $A_0$ in $M$(0) and $s_0$ in $G_M$(0);
4  Initialize current state dimension $d$ = 1;
5  while $d < d_{max}$ do
6     Find all state clusters $s_d$ at state dimension $d$;
7     for each state $s$ in $s_d$ do
8        Initialize environment $env_s$ with state $s$;
9        Train agent $A_s$ on for $L$ episodes;
10       Test agent $A_s$ in $env_s$ and get reward $R_A$;
11       Test current policy network $M$ in and get reward $R_M$;
12       if $R_A > R_M$ do
13          Add $A_s$ in $M$ and $s$ in $G_M(d)$;
14       end
15    end
16    $d = d + 1$;
17 end
```

The HSCG algorithm is depicted in Algorithm 1. Line 1-3 show the initialization of the main policy network and the training of the full state space model. Line 6-15 show the partitioning of the sub-state space of the current dimension, the comparison with the pre-training model after training the model on the sub space, and the update process of adding the successfully trained model to the main policy network.

D. Master Policy

The main work flow of the master policy is shown in Fig. 4. The master policy is a strategy used to match states with agents, selecting the most appropriate agent network for decision-making in the current round based on different state inputs.

First, these hierarchical metrics are extracted from the input states and form customized states. Then the master policy determines which of the sub-policies to use given the customized state, and sub-policies are responsible for selecting actions based on the state input. These sub-policies are the agents trained after HCSG-RL training.

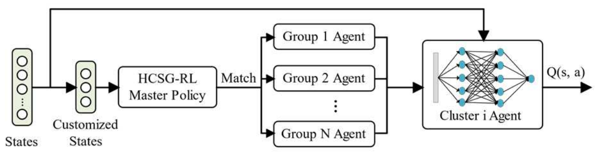


Fig. 4. Workflow diagram of the master policy.

IV. Experiment and Reslut

A. Experiment Setup

Our experiments are conducted in Mediator environment and run on a Intel Core i7-8750H CPU device with 8GB of RAM. The networks of DDQN agents are implemented base on the open source framework Pytorch. The training parameters of the DDQN algorithm with ReLU as activation function are seen in TABLE I.

TABLE I. TRAINING PARAMETERS OF THE DDQN ALGORITHM

Parameter	Value
Learning rate	0.0005
Batch size	8
Replay buffer size	10^5
Target Q network update period	4
$[\epsilon_{max}, \epsilon_{min}]$	[0.5, 0.02]
ϵ decay	0.995
τ	0.01
discount γ	0.99

As for the evaluation of the HCSG algorithm performance in training in multiple environments. We compare with transfer learning and decision tree algorithm. The two baseline methods are as described below:

- Base model: the DDQN model without state grouping that only uses Agent A_0 in Algorithm 2 Line 2.
- Tree model: the Rule-based model, where rules are pre-set based on practical experience.

We define four evaluation metrics, including the average episode reward, the case completion rate, driver unfit case ratio, driver unfit duration, and driver unfit time ratio.

Case completion rate

$$F = \sum_{n=0}^{N} f(n) \tag{2}$$

where the case completion rate is F, N is the number of testing rounds and $f(n)$ indicates whether the vehicle arrives at the destination. If the vehicle does not reach the destination due to an emergency stop in the n-th round, then $f(n) = 0$, otherwise, $f(n) = 1$.

Driver unfit case ratio, driver unfit duration and driver unfit time ratio are written as:

$$UCR_{unfit} = \frac{1}{N}\sum_{n=0}^{N} c(n) \tag{3}$$

$$UD_{unfit} = \frac{1}{N}\sum_{n=0}^{N}\sum_{t=0}^{T_n} u(t) \tag{4}$$

$$UTR_{unfit} = \frac{1}{N}\sum_{n=0}^{N}\frac{1}{T_n}\sum_{t=0}^{T_n} u(t) \tag{5}$$

where driver unfit case ratio is UCR_{unfit}, driver unfit duration is UD_{unfit} and driver unfit time ratio is UTR_{unfit}. N is the total episode number. T_n is the single episode length. We define the unfit state as $TTDU_t = 0$ and $autoLevel_t < L3$ at time step t. $c(n) = 1$ when the unfit state appears at episode n, otherwise $c(n) = 0$. $u(t) = 1$ when the comfort state is unfit at time step t, otherwise $u(t) = 0$.

B. State Grouping

In the HCSG algorithm, we select automation level (L0, L2, L3, L4) and distraction level (F0, F1, F2, F3) as the metric for states grouping, and train sub-models accordingly. The results are shown in Fig. 5, where each sub-state is trained separately based on the state grouping. The green blocks indicate that the sub-model trained in the current subspace outperformed the original model, and the white blocks represent failed training. In total, we obtained 13 models with 4 in the first dimension and 9 in the second dimension.

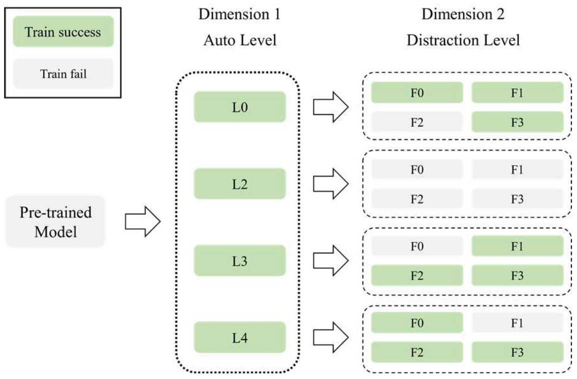


Fig. 5. The result of grouping states in different dimensions.

C. Model Performance

To compare the model performance of the proposed HCSG-RL method, the Base model and the Rule-based Tree model, we designed 1000 episodes of test experiments for each algorithm in random Mediator environment with different road lengths (5, 10, 20 and 30km), evaluating their performance using the average episode reward. The results are shown in the Fig. 6.

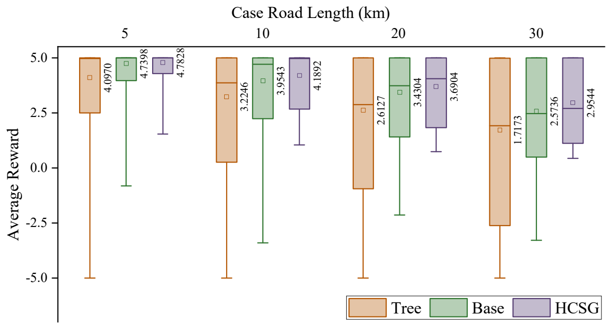


Fig. 6. The average episode reward with different road lengths.

As expected, longer road length results in decreased average rewards across all models, where the probability of tail states appearing in the "long-tail" distribution increases. Notably, the HCSG model consistently outperform the Base model and Tree model, exhibiting the highest mean and minimum values for the average reward, and a more stable reward distribution indicated by smaller boxplots over trials. Moreover, as shown in the TABLE II. , the advantages of the proposed model become more pronounced in challenging situations (i.e., longer road lengths), where there is a 72% improvement in rewards compared to the Tree model. By leveraging state grouping, the proposed model is able to thoroughly explore the entire state space, mitigating to some extent the issue of "long-tail" distribution in sampled data, and thereby achieving better performance limits.

TABLE II. DRIVER UNFIT DURATION ON THE BASE ENVIRONMENT FOR OUR HCSG MODELS AND BASELINE

Models	Average episode reward R on the base environments in different road length			
	5m	10m	20m	30m
HCSG	4.7828 (↑116.7%)	4.1892 (↑129.9%)	3.6904 (↑141.2%)	2.9544 (↑172.0%)
Base	4.7398 (↑115.7%)	3.9543 (↑122.6%)	3.4304 (↑131.3%)	2.5736 (↑149.9%)
Tree	4.0970 (100%)	3.2246 (100%)	2.6127 (100%)	1.7173 (100%)

D. Case Completion Rate

In the test environment of autonomous driving vehicles, there are cases where a vehicle makes an emergency stop during the trip, resulting in early termination of the round and failure to reach the destination. It usually means that the system encounters problems when making decisions and cannot complete the task smoothly while ensuring driving safety. We conducted test experiments at different road lengths (5, 10, 20, and 30 km) and observed whether different models could make reasonable decisions to avoid emergency stops by calculating task completion rates. The results are shown in Fig. 7.

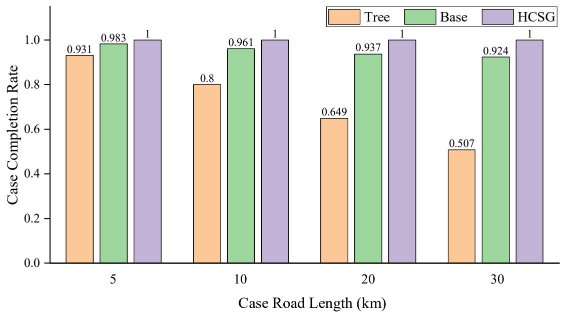


Fig. 7. The results of *case completion rate* with different road lengths.

The HCSG model effectively handled emergency situations by making appropriate decisions in driving mode switching, consequently avoiding emergency stops and ensuring smooth vehicle operation across all test scenarios. In contrast, the Base model and Tree model demonstrated reduced task completion rates as the road length increased. Notably, the Tree model exhibited the poorest adaptability, with a pass rate of 50.07% in the 30 km scenario.

E. Driver Fitness

Driver fitness test models' abilities in optimizing driving safety. We used 3 indicators to indicate driver fitness: *Driver Unfit Case Ratio*, *Driver Unfit Duration*, and *Driver Unfit Time Ratio*.

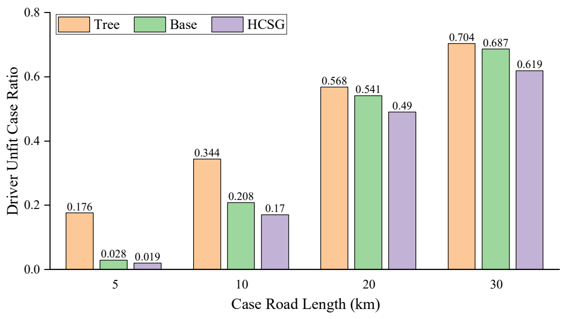


Fig. 8. The results of *driver unfit case ratio* with different road lengths.

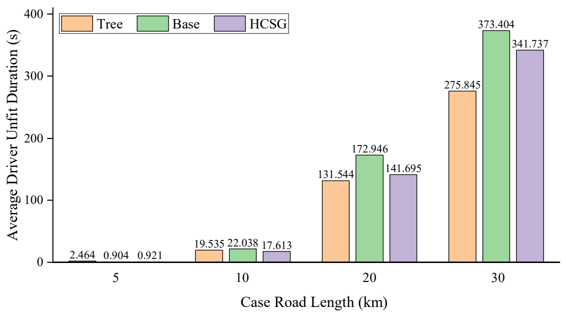


Fig. 9. The results of *driver unfit duration* with different road lengths.

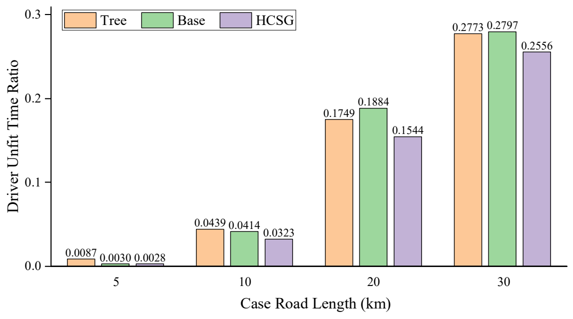


Fig. 10. The results of *driver unfit time ratio* in different case road length.

Fig. 8 shows the results of the UCR_{unfit}, which is defined as the proportion of discomfort episodes in total episodes. The driver is more likely to endure discomfort as the route lengthens and the driving environment becomes more complex. The proposed model demonstrates improvements of at least around 10% compared to the two baseline methods and the improvement compared to the base model is nearly 20% in shorter road length case, which can effectively reduce driver discomfort.

The results of UD_{unfit} are displayed in Fig. 9. It demonstrates the duration of driver discomfort increases with longer scenario lengths. The HCSG model, by selecting more appropriate driving modes, reduces driver discomfort duration and achieves a 10-20% improvement compared to the base model. Although the Tree model appears to have the lowest driver unfitness duration, considering the *Case Completion Rate*, it tends to apply emergency stops in unforeseen situations, resulting in early termination of the test.

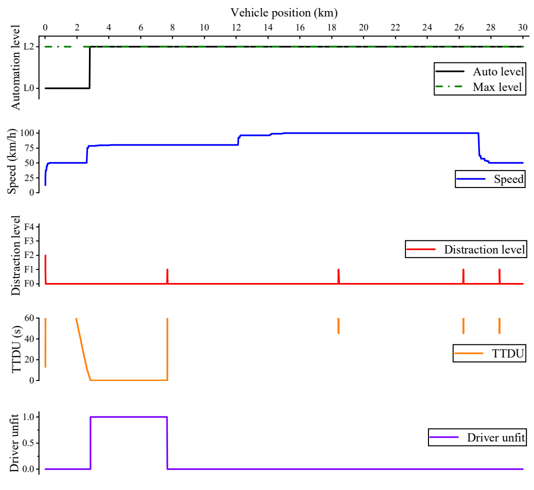
To account for differential driving distances and times among different models (where the Base and Tree models may not complete the entire scenario and therefore have reduced driving distance and time), we introduced UTR_{unfit}, which is defined as UD_{unfit} divided by the total driving time. This metric allows for a more realistic evaluation of model performance in terms of optimizing driver safety. As shown in the Fig. 10, the proposed model exhibited the lowest ratio of unfit time, reducing by 2-3% compared to other models when the road length reaches 20-30 km. For the Tree model, even though the UD_{unfit} is relatively short, it still account for a significant portion of their total driving time.

According to the above experiments, the HCSG-RL model can make reasonable decisions in response to emergency situations to alleviate driver discomfort and ensure that the vehicle can safely complete the test scenario. In general, the HCSG-RL model significantly outperformed the Tree model and Base model.

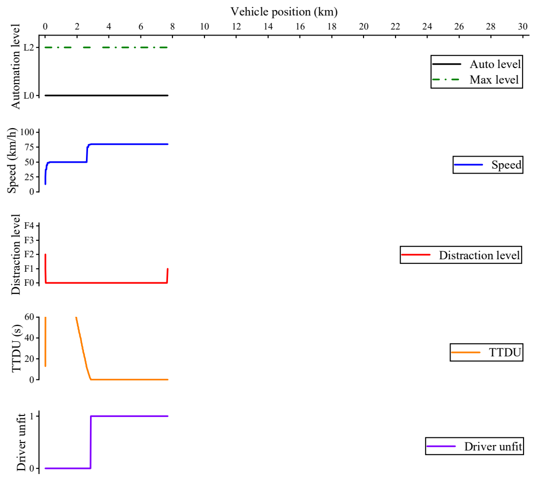
F. Case Study

In this section, we take a specific case as an example to intuitively observe the advantages of the HCSG model. The case comprised a 30-km road which contains urban road (0-2.6km), provincial road (2.6-12.1km), highway road (12.1-27.2km), urban road (27.2-30km), and different roads types have different driving speed limits.

Fig. 11 shows the performance of HCSG model and Tree model. When the vehicle transitions from an urban road to a provincial road (at 2.6 km), the HCSG agent determines to switch the driving level to L2. When the driver continues to be unfit and becomes distracted (at 7.5 km), the Tree model performs an emergency stop due to the current L0 Auto-level and the HCSG takes corrective action (reminding the driver) based on the current L2 Auto-level, allowing the vehicle safely pass through the entire scenario. The experimental results show that when higher Auto-level is available, the HCSG model can timely take into account the vehicle situation and switch Auto-level. This improves the ability to handle driver fatigue and prevents emergency stop caused by driver distraction, allowing the vehicle to operate safely and maintain normal performance. However, the rule-based Tree model is limited in finding solutions based on actual situations, which fails to take reasonable actions against driver unfit and leads to emergency stop.



(a)



(b)

Fig. 11. Model performance comparison on a case. (a) Test result of HCSG-RL agent on the base environment. (b) Test result of decision-tree agent on the base environment.

V. Conclusions

In this work, we propose an autonomous driving level switching decision method called hierarchical clustering-based state grouping reinforcement learning (HCSG-RL) to solve the "long-tail" issue and the cruse of dimensionality issue. This method partitions the whole state space into groups and trains corresponding sub-agents, allowing for a more fully exploration on the whole state space and achieving higher training efficiency for the sub-agents. As the simulation results show that the proposed method could perform better in environments with different settings compared with the two baseline models. Besides, the proposed method avoids driver unfit situations, improving driving safety. At last, we explain how the HCSG-RL model avoids driver unfit by a case representation.

In future work, the HCSG-RL model could expand more and deeper dimensions by introducing more state indicators. A smarter master algorithm might give more efficient results in higher HCSG dimensions. Besides, a deeper/wider neural net or more complicated network architecture might be more generalized than small neural nets.

References

[1] N. Zangi, R. Srour-Zreik, D. Ridel, H. Chassidim, and A. Borowsky, "Driver distraction and its effects on partially automated driving performance: A driving simulator study among young-experienced drivers," *Accident Analysis & Prevention*, vol. 166, p. 106565, Mar. 2022.

[2] I. Y. Noy, D. Shinar, and W. J. Horrey, "Automated driving: Safety blind spots," *Safety Science*, vol. 102, pp. 68–78, Feb. 2018.

[3] F. van Wyk, A. Khojandi, and N. Masoud, "Optimal switching policy between driving entities in semi-autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 517–531, May 2020.

[4] F. Biondi, I. Alvarez, and K.-A. Jeong, "Human–vehicle cooperation in automated driving: A multidisciplinary review and appraisal," *International Journal of Human–Computer Interaction*, vol. 35, no. 11, pp. 932–946, Jul. 2019.

[5] Z. Lu, R. Happee, C. D. D. Cabrall, M. Kyriakidis, and J. C. F. de Winter, "Human factors of transitions in automated driving: A general framework and literature survey", *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 43, pp. 183–198, Nov. 2016.

[6] F. Vanderhaegen, "Cooperation and learning to increase the autonomy of ADAS", *Cogn Tech Work*, vol. 14, no. 1, pp. 61–69, Mar. 2012

[7] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[8] J. Li, X. Wu, M. Xu, and Y. Liu, "Deep reinforcement learning and reward shaping based eco-driving control for automated HEVs among signalized intersections," *Energy*, vol. 251, p. 123924, Jul. 2022.

[9] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A Study on overfitting in deep reinforcement learning." *arXiv*, Apr. 20, 2018.

[10] J. Wang et al., "Synergistic effect of well-defined dual sites boosting the oxygen reduction reaction," *Energy Environ. Sci.*, vol. 11, no. 12, pp. 3375–3379, 2018.

[11] Y. Wang, W. Gan, J. Yang, W. Wu, and J. Yan, "Dynamic curriculum learning for imbalanced data classification," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 5016–5025.

[12] Y. Zang, C. Huang, and C. Change Loy, "FASA: Feature augmentation and sampling adaptation for long-tailed instance segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 3437–3446.

[13] X. Wang, L. Lian, Z. Miao, Z. Liu, and S. X. Yu, "Long-tailed recognition by routing diverse distribution-aware experts." *arXiv*, May 01, 2022.

[14] S. Park, J. Lim, Y. Jeon, and J. Y. Choi, "Influence-balanced loss for imbalanced visual classification." *arXiv*, Oct. 05, 2021.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[16] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning." *arXiv*, Dec. 08, 2015.

[18] P. Lu, X. Wu, W. Guo, and X. C. Zeng, "Strain-dependent electronic and magnetic properties of MoS2 monolayer, bilayer, nanoribbons and nanotubes," *Phys. Chem. Chem. Phys.*, vol. 14, no. 37, pp. 13035–13040, Aug. 2012.